Chetan Hiremath

Dr. Johnson

EECS 368

March 25, 2022

EECS 368 Assignment 5

JavaScript Source Code:

const {createServer} = require("http"); //This line shows that it is ready to start the server.

const methods = Object.create(null); //This line defines the variable for methods.

createServer((request, response) => {

let handler = methods[request.method] || notAllowed; //This line defines the variable for handler.

handler(request).catch(error => {

if (error.status != null) return error; //This line shows the if statement that works and throws the error if there is an error status.

return {body: String(error), status: 500}; //This line returns the status code of 500.

})

.then(({body, status = 200, type = "text/plain"}) => {

response.writeHead(status, {"Content-Type": type}); //This line invokes the type of the response.

if (body && body.pipe) body.pipe(response); //This line is the if statement that works and sends the data from the readable stream to a writeable stream if it is a readable stream.

else response.end(body); //This line is the else statement that works if it is not a readable stream.

});

}).listen(8000); //This line is used for the cycle server port 8000.

async function notAllowed(request) //This line defines the asynchronous function.

{

return

{

status: 405,

body: `Method ${request.method} is not supported.`

}; //These lines return the status code of 405 and the body that shows that the method is not supported.

}

const {parse} = require("url"); //This line shows that the url is required to parse it.

const {resolve, sep} = require("path"); //This line shows that the path is required to resolve it that is related to the directory.

const baseDirectory = process.cwd(); //This line defines the variable for the base directory.

function urlPath(url) //This line defined the url Path function.

{

let {pathname} = parse(url); //This line defines the variable for path name.

let path = resolve(decodeURIComponent(pathname).slice(1)); //This line defines the variable for path.

if (path != baseDirectory && !path.startsWith(baseDirectory + sep)) //This line shows the if statement that works if the path is not the base directory and doesn’t start with the base directory.

{

throw {status: 403, body: "Forbidden"}; //This line returns the status code of 403 if the if statement is true.

}

return path; //This line returns the path.

}

const {createReadStream} = require("fs"); //This line shows that the HTTP GET method is ready to create the read stream.

const {stat, readdir} = require("fs").promises; //This line shows that the HTTP GET method is ready.

const mime = require("mime"); //This line shows that the mime is required.

methods.GET = async function(request) //This line defines the HTTP GET method.

{

let path = urlPath(request.url); //This line translates the url into a file name.

let stats; //This line defines the variable for stats.

try

{

stats = await stat(path); //This line waits for stat to find the file.

}

catch (error)

{

if (error.code != "ENOENT") throw error; //This line shows the if statement that works and throws the error if it handles a non-existent file name.

else return {status: 404, body: "File not found"}; //This line shows the else statement that works and returns the status code of 404 if the if statement is false.

}

if (stats.isDirectory())

{

return {body: (await readdir(path)).join("\n")}; //This line shows the if statement that works and returns the body if the file name is the directory.

}

else

{

return {body: createReadStream(path), type: mime.getType(path)}; //This line shows the else statement that works and returns the body if the file name is not the directory.

}

};

const {rmdir, unlink} = require("fs").promises; //This line shows that the HTTP DELETE method is ready.

methods.DELETE = async function(request) //This line defines the HTTP DELETE method.

{

let path = urlPath(request.url); //This line translates the url into a file name.

let stats; //This line defines the variable for stats.

try

{

stats = await stat(path); //This line waits for stat to find the file.

}

catch (error)

{

if (error.code != "ENOENT") throw error; //This line shows the if statement that works and throws the error if it handles a non-existent file name.

else return {status: 204}; //This lines shows the else statement that works and returns the status code of 204 if the if statement is false.

}

if (stats.isDirectory()) await rmdir(path); //This line shows the if statement that works and removes the directory if the file name is a directory.

else await unlink(path); //This line shows the else statement that works and removes the file if the file name is not a directory.

return {status: 204}; //This line returns the status code of 204. It indicates that the file deletion is successful.

};

const {createWriteStream} = require("fs"); //This line shows that the HTTP PUT method is ready to create the write stream.

function pipeStream(from, to) //This line defines the pipe stream function.

{

return new Promise((resolve, reject) => {

from.on("error", reject); //This line returns a stream that fires an error.

to.on("error", reject); //This line will make the stream to reject the promise if the stream fails.

to.on("finish", resolve); //This line resolves the promise and closes the output stream when the pipe is done.

from.pipe(to); //This line moves the data from the readable stream to the writeable stream.

});

}

methods.PUT = async function(request) //This line defines the HTTP PUT method.

{

let path = urlPath(request.url); //This line translates the url into a file name.

await pipeStream(request, createWriteStream(path)); //This line waits for the pipe stream or a wrapper to create a promise.

return {status: 204}; //This line returns the status code of 204. It indicates that the file deletion is successful.

};

const {mkdir} = require("fs").promises; //This line shows that the HTTP MKCOL method is ready.

methods.MKCOL = async function(request) //This line defines the HTTP MKCOL method.

{

let path = urlPath(request.url); //This line translates the url into a file name.

let stats; //This line defines the variable for stats.

try

{

stats = await stat(path); //This line waits for stat to find the file.

}

catch (error)

{

if (error.code != "ENOENT") throw error; //This line shows the if statement that works and throws the error if it handles a non-existent file name.

await mkdir(path); //This line waits to find the file.

return {status: 204}; //This line returns the status code of 204. It indicates that the file deletion is successful.

}

if (stats.isDirectory()) return {status: 204}; //This line returns the status code of 204 if the file name is the directory.

else return {status: 400, body: "Not a directory"}; //This line shows the else statement that works and returns the status code of 400 if the file name is not a directory.

};