Notes

Advantages of Express

- Helps us to write clean code.
- Unless `http` module more can be done by just writing less code.
- The biggest advantage is that express has a lot of `middlewares`, even we can create custom one as well to literally do anything.

Middlewares

The middleware in `node.js` is **a function that will have all the access for requesting an object, responding to an object, and moving to the next middleware function in the application request-response cycle**

- It is just a function, which runs once the request is made but before the response is sent.
- It is written inside the server not between client and server.
- What ever the middleware we write, `app.use()` **will use that middle ware for every route below it.**

```
const express=require("express")

const app=express()

//Our first middleware
app.use((req,res,next)=>{
  console.log("Hello from Middleware")
  next()
})

app.get("/",(req,res)=>{
  console.log("Hello from the base route")
  res.send("Welcome")
})

app.get("/contacts",(req,res)=>{
  console.log("Hello from the contacts route")
  res.send("Contacts")
})

app.get("/about",(req,res)=>{
  console.log("Hello from the about route")
  res.send("About")
})
```

```
app.get("/blogs",(req,res)=>{
  console.log("Hello from the blogs route")
  res.send("Blogs")
})


app.listen(3500,()=>{
  console.log("Running on 3500")
})
```

- This function also has the access to `request` and `response` object.
- It has `next()` function as well. ⇒ Tells us where to go next
- If we go and visit the base route, first the middleware is going to be executed then `next()` will take it to the next thing.
- What will happen if i don't use `next()`?⇒ try it out.
- The middleware that has been written for all the routes, then keep it at top.
- **Try the below code.**

```
const express=require("express")

const app=express()

app.use((req,res,next)=>{
  console.log("Hello from Middleware")
  next()
  console.log("Bye from Middleware")
})

app.get("/",(req,res)=>{
  console.log("Hello from the base route")
  res.send("Welcome")
})

app.listen(3500,()=>{
  console.log("Running on 3500")
})
```

- This will show you how exactly `next()` is working.
- Now, lets see the use of `request` or `response` object
- **What will happen now?**

```
const express=require("express")

const app=express()

app.use((req,res,next)=>{
  if(true){
    res.send("BYE")
  } else {
    next()
  }
})

app.get("/",(req,res)=>{
  res.send("Welcome")
})

app.get("/contacts",(req,res)=>{
  res.send("Contacts")
})

app.get("/about",(req,res)=>{
  res.send("About")
})

app.get("/blogs",(req,res)=>{
  res.send("Blogs")
})


app.listen(3500,()=>{
  console.log("Running on 3500")
})
```

TimeLogger Middleware

- For this create a text file with some dummy data.

```
const express=require("express")

const app=express()

app.use((req,res,next)=>{
  const startTime=new Date().getTime()
  next()
  const endTime=new Date().getTime()
  console.log(endTime-startTime)
})

app.get("/",(req,res)=>{
  res.send("Welcome")
```

```
})

app.get("/contacts",(req,res)=>{
  res.send("Contacts")
})

app.get("/about",(req,res)=>{
  res.send("About")
})

app.get("/blogs",(req,res)=>{
  const data=fs.readFileSync("./lecture.txt","utf-8")
  res.send(data)
})


app.listen(3500,()=>{
  console.log("Running on 3500")
})
```

- Middleware is just a function, and we can declare it outside as well, and then use it inside the `app.use()`

WatchMan Middleware

```
//watchman middleware

const watchMan=(req,res,next)=>{
  if(req.url==="/about"){
    next()
  } else{
    res.send("Bye")
  }
}
```

Multiple Middlewares

```
const express=require("express")

const app=express()

app.use((req,res,next)=>{
  console.log("1")
```

```
  next()
  console.log("2")
})

app.use((req,res,next)=>{
  console.log("3")
  next()
  console.log("4")
})

app.get("/",(req,res)=>{
  console.log("Home")
  res.send("Welcome")
})

app.listen(3500,()=>{
  console.log("Running on 3500")
})
```

- Do the same with `timeLogger` and `WatchMan`
- Can we take out `timeLogger` and `watchMan` and then require them and use them?
- We can also call them within one `app.use()` only, the order will matter over there.

Logger Middleware

```
const logger=(req,res,next)=>{
  fs.appendFileSync("./logs.txt",`\n${req.method}+${req.url}`, "utf-8")
  next()
}
```

- The above middleware will log all the routes visited along with the methods in a separate file just to keep a record.
- We can do anything with the help of `middlewares`.
- **Can I add something to the body of request using middleware while making a post request?**

```
const addStamp=(req,res,next)=>{
  req.body.stamp="Masai Student"
  next()
}

app.post("/addatamp",(req,res)=>{
  console.log(req.body)
```

```
  res.send("Got the data")
})
```

- That means we can manipulate the `request` object as well using them, so you can just understand the power of these things.
- All of these things are `custom` middleware, as we have created them.
- There are `express inbuilt middlewares`. ⇒ `express.json()` ⇒ it just basically parse the json.
- If we need to parse a text only, then ⇒ `express.text()`
- One more important inbuilt middleware is ⇒ `express.Router()`
- make `/addstudent` `/students` `/addteacher` `/teachers` , Then we can actually separate them and route them.

```
//student.router.js

const express=require("express")

const studentRouter=express.Router()

studentRouter.get("/",(req,res)=>{
  res.send("All the students")
})

studentRouter.post("/addstudent",(req,res)=>{
  console.log(req.body)
  res.send("Added the student")
})

module.exports={
  studentRouter
}
```

```
//teacher.router.js

const express=require("express")

const teacherRouter=express.Router()

teacherRouter.get("/",(req,res)=>{
  res.send("All the teachers")
})

teacherRouter.post("/addstudent",(req,res)=>{
  console.log(req.body)
  res.send("Added the teacher")
```

```
})

module.exports={
  teacherRouter
}
```

```
//index.js

const express=require("express")
const {studentRouter} = require("./routes/student.router")
const {teacherRouter} = require("./routes/teacher.router")


const app=express()
app.use(express.json())

app.use("/students",studentRouter)
app.use("/teachers",teacherRouter)

app.get("/",(req,res)=>{
  res.send("This is the home page")
})
```

- There are `community middleware` as well. ⇒ `cors()` , `multer()`
- If i need to use `cors()` , i need to install it.
- `npm i cors`
- Why we use `cors`? ⇒ Research This
- What is `Cors`? ⇒ `Cross Origin Resource Sharing` ⇒ Research This

💡 **IMPORTANT**
⇒ Go through: https://expressjs.com/en/resources/middleware.html

Specifically:
1. CORS: http://expressjs.com/en/resources/middleware/cors.html
2. Multer: http://expressjs.com/en/resources/middleware/multer.html
3. Morgan: http://expressjs.com/en/resources/middleware/morgan.html