5) Create a Node.js file that replaces the content of an existing file

Ans.-

[**Node FS**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#node-fs)

In this post, we are going to discuss about Node JS Platform “fs” module. FS Stands for **File System**. This module is also known as **IO or FileSystem or Stream module**.

[**NodeJS FS Module Post Brief**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#nodejs-fs-module-post-brief)

1. Introduction to Node FS Module
2. Node JS Create File
3. Node JS Write to File
4. Node JS Read File

[**Introduction to Node FS Module**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#introduction-to-node-fs-module)

Node FS Module provides an API to interact with File System and to perform some IO Operations like create file, read File, delete file, update file etc. Like some Node modules for example “npm”, “http” etc, Node JS “fs” also comes with basic Node JS Platform. We don’t need to do anything to setup Node JS FS module.

[**Node FS Module import**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#node-fs-module-import)

We just need to import node fs module into our code and start writing IO Operations code. To import a node fs module;

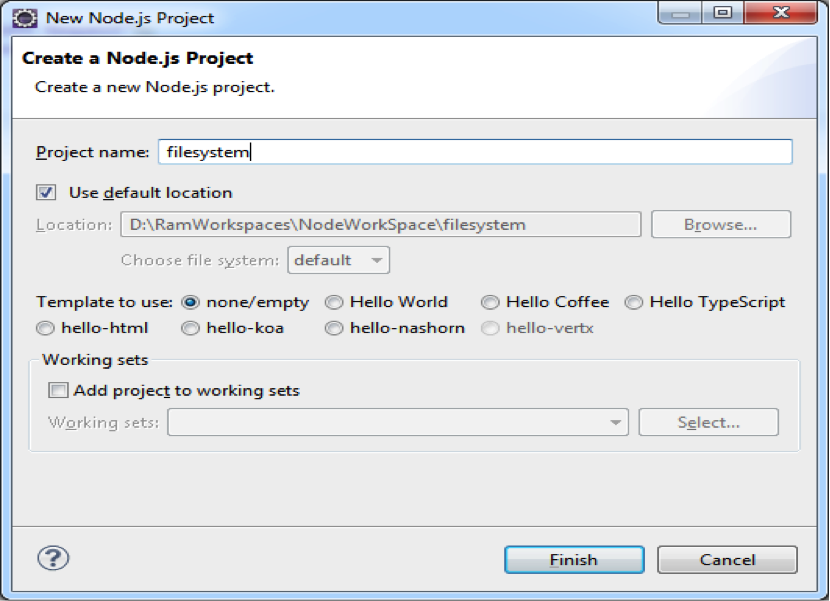
var fs = require("fs");

This require() call imports Node JS “fs” module into cache and creates an object of type Node FS module. Once it’s done, we can perform any IO Operation using node fs object. Let us assume that our ${Eclipse\_Workspace} refers to D:\RamWorkspaces\NodeWorkSpace. Now onwards, I’m going to use this variable to refer my Eclipse workspace. As a Java or DOT NET or C/C++ developers, we have already learned and wrote some IO Programs. IO or Streams are two types:

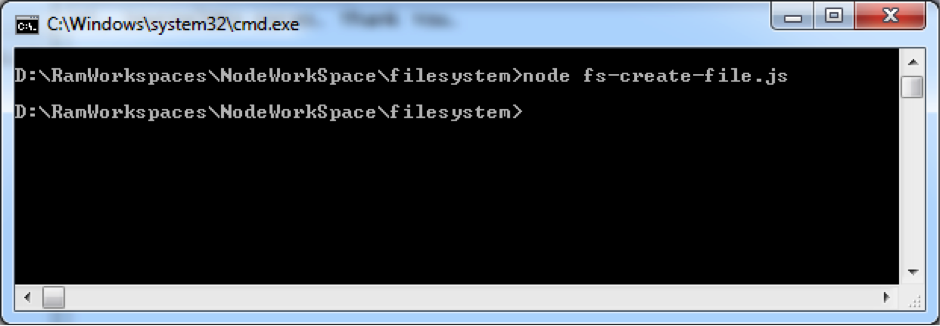
* Write Stream – To write data to a Stream.
* Read Stream – To read data from a Stream.

[**Node JS Create File**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#node-js-create-file)

Now we will discuss about how to create a new file using Node JS FS API.

1. Create a Node JS Project in Eclipse IDE.
2. Copy package.json file from previous examples and update the required things.
3. {
4. "name": "filesystem",
5. "version": "1.0.0",
6. "description": "File System Example",
7. "main": "filesystem",
8. "author": "JournalDEV",
9. "engines":{
10. "node":"\*"
11. }
12. }
13. Create a JavaScript file with the following content; fs-create-file.js
14. /\*\*
15. \* Node FS Example
16. \* Node JS Create File
17. \*/
18. var fs = require("fs");
19. var createStream = fs.createWriteStream("JournalDEV.txt");
20. createStream.end();

**Code Description**: var fs = require("fs") require() call loads specified Node FS module into cache and assign that to an object named as fs. fs.createWriteStream(filename) call is used to create a Write Stream and file with given filename. createStream.end() call ends or closes the opened stream.

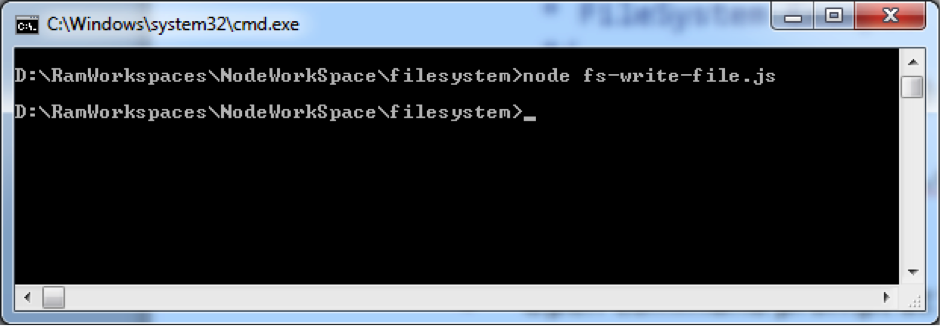
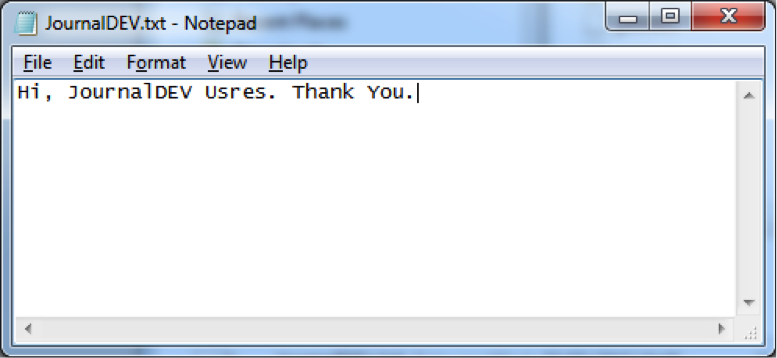
1. Before executing fs-create-file.js, first observe the filesystem project content and you will notice that “JournalDEV.txt” file is not available.
2. Open command prompt at ${Eclipse\_Workspace}/filesystem and run node commend to execute fs-create-file.js file as shown in below image.Notice your project directory contents now, you will notice an empty file named “JournalDEV.txt”.

[**Node JS Write to File**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#node-js-write-to-file)

We will use Node FS API to create a new file and write some data into that. It is continuation to our previous example.

1. Remove previously created “JournalDEV.txt” from ${Eclipse\_Workspace}/filesystem folder
2. Create a Java Script file with the following content: fs-write-file.js
3. /\*\*
4. \* Node FS Example
5. \* Node JS Write to File
6. \*/
7. var fs = require("fs");
8. var writeStream = fs.createWriteStream("JournalDEV.txt");
9. writeStream.write("Hi, JournalDEV Users. ");
10. writeStream.write("Thank You.");
11. writeStream.end();

createStream.write(sometext) call is used to write some text to a file.

1. Open command prompt at ${Eclipse\_Workspace}/filesystem and run node commend to execute fs-write-file.js file as shown below.
2. Go to ${Eclipse\_Workspace}/filesystem folder and open “JournalDEV.txt” to verify its content.Now we have created a new file and write some data into that file.

[**Node JS Read File**](https://www.digitalocean.com/community/tutorials/node-fs-js-create-file-read-write#node-js-read-file)

We will use Node FS API to open and read an existing file content and write that content to the console. It is continuation to our previous examples. Here we are going to use named JavaScript function. Go through this example to understand this.

1. Create a Java Script file with the following content. fs-read-file1.js
2. /\*\*
3. \* Node FS Read File
4. \* Node JS Read File
5. \*/
6. var fs = require("fs");
7. function readData(err, data) {
8. console.log(data);
9. }
10. fs.readFile('JournalDEV.txt', 'utf8', readData);

**Code Description**: readData() is a JavaScript function which takes two parameters;

* 1. **err**: it’s an error object. When our program fails to open or read data from a file, then FS module writes some error message into this parameter.
  2. **data**: it’s a variable to hold some data.

This function takes data parameter and prints that data to a console. fs.readFile() is Node JS FS API. It takes three parameters;

* 1. file name
  2. file data format to read
  3. A JavaScript function or JavaScript anonymous functionreadFile() reads data from a filename (first parameter) in given format (second parameter) and uses function given at third parameter to do some operation. Here we are using a plain JavaScript function as the third Parameter. We will see how to use JavaScript anonymous function in next example. In our example, readFile() reads data from “JournalDEV.txt” file and writes to console. If we don’t use ‘utf8’ format, then we will get binary data. We will verify this in few moments.