How to Get All Files in Directories (Recursively) with Node.js

Robin Node.js Updated on June 10, 2022

In Node.js, it is very easy to work with files and folders. Node.js provides `fs.readdir()` function to get all files present in a directory. If the parent directory contains sub-directories, you can scan those sub-directories to get files recursively.

In this article, you will know how to get all files in directories and sub-directories with Node.js modules.

We can get all contents from a directory asynchronously or synchronously. If you don't know the differences between them then I will highly suggest you understand the **differences between synchronous and asynchronous programming**.

But in short, in asynchronous programming, an application performs some tasks without blocking the main thread. In this way, an asynchronous program can perform multiple tasks at the same time.

In synchronous programming, an application performs tasks one by one. It blocks the main thread and starts a new task when the previous one completes. That means it can't perform more than one task at the same time.

We will use the **Node.js core fs module** to get all files from a

- **fs.readdir(path, options)** - This method reads the content in a directory asynchronously. By default, it returns the contents of the parent directory which we specify in the first argument. But you can pass some options to also read sub-directories for getting all files.
- **fs.readdirSync(path, options)** - This method reads the content in a directory but in a synchronous way. We need to pass a directory path as its first argument. It will return the contents of that directory. You can also pass some options in the second argument to get files for the sub-directories in the parent directory.

> ***Also Read: Easy Ways to Download Any File From Node.js Server Using Express***

The main difference between `fs.readdir()` and `fs.readdirSync()` methods are that the `fs.readdir()` method works asynchronously and `fs.readdirSync()` method works synchronously. That means the asynchronous method will read a directory without blocking the main thread. If the directory size is larger, it might take some time to complete. Within this time the main application will also work using the main thread.

completes.

There are the steps to get all files from a directory in Node.js

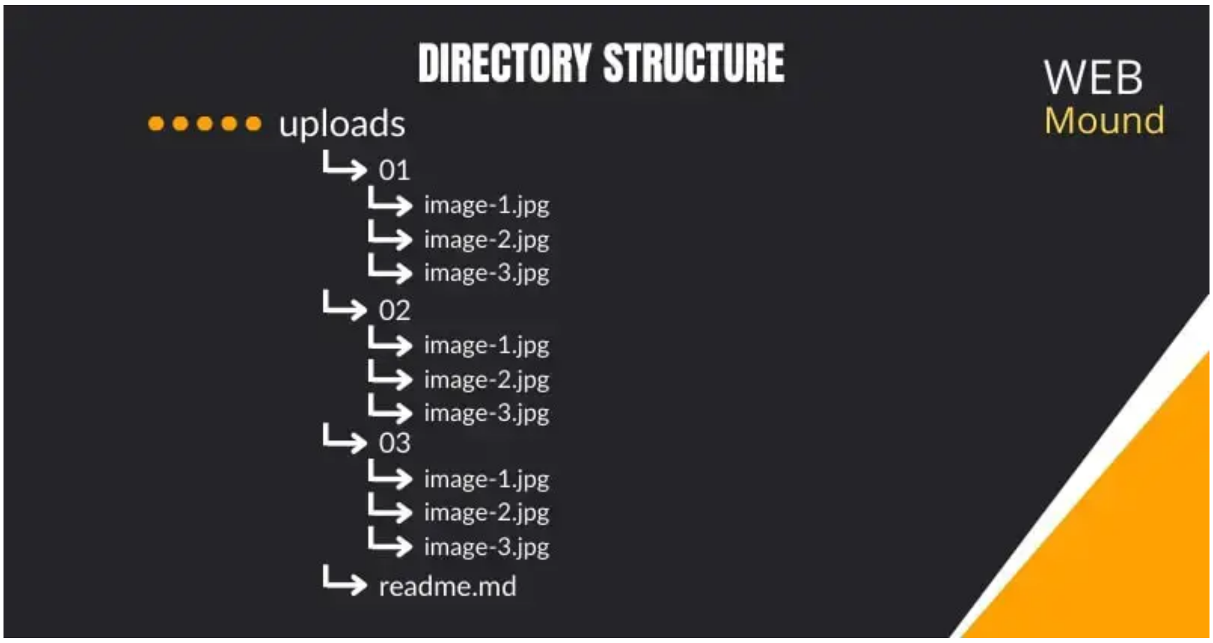
- Import the necessary Node.js package using "**require()**".
- Pass the path and options arguments in the **readdir(path, options)** method. It will return an array with all the contents.
- Run **for** loop on the array of files list returned from the directory.
- Use each file path inside the loop for our application.

If you don't want to get all files from a directory, you can also **search files by names, extensions, or patterns in Node.js** to get specific types of files.

How to Get Files Asynchronously in Directories Using Node.js

First, I will show you how you can get all files asynchronously in a directory and its sub-directories. I always prefer to use this method.

But if your application needs to use the synchronous way for any reason, don't worry. I will also teach you that in the later section.



I have a directory called `uploads` that I want to read. Inside this directory, I have 3 sub-directories `01`, `02`, and `03`. It also has a `readme.md` file. Each sub-directory contains 3 image files.

fsPromises.readdir(): Getting Files in a Directory (without Sub-Directories)

If you have a directory that contains only files not any sub-directory then this method will work for you to get all the files. Here, you don't need to pass any options to the `readdir()` method.

There are 2 versions of the `readdir()` method. One uses a callback function and another uses a promise. We will use the second version in this article. It is very easy to use and looks very clean as well.

```
const { readdir } = require('fs').promises;

const getFileList = async (dirName) => {
```

```
getFileList('uploads').then((files) => {
    console.log(files);
});
```

```
//Output
[ '01', '02', '03', 'readme.md' ]
```

In this example, I have created `getFileList()` function that accepts one argument for a directory path. It is an async function. Inside it, I am using `readdir()` method required from `require("fs").promises` with the `await` keyword.

If we look into the array returned from `readdir()` method, we will see it contains the filename as well as the sub-directory names. But it doesn't return filenames from those sub-directories.

Also Read: Perfect Ways to Serve Static Files in Express and Node.js

That's why this method is suitable for a directory that contains only files.

the `upload` folder.

```
const { statSync } = require('fs');
const { readdir } = require('fs').promises;

const getFileList = async (dirName) => {
    let files = [];
    const items = await readdir(dirName);

    for (const item of items) {
        if (!statSync(`${dirName}/${item}`).isDirectory()) {
            files.push(item);
        }
    }

    return files;
};

getFileList('uploads').then((files) => {
    console.log(files);
});
```

```
//Output
[ 'readme.md' ]
```

Here, I am using `for` loop on the array and checking if the array item is a directory or file using the `statSync()` method. You have to require this method from the Node.js `fs` module.

If the item is not a directory, push it to the `files` array. So the output array will only contain the files.

fsPromises.readdir(): Getting Files Recursively in a Directory (with Sub-Directories)

When we need to get all files recursively including the sub-directories, you have to use `readdir()` method with an option.

In our uploads directory, we have 3 sub-directories. Therefore, this method is suitable for our requirements.

```
const { readdir } = require('fs').promises;

const getFileList = async (dirName) => {
    let files = [];
    const items = await readdir(dirName, { withFileTypes: true });

    for (const item of items) {
        if (item.isDirectory()) {
            files = [
                ...files,
                ...(await getFileList(`${dirName}/${item.name}`)),
            ];
        } else {
            files.push(`${dirName}/${item.name}`);
};

getFileList('uploads').then((files) => {
    console.log(files);
});
```

```
// Output
[
  'uploads/01/image-1.jpg',
  'uploads/01/image-2.jpg',
  'uploads/01/image-3.jpg',
  'uploads/02/image-1.jpg',
  'uploads/02/image-2.jpg',
  'uploads/02/image-3.jpg',
  'uploads/03/image-1.jpg',
  'uploads/03/image-2.jpg',
  'uploads/03/image-3.jpg',
  'uploads/readme.md'
]
```

While using the `readdir()` method with a directory path as the first argument, we have to pass an object with a property `withFileTypes: true` as the second argument.

When we pass this option, the `readdir()` method doesn't just return an array to strings. Instead, the array items contain the metadata about the files and directories.

That's why we can call `isDirectory()` method directly on each item inside the `for` loop. If the current item in the loop is a directory, we reassign the files array by spreading the existing files array items and calling `getFileList()` function with the path of the sub-directory.

You can access the directory or file name using the `name` property on each `item` in the loop. But if the current `item` is not a directory, in the `else` block you push the file path to the `files` array.

When we look into the output, we can see the paths for all the files including the sub-directories.

> ***Also Read: Best Setup to Use TypeScript with Node.js and Express Project***

How to Get Files Synchronously in Directories Using Node.js

Most of the time an asynchronous method is the best choice. But if you want or your application needs to read a directory synchronously then you can use `readdirSync()` method.

As I said before if your directory structure is complex or large in size, this might stop your application from working while scanning the directory. JavaScript is a single-threaded programming language so blocking the main thread will hurt your application get files in a directory and sub-directories.

fs.readdirSync(): Getting Files in a Directory (without Sub-Directories)

This method works exactly like `fs.readdir()` method. It accepts 2 arguments. The first is the directory path and the second is the options.

```
const { readdirSync } = require('fs');

const getFileList = (dirName) => {
    const items = readdirSync(dirName);
    return items;
};

const files = getFileList('uploads');

console.log(files);
```

```
//Output
[ '01', '02', '03', 'readme.md' ]
```

You have to require the `readdirSync()` method from the Node.js `fs` module. Here I am reading the sub-directories so no need to provide options.

If we see the output, we will see that it returns directory and file names in an array. To get only the filenames use the following function.

```
const { readdirSync } = require('fs');

const getFileList = (dirName) => {
    let files = [];
    const items = readdirSync(dirName, { withFileTypes: true });

    for (const item of items) {
        if (!item.isDirectory()) {
            files.push(`${dirName}/${item.name}`);
        }
    }

    return files;
};

const files = getFileList('uploads');

console.log(files);
```

```
// Output
[ 'uploads/readme.md' ]
```

In this function, I am passing `withFileTypes: true` as the option for `readdirSync()` method. Therefore, now it will return an array of metadata about the files and directories.

In the `for` loop, check if the current item is a directory or not. If it is not a directory then push the file path in the `files` array.

In the output, you will get an array with only the file paths.

fs.readdirSync(): Getting Files Recursively in a Directory (with Sub-Directories)

The `fs.readdirSync()` method can be used to get all files recursively in a directory and sub-directories. In this case, you have to pass `withFileTypes: true` option in the method.

```
const { readdirSync } = require('fs');

const getFileList = (dirName) => {
    let files = [];
```

```
            if (item.isDirectory()) {
                files = [...files, ...getFileList(`${dirName}/${item.name
            } else {
                files.push(`${dirName}/${item.name}`);
            }
        }

        return files;
    };

    const files = getFileList('uploads');

    console.log(files);
```

```
// Output
[
  'uploads/01/image-1.jpg',
  'uploads/01/image-2.jpg',
  'uploads/01/image-3.jpg',
  'uploads/02/image-1.jpg',
  'uploads/02/image-2.jpg',
  'uploads/02/image-3.jpg',
  'uploads/03/image-1.jpg',
  'uploads/03/image-2.jpg',
  'uploads/03/image-3.jpg',
  'uploads/readme.md'
]
```

This is exactly the same as you have seen earlier for `readdir()` method. I am checking if the current item is a directory or not inside the `for` loop.

If it is a directory, reassign the `files` array by copping the previous items present in this array using the **spread operator** and calling the `getFileList()` function again with the current sub-directory path.

In the output, you will get an array with a complete list of file paths including the sub-directories.

Conclusion

Reading a directory or folder is not that hard in Node.js. You have built-in methods that you can use. You don't have to install any 3rd party package for it.

That's it, now you know how to get files in a directory and sub-directories with Node.js recursively. You can use the asynchronous or synchronous method. But I always prefer the asynchronous method.

Because it doesn't block the main thread of JavaScript and also speeds up the performance of my application.

I hope it was helpful for you. Now you can scan any directory in your next Node.js project.

Thank you so much for reading. Happy coding!

Related Posts

Robin Node.js

Best Setup to Use TypeScript with Node.js and Express Project

Robin Node.js

How to Connect MongoDB to Node.js Express Using Mongoose

Robin Node.js

Getting Data From req.body in Node.js & Express Server

Recent Posts

Categories

JavaScript

CSS

HTML

Web Dev

Vue

Node.js

TypeScript

Golang

Blog About Contact Terms Privacy Policy Disclaimer