Node and HTTP

✓ **Done:** View

Learn about Node's core HTTP module.

3. Exercise: Node HTTP Module

https://www.youtube.com/watch?v=2bseGpuLSDI

Click here if you wish to open this video in its own window.

Objectives

- Explore the use of three core Node modules: http, fs, and path.
- Implement a simple HTTP Server that can respond to server requests.
- Serves static HTML files from a designated folder.
- Learn to use the Postman tool to test the server.

Instructions

Part 1: Set up a basic server using http module

- Create a folder named **node-http** in the **NucampFolder/5-NodeJS-Express-MongoDB** folder.
- Open VS Code to this folder.
- Inside the **node-http** folder, create a **package.json** file with the following as its content:

```
{
    "name": "node-http",
    "version": "1.0.0",
    "description": "Node HTTP Module Example",
    "main": "server.js",
    "scripts": {
        "test": "echo \"Error: no test specified\" && exit 1",
        "start": "node server"
    }
}
```

- Inside the **node-http** folder, create a file named **server.js** with the following as its content:

```
const http = require('http');

const hostname = 'localhost';
const port = 3000;

const server = http.createServer((req, res) => {
    console.log(req.headers);
    res.statusCode = 200;
    res.setHeader('Content-Type', 'text/html');
    res.end('<html><body><h1>Hello World!</h1></body></html>');
});

server.listen(port, hostname, () => {
    console.log(`Server running at http://${hostname}:${port}/`);
});
```

- Start the server by typing the following at the prompt:

```
npm start
```

- Then type localhost:3000 in your browser address bar and observe the results:
- Use the Postman desktop application that you installed at the beginning of this course to send a test request to your server and observe the results. View the exercise video for details on using Postman.
- **Optional:** If using Git, initialize a local Git repository in the node-http folder, then add and commit the files with the message "Node HTTP Example 1".

Part 2: Serve static HTML files using path and fs modules

- Create a folder called **public** in the **node-http** folder.
- In the **public** folder, create a file named **index.html** and add the following code to it:

```
<html>
<head>
    <title>This is index.html</title>
</head>
<body>
    <h1>Index</h1>
    <p>This is the content of index.html.</p>
</body>
</html>
```

- Similarly, create an **aboutus.html** file and add the following code to it:

```
<html>
<head>
    <title>This is aboutus.html</title>
</head>
<body>
    <h1>About Us</h1>
    <p>This is the content of aboutus.html.</p>
</body>
</html>
```

- Then update **server.js** as follows:

```
. . .

const path = require('path');
const fs = require('fs');

const server = http.createServer((req, res) => {
    console.log(`Request for ${req.url} by method ${req.method}`);

    if (req.method === 'GET') {
        let fileUrl = req.url;
        if (fileUrl === '/') {
            fileUrl = '/index.html';
        }

        const filePath = path.resolve('./public' + fileUrl);
        const fileExt = path.extname(filePath);

        if (fileExt === '.html') {
            fs.access(filePath, err => {
                if (err) {
                    res.statusCode = 404;
                    res.setHeader('Content-Type', 'text/html');
                    res.end(`<html><body><h1>Error 404: ${fileUrl} not found</h1></body></html>`);
                    return;
                }
                res.statusCode = 200;
                res.setHeader('Content-Type', 'text/html');

                fs.createReadStream(filePath).pipe(res);
            });
        } else {
            res.statusCode = 404;
            res.setHeader('Content-Type', 'text/html');
            res.end(`<html><body><h1>Error 404: ${fileUrl} is not an HTML file</h1></body></html>`);
        }
    } else {
        res.statusCode = 404;
        res.setHeader('Content-Type', 'text/html');
        res.end(`<html><body><h1>Error 404: ${req.method} not supported</h1></body></html>`);
    }
});

. . .
```

- Start the server, and send various requests to it and see the corresponding responses. Use both your browser and Postman to send requests. View the exercise video for details on this.
- **Optional:** Do a Git commit with the message "Node HTTP Example 2".

Summary

- In this exercise, you learned about using the core Node **http** module to implement a basic HTTP server, as well as using two other core modules, **fs** and **path**, to access filesystem and path utility methods provided by Node to write the code to handle server requests.

Additional Resources

- NodeJS - Anatomy of an HTTP Transaction
- NodeJS - HTTP, Path, File System (fs)
- NodeJS - Class http.IncomingMessage (HTTP request)
- NodeJS - Class http.ServerResponse (HTTP response)
- NodeJS - Stream
- NodeSource - Understanding Streams in Node.js
- FlavioCopes - Node.js Streams

«
PREVIOUS ACTIVITY
Challenge Question: Import Export

NEXT ACTIVITY
Code Challenge: Callbacks
»