Lesson 04 Demo 03

Working with Requests and Responses

Objective: To perform fundamental Node.js server operations, including handling HTTP requests and responses, managing headers, routing, redirection, and parsing query parameters

Tools required: Visual Studio Code and Node Package Manager

Prerequisites: Basic Linux commands and NPM commands

Steps to be followed:

1. Create a request
2. Send a response
3. Create the request and response headers
4. Route the requests
5. Redirect the requests
6. Parse the request body

Step 1: Create a request

1.1 Open the Node.js project with VS Code and enter the following code inside the **index.js** file to create an HTTP request:

```
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";
const server = http.createServer();
server.on("listening", () => console.log("Server Listening"))
server.on("error", () => console.log("Error while handling request"))
server.on("request", (req, res) => { /** handling requests */  })

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

```
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("listening", () => console.log("Server Listening"))
server.on("error", () => console.log("Error Occur while handling request"))
server.on("request", (req, res) => { /** handling requests */  })

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
    console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

1.2 Open the terminal and navigate inside the project directory, then execute the following command to run the server:
node index.js

```
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("listening", () => console.log("Server Listening"))
server.on("error", () => console.log("Error Occur while handling request"))
server.on("request", (req, res) => { /** handling requests */  })

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
    console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL
demopythonlyopm@ip-172-31-16-204:
demopythonlyopm@ip-172-31-16-204:~/Desktop/nodeProje
demopythonlyopm@ip-172-31-16-204:~/Desktop/nodeProjec/demo1$ node index
Server Listening
Server is up and listening on port 3000
```

The server started and is listening to port 3000.

Step 2: Send a response

2.1 Add the following code to send a response:

```
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
   res.setHeader("Content-Type", "text/plain")


   res.end("Hello, this is plain response")
})
```

```
server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
    res.setHeader("Content-Type", "text/plain")
    res.end("Hello, this is plain response")
})
server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

2.2 Modify the following code to send an HTML response:

```
server.on("request", (req, res) => {
   res.setHeader("Content-Type", "text/html")
   res.end("<html><body><h2>Node Server</h2></body></html>")
})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
  res.setHeader("Content-Type", "text/html")
  res.end("<html><body><h2>Node Server</h2></body></html>")
})

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

The output is obtained when the server is started.

2.3 Use the following code to send a JSON response:

```
const server = http.createServer();
server.on("request", (req, res) => {
   res.setHeader("Content-Type", "application/json")
   res.end(JSON.stringify({
      "platform": process.platform,
      "date": new Date(),
      "message": "Hellos"
   }));
})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
  res.setHeader("Content-Type", "application/json")
  res.end(JSON.stringify({
    "platform": process.platform,
    "date": new Date(),
    "message": "Hellos"
  }));
})


server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

127.0.0.1:3000

127.0.0.1:3000

{"platform":"linux","date":"2023-01-08T07:56:26.615Z","message":"Hellos"}

The JSON response is obtained when the server is started.

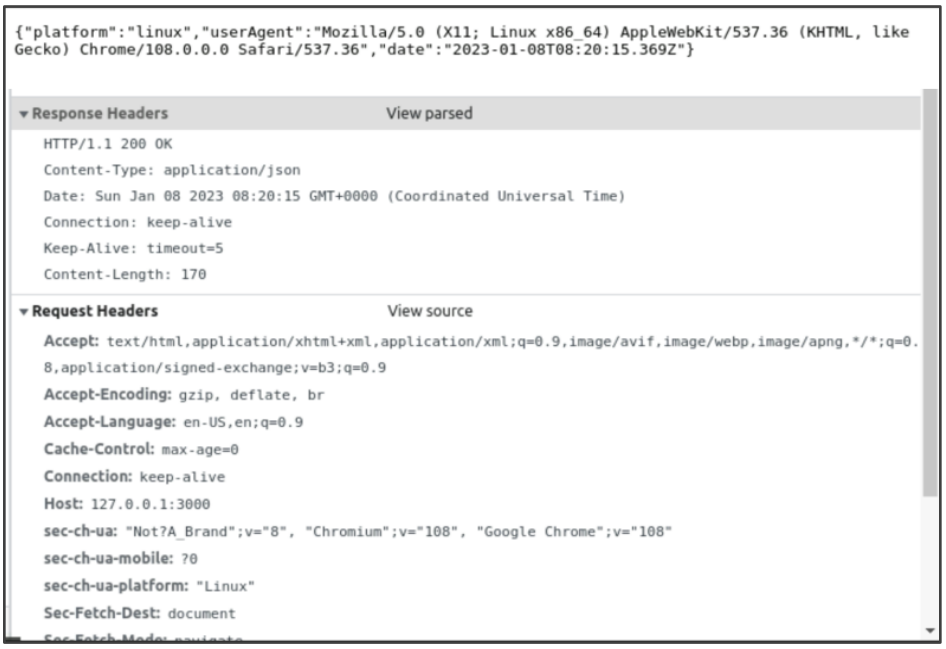
Step 3: Create the request and response headers

3.1 Use the following code to request headers and set response headers in node servers:

```
const server = http.createServer();
server.on("request", (req, res) => {
   // Accessing Request Headers of coming request
   const { headers } = req;

   // Getting value of user agent from request header
   const userAgent = headers['user-agent'];

   // Setting Response Header
   res.setHeader('Content-Type', 'application/json')
   res.setHeader('Date', new Date())
   res.end(JSON.stringify({
      "platform": process.platform,
      "userAgent": userAgent,
      "date": new Date()
   }))

})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
    // Accessing Request Headers of coming request
    const { headers } = req;

    // Getting value of user agent from request header
    const userAgent = headers['user-agent'];

    // Setting Response Header
    res.setHeader('Content-Type', 'application/json')
    res.setHeader('Date', new Date())
    res.end(JSON.stringify({
        "platform": process.platform,
        "userAgent": userAgent,
        "date": new Date()
    }))

})


server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

The output obtained after executing the **index.js** file as a Node.js app is as shown below:



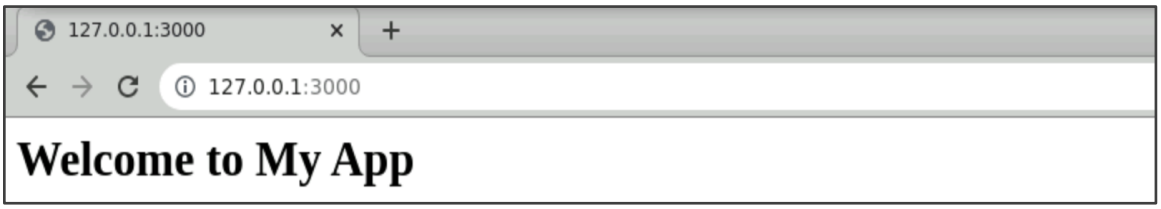
Step 4: Route the requests

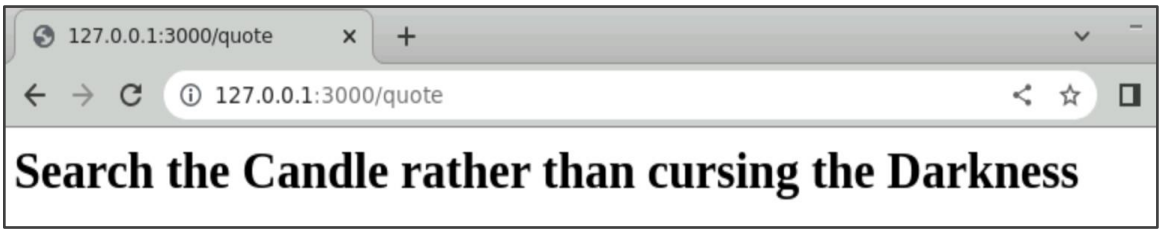
4.1 Use the following code to route the HTTP requests that helps to execute the business rules based on the request URL:

```
const server = http.createServer();
server.on("request", (req, res) => {
  const url = req.url;
  if (url === '/') {
    res.setHeader("Content-Type", "text/html");
    res.writeHead(200);
    res.end('<html><body><h1>Welcome to My App</h1></body></html>');
  }

  if (url === '/quote') {
    res.setHeader("Content-Type", "text/html");
    res.writeHead(200);
    res.end('<html><body><h1>Search the Candle rather than cursing the
Darkness</h1></body></html>');
  }
})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
    const url = req.url;
    if (url === '/') {
        res.setHeader("Content-Type", "text/html");
        res.writeHead(200);
        res.end('<html><body><h1>Welcome to My App</h1></body></html>');
    }

    if (url === '/quote') {
        res.setHeader("Content-Type", "text/html");
        res.writeHead(200);
        res.end('<html><body><h1>Search the Candle rather than cursing the Darkness</h1></body></html>');
    }
})

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

The output obtained after executing the **index.js** file as a Node.js app is as shown below:





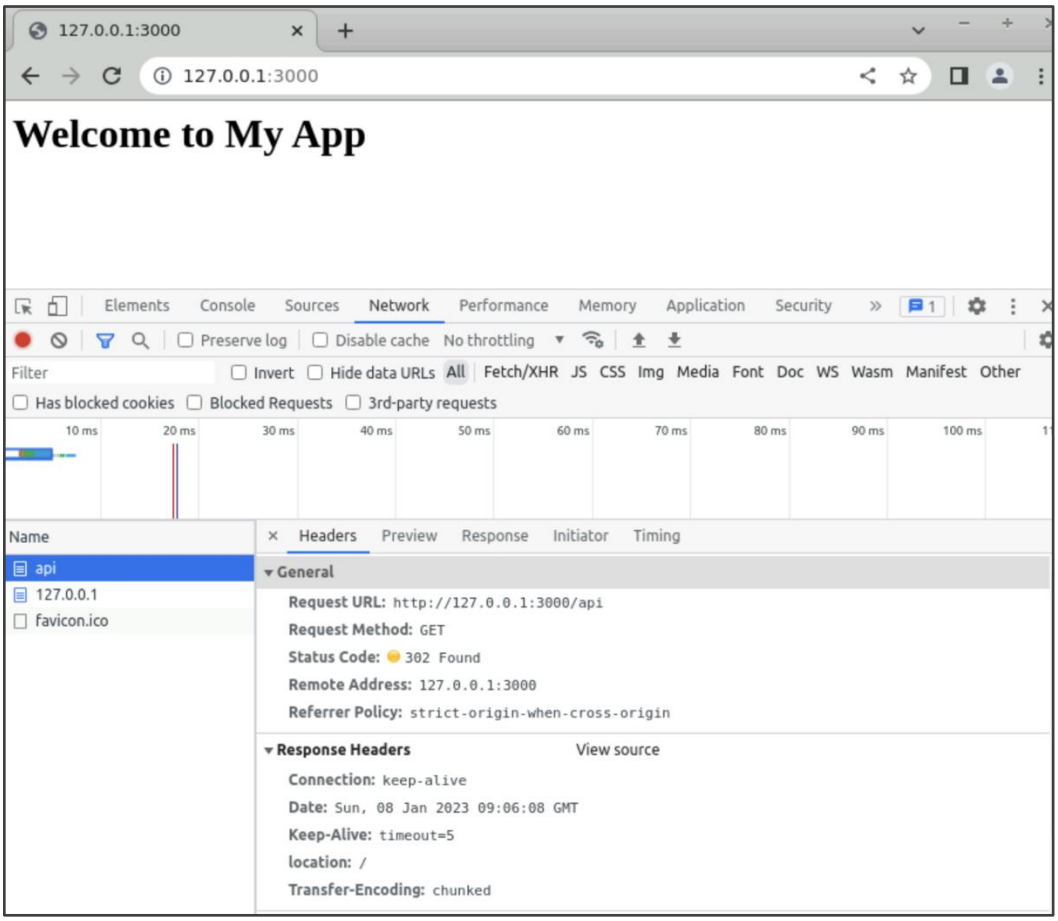
Step 5: Redirect the requests

5.1 Use the following code to redirect the HTTP requests:

```
const server = http.createServer();
server.on("request", (req, res) => {
  // ... continuing the previous example of the routing request

  if (url === '/api') {
    //redirect
    res.writeHead(302, {
      location: '/'
    })
    return res.end();
  }
})
```

```
demo1 > JS index.js > ...
const http = require('http');
const SERVER_PORT = 3000;
const SERVER_HOSTNAME = "127.0.0.1";

const server = http.createServer();
server.on("request", (req, res) => {
    // ... continuing the previous example of the routing request

    if (url === '/api') {
        //redirect
        res.writeHead(302, {
            location: '/'
        })
        return res.end();
    }
})

server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
  console.log(`Server is up and listening on port ${SERVER_PORT}`);
})
```

The output obtained after executing the **index.js** file as a Node.js app is as shown below:



Step 6: Parse the request body

6.1 Use the following code to convert the query parameters as the JSON object:

```
const http = require('http');
const url = require('url');
const server = http.createServer();
server.on("request", (req, res) => {
   const query = url.parse(req.url).query;
   const queryObj = query.split("&").reduce((prev, next) => {
     let [key, value] = next.split("=");
     return { ...prev, [key]: value }
   }, {});

   res.statusCode = 200;
   res.setHeader("Content-Type", "application/json")
```

```
    return res.end(JSON.stringify(queryObj));
})
```

```
demo1 > JS index.js > ...
1   const http = require('http');
2   const SERVER_PORT = 3000;
3   const SERVER_HOSTNAME = "127.0.0.1";
4   const url = require('url');
5
6   const server = http.createServer();
7   server.on("request", (req, res) => {
8     const query = url.parse(req.url).query;
9     const queryObj = query.split("&").reduce((prev, next) => {
10      let [key, value] = next.split("=");
11      return { ...prev, [key]: value }
12    }, {});
13
14    res.statusCode = 200;
15    res.setHeader("Content-Type", "application/json")
16    return res.end(JSON.stringify(queryObj));
17  })
18
19  server.listen(SERVER_PORT, SERVER_HOSTNAME, () => {
20    console.log(`Server is up and listening on port ${SERVER_PORT}`);
21  })
22
```

The output obtained after executing the **index.js** file as a Node.js app is as shown below:

By following these steps, you have successfully performed fundamental Node.js server operations, including handling HTTP requests and responses, managing headers, routing, redirection, and parsing query parameters.