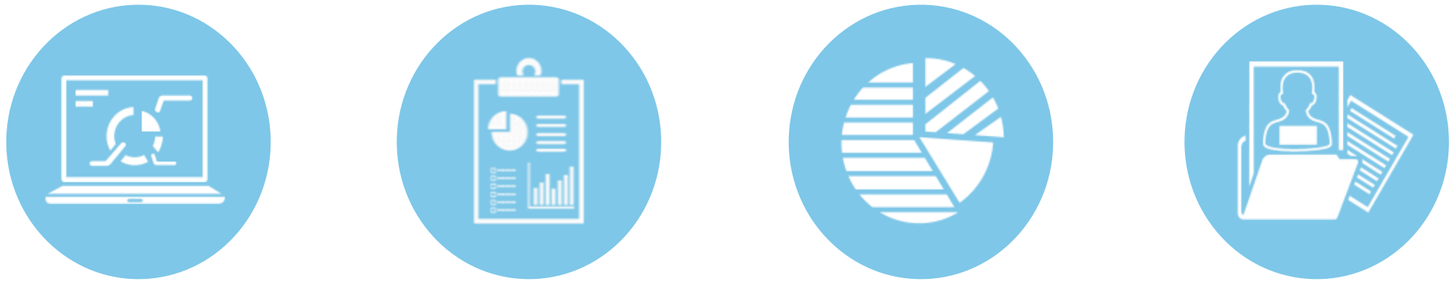
Node.js

Lesson 06—Working with Node.js File System



Lesson Overview

In this lesson, you will be able to understand Node.js file system, differentiate Synchronous and Asynchronous I/O, and understand __dirname and __filename.

Learning Objectives

- Learn about Node.js File System(FS)
- Distinguish between Synchronous and Asynchronous I/O
- Perform Path and Directory operations
- Understand __dirname and __filename
- Understand Asynchronous file reads and writes

Working with Node.js File System

Topic 1—Introduction to Node.js File System(FS)

Working with Node.js File System

INTRODUCTION

- File system (fs) module comes by default with Node.js installation.
- fs module provides a standard wrapper over the standard file I/O operations.
- Most of the scenarios uses fs module to read the contents of a file into memory.
- fs module accesses the physical file system of your computer and is responsible for all the asynchronous and synchronous file I/O operations.
- Its fs.readFile() and fs.writeFile() methods are designed to R/W the file asynchronously.

Standard Method Signature: *fs.readFile(sourceFName [,option], callbackfunction)*

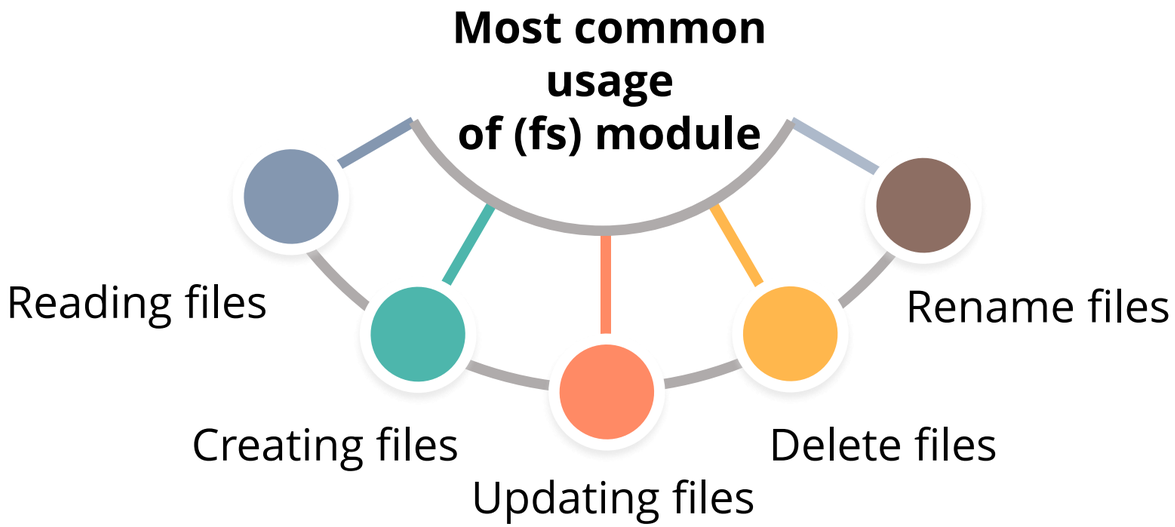
File System

SYNCHRONOUS VS ASYNCHRONOUS OPERATIONS

Every method in the file system executes synchronous and asynchronous functions.

Functions in fs module that end with ‘Sync’ represent synchronous functions.

It is always a good practice to choose an asynchronous method over a synchronous method, as the asynchronous method never blocks a program during the execution.



File System

SYNCHRONOUS VS ASYNCHRONOUS OPERATIONS - EXAMPLE

These examples show how to read a physical file in memory synchronously and asynchronously.

Asynchronous Reading

```
var fs = require("fs");

fs.readFile("source.txt", "utf8", function(error, data) {
 console.log(data);
});
```

Synchronous Reading

```
var fs = require("fs");
var data = fs.readFileSync("source.txt", "utf8");

console.log(data);
```

Demo for File System

Working with File System

Topic 2—Path and Directory Operations

Path and Directory Operations

Node.js fs module is an important aspect of any application that deals with accessing files paths for loading, manipulating, or sharing data.

fs module provides essential handy methods for working with files, paths, and directories.

Mostly, fs module finds out where all files and directories are placed and executes them in certain contexts.

Other languages may have these standard methods; however, node.js has a few important features that might not be available in any other language.

Demo for Path and Directory

Working with File System

Topic 3—__dirname and __filename

__dirname and __filename

Node module informs the developer about the existing file system work by implementing the two standard variables.

`_dirname:` This describes the absolute path of the file that is being executed currently.

`_filename:` This describes the absolute path of the current working directory of the file that is being executed.

Example that uses both *_filename* and *_dirname*

```
console.log(" file name is " + __filename);

console.log("file location is" + __dirname);
```

__dirname and __filename

The output of this code from the machine is:

file name is `C:\Users\appadmin\nodetraining\Intro\demo.js`

file location is `C:\Users\appadmin\nodetraining\Intro`

Object's cwd() method can be implemented to find the current working directory:

```
console.log("current directory " + process.cwd());
```

Demo for __dirname and __filename

Working with File System

Topic 4—Asynchronous File Reading & Writing

Asynchronous File Reading

The code is used to read the file asynchronously using the ReadStream API

```
var fs = require('fs');
var rs = 'D:/userdata.txt';
var readStreamObject = fs.createReadStream(rs, { flags: 'r', encoding:"utf8",fd:
null,mode: 0666, autoClose: true
});
readStreamObject.on('readable', function(){
 console.log("*** Reading data using ReadStream");
});
rs.on('data', function(data){
 console.log(data);
});
readStreamObject.on('end', function(){
 console.log("*** reading completedusing ReadStream");
});
```

Asynchronous File Writing

The code is used to read the file asynchronously using the WriteStream API

```
var fs = require('fs');
var writeSource = D:/userdata.txt’;

var writeStreamObject = fs.createWriteStream(writeSource);

 writeStreamObject.write("Writing to a destination", "utf8");

 fs.readFile(writeSource, "utf8", function(err, data){

 if ( err ){ throw err;}
 console.log(“ Reading file");
 console.log(data);
});
```

Demo for Asynchronous File Read and Write

Quiz

QUIZ 1

Which of the following is the correct way to get an absolute path?

a. os.resolve('main.js')

b. path.resolve('main.js')

c. fs.resolve('main.js')

d. None of the above

QUIZ 1

Which of the following is the correct way to get an absolute path?

a. os.resolve('main.js')

b. path.resolve('main.js')

c. fs.resolve('main.js')

d. None of the above

The correct answer is **b. path.resolve('main.js').**

path.resolve('main.js') is the correct way to get an absolute path.

QUIZ 2

Which method of fs module is used to close a file?

a. fs.close(fd, callback)

b. fs.closeFile(fd, callback)

c. fs.closePath(fd, callback)

d. None of the above

QUIZ 2

Which method of fs module is used to close a file?

a. fs.close(fd, callback)

b. fs.closeFile(fd, callback)

c. fs.closePath(fd, callback)

d. None of the above

The correct answer is **a. fs.close(fd, callback).**

fs.close(fd, callback) is a method of fs module which is used to close a file.

QUIZ 3

Which of the following is true about File I/O in Node applications?

a. Node implements File I/O using simple wrappers around standard POSIX functions

b. Node File System (fs) module should be imported for File I/O operations

c. Both A & B

d. None of the above

QUIZ 3

Which of the following is true about File I/O in Node applications?

a. Node implements File I/O using simple wrappers around standard POSIX functions

b. Node File System (fs) module should be imported for File I/O operations

c. Both A & B

d. None of the above

The correct answer is **c. Both A & B**

In Node Application, Node implements File I/O using simple wrappers around standard POSIX functions and Node File System (fs) module should be imported for File I/O operations.

QUIZ 4

Which of the following is true about the global object “__filename”?

a. __filename represents the filename of the code being executed

b. __filename represents the resolved absolute path of code file

c. Both A & B

d. None of the above

QUIZ 4

Which of the following is true about the global object "__filename"?

a. __filename represents the filename of the code being executed

b. __filename represents the resolved absolute path of code file

c. Both A & B

d. None of the above

The correct answer is **c. Both A & B**

__filename represents the filename of the code being executed and the resolved absolute path of code file.

Key Takeaways

- File system (fs) module comes by default with Node.js installation.
- Functions in fs module that end with ‘Sync’ represent synchronous functions
- fs module provides essential handy methods for working with files, paths, and directories.
- `_filename`: this describes the absolute path of the file that is being executed currently.

  `_dirname`: this describes the absolute path of the current working directory of the file that is being executed.

Thank You