**Day 46**





**“Web Development + Security”**

**Middleware in Express Js:**

**What is Middleware?**

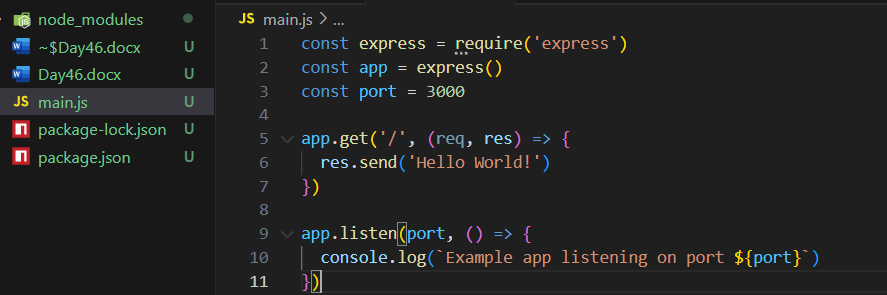
A middleware is a function that runs during the request-response cycle in Express.js.  
It has access to:

* req → the request object
* res → the response object
* next() → a function to pass control to the next middleware or route handler

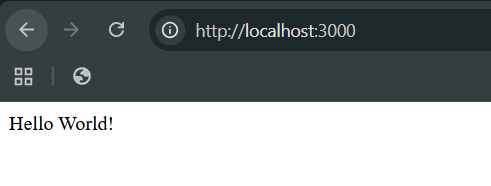
Think of it like a pipeline: every request passes through middleware before reaching the final route handler.

A general template to create:

Main.js:

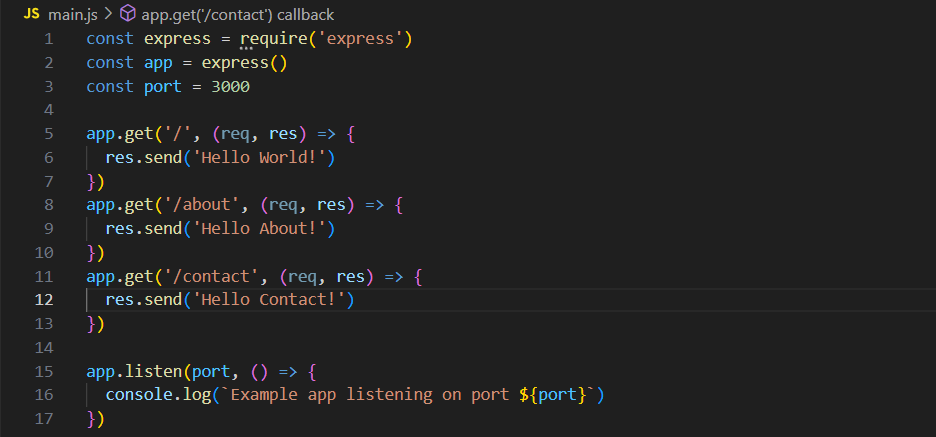


Output:

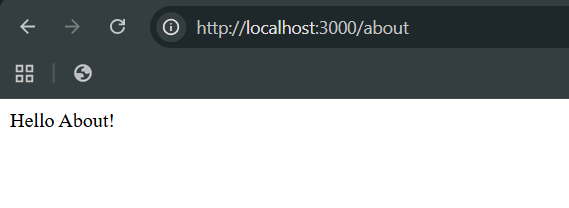


Also, we can create something like this as well:

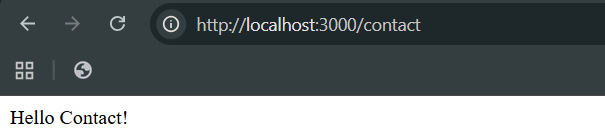
Main.js:



Output: at “/about”



Output: at “/contact”



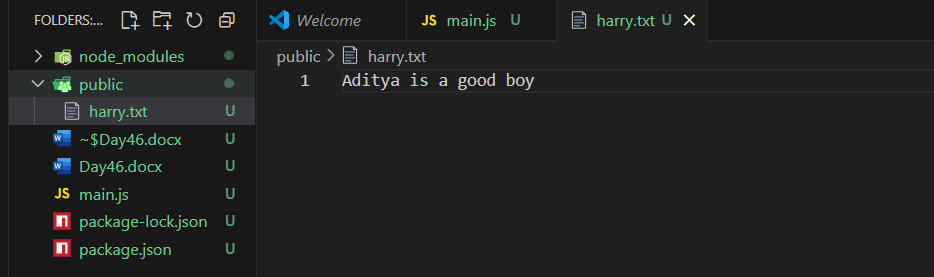
**Why Middleware is Useful**

* Logging requests
* Authenticating users
* Parsing request bodies (JSON, URL-encoded)
* Handling errors
* Serving static files

Without middleware, you’d have to manually write the same logic inside every route — which is messy and repetitive.

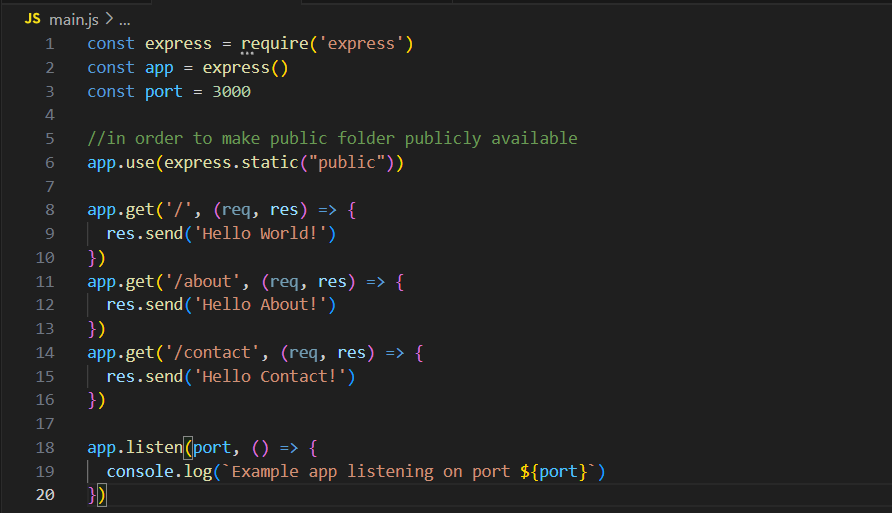
Now, we are using the built in middleware of express js. For which we will first create a “public” folder and inside it we will create “harry.txt” and some content in it.

Public/harry.txt:

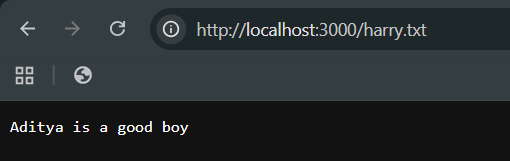


Now, we will be allowing our browser to use it using the following code:

Main.js:

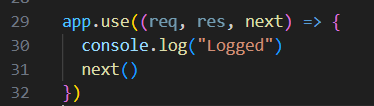


Output: at “/harry.txt”

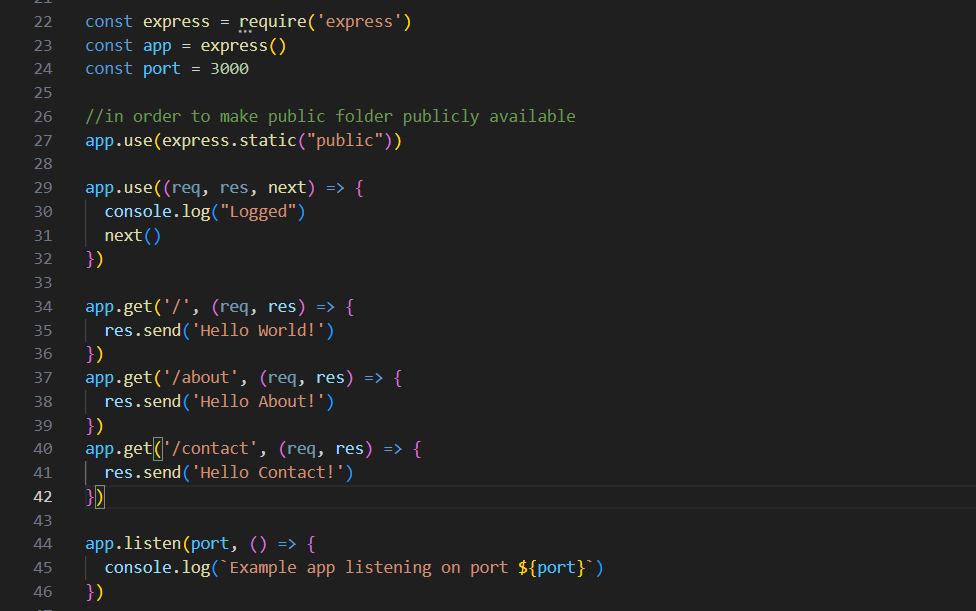


What basically, happened in this example is that first the request is made, then the request Is checked if it is in public folder or not, since it is in public folder. It sends the data to the response form.

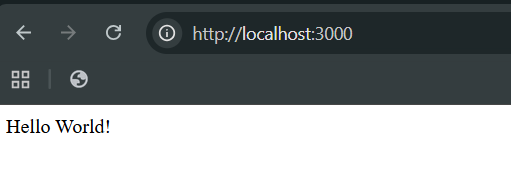
Now, we can write our own middleware using the following part of code:



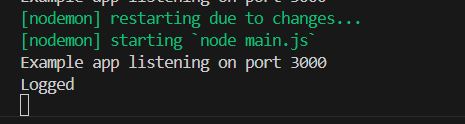
We can see the whole code as:



Output: browser



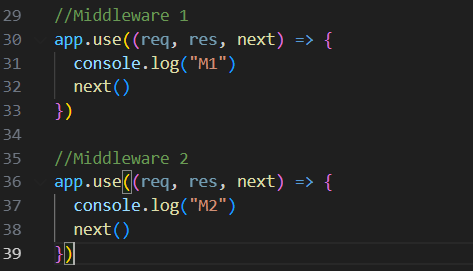
Terminal:



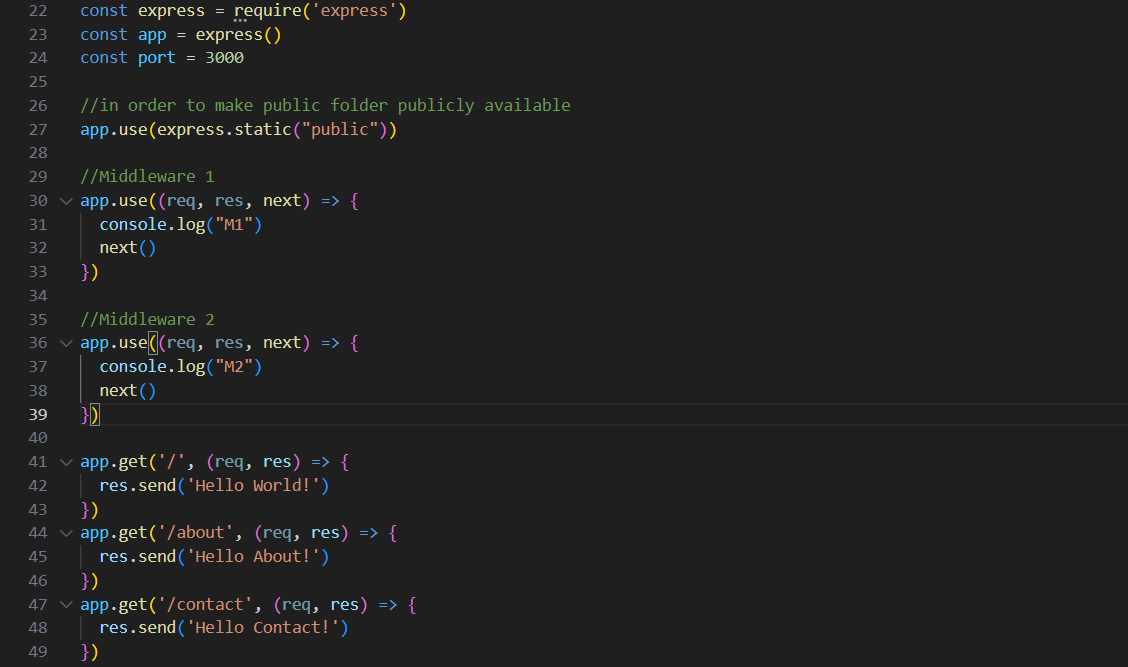
So, what basically happened? We used the middleware, and as since it has the parameters, req it deals with the request, res for response, and next to say that move to the next middleware. So, when website was loaded at main page, then the “hello world!” got printed while “Logged” is observed on the terminal.

Now, to better understand the next() we will add one more middleware as shown below:

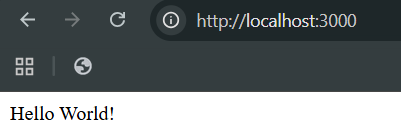
Snippet:



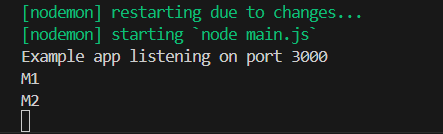
Whole code:



Output: browser



Output: terminal



Clearly, first middleware 1 ran, then printed M1 and then due to next() it says to go to the next middleware, and that’s why M2 printed.

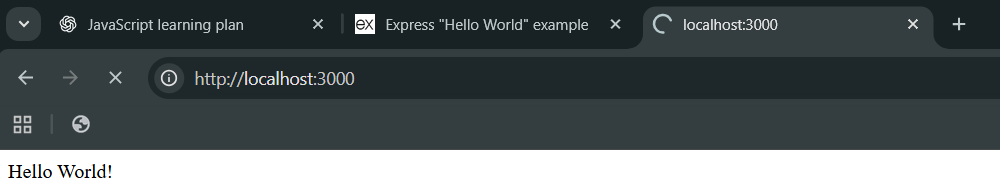
What if we remove the “next()”?

Then we will observe that the response will not come, although the terminal will show the M1 but in browser it will remain loading.

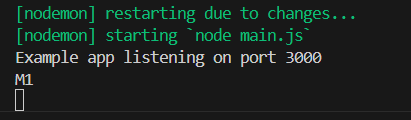
Main.js:



Output: see it is loading.



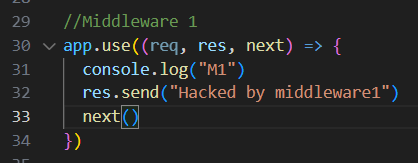
Terminal:



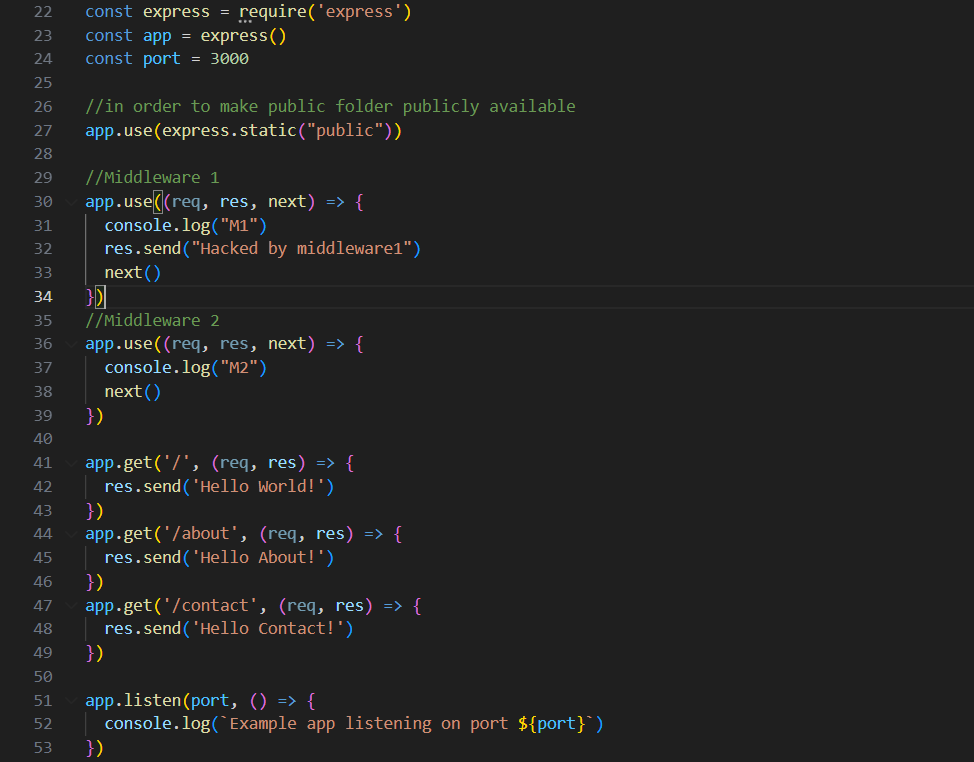
So, basically the control never went from middleware 1 as we had removed “next()”.

Now, suppose if request is sent and still you are giving the controls then it will load the output at browser but at terminal it will show error.

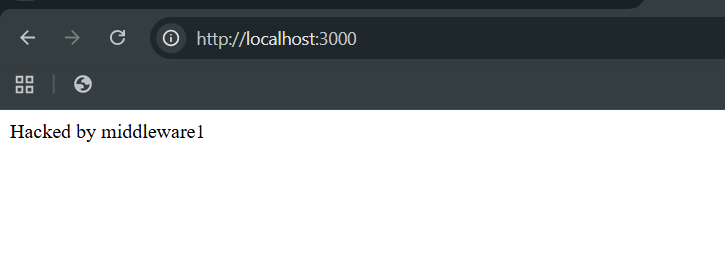
Snippet:



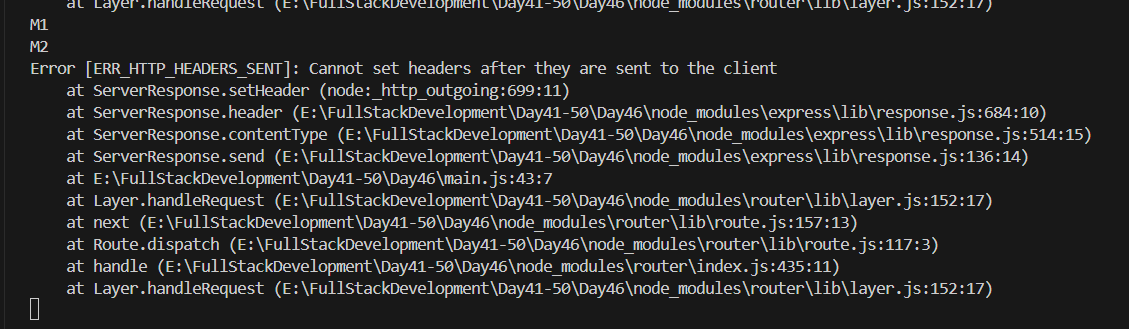
Code:



Output: at browser

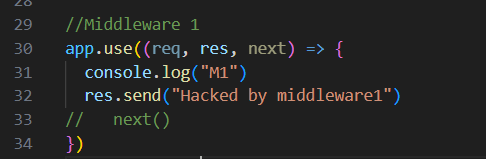


At terminal: error



But what if we want that response is sent by the middleware 1 and the error doesn’t occur, for this we will remove the next() from the middleware 1. (rest code remains same)

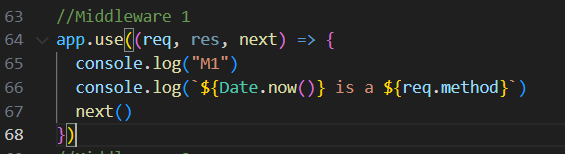
Snippet:



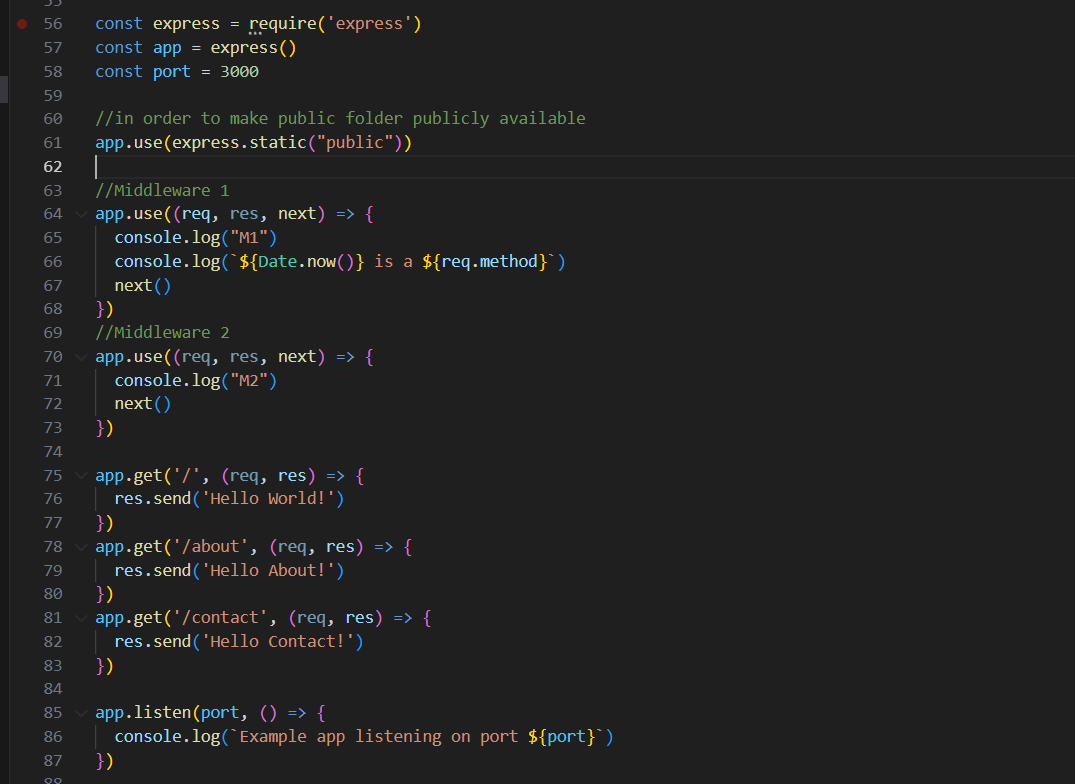
*“So, basically till now, we saw that middleware can change the request, or can also send the request and also can give control to the next middleware.”*

So, why we need middleware? In below example we will understand that we need middleware in order to log the data.

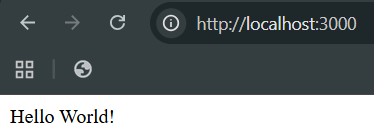
Snippet: we are basically logging the date and request method.



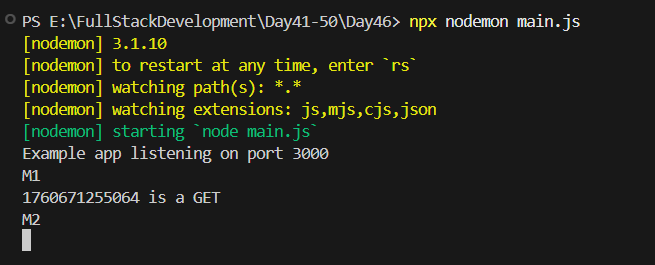
Code:



Output: browser

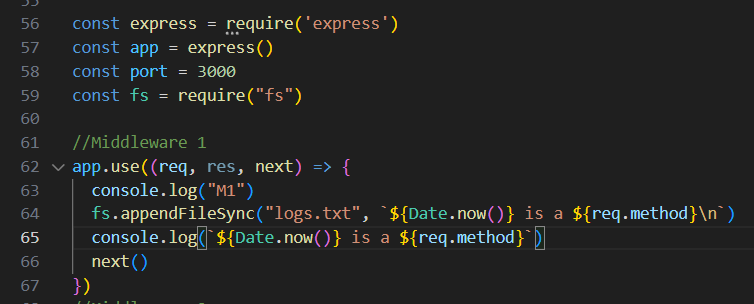


Output: terminal

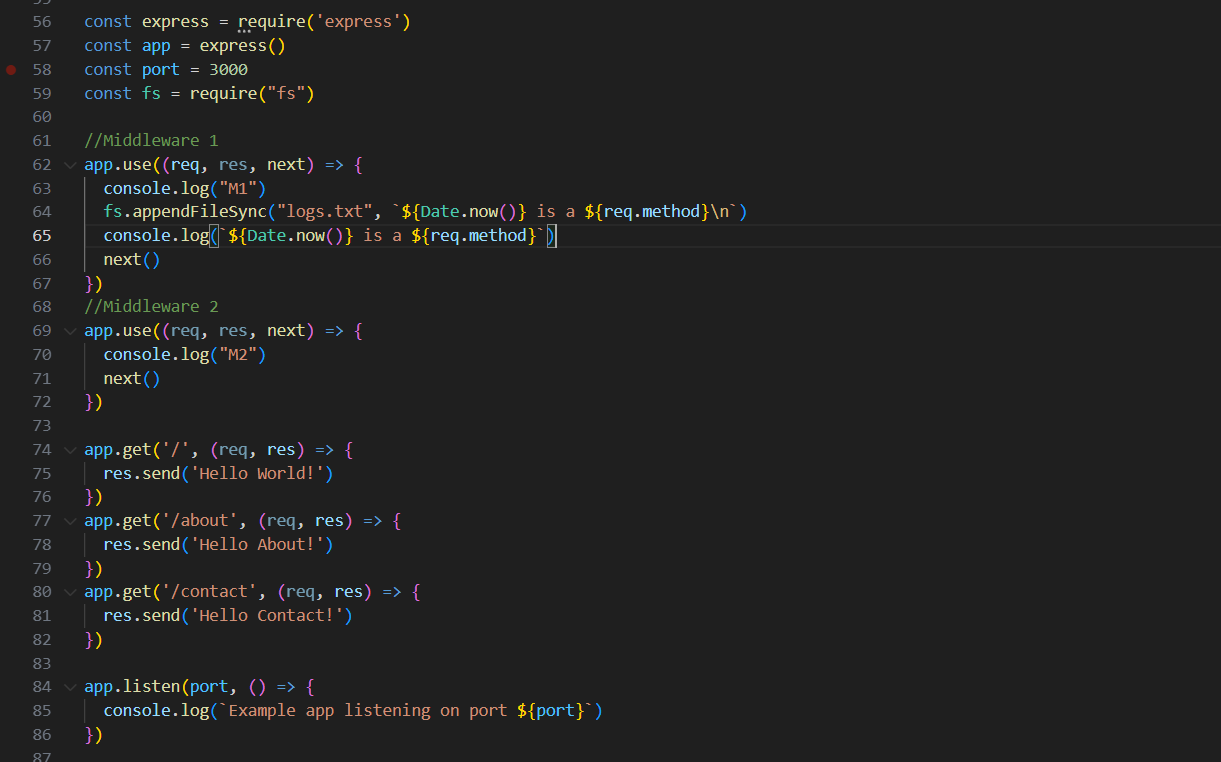


Now, we can also write and append this logged data in a text file: for which we will use “fs” module.

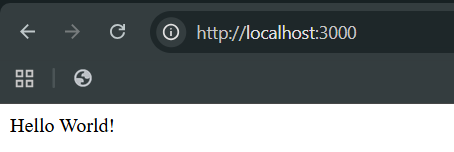
Snippet:



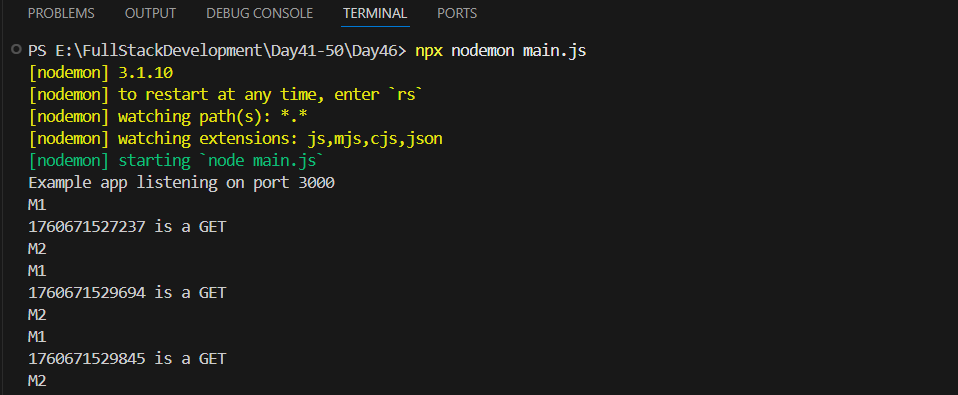
Code:



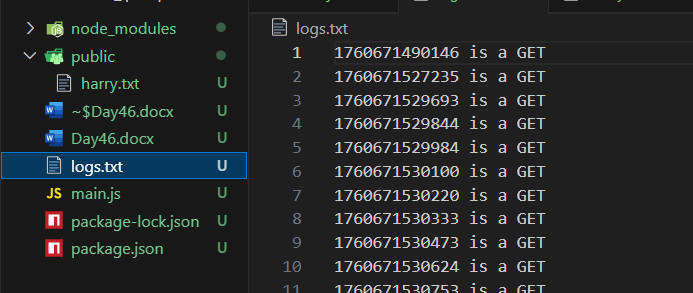
Output: browser



Output: terminal

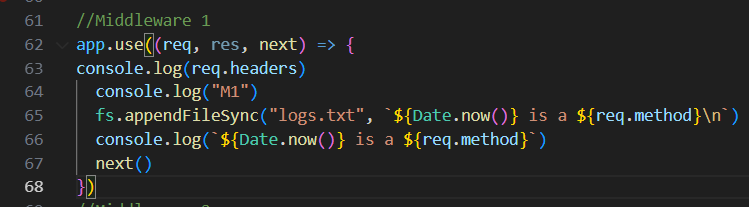


Output: logs.txt



Now, we can also get the headers:

Snippet: we used req.headers and logged it



Code:



Output: terminal



**Types of Middleware**

| **Type** | **Description** | **Example** |
| --- | --- | --- |
| **Application-level** | Used for all routes or specific routes | app.use() |
| **Router-level** | Used for routes in a router | router.use() |
| **Third-party** | Middleware from npm packages | body-parser, cors, helmet |
| **Error-handling** | Handles errors in routes | (err, req, res, next) => {...} |
| **Built-in** | Middleware provided by Express | express.json(), express.static() |

--The End--