# @nodelib/fs.walk

A library for efficiently walking a directory recursively.

## :bulb: Highlights

* :moneybag: Returns useful information: name, path, dirent and stats (optional).
* :rocket: On Node.js 10.10+ uses the mechanism without additional calls to determine the entry type for performance reasons. See [old and modern mode](https://github.com/nodelib/nodelib/blob/master/packages/fs/fs.scandir/README.md#old-and-modern-mode).
* :gear: Built-in directories/files and error filtering system.
* :link: Can safely work with broken symbolic links.

## Install

npm install @nodelib/fs.walk

## Usage

import \* as fsWalk from '@nodelib/fs.walk';

fsWalk.walk('path', (error, entries) => { /\* … \*/ });

## API

### .walk(path, [optionsOrSettings], callback)

Reads the directory recursively and asynchronously. Requires a callback function.

:book: If you want to use the Promise API, use util.promisify.

fsWalk.walk('path', (error, entries) => { /\* … \*/ });

fsWalk.walk('path', {}, (error, entries) => { /\* … \*/ });

fsWalk.walk('path', new fsWalk.Settings(), (error, entries) => { /\* … \*/ });

### .walkStream(path, [optionsOrSettings])

Reads the directory recursively and asynchronously. [Readable Stream](https://nodejs.org/dist/latest-v12.x/docs/api/stream.html#stream_readable_streams) is used as a provider.

const stream = fsWalk.walkStream('path');

const stream = fsWalk.walkStream('path', {});

const stream = fsWalk.walkStream('path', new fsWalk.Settings());

### .walkSync(path, [optionsOrSettings])

Reads the directory recursively and synchronously. Returns an array of entries.

const entries = fsWalk.walkSync('path');

const entries = fsWalk.walkSync('path', {});

const entries = fsWalk.walkSync('path', new fsWalk.Settings());

#### path

* Required: true
* Type: string | Buffer | URL

A path to a file. If a URL is provided, it must use the file: protocol.

#### optionsOrSettings

* Required: false
* Type: Options | Settings
* Default: An instance of Settings class

An [Options](#options) object or an instance of [Settings](#settings) class.

:book: When you pass a plain object, an instance of the Settings class will be created automatically. If you plan to call the method frequently, use a pre-created instance of the Settings class.

### Settings([options])

A class of full settings of the package.

const settings = new fsWalk.Settings({ followSymbolicLinks: true });

const entries = fsWalk.walkSync('path', settings);

## Entry

* name — The name of the entry (unknown.txt).
* path — The path of the entry relative to call directory (root/unknown.txt).
* dirent — An instance of [fs.Dirent](http://./src/types/index.ts) class.
* [stats] — An instance of fs.Stats class.

## Options

### basePath

* Type: string
* Default: undefined

By default, all paths are built relative to the root path. You can use this option to set custom root path.

In the example below we read the files from the root directory, but in the results the root path will be custom.

fsWalk.walkSync('root'); // → ['root/file.txt']

fsWalk.walkSync('root', { basePath: 'custom' }); // → ['custom/file.txt']

### concurrency

* Type: number
* Default: Infinity

The maximum number of concurrent calls to fs.readdir.

:book: The higher the number, the higher performance and the load on the File System. If you want to read in quiet mode, set the value to 4 \* os.cpus().length (4 is default size of [thread pool work scheduling](http://docs.libuv.org/en/v1.x/threadpool.html#thread-pool-work-scheduling)).

### deepFilter

* Type: [DeepFilterFunction](http://./src/settings.ts)
* Default: undefined

A function that indicates whether the directory will be read deep or not.

// Skip all directories that starts with `node\_modules`

const filter: DeepFilterFunction = (entry) => !entry.path.startsWith('node\_modules');

### entryFilter

* Type: [EntryFilterFunction](http://./src/settings.ts)
* Default: undefined

A function that indicates whether the entry will be included to results or not.

// Exclude all `.js` files from results

const filter: EntryFilterFunction = (entry) => !entry.name.endsWith('.js');

### errorFilter

* Type: [ErrorFilterFunction](http://./src/settings.ts)
* Default: undefined

A function that allows you to skip errors that occur when reading directories.

For example, you can skip ENOENT errors if required:

// Skip all ENOENT errors

const filter: ErrorFilterFunction = (error) => error.code == 'ENOENT';

### stats

* Type: boolean
* Default: false

Adds an instance of fs.Stats class to the [Entry](#entry).

:book: Always use fs.readdir with additional fs.lstat/fs.stat calls to determine the entry type.

### followSymbolicLinks

* Type: boolean
* Default: false

Follow symbolic links or not. Call fs.stat on symbolic link if true.

### throwErrorOnBrokenSymbolicLink

* Type: boolean
* Default: true

Throw an error when symbolic link is broken if true or safely return lstat call if false.

### pathSegmentSeparator

* Type: string
* Default: path.sep

By default, this package uses the correct path separator for your OS (\ on Windows, / on Unix-like systems). But you can set this option to any separator character(s) that you want to use instead.

### fs

* Type: FileSystemAdapter
* Default: A default FS methods

By default, the built-in Node.js module (fs) is used to work with the file system. You can replace any method with your own.

interface FileSystemAdapter {

lstat: typeof fs.lstat;

stat: typeof fs.stat;

lstatSync: typeof fs.lstatSync;

statSync: typeof fs.statSync;

readdir: typeof fs.readdir;

readdirSync: typeof fs.readdirSync;

}

const settings = new fsWalk.Settings({

fs: { lstat: fakeLstat }

});

## Changelog

See the [Releases section of our GitHub project](https://github.com/nodelib/nodelib/releases) for changelog for each release version.

## License

This software is released under the terms of the MIT license.