Notes

Post Request with HTTP

- Till now we have seen how to make `GET` request using `http` .
- For posting something to the server there should be some `data` that client has to post while making the request.
- Where exactly that `data` is? ⇒ It will be in the **request object** as that is the only way by which the client can interact with server

```
if(req.url === "/adddata" && req.method === "POST"){
    //some logic to get the payload sent by client.
    res.end("Data has been recorded");
}
```

- If we go to browser and hit this endpoint, it will show us invalid endpoint as the browser by default makes get request.
- Use thunder client for `POST`.
- We will get the response that `Data has been recorded`.
- We can use **POSTMAN** as well to work around these things.
- How to get what client is sending?
- `req.body` will not work over here in `http` module.
- It will give us `undefined`.
- **How to actually do it?**

```
if(req.url === "/adddata" && req.method === "POST"){
    //some logic to get the payload sent by client.
    let str = ""
    req.on("data", (chunk)=>{
        str += chunk
    })
    //console.log(str)// will not print it as the event has not
    req.on("end", () => {
        console.log(str) //now we can get the data
    })
    res.end("data has been sent");
}
```

Stream

- A stream is **a sequence of bytes used to hold file data.**
- Go through https://nodejs.org/dist/latest-v18.x/docs/api/stream.html
    - `createReadStream()`
    - `createWriteStream()`

```
//without stream
if(req.url === "/movies"){
    const movie = fs.readFileSync("./dummy.txt", "utf-8")
    res.end(movie)
}

//with Stream
if(req.url === "/movies"){
    const movieStream = fs.createReadStream("./dummy.txt","utf-
    movieStream.pipe(res)
}
//This will give us the same response but the load on server is
```

Express

Express is just a framework, that can help us in creating the server in very easy way.

- In `http` we were handling various methods and routes, the code was really messy. We can use express to get it rid all of that.
- Basically express is built over `http` module of node only
- It is not an inbuilt module of node, so we have to install it using `npm`.
- Initialise a node project and install `nodemon` .
- create an `index.js` file.
- Install `express` .

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

const app = express()

//this is a middleware we will see these in detail in the upcomi
app.use(express.json()) //this will parse the data in the req.bo

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

app.post("/",(req,res) => {
    console.log(req.body)
    res.send("data has been accepted")
})

//to send all the details of the students that are added
app.get("/details", (req,res) => {
    res.send("All details so far...")
})

app.listen(4500, () => {
    console.log("running on port 4500")
})
```

CRUD Operations

- create a file called `db.json`.

```
//sample json file

{
    "students": [
        {
            "name": "Chunnu",
            "city": "Pune"
        },
        {
            "name": "Munnu",
            "city": "Delhi"
        }
    ],

    "teachers": [
        {
            "name": "Albert",
            "sub": "Coding"
        },
        {
            "name": "Ankush",
            "sub": "DSA"
        }
    ]
}
```

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

const app = express()

app.use(express.json())

app.get("/student", (req,res) => {
    const data=JSON.parse(fs.readFileSync("/db.json","utf-8"))
    console.log(data.students)
    res.json(data.students)
})

app.post("/student", (req,res) => {
    const data = JSON.parse(fs.readFileSync("/db.json","utf-8")
    data.students.push(req.body)
    fs.writeFileSync("./db.json",data)
})

app.delete("") ==> //try it by your own as everything boils dow
app.listen(4500, () => {
    console.log("running on port 4500")
})
```

💡 **Homework:**
1.
Go through the express documentation, and see how can we `DELETE`, `PUT` something.