Version: v4

Initialization

The main entry point of NextAuth.js is the `NextAuth` method that you import from `next-auth`. It handles different types of requests, as defined in the REST API section.

> **INFO**
>
> NextAuth.js cannot use the run Edge Runtime for initialization. The upcoming `@auth/nextjs` library (which will replace `next-auth`) on the other hand will be fully compatible.

You can initialize NextAuth.js in a few different ways.

Simple initialization

API Routes (`pages`)

In Next.js, you can define an API route that will catch all requests that begin with a certain path. Conveniently, this is called Catch all API routes.

When you define a `/pages/api/auth/[...nextauth]` JS/TS file, you instruct NextAuth.js that every API request beginning with `/api/auth/*` should be handled by the code written in the `[...nextauth]` file.

/pages/api/auth/[...nextauth].ts

```
import NextAuth from "next-auth"

export default NextAuth({
  ...
})
```

Here, you only need to pass your **options** to `NextAuth`, and `NextAuth` does the rest.

This is the preferred initialization in tutorials/other parts of the documentation, as it simplifies the code and reduces potential errors in the authentication flow.

Route Handlers (`app/`)

Next.js 13.2 introduced **Route Handlers**, the preferred way to handle REST-like requests in App Router (`app/`).

You can initialize NextAuth.js with a Route Handler too, very similar to API Routes.

/app/api/auth/[...nextauth]/route.ts

```
import NextAuth from "next-auth"

const handler = NextAuth({
  ...
})

export { handler as GET, handler as POST }
```

Internally, NextAuth.js detects that it is being initialized in a Route Handler (by understanding that it is passed a Web **`Request` instance**), and will return a handler that returns a **`Response` instance**. A Route Handler file expects you to export some named handler functions that handle a request and return a response. NextAuth.js needs the `GET` and `POST` handlers to function properly, so we export those two.

> **INFO**
>
> Technically, in a Route Handler, the `api/` prefix is not necessary, but we decided to keep it required for an easier migration.

Advanced initialization

> **INFO**

> The following describes the advanced initialization with API Routes, but everything will apply similarily when using Route Handlers too. Instead, `NextAuth` will receive the first two arguments of a Route Handler, and the third argument will be the auth options

If you have a specific use case and need to make NextAuth.js do something slightly different than what it is designed for, keep in mind, the `[...nextauth].ts` config file is just **a regular** API Route.

That said, you can initialize NextAuth.js like this:

/pages/api/auth/[...nextauth].ts

```
import type { NextApiRequest, NextApiResponse } from "next"
import NextAuth from "next-auth"

export default async function auth(req: NextApiRequest, res: NextApiResponse) {
  // Do whatever you want here, before the request is passed down to `NextAuth`
  return await NextAuth(req, res, {
    ...
  })
}
```

The `...` section will still be your **options**, but you now have the possibility to execute/modify certain things on the request.

You could for example log the request, add headers, read `query` or `body` parameters, whatever you would do in an API route.

> **TIP**
>
> Since this is a catch-all route, remember to check what kind of NextAuth.js "action" is running. Compare the REST API with the `req.query.nextauth` parameter.
>
> For example to execute something on the "callback" action when the request is a POST method, you can check for `req.query.nextauth.includes("callback") && req.method === "POST"`

> **NOTE**

> `NextAuth` will implicitly close the response (by calling `res.end`, `res.send` or similar), so you should not run code **after** `NextAuth` in the function body. Using `return NextAuth` makes sure you don't forget that.

Any variable you create this way will be available in the `NextAuth` options as well, since they are in the same scope.

/pages/api/auth/[...nextauth].ts

```
import type { NextApiRequest, NextApiResponse } from "next"
import NextAuth from "next-auth"

export default async function auth(req: NextApiRequest, res: NextApiResponse) {

  if(req.query.nextauth.includes("callback") && req.method === "POST") {
    console.log(
      "Handling callback request from my Identity Provider",
      req.body
    )
  }

  // Get a custom cookie value from the request
  const someCookie = req.cookies["some-custom-cookie"]

  return await NextAuth(req, res, {
    ...
    callbacks: {
      session({ session, token }) {
        // Return a cookie value as part of the session
        // This is read when `req.query.nextauth.includes("session") &&
req.method === "GET"`
        session.someCookie = someCookie
        return session
      }
    }
  })
}
```

A practical example could be to not show a certain provider on the default sign-in page, but still be able to sign in with it. (The idea is taken from this discussion):

/pages/api/auth/[...nextauth].ts

```
import NextAuth from "next-auth"
import CredentialsProvider from "next-auth/providers/credentials"
import GoogleProvider from "next-auth/providers/google"

export default async function auth(req, res) {
  const providers = [
    CredentialsProvider(...),
    GoogleProvider(...),
  ]

  const isDefaultSigninPage = req.method === "GET" &&
req.query.nextauth.includes("signin")

  // Will hide the `GoogleProvider` when you visit `/api/auth/signin`
  if (isDefaultSigninPage) providers.pop()

  return await NextAuth(req, res, {
    providers,
    ...
  })
}
```

For more details on all available actions and which methods are supported, please check out the **REST API documentation** or the appropriate area in **the source code**

This way of initializing `NextAuth` is very powerful, but should be used sparingly.

> **DANGER**
>
> Changing parts of the request that is essential to `NextAuth` to do its job - like messing with the default cookies - can have unforeseen consequences, and have the potential to introduce security holes if done incorrectly. Only change those if you understand consequences.

Edit this page

*Last updated on **May 16, 2024** by **Codie Newark***