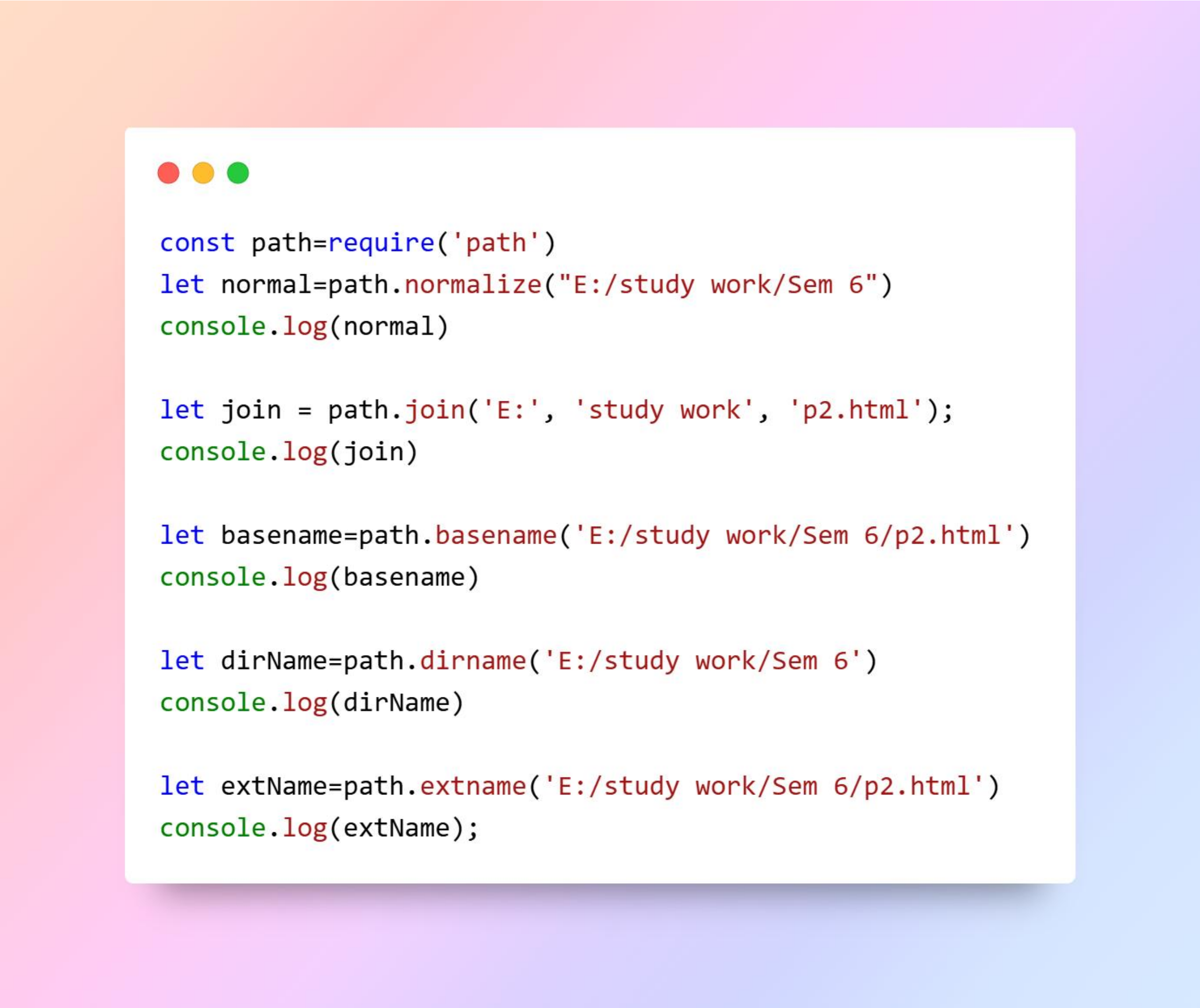
- **Lab-10**
- **1. Demonstrate the use of Node Package Manage (NPM).**

NPM (Node Package Manager) is used to manage dependencies (libraries and tools) for Node.js applications. Below is a step-by-step demonstration of how to use NPM.

Summary of NPM Commands

Command	Description
npm init -y	Initializes a project with a default package.json
npm install <package>	Installs a package and saves it in dependencies
npm install <package> --save-dev	Installs a package as a dev dependency
npm list	Lists installed packages
npm uninstall <package>	Removes a package
npm install	Installs all dependencies from package.json
npm update	Updates all installed packages
npm outdated	Shows outdated packages
npm install -g <package>	Installs a package globally

➢ 2. Demonstrate “path” core module in NodeJS

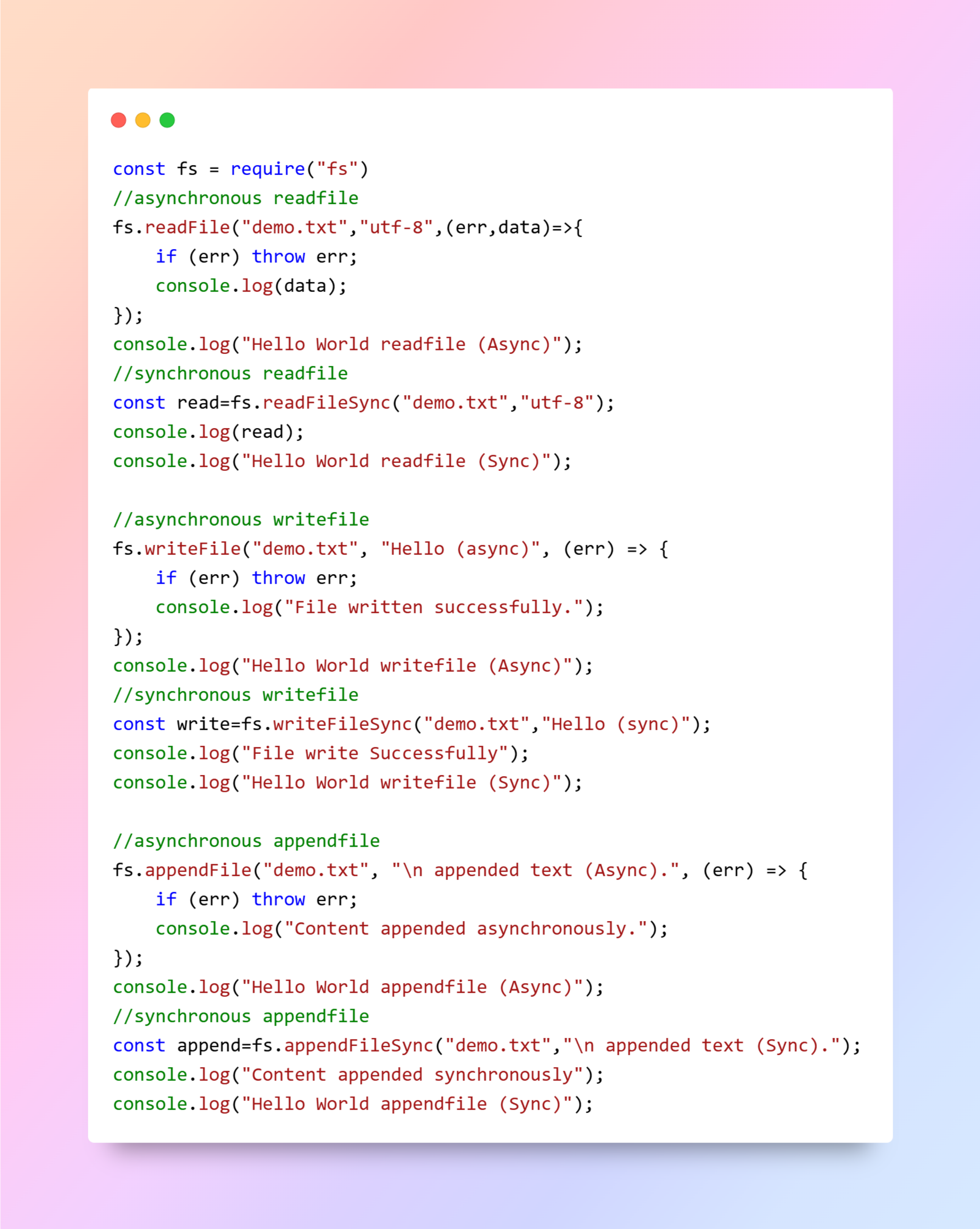


```
const path=require('path')
let normal=path.normalize("E:/study work/Sem 6")
console.log(normal)

let join = path.join('E:', 'study work', 'p2.html');
console.log(join)

let basename=path.basename('E:/study work/Sem 6/p2.html')
console.log(basename)

let dirName=path.dirname('E:/study work/Sem 6')
console.log(dirName)

let extName=path.extname('E:/study work/Sem 6/p2.html')
console.log(extName);
```

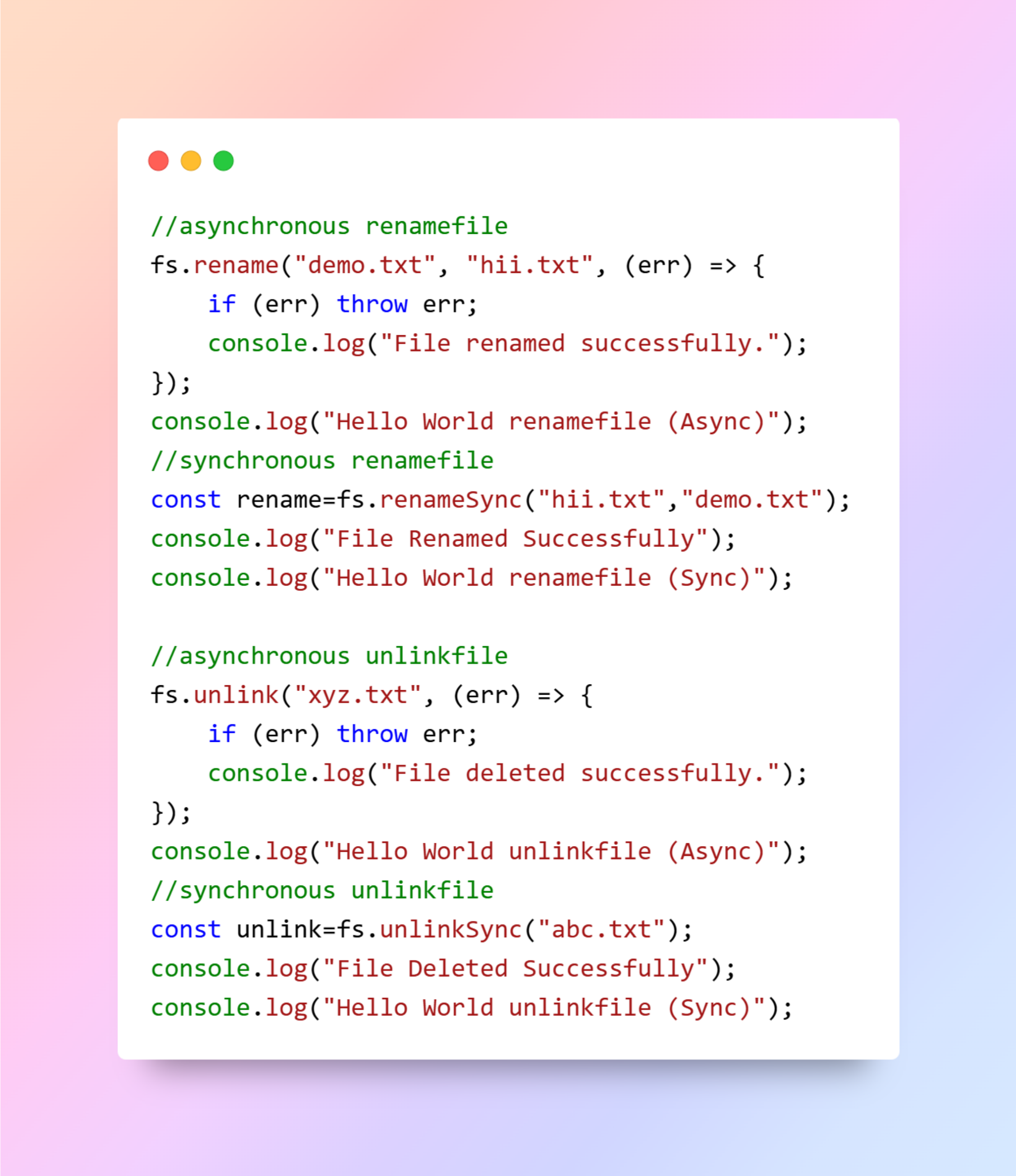
1. ***path.dirname(p) → Returns the directory name of the path.***
2. ***path.basename(p) → Returns the last portion of a path (filename).***
3. ***path.extname(p) → Returns the file extension of a path.***
4. ***path.join([...paths]) → Joins multiple path segments into one.***
5. ***path.resolve([...paths]) → Resolves a sequence of paths into an absolute path.***
6. ***path.parse(p) → Parses a path into an object with root, dir, base, ext, and name.***
7. ***path.format(obj) → Formats a parsed object back into a string.***
8. ***path.normalize(p) → Normalizes a path, resolving .. and . segments.***

```
E:\study work\Sem 6
E:\study work\p2.html
p2.html
E:/study work.html
```

➢ 3. Demonstrate “fs” core module in NodeJS.

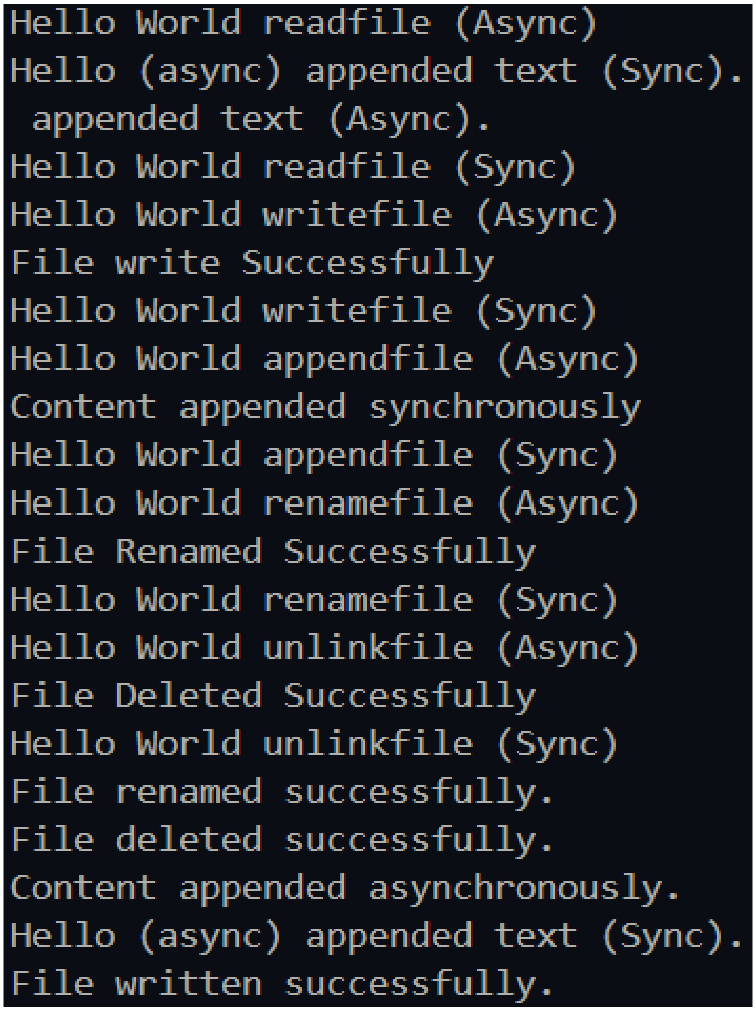


```
const fs = require("fs")
//asynchronous readfile
fs.readFile("demo.txt","utf-8",(err,data)=>{
    if (err) throw err;
    console.log(data);
});
console.log("Hello World readfile (Async)");
//synchronous readfile
const read=fs.readFileSync("demo.txt","utf-8");
console.log(read);
console.log("Hello World readfile (Sync)");

//asynchronous writefile
fs.writeFile("demo.txt", "Hello (async)", (err) => {
    if (err) throw err;
    console.log("File written successfully.");
});
console.log("Hello World writefile (Async)");
//synchronous writefile
const write=fs.writeFileSync("demo.txt","Hello (sync)");
console.log("File write Successfully");
console.log("Hello World writefile (Sync)");

//asynchronous appendfile
fs.appendFile("demo.txt", "\n appended text (Async).", (err) => {
    if (err) throw err;
    console.log("Content appended asynchronously.");
});
console.log("Hello World appendfile (Async)");
//synchronous appendfile
const append=fs.appendFileSync("demo.txt","\n appended text (Sync).");
console.log("Content appended synchronously");
console.log("Hello World appendfile (Sync)");
```



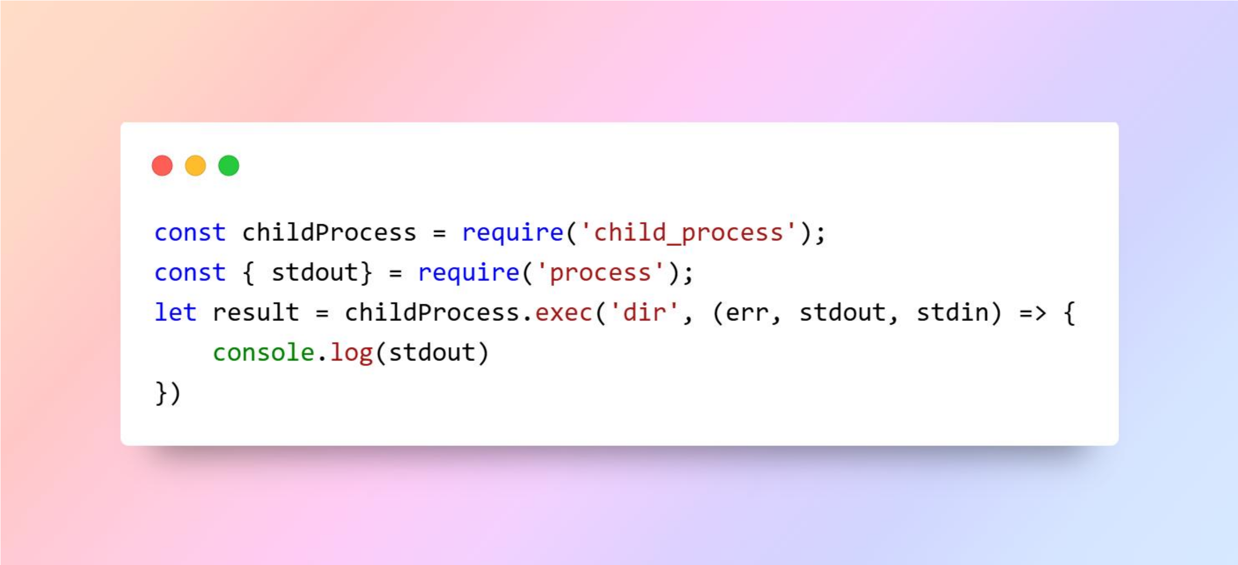
```
//asynchronous renamefile
fs.rename("demo.txt", "hii.txt", (err) => {
    if (err) throw err;
    console.log("File renamed successfully.");
});
console.log("Hello World renamefile (Async)");
//synchronous renamefile
const rename=fs.renameSync("hii.txt","demo.txt");
console.log("File Renamed Successfully");
console.log("Hello World renamefile (Sync)");

//asynchronous unlinkfile
fs.unlink("xyz.txt", (err) => {
    if (err) throw err;
    console.log("File deleted successfully.");
});
console.log("Hello World unlinkfile (Async)");
//synchronous unlinkfile
const unlink=fs.unlinkSync("abc.txt");
console.log("File Deleted Successfully");
console.log("Hello World unlinkfile (Sync)");
```

OUTPUT

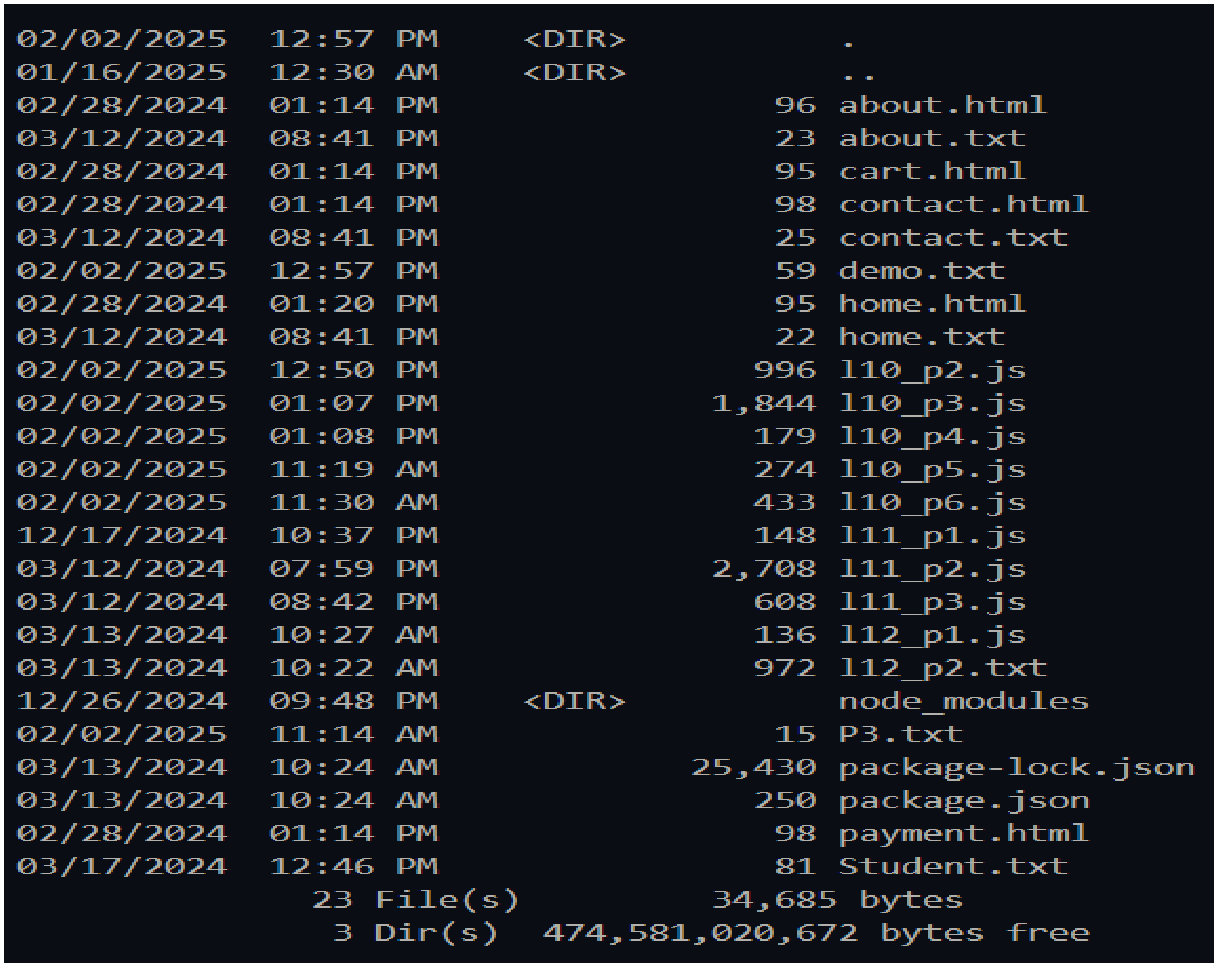


```
Hello World readfile (Async)
Hello (async) appended text (Sync).
 appended text (Async).
Hello World readfile (Sync)
Hello World writefile (Async)
File write Successfully
Hello World writefile (Sync)
Hello World appendfile (Async)
Content appended synchronously
Hello World appendfile (Sync)
Hello World renamefile (Async)
File Renamed Successfully
Hello World renamefile (Sync)
Hello World unlinkfile (Async)
File Deleted Successfully
Hello World unlinkfile (Sync)
File renamed successfully.
File deleted successfully.
Content appended asynchronously.
Hello (async) appended text (Sync).
File written successfully.
```

➢ 4. Demonstrate “child_process” core module in NodeJS.

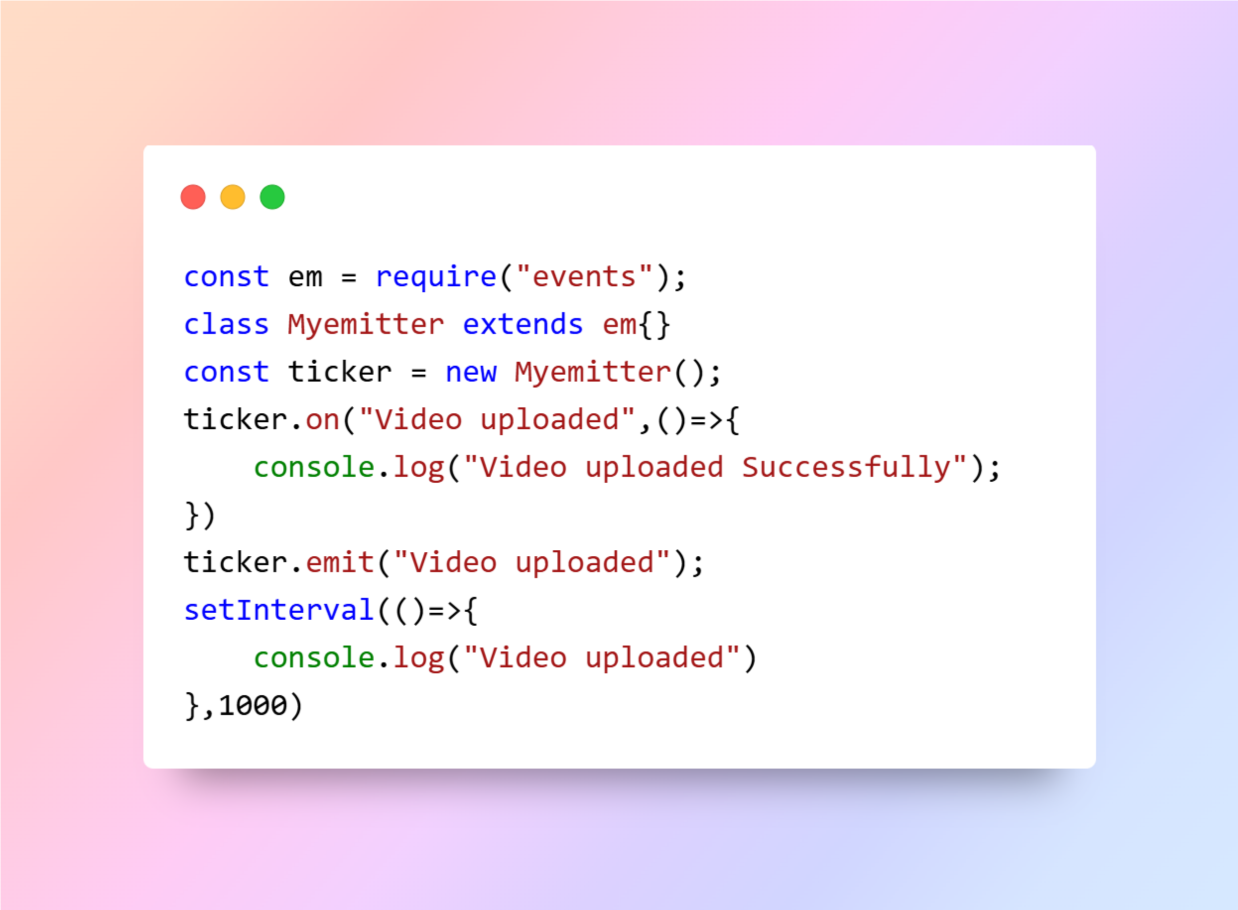


```
const childProcess = require('child_process');
const { stdout} = require('process');
let result = childProcess.exec('dir', (err, stdout, stdin) => {
    console.log(stdout)
})
```

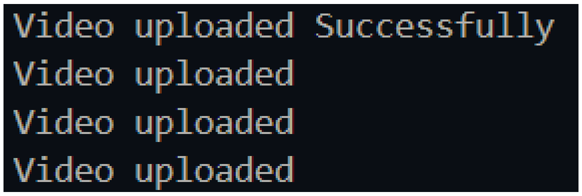


```
02/02/2025  12:57 PM    <DIR>          .
01/16/2025  12:30 AM    <DIR>          ..
02/28/2024  01:14 PM                96 about.html
03/12/2024  08:41 PM                23 about.txt
02/28/2024  01:14 PM                95 cart.html
02/28/2024  01:14 PM                98 contact.html
03/12/2024  08:41 PM                25 contact.txt
02/02/2025  12:57 PM                59 demo.txt
02/28/2024  01:20 PM                95 home.html
03/12/2024  08:41 PM                22 home.txt
02/02/2025  12:50 PM               996 l10_p2.js
02/02/2025  01:07 PM             1,844 l10_p3.js
02/02/2025  01:08 PM               179 l10_p4.js
02/02/2025  11:19 AM               274 l10_p5.js
02/02/2025  11:30 AM               433 l10_p6.js
12/17/2024  10:37 PM               148 l11_p1.js
03/12/2024  07:59 PM             2,708 l11_p2.js
03/12/2024  08:42 PM               608 l11_p3.js
03/13/2024  10:27 AM               136 l12_p1.js
03/13/2024  10:22 AM               972 l12_p2.txt
12/26/2024  09:48 PM    <DIR>          node_modules
02/02/2025  11:14 AM                15 P3.txt
03/13/2024  10:24 AM            25,430 package-lock.json
03/13/2024  10:24 AM               250 package.json
02/28/2024  01:14 PM                98 payment.html
03/17/2024  12:46 PM                81 Student.txt
              23 File(s)         34,685 bytes
               3 Dir(s)  474,581,020,672 bytes free
```

➢ 5. Demonstrate the use of EventEmmiter in NodeJS.

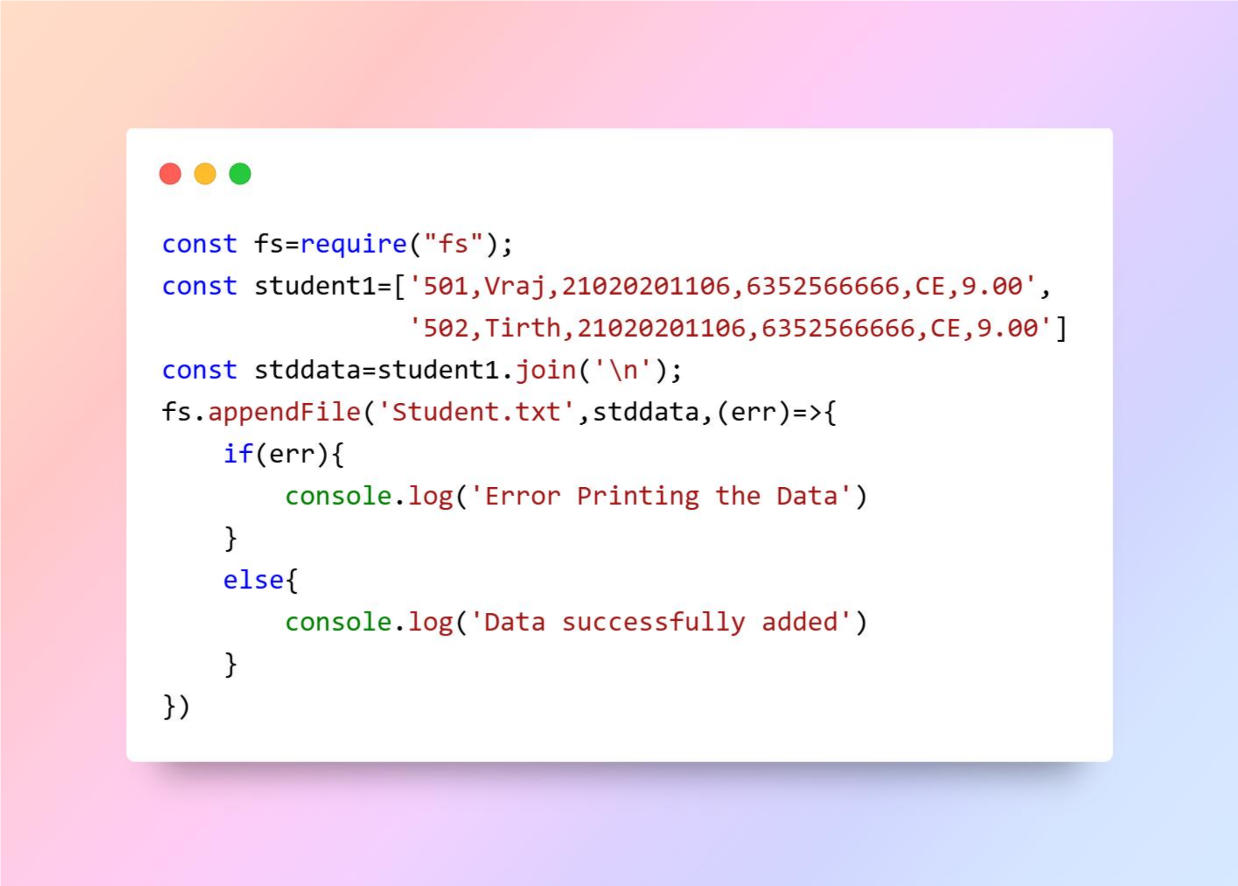


```
const em = require("events");
class Myemitter extends em{}
const ticker = new Myemitter();
ticker.on("Video uploaded",()=>{
    console.log("Video uploaded Successfully");
})
ticker.emit("Video uploaded");
setInterval(()=>{
    console.log("Video uploaded")
},1000)
```



```
Video uploaded Successfully
Video uploaded
Video uploaded
Video uploaded
```

➢ 6. W.A.P. in NodeJS to store the student details in text file.



```
const fs=require("fs");
const student1=['501,Vraj,21020201106,6352566666,CE,9.00',
                '502,Tirth,21020201106,6352566666,CE,9.00']
const stddata=student1.join('\n');
fs.appendFile('Student.txt',stddata,(err)=>{
    if(err){
        console.log('Error Printing the Data')
    }
    else{
        console.log('Data successfully added')
    }
})
```

```
Data successfully added
```

Student.txt

```
501,Vraj,21020201106,6352566666,CE,9.00
502,Tirth,21020201106,6352566666,CE,9.00
```