Exercise (Instructions): Node and the HTTP Module

Objectives and Outcomes

In this exercise, you will explore three core Node modules: HTTP, fs and path. At the end of this exercise, you will be able to:

- Implement a simple HTTP Server
- Implement a server that returns html files from a folder

A Simple HTTP Server

- Create a folder named *node-http* in the *NodeJS* folder and move into the folder.
- In the *node-http* folder, create a subfolder named *public*.
- At the prompt, type the following to initialize a package.json file in the node-examples folder:

```
npm init
```

- Accept the standard defaults suggested until you end up with a package.json file containing the following:

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

- Create a file named index.*js* and add the following code to it:

```
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 you can type http://localhost:3000 in your browser address bar and see the result.
- You can also use postman chrome extension to send requests to the server and see the response. Alternately, you can download the stand-alone Postman tool from http://getpostman.com and install it on your computer.
- Initialize a Git repository, check in the files and do a Git commit with the message "Node HTTP Example 1".

Serving HTML Files

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

```
<html>
<title>This is index.html</title>
<body>
<h1>Index.html</h1>
<p>This is the contents of this file</p>
</body>
</html>
```

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

```
<html>
<title>This is aboutus.html</title>
<body>
<h1>Aboutus.html</h1>
<p>This is the contents of the aboutus.html file</p>
</body>
</html>
```

- Then update *index.js* as follows:

```
. . .
const fs = require('fs');
const path = require('path');
. . .
const server = http.createServer((req, res) => {
  console.log('Request for ' + req.url + ' by method ' + req.method);
  if (req.method == 'GET') {
    var fileUrl;
    if (req.url == '/') fileUrl = '/index.html';
```

```
    else fileUrl = req.url;
    var filePath = path.resolve('./public'+fileUrl);
    const fileExt = path.extname(filePath);
    if (fileExt == '.html') {
      fs.exists(filePath, (exists) => {
        if (!exists) {
          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 +
              ' not a 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.
- Do a Git commit with the message "Node HTTP Example 2".

Conclusions

In this exercise you learnt about using the Node HTTP module to implement a HTTP server.