Episode - 5

Middlewares & Error Handlers

1) What is app.use() method?

→ app.use methods are used to add middlewares in request response cycle.

→ It accept all the HTTP methods by default. if we want any specific HTTP method for a particular use we can use methods like app.get(), app.post() instead of app.use.

→ The order of app.use() calls are matters becoz requests will go through the middleware in the order they are defined.

2) what happens when you don't send the response back inside a route handler?

→ when you don't send a response back the request will be left hanging & client making the request keep on waiting for a response from a server until the timeout occurs.

→ whenever we don't want to send any response back to the client we can call next() method to pass control to the next middleware function.

3) what happens when you call res.send()?

→ when we call res.send() method it sends response back to the client and ends the request-response cycle.

→ Not only that it also sets the necessary headers and status code automatically based on the type of data being sent.

4) what is middleware functions ?

→ function that execute for every request sent to the server.

→ They can modify the request & response objects, ends the req-res cycle (or) pass the control to the next middleware function using next() . It is used for authentication, logging, modifying req/res etc.

→ we can specify that middleware should run for all the routes (or) specific routes.

→ It consits 3 Parameters request, response & next fn.

eg: Middleware for all request

```
app.use ((req, res, next) =>{
        next()
})
```

Middleware for specific routes

```
app.use ("/user", (req, res, next) =>{
        next()
})
```

5) How many middleware fn/ route handlers can one Route have?

→ Single route can have multiple middleware fn/route handlers but it should send only one response back. we can also pass them in array [ ].

→ These handlers are executed in the order they define & move to next handlers using next().

eg:

```
app.use ("/user", (req, res, next) =>{
        console.log("Handler 1")
        next()
},
(req, res, next) =>{
        console.log("Handler 2")
        res.send("Response")
})
```

6) what happens when you send a response in 1st route handler will it still execute the second handler function?

→ when we sent a res in 1st handler fn it sends a res to the client and ends the request so it will not execute the other handler fn defined below.

7) can we call next() function before (or) after sending a response?

→ Calling next() after response send:

→ we can call next() fn after the res is send back to the client to move to the next handler fn but it is unneccessary as the response is already sent we cannot modify it (or) send a new response in the 2nd handler.

→ if we try to send response in both handler function it will result in error: Cannot set headers after they are sent to the client:

→ calling next() before response send:

→ when next() is called before res.send() in 1st handler function it moves to the 2nd handler function and starts executing it once it finishes it return back & send the response back to client.

→ It is also bad practice if the 2nd handler attempts to modify the response.

Only call next() if we are not sending the response in the current handler. If you call res.send() there is no need for next()

8) what happens when you don't send response in any of the handler function just keep calling next()?

→ If we don't send a response and just keep calling next() in all the route handlers for a particular route, the request will continue and executes all the handlers associated with the route. However, if no response is ever sent it will result in a error 404 Not-found by express.

9) what is HTTP response status code & how to set it manually?

→HTTP response status code indicate whether a specific HTTP request has been successfully completed (or) not.

→ Commonly used Status Codes

200 → Success
201 → Created
400 → Bad request
401 → Unauthorized
403 → forbidden
404 → Not found
500 → Internal Server Error.

→ By default status code is 200.

→ we can set the status code manually in the response using status() function.

eg: res.status(401).send("Page Not Found").

10) what is the difference between app.use() & app.all()?

→ Both of these method accept all kind of HTTP Methods but the key difference is app.use() can matches all the routes if no path

is specified (or) to a specific route

→ app.all() always tied to a specific route.

11) How to handle error in Node.js?

There are 2 ways of handling errors

*) using try/catch block.

*) error handling middleware

error handling middleware:

→ It is similar to normal middleware fn but err should be the 1st parameter.

eg:
```
app.use("/", (err, req, res, next) => {
    if (err) {
        res.status(500).send("something went wrong")
    }
    next()
})
```

→ the above code catches the error wherever the error occurs in the entire app by placing this middleware fn at the end.

Always prefer using try/catch block
