**Experiment-3**

**The fs module in Node.js provides an interface for working**with the file system. It allows you to perform various operations such as reading from and writing to files, manipulating directories

**What is the fs Module?**

The fs module is a built-in module in Node.js that provides file system-related functionality.

The fs module provides a wide range of functions for working with files and directories. Some of the most commonly used functions include:

* [**fs.readFile():**](https://www.geeksforgeeks.org/node-js-fs-readfile-method) Reads the contents of a file asynchronously.
* [**fs.writeFile():**](https://www.geeksforgeeks.org/node-js-fs-writefile-method) Writes data to a file asynchronously, replacing the file if it already exists.
* [**fs.appendFile()**](https://www.geeksforgeeks.org/node-js-fs-appendfile-function)**:** Appends data to a file asynchronously, creating the file if it does not exist.
* [**fs.unlink()**](https://www.geeksforgeeks.org/node-js-fs-unlink-method)**:** Deletes a file asynchronously.
* [**fs.mkdir()**](https://www.geeksforgeeks.org/node-js-fs-mkdir-method)**:** Creates a directory asynchronously.
* [**fs.rmdir()**](https://www.geeksforgeeks.org/node-js-fs-rmdir-method)**:** Removes a directory asynchronously.
* **importing the fs Module**

To use the fs module in your Node.js application, you need to import it first

const fs = require('fs');

* **Reading Files Asynchronously**

The fs.readFile() method reads the contents of a file asynchronously.

It takes the file path, encoding (optional), and a callback function as arguments.

fs.readFile('example.txt', 'utf8', (err, data) => {  
 if (err) {  
 console.error('Error reading file:', err);  
 return;  
 }  
 console.log('File content:', data);  
});

* **Synchronously**

The fs.readFileSync() method reads the contents of a file synchronously. It blocks the execution until the file is read.

try {  
 const data = fs.readFileSync('example.txt', 'utf8');  
 console.log('File content:', data);  
} catch (err) {  
 console.error('Error reading file:', err);  
}

**Renaming Files Asynchronously**

The fs.rename() method renames a file asynchronously.

fs.rename('example.txt', 'newname.txt', (err) => {  
 if (err) {  
 console.error('Error renaming file:', err);  
 return;  
 }  
 console.log('File renamed successfully');  
});

Synchronously

The fs.renameSync() method renames a file synchronously.

try {  
 fs.renameSync('example.txt', 'newname.txt');  
 console.log('File renamed successfully');  
} catch (err) {  
 console.error('Error renaming file:', err);  
}

1. Reading a File

You can read a file asynchronously using fs.readFile:

let fs=require('fs')

fs.readFile('./content.txt',(err,data)=>{

    if(err){

        console.log("file not found")

    }

    else{

        console.log(data)

    }

    }

)

2. Writing to a File

To write data to a file, you can use fs.writeFile

Writing File;

let fs=require('fs')

fs.writeFile('./content1.txt',"updating program",(err)=>{

    if(err){

        console.log("Not updated")

    }

    else{

        console.log("Updated Successfully")

    }})

3. Appending to a File

If you want to append data to an existing file, use fs.appendFile:

console.log("before writing")

let fs=require('fs')

fs.appendFile('./topics.txt',"adding extra",(err)=>{

    if(err){

        console.log("Not Written")

            }

            else{

                console.log("File has been written successfully")

            }

})

4. Deleting a File

To delete a file, you can use fs.unlink:

let fs=require('fs')

fs.unlike('./content.txt'),(err)=>{

    if(err){

        console.log("File not deleted");

    }

    else{

        console.log("File has been deleted successfully")

    }

}

5. Creating a Directory

To create a new directory, use fs.mkdir:

let fs=require('fs')

fs.mkdir('./test2',(err)=>{

    if(err){

        console.log("Directory not created")

    }

    else{

        console.log("Directory created successfully")

    }

})

console.log("before writing")