SHREE H.N.SHUKLA COLLEGE OF I.T & MGMT

(Affiliated to Saurashtra University and G.T.U)

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Lt. Shree Chimanbhai Shukla

MScIT SEM-3 :: CS-14: Node.js

Shree H.N.Shukla College
Street No. 2, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2440478, 2472590

Shree H.N.Shukla College
Street No. 3, Vaishali Nagar,
Nr. Amrapali Under Bridge,
Raiya Road, Rajkot.
Ph. (0281)2471645

Shree H.N.Shukla College of I.T & Management "Sky is the Limit"

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

CS – 14: NODE JS

MSc(IT) SEMESTER : 03 :: CS-14: Node.js		
No	**Topics**	**Details**
3.	**Creating Web Server, File System, Debugging Node.js Application**	• Handling HTTP requests • Sending requests • Reading, Writing a File • Writing a file asynchronously • Opening a file • deleting a file • Other IO Operations: Append, Rename, Truncate • File System Module with URL Module Create, Read, Remove a Directory

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

1) **What is the purpose of a web server in web applications?**
2) **Why is it recommended to use Node.js web server for Node.js applications?**
3) **What are the steps to create a simple web server using Node.js?**

Introduction to Web Servers:

- To access web pages of any web application, a web server is needed.
- The web server handles all HTTP requests for the web application.
- Examples:
  - IIS is a web server for ASP.NET web applications.
  - Apache is a web server for PHP or Java web applications.

Node.js Web Server:

- Node.js allows you to create your own web server to handle HTTP requests asynchronously.
- While you can use IIS or Apache to run Node.js web applications, it is recommended to use the Node.js web server.

Benefits of Using Node.js Web Server:

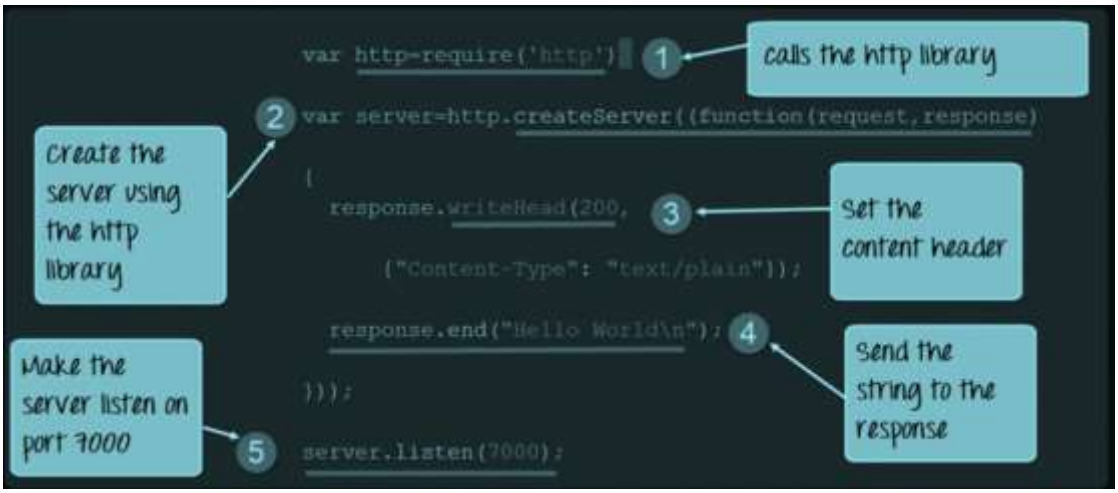
- Handles HTTP requests asynchronously.
- Provides better performance and scalability for Node.js applications.
- Simplifies the setup and configuration for Node.js applications.

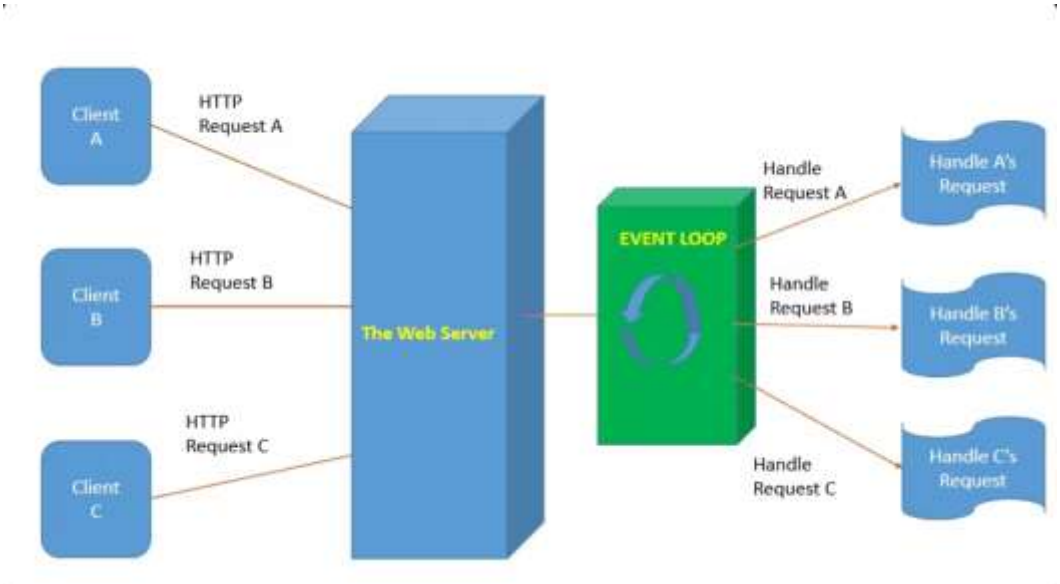
Steps to Create a Simple Node.js Web Server:

- Import the http module using require('http').
- Create a server using http.createServer().
- Define a callback function to handle incoming requests and send responses.
- Make the server listen on a specified port using server.listen(port).

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645





2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Creating a Web Server

How do you create a simple web server in Node.js?

Creating a simple web server in Node.js involves using the built-in `http` module. Here is a step-by-step example:

1. **Import the HTTP module:**

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

2. **Create a server:**

```
const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello, World!\n');
});
```

3. **Make the server listen on a specific port:**

```
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```

4. **Run the server:** Save the above code in a file (e.g., server.js) and run it using the command:

```
node server.js
```

Output:

When you navigate to http://localhost:3000/ in your browser, you should see "Hello, World!" displayed.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Handling HTTP Requests

How do you handle different HTTP request methods in Node.js?

Handling different HTTP request methods involves checking the req.method property.

Here's an example:

1. Import the HTTP module:

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

2. Create a server that handles GET and POST requests:

```
const server = http.createServer((req, res) => {
  if (req.method === 'GET') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('GET request received\n');
  } else if (req.method === 'POST') {
    res.writeHead(200, { 'Content-Type': 'text/plain' });
    res.end('POST request received\n');
  } else {
    res.writeHead(405, { 'Content-Type': 'text/plain' });
    res.end(`${req.method} not supported\n`);
  }
});
```

3. Run the server:

```
const PORT = 3000;
server.listen(PORT, () => {
  console.log(`Server running at http://localhost:${PORT}/`);
});
```

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Output:

- `GET` request to `http://localhost:3000/` will return "GET request received".
- `POST` request to `http://localhost:3000/` will return "POST request received".

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Sending Requests

How can you send an HTTP request from a Node.js application?

You can use the http or https module to send requests. Here's an example using the https module:

1. **Import the HTTPS module:**

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

2. **Send a GET request to an API:**

```
https.get('https://jsonplaceholder.typicode.com/todos/1', (res) => {
  let data = '';

  // A chunk of data has been received.
  res.on('data', (chunk) => {
    data += chunk;
  });

  // The whole response has been received.
  res.on('end', () => {
    console.log(JSON.parse(data));
  });
}).on("error", (err) => {
  console.log("Error: " + err.message);
});
```

Output:

This code will fetch and display a TODO item from the JSONPlaceholder API.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Reading and Writing a File

How do you read and write a file in Node.js?

Using the fs (File System) module, you can read and write files.

Here's an **example:**

1. Import the FS module:

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

2. Read a file:

```
fs.readFile('example.txt', 'utf8', (err, data) => {
  if (err) throw err;
  console.log(data);
});
```

3. Write to a file:

```
fs.writeFile('example.txt', 'Hello, Node.js!', (err) => {
  if (err) throw err;
  console.log('The file has been saved!');
});
```

Output:

- Reading will output the contents of `example.txt`.
- Writing will output "The file has been saved!" and update `example.txt` with "Hello, Node.js!".

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Writing a File Asynchronously

How do you write a file asynchronously in Node.js?

Writing a file asynchronously ensures non-blocking I/O operations.

1. Import the FS module:

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

2. Write a file asynchronously:

```
fs.writeFile('async_example.txt', 'This is asynchronous!', 'utf8', (err) => {
  if (err) {
    console.error('Error writing file:', err);
    return;
  }
  console.log('File has been written asynchronously.');
});
```

***Output*:**

The console will display "File has been written asynchronously." once the file operation is complete.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Opening a File

How do you open a file in Node.js?

You can open a file using fs.open().

1. Import the FS module:

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

2. Open a file:

```
fs.open('open_example.txt', 'r', (err, fd) => {
  if (err) {
    console.error('Error opening file:', err);
    return;
  }
  console.log('File descriptor:', fd);
});
```

Output:

The console will display the file descriptor if the file is successfully opened.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Deleting a File

How do you delete a file in Node.js?

Use the fs.unlink() method to delete a file.

1. Import the FS module:

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

2. Delete a file:

```
fs.unlink('delete_example.txt', (err) => {
  if (err) {
    console.error('Error deleting file:', err);
    return;
  }
  console.log('File deleted successfully.');
});
```

Output:

The console will display "File deleted successfully." if the file is deleted without errors.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Other I/O Operations

What are some other common I/O operations in Node.js?

Node.js provides various I/O operations in the fs module.

Operation	Code Example	Description
Append	fs.appendFile('append_example.txt', 'Appended text', (err) => { if (err) throw err; });	Appends data to a file
Rename	fs.rename('oldname.txt', 'newname.txt', (err) => { if (err) throw err; });	Renames a file
Truncate	fs.truncate('truncate_example.txt', 10, (err) => { if (err) throw err; });	Truncates a file to a specified length

Append:

```
fs.appendFile('append_example.txt', ' This is appended text.', (err) => {
  if (err) throw err;
  console.log('Data appended to file.');
});
```

Output:

"Data appended to file." and append_example.txt contains the appended text.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Rename:

```
fs.rename('example.txt', 'renamed_example.txt', (err) => {
  if (err) throw err;
  console.log('File renamed.');
});
```

Output: "File renamed."

Truncate:

```
fs.truncate('truncate_example.txt', 5, (err) => {
  if (err) throw err;
  console.log('File truncated.');
});
```

Output:

"File truncated." and **truncate_example.txt** is shortened to 5 bytes.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

File System Module with URL Module: Create, Read, Remove a Directory

How do you create, read, and remove a directory in Node.js using the File System module along with the URL module?

You can perform directory operations using the fs module and handle paths using the url module in Node.js. Here's how you can create, read, and remove directories.

Create a Directory

1. Import the FS and URL modules:

```
const fs = require('fs');
const url = require('url');
```

2. Create a directory:

```
const dirURL = new URL('file:///tmp/mydir');
const dirPath = dirURL.pathname;

fs.mkdir(dirPath, { recursive: true }, (err) => {
  if (err) {
    return console.error('Error creating directory:', err);
  }
  console.log('Directory created successfully:', dirPath);
});
```

Output:

The console will display "Directory created successfully: /tmp/mydir" if the directory is created without errors.

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

Read a Directory

1. Import the FS and URL modules:

```
const fs = require('fs');
const url = require('url');
```

2. Read a directory:

```
const dirURL = new URL('file:///tmp/mydir');
const dirPath = dirURL.pathname;

fs.readdir(dirPath, (err, files) => {
  if (err) {
    return console.error('Error reading directory:', err);
  }
  console.log('Directory contents:', files);
});
```

Output:

The console will display the contents of the directory. For example, "Directory contents: ['file1.txt', 'file2.txt']".

Remove a Directory

1. Import the FS and URL modules:

```
const fs = require('fs');
const url = require('url');
```

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

2. Remove a directory:

```
const dirURL = new URL('file:///tmp/mydir');
const dirPath = dirURL.pathname;

fs.rmdir(dirPath, { recursive: true }, (err) => {
  if (err) {
    return console.error('Error removing directory:', err);
  }
  console.log('Directory removed successfully:', dirPath);
});
```

Output:

The console will display "Directory removed successfully: /tmp/mydir" if the directory is removed without errors.

Example Code with Step-by-Step Explanation:

```
const fs = require('fs');
const url = require('url');

// Define the directory URL and path
const dirURL = new URL('file:///tmp/mydir');
const dirPath = dirURL.pathname;

// Step 1: Create a directory
fs.mkdir(dirPath, { recursive: true }, (err) => {
  if (err) {
    return console.error('Error creating directory:', err);
```

2 – Vaishalinagar,
Nr.Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2440478

3 – Vaishalinagar,
Nr. Amrapali Railway Under Bridge
Raiya Road,
Rajkot – 360001.
Ph No. 0281- 2471645

```
  }
  console.log('Directory created successfully:', dirPath);

  // Step 2: Read the directory contents
  fs.readdir(dirPath, (err, files) => {
    if (err) {
      return console.error('Error reading directory:', err);
    }
    console.log('Directory contents:', files);

    // Step 3: Remove the directory
    fs.rmdir(dirPath, { recursive: true }, (err) => {
      if (err) {
        return console.error('Error removing directory:', err);
      }
      console.log('Directory removed successfully:', dirPath);
    });
  });
});
```

SUMMARY:
- **Creating a Directory:** Use `fs.mkdir`.
- **Reading a Directory:** Use `fs.readdir`.
- **Removing a Directory:** Use `fs.rmdir`.
- **Handling Paths:** Use the `url` module to handle and convert URLs to paths.

Explanation:

1. **Create a Directory:**
   - Use `fs.mkdir` with the `recursive: true` option to create the directory and any necessary parent directories.
   - Log success or error messages.
2. **Read a Directory:**
   - Use `fs.readdir` to read the contents of the directory.
   - Log the directory contents or error messages.
3. **Remove a Directory:**
   - Use `fs.rmdir` with the `recursive: true` option to remove the directory and its contents.
   - Log success or error messages.