# @nodelib/fs.scandir

List files and directories inside the specified directory.

## :bulb: Highlights

The package is aimed at obtaining information about entries in the directory.

* :moneybag: Returns useful information: name, path, dirent and stats (optional).
* :gear: On Node.js 10.10+ uses the mechanism without additional calls to determine the entry type. See [old and modern mode](#old-and-modern-mode).
* :link: Can safely work with broken symbolic links.

## Install

npm install @nodelib/fs.scandir

## Usage

import \* as fsScandir from '@nodelib/fs.scandir';

fsScandir.scandir('path', (error, stats) => { /\* … \*/ });

## API

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

Returns an array of plain objects ([Entry](#entry)) with information about entry for provided path with standard callback-style.

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

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

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

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

Returns an array of plain objects ([Entry](#entry)) with information about entry for provided path.

const entries = fsScandir.scandirSync('path');

const entries = fsScandir.scandirSync('path', {});

const entries = fsScandir.scandirSync(('path', new fsScandir.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](#settingsoptions) 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 fsScandir.Settings({ followSymbolicLinks: false });

const entries = fsScandir.scandirSync('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. On Node.js below 10.10 will be emulated by [DirentFromStats](http://./src/utils/fs.ts) class.
* stats (optional) — An instance of fs.Stats class.

For example, the scandir call for tools directory with one directory inside:

{

dirent: Dirent { name: 'typedoc', /\* … \*/ },

name: 'typedoc',

path: 'tools/typedoc'

}

## Options

### stats

* Type: boolean
* Default: false

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

:book: Always use fs.readdir without the withFileTypes option. ??TODO??

### 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 use 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](http://./src/adapters/fs.ts)
* 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 fsScandir.Settings({

fs: { lstat: fakeLstat }

});

## old and modern mode

This package has two modes that are used depending on the environment and parameters of use.

### old

* Node.js below 10.10 or when the stats option is enabled

When working in the old mode, the directory is read first (fs.readdir), then the type of entries is determined (fs.lstat and/or fs.stat for symbolic links).

### modern

* Node.js 10.10+ and the stats option is disabled

In the modern mode, reading the directory (fs.readdir with the withFileTypes option) is combined with obtaining information about its entries. An additional call for symbolic links (fs.stat) is still present.

This mode makes fewer calls to the file system. It's faster.

## 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.