Node.js File System (FS)

In Node.js, file I/O is provided by simple wrappers around standard POSIX functions. Node File System (fs) module can be imported using following syntax:

**Syntax:**

1. var fs = require("fs")

Node.js FS Reading File

Every method in fs module has synchronous and asynchronous forms.

Asynchronous methods take a last parameter as completion function callback. Asynchronous method is preferred over synchronous method because it never blocks the program execution where as the synchronous method blocks.

**Let's take an example:**

Create a text file named "input.txt" having the following content.

*File: input.txt*

1. Javatpoint is a one of the best online tutorial website to learn different technologies
2. in a very easy and efficient manner.

Let's take an example to create a JavaScript file named "main.js" having the following code:

*File: main.js*

1. var fs = require("fs");
2. // Asynchronous read
3. fs.readFile('input.txt', function (err, data) {
4. if (err) {
5. return console.error(err);
6. }
7. console.log("Asynchronous read: " + data.toString());
8. });
9. // Synchronous read
10. var data = fs.readFileSync('input.txt');
11. console.log("Synchronous read: " + data.toString());
12. console.log("Program Ended");

Open Node.js command prompt and run the main.js:

1. node main.js

Node.js Open a file

**Syntax:**

Following is the syntax of the method to open a file in asynchronous mode:

1. fs.open(path, flags[, mode], callback)

**Parameter explanation:**

Following is the description of parameters used in the above syntax:

**path:** This is a string having file name including path.

**flags:** Flag specifies the behavior of the file to be opened. All possible values have been mentioned below.

**mode:** This sets the file mode (permission and sticky bits), but only if the file was created. It defaults to 0666, readable and writeable.

**callback:**This is the callback function which gets two arguments (err, fd).

Node.js Flags for Read/Write

Following is a list of flags for read/write operation:

|  |  |
| --- | --- |
| **Flag** | **Description** |
| r | open file for reading. an exception occurs if the file does not exist. |
| r+ | open file for reading and writing. an exception occurs if the file does not exist. |
| rs | open file for reading in synchronous mode. |
| rs+ | open file for reading and writing, telling the os to open it synchronously. see notes for 'rs' about using this with caution. |
| w | open file for writing. the file is created (if it does not exist) or truncated (if it exists). |
| wx | like 'w' but fails if path exists. |
| w+ | open file for reading and writing. the file is created (if it does not exist) or truncated (if it exists). |
| wx+ | like 'w+' but fails if path exists. |
| a | open file for appending. the file is created if it does not exist. |
| ax | like 'a' but fails if path exists. |
| a+ | open file for reading and appending. the file is created if it does not exist. |
| ax+ | open file for reading and appending. the file is created if it does not exist. |

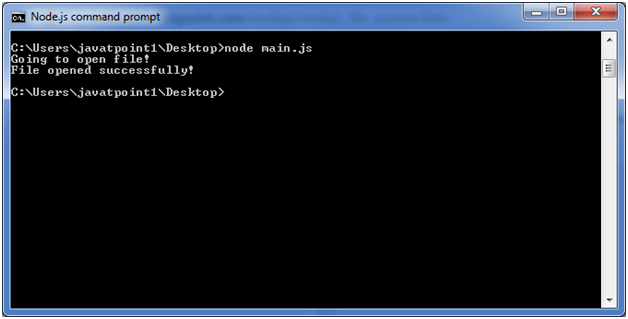
Create a JavaScript file named "main.js" having the following code to open a file input.txt for reading and writing.

*File: main.js*

1. var fs = require("fs");
2. // Asynchronous - Opening File
3. console.log("Going to open file!");
4. fs.open('input.txt', 'r+', function(err, fd) {
5. if (err) {
6. return console.error(err);
7. }
8. console.log("File opened successfully!");
9. });

Open Node.js command prompt and run the main.js:

1. node main.js



Node.js File Information Method

**Syntax:**

Following is syntax of the method to get file information.

1. fs.stat(path, callback)

**Parameter explanation:**

**Path:**This is string having file name including path.

**Callback:**This is the callback function which gets two arguments (err, stats) where stats is an object of fs.Stats type.

Node.js fs.Stats class Methods

|  |  |
| --- | --- |
| **Method** | **Description** |
| stats.isfile() | returns true if file type of a simple file. |
| stats.isdirectory() | returns true if file type of a directory. |
| stats.isblockdevice() | returns true if file type of a block device. |
| stats.ischaracterdevice() | returns true if file type of a character device. |
| stats.issymboliclink() | returns true if file type of a symbolic link. |
| stats.isfifo() | returns true if file type of a fifo. |
| stats.issocket() | returns true if file type of asocket. |

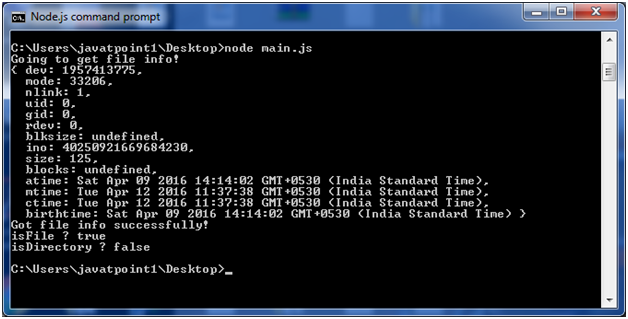
Let's take an example to create a JavaScript file named main.js having the following code:

*File: main.js*

1. var fs = require("fs");
2. console.log("Going to get file info!");
3. fs.stat('input.txt', function (err, stats) {
4. if (err) {
5. return console.error(err);
6. }
7. console.log(stats);
8. console.log("Got file info successfully!");
9. // Check file type
10. console.log("isFile ? " + stats.isFile());
11. console.log("isDirectory ? " + stats.isDirectory());
12. });

Now open the Node.js command prompt and run the main.js

1. node main.js



Node.js Path

The Node.js path module is used to handle and transform files paths. This module can be imported by using the following syntax:

**Syntax:**

1. var path =  require ("path")

Node.js Path Methods

Let's see the list of methods used in path module:

|  |  |  |
| --- | --- | --- |
| **Index** | **Method** | **Description** |
| 1. | path.normalize(p) | It is used to normalize a string path, taking care of '..' and '.' parts. |
| 2. | path.join([path1][, path2][, ...]) | It is used to join all arguments together and normalize the resulting path. |
| 3. | path.resolve([from ...], to) | It is used to resolve an absolute path. |
| 4. | path.isabsolute(path) | It determines whether path is an absolute path. an absolute path will always resolve to the same location, regardless of the working directory. |
| 5. | path.relative(from, to) | It is used to solve the relative path from "from" to "to". |
| 6. | path.dirname(p) | It return the directory name of a path. It is similar to the unix dirname command |
| 7. | path.basename(p[, ext]) | It returns the last portion of a path. It is similar to the Unix basename command. |
| 8. | path.extname(p) | It returns the extension of the path, from the last '.' to end of string in the last portion of the path. if there is no '.' in the last portion of the path or the first character of it is '.', then it returns an empty string. |
| 9. | path.parse(pathstring) | It returns an object from a path string. |
| 10. | path.format(pathobject) | It returns a path string from an object, the opposite of path.parse above. |

Node.js Path Example

*File: path\_example.js*

1. var path = require("path");
2. // Normalization
3. console.log('normalization : ' + path.normalize('/sssit/javatpoint//node/newfolder/tab/..'));
4. // Join
5. console.log('joint path : ' + path.join('/sssit', 'javatpoint', 'node/newfolder', 'tab', '..'));
6. // Resolve
7. console.log('resolve : ' + path.resolve('path\_example.js'));
8. // Extension
9. console.log('ext name: ' + path.extname('path\_example.js'));

Open Node.js command prompt and run the path\_example.js:

1. node path\_example.js

