**Let’s Learn About Middleware In Express.js -** [Total nerd](https://devdotpy.medium.com/?source=post_page-----174e839ffffa--------------------------------) Jun 16, 2022

A brief and fun introduction to the concept of middleware in Express.js

You’ve probably heard of middleware if you’ve been around Node.js long enough, and I’m not going to lie, I was confused as f\*\*\* when I first came across the WORD ITSELF so let alone the implementation of middleware but you’ve got to get your hands dirty to learn so please allow me to gently teach you what you need to know about middleware as simple as possible.

**SETTING UP OUR DEVELOPMENT ENVIRONMENT**

Open up your terminal if you’re on Linux or Mac, command prompt if you’re on Windows

mkdir middleware && cd middleware to create a new directory and navigate into it

npm init --y to initialize a new npm project with the default settings ( you can play around with the package.json later as I’ll show you )

npm i express && npm i --save-dev nodemon this will firstly install express as we need it to construct our backend, then we’ll install nodemon and save it as a dev dependency as we don’t really need it in a real production environment

**ADDING A SCRIPT TO package.json**

If you open up your package.json file you’ll find a section named scripts , this section contains “shortcuts” to run commands in the terminal and it’s crazy useful so let’s add a tiny script to run nodemon just to look like we’re cool n all that.

"dev": "nodemon index.js" add this after the “test” script

To trigger those scripts we simply type in the terminal npm run followed by the script name as defined in our scripts section so in our case we type npm run dev but let’s use this a little later :)

**WHAT IS A MIDDLEWARE ANYWAY.**

A middleware is basically a function that executes **BEFORE**the route handler function……. ik this wasn’t so simple so let’s have a look at a very basic express server.



OK so nothing’s too fancy there right ? We have three routes / , /users , /articles and we have the *route handler function* for each one of those routes

Run npm run dev and you should see the little message saying “up and running”

Our middleware will execute before every request to those routes or before a specific route as we have the choice to specify that

A middleware is basically a function with access to the request and response object just like a route handler, and it could be used to maybe log every request made to our server or to check if a user is logged in, authenticate and authorize users to protect certain resources and many other purposes

**MIDDLEWARE VS. ROUTE HANDLER**

The big difference about a middleware and a route handler **is the next() function**

next() function is the third parameter to a middleware and it basically tells the backend to “go on with the next middleware” or in other words “give it the green light to continue with the next routine” so our page doesn’t just freeze and never finish loading so if we forget to put it, the program won’t continue because it’s stuck in the middleware

**TYPES OF MIDDLEWARE**

As mentioned above we have two types of middleware, one that runs at the top level of the app so it’s applied to all the route handlers like a middleware to check if the user is logged in and we can apply it to all the posts routes of some social media app we’re writing

Example of a default express middleware is the express.json() middleware which we use at the top all the time to enable accepting and parsing json

The other type is a middleware that runs before a specific route, maybe you need to protect some route that has access to admin-only articles or some VIP content access for your app

**WRITING OUR FIRST MIDDLEWARE**

So enough talking and let’s see how to write our own middleware which will give us a delicious cookie.

Add this code snippet after our imports

const say\_hi = (req, res, next) => {  
 console.log("have a yummy cookie 🍪");  
 next();  
}

To “enable” the middleware at the app level we use express.use(say\_hi); and pay attention we don’t add parenthesis because we’re not calling the function

Now head over to any route and check your terminal for a 🍪

Your index.js now should look like this

const express = require("express"); const app = express();   
const say\_hi = (req, res, next) => {  
 console.log("have a yummy cookie 🍪");  
 next();  
};  
app.use(say\_hi); app.get("", (req, res) => res.send("you've reached the home page")); app.get("/users", (req, res) => res.send("this is the users page")); app.get("/articles", (req, res) => res.send("this is the articles page")); app.get("\*", (req, res) => res.status(404).send("Page not found")); app.listen(3000, () => console.log("up and running"));

**WRITING A MIDDLEWARE FOR A SPECIFIC ROUTE**

Now that we’ve seen how to write a middleware that runs at the app level aka runs before every route, let’s write two more middlewares, one that runs before the users route telling us “i love my users 💕” and one for the articles that says “wow look at all those articles 😮” so leggoooooo

const users\_middleware = (req, res, next) => {  
 console.log("i love my users 💕");  
 next();  
};const articles\_middleware = (req, res, next) => {  
 console.log("wow look at all those articles 😮");  
};

Now in order for us to specifically use those for our routes we need to respectively pass each one as a parameter of the route’s handler function so it looks now something like this



**QUICK NOTES**

* Middleware is just a function that executes after the server receives the request and before the route handles responds to it
* They have access to three parameters, req res and a third parameter which is a function conveniently called next() to allow the next middleware to run
* They can be either at app level so they run before every request to every endpoint or can be specified to a certain route

**CONCLUSION**

Middleware can come off as a complicated concept at first for those who are new to express but they’re quite easy to implement once you find out the true need for them, they’re a powerful tool to regulate your server resources and access control among many other needs since you understand the basic idea behind them.