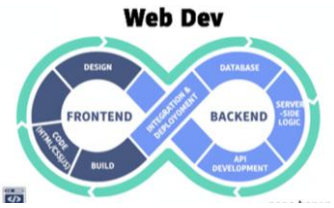

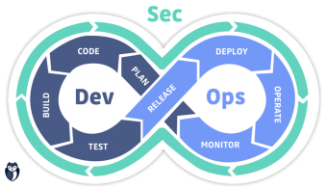

Day 43

"Web Development + Security"

Working with Files: fs and path Modules:

What is the fs Module?

- The fs (File System) module in Node.js lets you work with files — read, write, update, delete, and manage them.
- It's a built-in module, so no need to install anything.
- Works only in Node.js, not in the browser.

Types of File Operations:

fs supports both:

- Synchronous (blocking) – code waits until the file task finishes
- Asynchronous (non-blocking) – code continues running without waiting

Basic code to import and print the functions in the fs module:

```
JS main.js > ...
1   const fs = require("fs"); //to import it
2
3   console.log(fs);

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\FullStackDevelopment\Day41-50\Day43> node main.js
{
  appendFile: [Function: appendFile],
  appendFileSync: [Function: appendFileSync],
  access: [Function: access],
  accessSync: [Function: accessSync],
```

Now, in order that we shall create a file (say .txt) then write in it we will use the ".writeFileSync".

Example:

Main.js:

```
~$Day43.docx
Day43.docx
harry.txt
main.js
package.json

main.js > ...
const fs = require("fs"); //to import it

// console.log(fs);
console.log("starting");
fs.writeFileSync("harry.txt", "Harry is a good boy");
console.log("ending");

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS E:\FullStackDevelopment\Day41-50\Day43> node main.js
starting
ending
```

Since, JS is the asynchronous language, we want that it should work synchronously then we will do like this: we used ".writeFile"

```
~$Day43.docx
Day43.docx
harry.txt
harry2.txt
main.js
package.json

main.js > ...
const fs = require("fs"); //to import it
console.log("starting");
// fs.writeFileSync("harry.txt", "Harry is a good boy");
fs.writeFile("harry2.txt", "Aditya is a good boy",()=>{
    console.log("done");
})
console.log("ending");

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS E:\FullStackDevelopment\Day41-50\Day43> node main.js
starting
ending
done
PS E:\FullStackDevelopment\Day41-50\Day43>
```

Now, suppose we want to read the file as well after writing: then we will use .readFile:

```
const fs = require("fs"); //to import it
console.log("starting");
fs.writeFile("harry2.txt", "Aditya is a good boy",()=>{
    console.log("done");
    fs.readFile("harry2.txt", (error,data)=>{
        console.log(error, data.toString());
    })
})
console.log("ending");

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   PORTS

PS E:\FullStackDevelopment\Day41-50\Day43> node main.js
starting
ending
done
null Aditya is a good boy
```

Now, using append: “.appendFile”

```
const fs = require("fs"); //to import it
console.log("starting");
fs.writeFile("harry2.txt", "Aditya is a good boy",()=>{
    console.log("done");
    fs.readFile("harry2.txt", (error,data)=>{
        console.log(error, data.toString());
    })
})

fs.appendFile("harry.txt", "Aditya", (error,data)=>{
    console.log(data);
})
console.log("ending");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day41-50\Day43> node main.js
starting
ending
undefined
done
null Aditya is a good boy
```

Now, using the promises in the fs modules: we are reading the file

JS mainpromises.js > ...

```
import fs from "fs/promises";

let a = await fs.readFile("harry.txt");

console.log(a.toString());
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS E:\FullStackDevelopment\Day41-50\Day43> node .\mainpromises.js
(node:25172) [MODULE_TYPELESS_PACKAGE_JSON] Warning: Module type of file:///E:/FullStackDevelopment/Day41-
pecified and it doesn't parse as CommonJS.
Reparsing as ES module because module syntax was detected. This incurs a performance overhead.
To eliminate this warning, add "type": "module" to E:\FullStackDevelopment\Day41-50\Day43\package.json.
(Use `node --trace-warnings ...` to show where the warning was created)
Harry is a good boyAdityaAditya
PS E:\FullStackDevelopment\Day41-50\Day43>
```

Now, in case we want to write in the file:

```
JS mainpromises.js > ...
1   import fs from "fs/promises";
2
3   let a = await fs.readFile("harry.txt");
4   let b = await fs.writeFile("harry.txt", "yeashhh");
5   console.log(a.toString());
```

Now, in case we want to append in the file:

```
JS mainpromises.js > b
1   import fs from "fs/promises";
2
3   let a = await fs.readFile("harry.txt");
4   let b = await fs.appendFile("harry.txt", "yeashhh");
5   console.log(a.toString());
```

Operation	**Method**	**Description**
Read File	fs.readFile()	Read file content
Write File	fs.writeFile()	Create or replace a file
Append Data	fs.appendFile()	Add data to end of file
Delete File	fs.unlink()	Delete a file
Rename File	fs.rename()	Change file name
Check Existence	fs.existsSync()	Check if file exists

What is the path Module in Node.js?

- The path module helps you work with file and directory paths easily and safely.
- It is a core (built-in) module — you don't need to install it.
- It works on all operating systems (Windows, macOS, Linux) — automatically adjusts path separators (\ or /).

Example: importing path module

```
E:\FullStackDevelopment\Day41-50\Day43\main.js
1   import path from "path";
2
3   console.log(path);

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS E:\FullStackDevelopment\Day41-50\Day43> node .\aboutpath.js
<ref *1> {
  resolve: [Function: resolve],
  normalize: [Function: normalize],
  isAbsolute: [Function: isAbsolute],
```

--The End--