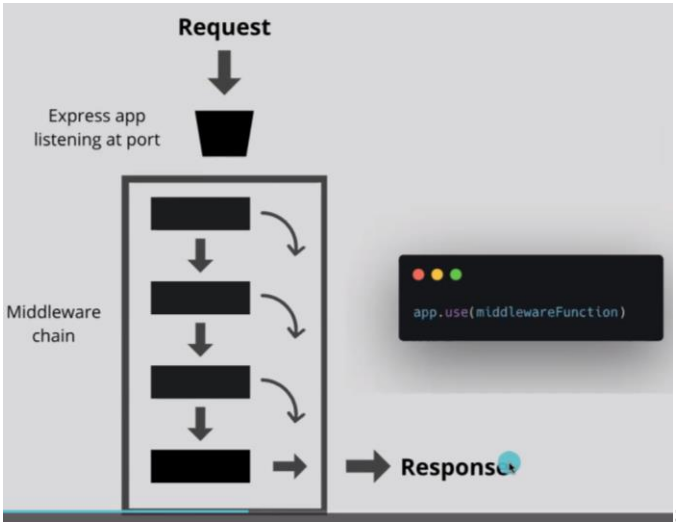
Middleware

A simple server is an oreo cookie – request comes in that's the top layer, cream filling is the stuff that's happening as a result of the request and the response comes back out that's the bottom layer. We can have software that pops in to the chain, during the cream filling to help process the request – this is called middleware.

Restaurant – clear menu which is what your API/docs should be, so the user knows what to expect. Route = menu, Response = what comes to your table on the plate, but lots of things have to happen in between to get the meal ready (e.g. waiter goes to kitchen to give order, kitchen has lots of steps to prepare order)

The process at a glance:

Request comes in
Express app listening at port *x* app.listen(5000)
Middleware chain starts here
Can leave the middleware chain at any point or carry on through depending on logic etc.
But always ends in the response



app.use(middlewareFunction)

We've used app.use before – to set up our Router yesterday
Some middleware is built in to Express already – e.g. App.get and app.listen are types of app.use with specific functions!
You can also write your own middleware as well

Middleware doesn't give us the final response, just a step in the chain to do something then passes the request through to the next bit in the chain – e.g. console.log() won't send a response to the client

Define them before your routers!

Using middleware function?

We use the next argument and next() function to pass request on to the next handler in the chain:

```
function (req, res, next) {
  // Middleware code
  next();
}
```

app.use(function (req, res, next) {
    console.log(req.path);
    next();
})

not specified a path so it applies to all requests!
prints the path request to the console
passes on to the next middleware in the chain

Without the next() function it would hang on this function and not go to the route handler (to give the response!)
Next is not part of the request but is built in to express so can be used as an optional argument in the function

Don't have to define in-line function, can pre-define then call it with app.use()

```
21  function logger(req, res, next) {
22    console.log(req.path);
23    next();
24  }
25
26  app.use(logger);
```

Using POST

Get requests can be without a body, but for POST/PUT/PATCH you need to have some data with it – the body should be a JSON

You can't include this data using browsers like Goole Chrome. Postman is a tool that lets us mock requests to our API without a frontend – download the app or use it on the web: postman.co

Handling POST in your express server

The plan:

- Set up a handler in books router for POST requests to "/books"
- Access the body out of the request object (the data about the new book)
- Add the new book's data to the existing books
- Respond to the client using a responseObject with the info about the new book and the updated books

Need to add first line of code – gives our app the ability to parse the body from the request object first (before we hit the router), using the built-in middleware

This goes towards the top of app.js

App.use(express.json())

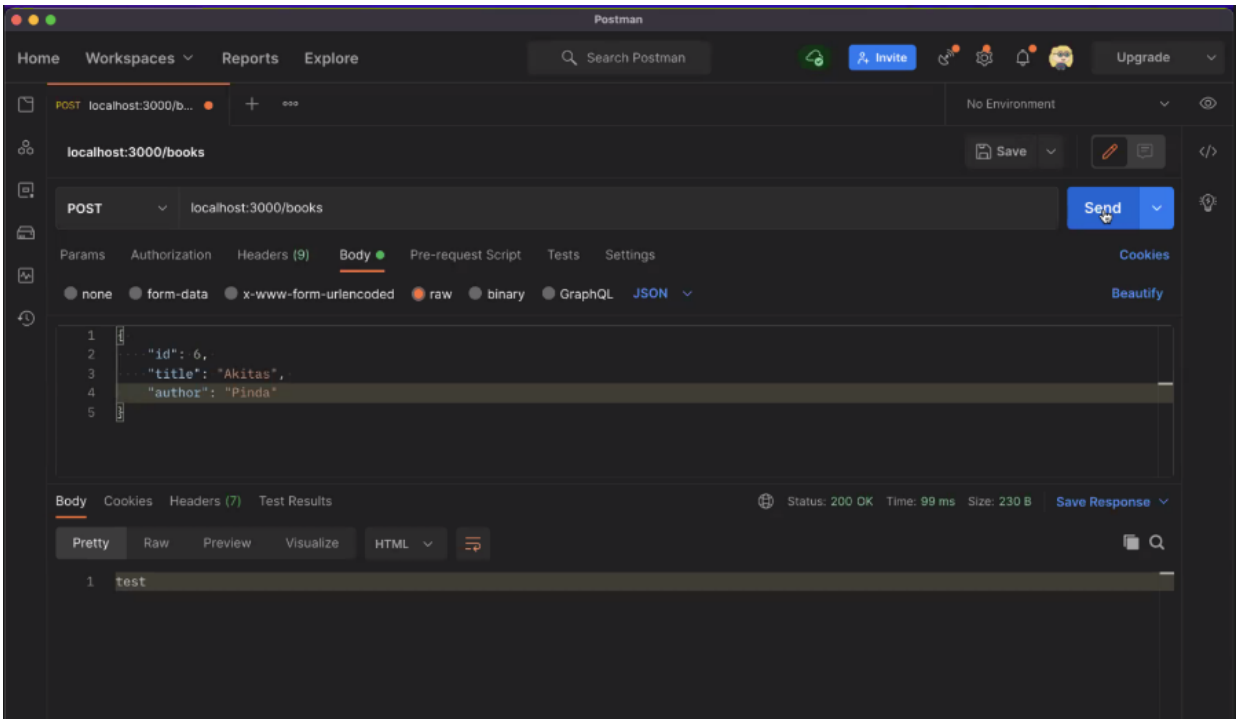
This is middleware that parses request bodies
What this does:
Request comes in
Middleware checks if there's a body
If there's a body, parses and adds to req.body
If no body, does nothing
Calls next() to move on to next step in chain
Screenshot showing what this does under the hood:

This goes in books.js which is handling the /books path

```
router.post("/", function (req, res) {
        const body = req.body;                          access the body from the request object
        books.push(body);      Add the new book's data to the existing books

        const responseObject = {
                success: true,
                message: `added book ${body.title} with id ${body.id} to books data`,
                data: books,
        };
        res.json(responseObject)
})
```

Using Postman as a client to test the above code:



This is only temporary whilst the app is open. Books-data will revert when you close. To make it permanent, we need databases (which is coming next week). If you get a chance, you can use FS (file system) to modify your files permanently!