## Page:1

# **Express**

## **🌐 HTTP Post Request**

### 📚 Introduction

So far, we've explored how to make **GET** requests using HTTP. To post data to a server, the client must include some data in the request. This data is included in the request object, the only means by which the client can interact with the server.

## **🚀 Implementation**

javascriptCopy code

if (req.url === "/adddata" && req.method === "POST") {

// Logic to retrieve the payload sent by the client.

res.end("Data has been recorded");

}

If you try accessing this endpoint via a browser, you'll see an "Invalid Endpoint" message since browsers default to GET requests. Use tools like Thunder Client or Postman to test POST requests.

## **🛠 Handling Client Data**

The typical req.body won't work here in the native HTTP module; it will return undefined. Instead, we use event listeners:

javascriptCopy code

if (req.url === "/adddata" && req.method === "POST") {

let str = "";

req.on("data", (chunk) => {

str += chunk;

});

req.on("end", () => {

console.log(str); // Now we can access the data

});

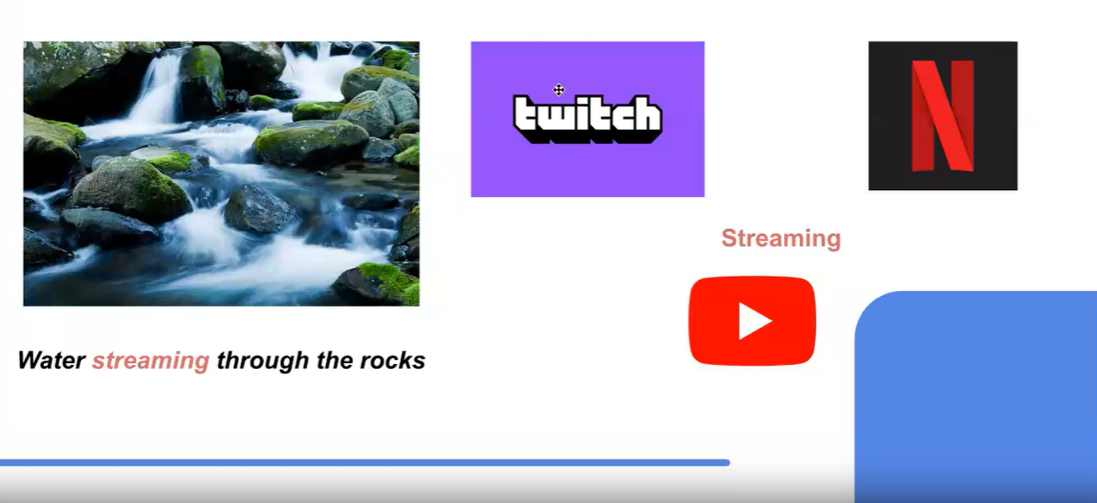
res.end("Data has been sent");

}

## Page:2

## **📜 Stream**

A stream is a sequence of bytes used to handle data, especially useful for managing large files.



## **🎥 Example**

## **Without Stream**

**javascriptCopy code**

**if (req.url === "/movies") {**

**const movie = fs.readFileSync("./dummy.txt", "utf-8");**

**res.end(movie);**

**}**

## **With Stream**

javascriptCopy code

if (req.url === "/movies") {

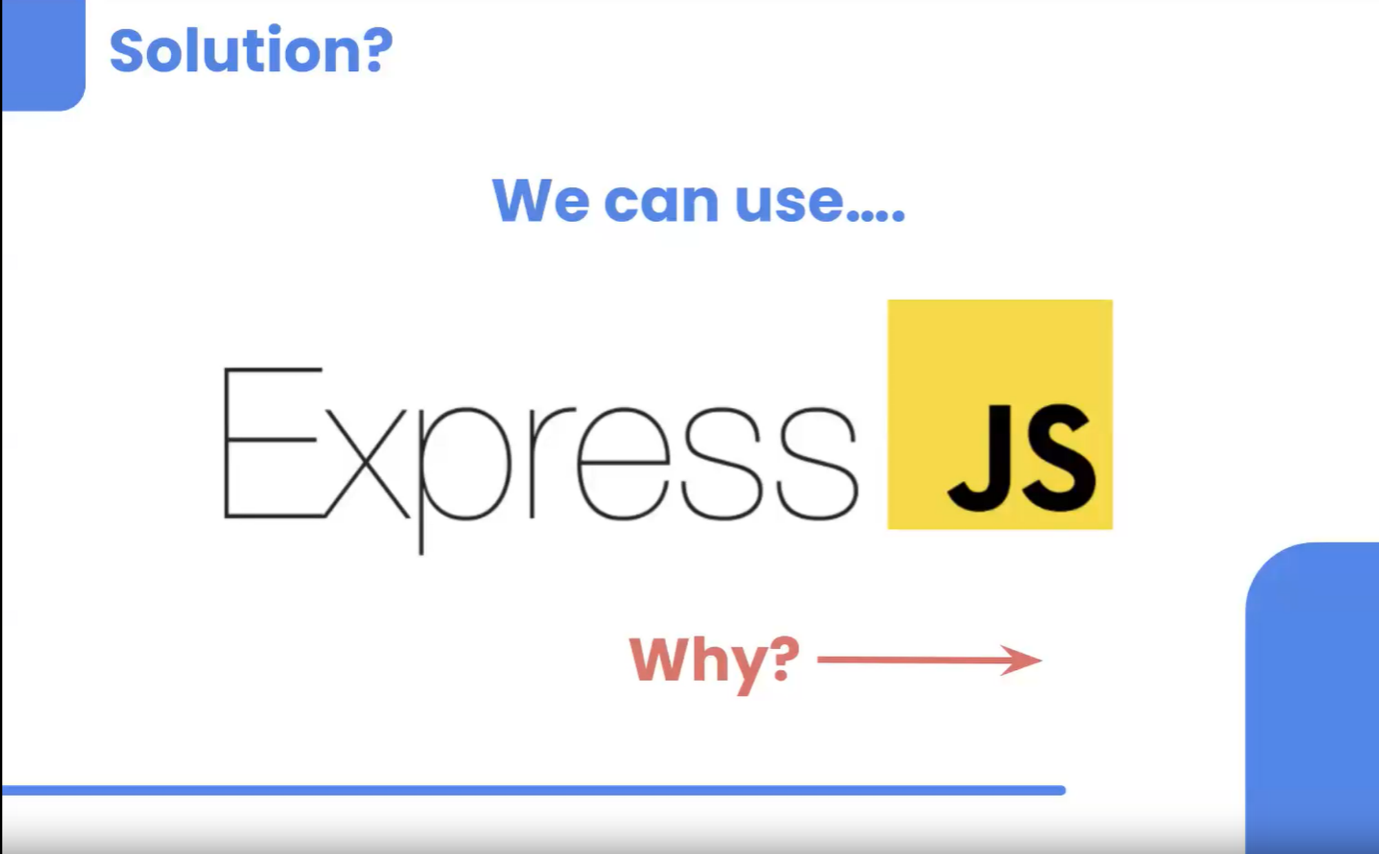
const movieStream = fs.createReadStream("./dummy.txt", "utf-8");

movieStream.pipe(res);

}

Using streams reduces server load, especially with large files. For more details, visit the [Node.js Stream Documentation](https://nodejs.org/dist/latest-v18.x/docs/api/stream.html).

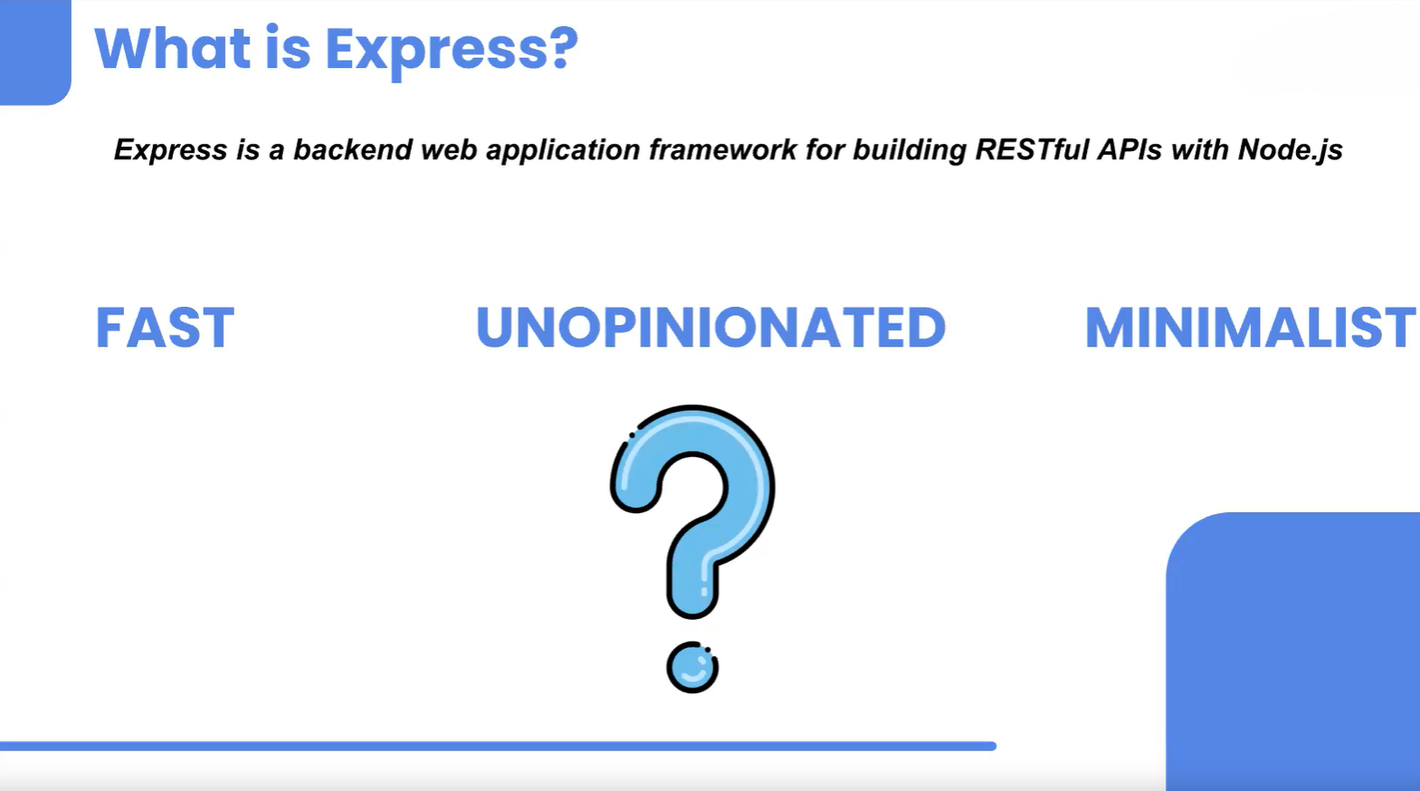
## Page:3



## **🌟 Express**

### **🛠️ Introduction**

Express is a powerful framework that simplifies creating a server. It helps clean up the code that handles various methods and routes, providing a more organized structure. While built over Node's HTTP module, Express is not a built-in module and needs to be installed via npm.



## Page:4

### **🚀 Setting Up**

1. **Initialize** a Node project and install **nodemon**.
2. Create an index.js file.
3. **Install Express**:

bashCopy code

npm install express

## **📚 Example**

javascriptCopy code

const express = require("express");

const app = express();

// Middleware to parse JSON data in the request body

app.use(express.json());

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

res.send("Hello");

});

app.post("/adddetails", (req, res) => {

console.log(req.body);

res.send("Data has been accepted");

});

// To send all details of the students that are added

app.get("/details", (req, res) => {

res.send("All details so far...");

});

## Page:5

app.listen(4500, () => {

console.log("Running on port 4500");

});

## **📁 CRUD Operations**

**🛠️ Setup**

1. Create a file named db.json with sample data:

jsonCopy code

{

"students": [

{ "name": "Chunnu", "city": "Pune" },

{ "name": "Munnu", "city": "Delhi" }

],

"teachers": [

{ "name": "Albert", "sub": "Coding" },

{ "name": "Ankush", "sub": "DSA" }

]

}

## **📚 Example**

javascriptCopy code

const express = require("express");

const app = express();

app.use(express.json());

app.get("/students", (req, res) => {

const data = JSON.parse(fs.readFileSync("/db.json", "utf-8"));

console.log(data.students);

res.json(data.students);

});

## Page:6

app.post("/addstudent", (req, res) => {

const data = JSON.parse(fs.readFileSync("/db.json", "utf-8"));

data.students.push(req.body);

fs.writeFileSync("./db.json", JSON.stringify(data));

res.send("Student data has been added");

});

// To be implemented: app.delete() for deleting a student

app.listen(4500, () => {

console.log("Running on port 4500");

});

## **💡 Homework**

1. Explore the Express documentation and learn how to implement **DELETE** and **PUT** operations.