Read the File record in Node.js

In the previous chapters, you found that you used a lot of require (fs) syntax. So what is the syntax to do? This is the syntax to declare fs Module to deploy File I / O operations in Node.js.

In the previous chapters, you found that you used a lot of **require** syntax ("fs"). So what is the syntax to do? This is the syntax to declare **fs** Module to deploy File I / O operations in Node.js. The syntax is as follows:

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

The Concept of Sync vs. Asynchronous in Node.js

Each method in fs Module has synchronous forms and asynchronous forms. Asynchronous methods take a final parameter as an end callback function and get the first parameter as a callback function to handle the error. Using asynchronous methods is better than synchronous methods, because asynchronous methods never lock the program runtime while the sync method does.

For example

To illustrate the I / O operation in Node.js, first create **input.txt** with the content:

```
QTM is a Web page that contains all the scripts
Welcome to the world !!!!!!
```

Create **main.js.** As mentioned above, each method of **fs** Module has two forms that are synchronous and asynchronous. To read the data, I use the asynchronous form readFile () method and readFileSync () of the synchronized form to read the data. These two methods take the first parameter as the file name to read the data from.

```
var fs = require ( "fs" );  // The file method is not available .  readFile (
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
File sharing method: QTM is a Web page that contains all files
Welcome to the world !!!!!!

Let's get married
The file method does not work: QTM is a Web page that contains all files
Welcome to the world !!!!!!
```

In the next section, I will detail the common file I / O activities:

Open a File in Node.js

Syntax

To open a file in asynchronous mode, you use the open () method with the syntax:

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

Parameters

path - This is a string that represents the file name as well as the path to the file.

flags - **Shows** the behavior of the file opened. All values ??may be presented in the table below.

mode - Set the mode for the file, these modes are set only when the file has been created. The default value is 0666, ie readable and writeable.

callback - The callback function takes two parameters, for example (err, fd).

Flags are used for Read / Write files in Node.js

Flag Description Open the file to read. Exception occurs if the file does not exist. r + Open file to read and write. Exception occurs if the file does not exist. rs Open the file to read in synchronous mode. rs + Open file to read and write, tell OS to open it in synchronous mode. w Open the file to record. If the file does not exist, it will create a new file. wx Like 'w' but this operation fails if the file does not exist (ie it does not create a new file). w + Open the file to read and write. If the file does not exist, it will create a new file. wx + Like 'w +' but this operation fails if the file does not exist a Open the file to append. The file will be created if it does not exist. ax Same 'a' but this operation fails if the file does not exist. a + Open the file to read and append. The file will be created if it does not exist. ax + Same as 'a +' but this operation fails if the file does not exist.
For example

The following example illustrates how to open a file to read and write. First, create **main.js** with the following content. The content of the file is quite similar to the above example, you pay attention to the flag used here.

```
  var fs = require ( "fs" );  // File download by using the console .  log ( "Chu
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
  Chuan bi file File download!
  File is a curvy!
```

Get the File information in Node.js

Syntax

To get information about a file in Node.js, you use the stat () method of **fs** Module with the syntax:

```
fs . stat ( path ,  callback )
```

Details about parameters

path - This is a string that represents the file name as well as the path to the file.

callback - A callback function that takes two parameters (err, stats), where stats is an object of fs.Stats printed as in the following example.

In addition to the important properties printed as in the following example, the fs.Stats class also has some useful methods that can be used to check file types. That is:

Method Descriptionstats.isFile () Returns true if it is a file stats.isDirectory () Returns true if it is a stats.isBlockDevice directory () Returns true if it is a Block Device. stats.isCharacterDevice () Returns true if it is a Character Device. stats.isSymbolicLink () Returns true if it is a Symbolic Link. stats.isFIFO () Returns true if it is a FIFO type. stats.isSocket () Returns true if it's a Socket type.
For example

Here is an example to illustrate how to get information about a file. Create **main.js** and use the stat () method of **fs** Module shown above:

```
var fs = require ( "fs" );  console .  log ( "How to save file information!" )
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
 You will be able to file information directly!
 {dev: 1792,
mode: 33188, mode: 33188,
nlink: 1, nlink: 1,
uid: 48, uid: 48,
gid: 48, gid: 48,
rdev: 0, rdev: 0,
blksize: 4096, blksize: 4096,
ino: 4318127, ino: 4318127,
size: 97, size: 97,
blocks: 8, blocks: 8,
atime: Sun Mar 22 2015 13:40:00 GMT-0500 (CDT), atime: Sun Mar 22 2015 13:40:00
mtime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT), mtime: Sun Mar 22 2015 13:40:57
ctime: Sun Mar 22 2015 13:40:57 GMT-0500 (CDT) } ctime: Sun Mar 22 2015 13:40:5
 Lay information File bar!
 isFile?  true
 isDirectory?  false
```

Write data to File in Node.js

Syntax

To write data to File in Node.js, you can use the writeFile () method of fs Module as follows:

```
fs . writeFile ( filename ,  data [,  options ],  callback )
```

This method will override if the file already exists.

Details about parameters

path - This is a string that represents the file name as well as the path to the file.

data - Data as String or Buffer to write to File.

options - This parameter is an object that holds {encoding, mode, flag}. By default, encoding is utf8, mode is 0666 and flag is 'w'

callback - The callback function takes one parameter err and is used to return an error if any error occurs in the write operation.

For example

The following example illustrates how to write data to a file. Creating **main.js** has the following content:

```
var fs = require ( "fs" );  console .  log ( "Earlier file recording" );  fs .
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
The file is written into the file
Write down the file in the bar file!
Doc du lieu king recorded
The file method is not available: Learning Node.js with QTM board!
```

Read data from File in Node.js

Syntax

To read data from a File, you use the read () method with the following syntax:

```
fs . read ( fd ,  buffer ,  offset ,  length ,  position ,  callback )
```

This method will use the parameter fd (short for File Descriptor) to read the file. If you want to read the file using the file name directly, you should use another method.

Details about parameters

fd - An abbreviation of file descriptor returned by the method fs.open ().

buffer - This is the Buffer, where the data is written.

offset - This is the offset in the Buffer so that data begins to record from that location.

length - An integer specifying the number of bytes to read.

position - An integer specifying where to start reading from within the file. If the location is null, the data will be read from the current location of the file.

callback - A callback function that takes three parameters, shaped (err, bytesRead, buffer).

For example

Create **main.js** with the following content:

```
var fs = require ( "fs" );  var buf = new Buffer ( 1024 );  console .  log ( "
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Chuan is a file
File is a curvy!
I want to study the file of mo
97 bytes read
QTM is a Web page that contains all the scripts
Welcome to the world !!!!!!
```

Close File in Node.js

Syntax

To close a file after opening, you use the close () method with the syntax:

```
fs . close ( fd ,  callback )
```

Details about parameters

fd - An abbreviation of file descriptor returned by the method fs.open ().

callback - The callback function takes a parameter to handle the case if there is an exception.

For example

Create **main.js** with content:

```
var fs = require ( "fs" );  var buf = new Buffer ( 1024 );  console .  log ( "
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Chuan is a file
File is a curvy!
I want to study the file of mo
QTM is a Web page that contains all the scripts
Welcome to the world !!!!!!

The file is scrolled.
```

Truncate a File in Node.js

Syntax

To truncate an open file, you use the ftruncate () method with the syntax:

```
fs . ftruncate ( fd ,  len ,  callback )
```

Details about parameters

fd - An abbreviation of file descriptor returned by the method fs.open ().

len - The length of the file after it has been truncate.

callback - The callback function takes a parameter to handle the case if there is an exception.

For example

Create **main.js** with the following content:

```
var fs = require ( "fs" );  var buf = new Buffer ( 1024 );  console .  log ( "
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Chuan is a file
File is a curvy!
You can truncate the file
The file must be truncate the curved bar.
Please read the file
VietNamVo
The file is scrolled.
```

Delete File in Node.js

Syntax

To delete a file in Node.js, you use the unlink () method with the syntax:

```
fs . unlink ( path ,  callback )
```

Details about parameters

path - The file name or path name that points to the file.

callback - The callback function takes a parameter to handle the case if there is an exception.

For example

Create **main.js** with the following content:

```
  var fs = require ( "fs" );  console .  log ( "Chuan is using a file" );  fs .
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
  Chuan is a file
  Rub File bar!
```

Create a directory in Node.js

Syntax

To create a directory in Node.js, you use the mkdir () method with the syntax:

```
fs . mkdir ( path [,  mode ],  callback )
```

Details about parameters

path - This is the directory name that includes the path to the directory.

mode - Mode that determines permissions when accessing folders. The default value is 0777.

callback - The callback function takes a parameter to handle the case if there is an exception.

For example

Create **main.js** with the following content:

```
  var fs = require ( "fs" );  console .  log ( "Chuan is creating a folder / tmp
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Chuan is creating a folder / tmp / test
The folder has a curved bar!
```

Read the directory in Node.js

Syntax

To read the directory in Node.js, you use the readdir () method with the syntax:

```
fs . readdir ( path ,  callback )
```

Details about parameters

path - This is the directory name that includes the path to the directory.

callback - The callback function takes two parameters, form (err, files) in which files are an array containing file names in the directory.

For example

Create main.js with the following content:

```
var fs = require ( "fs" );  console .  log ( "Please read the file / tmp" );
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Please read the news file / tmp
ccmzx99o.out
ccyCSbkF.out
employee.ser
hsperfdata_apache
ki?m TRA
test.txt
```

Delete the directory in Node.js

Syntax

To delete a directory in Node.js, you use the rmdir () method with the syntax:

```
fs . rmdir ( path ,  callback )
```

Details about parameters

path - This is the directory name that includes the path to the directory.

callback - The callback function takes a parameter to handle the case if there is an exception.

For example

Create **main.js** with the following content:

```
var fs = require ( "fs" );  console .  log ( "Chuan bi rub the folder / tmp / t
```

Run main.js to see the result:

```
$ node main . js
```

Check the result:

```
Please read the news file / tmp
ccmzx99o.out
ccyCSbkF.out
employee.ser
hsperfdata_apache
test.txt
```

According to Tutorialspoint

Previous post: Stream in Node.js

Next lesson: Global object in Node.js

You finished reading the article "**Read the File record in Node.js**" edited by the **TipsMake** team. We hope this article has provided you with many useful tech tips and tricks. You can search for similar articles on tips and guides. Thank you for reading and for following us regularly.

© 2019 TipsMake.com