## Page:1

# HTTP Assignment

This code is a simple HTTP server built using Node.js that handles different routes and interacts with a JSON file (db.json) to fetch data. Here's a line-by-line explanation:

## Importing Required Modules

const http = require("http");

console.log(http);

const http = require("http");: This line imports the http module, which is a built-in Node.js module used to create an HTTP server.

console.log(http);: This line logs the entire http module to the console. It's often used for debugging purposes to see what's included in the module.

const fs = require("fs");

const fs = require("fs");: This line imports the fs (File System) module, another built-in Node.js module used to interact with the file system, such as reading and writing files.

## Creating the Server

const server = http.createServer((req, res) => {

const server = http.createServer((req, res) => {...});: This creates an HTTP server instance. The callback function inside it will be executed every time the server receives a request. req is the request object (containing details about the client's request), and res is the response object (used to send data back to the client).

## Handling Different Routes

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

res.end("welcome");

} else if (req.url == "/about") {

res.end("aboutpage");

## Page:2

}

if (req.url == "/home") {...}: This checks if the request's URL is /home. If it is, the server responds with the string "welcome" and ends the response.

else if (req.url == "/about") {...}: Similarly, this checks if the request's URL is /about. If it is, the server responds with the string "aboutpage".

## Handling /getproductdata Route

else if (req.url == "/getproductdata") {

let str = "";

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

str += chunk;

});

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

fs.readFile("./db.json", "utf-8", (err, data) => {

if (err) {

res.end(err);

} else {

const getdatafromdb = JSON.parse(data);

console.log(getdatafromdb.products);

res.end(JSON.stringify(getdatafromdb.products));

}

res.end();

});

});

}

else if (req.url == "/getproductdata") {...}: This checks if the request's URL is /getproductdata.

let str = "";: Initializes an empty string to accumulate incoming data.

req.on("data", (chunk) => {...});: This event listener listens for data chunks being sent from the client and appends each chunk to the str variable.

req.on("close", () => {...});: This event listener triggers when the request is closed (i.e., when all data has been received). Inside it:

## Page:3

fs.readFile("./db.json", "utf-8", (err, data) => {...});: Reads the db.json file asynchronously. If an error occurs, it sends the error message as the response. Otherwise:

const getdatafromdb = JSON.parse(data);: Parses the JSON data from the file.

console.log(getdatafromdb.products);: Logs the products array from the parsed data to the console.

res.end(JSON.stringify(getdatafromdb.products));: Sends the products array back to the client as the response, converting it to a JSON string.

res.end();: Ensures that the response is properly ended.

## Handling /user Route

else if (req.url == "/user") {

fs.readFile("./db.json", "utf-8", (err, data) => {

if (err) {

res.end(err);

} else {

const getdatafromdb = JSON.parse(data);

console.log(getdatafromdb.user);

res.end(JSON.stringify(getdatafromdb.user));

}

res.end();

});

}

else if (req.url == "/user") {...}: This checks if the request's URL is /user.

fs.readFile("./db.json", "utf-8", (err, data) => {...});: Reads the db.json file asynchronously.

if (err) { res.end(err); }: If there's an error reading the file, it sends the error as the response.

else {...}: If the file is read successfully:

const getdatafromdb = JSON.parse(data);: Parses the JSON data from the file.

console.log(getdatafromdb.user);: Logs the user data from the parsed JSON to the console.

res.end(JSON.stringify(getdatafromdb.user));: Sends the user data back to the client as a JSON string.

res.end();: Ends the response.

## Page:4

## Handling Unknown Routes

else {

res.end("Not Fond");

}

else { res.end("Not Fond"); }: If the request URL doesn't match any of the predefined routes, the server responds with the string "Not Fond" (likely a typo of "Not Found").

## Starting the Server

server.listen(8090, () => {

console.log("server is running");

});

server.listen(8090, () => {...});: This starts the server on port 8090. The callback function logs "server is running" to the console once the server starts.

## Summary:

The server can handle different routes (/home, /about, /getproductdata, /user).

It reads data from a db.json file and returns specific parts of it (products or user) based on the request URL.

If the request doesn't match any of the routes, it responds with "Not Fond".

The server listens on port 8090 for incoming requests.

## Page:5

# Program

const http = require("http");

console.log(http);

const fs = require("fs");

const server = http.createServer((req, res) => {

// console.log(req.url);

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

res.end("welcome");

} else if (req.url == "/about") {

res.end("aboutpage");

} else if (req.url == "/getproductdata") {

let str = "";

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

str += chunk;

});

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

fs.readFile("./db.json", "utf-8", (err, data) => {

if (err) {

res.end(err);

} else {

const getdatafromdb = JSON.parse(data);

console.log(getdatafromdb.products);

res.end(JSON.stringify(getdatafromdb.products));

}

res.end();

});

});

} else if (req.url == "/user") {

fs.readFile("./db.json", "utf-8", (err, data) => {

if (err) {

## Page:6

res.end(err);

} else {

const getdatafromdb = JSON.parse(data);

console.log(getdatafromdb.user);

res.end(JSON.stringify(getdatafromdb.user));

}

res.end();

});

} else {

res.end("Not Fond");

}

});

server.listen(8090, () => {

console.log("server is running");

});

//http://localhost:8090

## Package.json

{

  "name": "nodedemo",

  "version": "1.0.0",

  "description": "",

  "main": "index.js",

  "server": "nodemon index.js",

  "scripts": {

    "server": "nodemon index.js",

    "test": "echo \"Error: no test specified\" && exit 1"

  },

## Page:7

  "keywords": [],

  "author": "",

  "license": "ISC",

  "dependencies": {

    "nodemon": "^3.1.4"

  }

}