

Ex. No: 8	Node JS

Aim: To create a Node JS server that has 2 pages - Homepage, and About and to implement proper error handling for invalid routes (e.g., return a 404 status code for unknown routes), and also, Implement a middleware to log incoming requests to the console.

Target Design:



Algorithm:

1. Import necessary modules: http, fs, path.
2. Construct file path based on request URL (filePath).
3. If URL is root or index.html, set path to public/index.html.
4. If URL is about.html, set path to public/about.html.
5. Read file asynchronously.
6. If file not found (error), return 404 response.
7. Determine file extension and set content type accordingly.

8. Write 200 OK header with appropriate content type.
9. Send file content as response using binary encoding.
10. Create an HTTP server instance using `http.createServer()`.
11. Handle server errors and log error messages.
12. Server listens on port 3000.
13. Pass request and response objects to `handleRequest` function.
14. If requested file not found, respond with 404 status code.
15. Use appropriate content type for valid files (text/html, images).
16. Serve `index.html` for root and `about.html` for `/about.html` URL.
17. Access the server at `http://localhost:3000/` for Homepage.
18. Access the server at `http://localhost:3000/about.html` for About page.
19. Invalid routes will return a 404 status code as expected.

Program :

./public/index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Koenigsegg</title>
  <link rel="icon" href="./logo_preview_rev_1.png" type="image/x-icon">
  <link rel="stylesheet" href="./index.css">
</head>
<body>
  
  
  
</body>
</html>
```

./public/index.css

```
body {
  background-color: black;
}
.car {
  height: 50%;
  width: 50%;
}
.engine {
  height: 60%;
  width: 30%;
}
.logo {
  height: 100px;
  width: 150px;
}
.logo:hover {
  filter: drop-shadow(0 0 2em #ffffff);
}
```

./public/about.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>About</title>
  <link rel="icon" href="./logo_preview_rev_1.png" type="image/x-icon">
  <link rel="stylesheet" href="./about.css">
</head>
<body>
  
  <div class="text">
    <p>
      Koenigsegg is a Swedish avante-garde megacar and innovative
      technology company. <br \>
```

```
        A record-breaking tradition courses through our history – the  
result of pushing <br \>  
        the boundaries of automotive technology to achieve higher  
standards in ultimate performance.  
    </p>  
</div>  
  
</body>  
</html>
```

./public/about.css

```
body {  
    background-color: black;  
}  
  
p {  
    font-family: Impact, Haettenschweiler, 'Arial Narrow Bold', sans-serif;  
    color: #dfdede;  
}  
  
.text {  
    display: flexbox;  
    background-color: rgb(132, 132, 209, 0.50);  
    height: 60px;  
    text-align: center;  
}  
  
.logo {  
    height: 100px;  
    width: 150px;  
    padding-left: 42%;  
}  
  
.logo:hover {  
    filter: drop-shadow(0 0 2em #ffffff);  
}
```

server.js

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

const server = http.createServer((req, res) => {
  // Middleware: Log incoming requests to the console
  console.log(`Request for ${req.url}`);

  // Determine the file path based on the requested URL
  let filePath = '.' + req.url;
  if (filePath === './') {
    filePath = './public/index.html';
  }

  // Read the file and serve it
  fs.readFile(filePath, (err, data) => {
    if (err) {
      if (err.code === 'ENOENT') {
        // File not found, return a 404 response
        res.writeHead(404, { 'Content-Type': 'text/plain' });
        res.end('404 Not Found');
      } else {
        // Server error, return a 500 response
        res.writeHead(500, { 'Content-Type': 'text/plain' });
        res.end('500 Internal Server Error');
      }
    } else {
      // Set a basic content type for all files
      res.writeHead(200, { 'Content-Type': 'text/html' });
      res.end(data);
    }
  });
});

const PORT = process.env.PORT || 3000;

server.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

Output Screenshots:



Result: Using Node JS, a web server of a website, containing Home and About pages were successfully implemented and errors were successfully handled.