fs-extra to fs

07 Mar 2021

I'm a big fan of the fs-extra module for Node.js. It has made my life much easier over the years. However, as I migrate my modules from CommonJS to ECMAScript Modules (ESM) and as I'm looking at bundling Place into a single JS file for deployment using esbuild, I've started removing third-party dependencies wherever I can easily replicate their behaviour using modules from the Node standard library.

For the move from `fs-extra` to `fs`, this is helped by Node.js implementing some of the features that were hitherto unique to `fs-extra` in version 14.x (LTS).

(There still isn't an equivalent for recursive *copying* in the standard library. So you're probably better off sticking to `fs-extra` is you need that functionality.)

So, without further ado, here's a short summary of `fs-extra` methods and their equivalents in Node's standard `fs` module that I've encountered, both as a reference for you and for my future self. (I'll add to the list as I find more.)

Methods that need migrating

Not every `fs` method is re-implemented in `fs-extra`. It simply proxies calls to the built-in module for any methods it does not implement itself. These are the methods that have custom implementations that will need migration:

```
copy, emptyDir, ensureFile, ensureDir, ensureLink, ensureSymlink,
mkdirp, mkdirs, move, outputFile, outputJson, pathExists, readJson,
remove, writeJson
```

This post is not an exhaustive give to every method, just the ones I've had to migrate myself.

For simplicity, sync methods are shown, but the changes apply equally to the async methods.

Creating recursive directories

fs-extra

```
fs.mkdirpSync('/some/directory/structure/')
```

fs

```
fs.mkdirSync('/some/directory/structure/', { recursive: true })
```

Removing a directory and all its contents and not failing if the directory doesn't exist

fs-extra

```
fs.removeSync('/some/directory')
```

fs

```
fs.rmSync('/some/directory', { recursive: true, force: true })
```

Regular expression find/replace

Find:

```
fs.removeSync\((.*?)\)
```

Replace:

```
fs.rmSync($1, {recursive: true, force: true})
```

Moving a directory on the same device (and overwriting the destination if it exists)

fs-extra

```
fs.moveSync(sourcePath, destinationPath, { overwrite: true })
```

fs

```
fs.renameSync(sourcePath, destinationPath)
```

Note: these are not strictly equivalent. The `move` methods in `fs-extra` act like the `mv` command and they will work across devices

also. The `rename` methods in `fs` work like the `rename` system call and will not work across devices. Having Node's `rename` methods work across devices is currently marked as a "won't fix."

Also, the `overwrite: true` behaviour is the default/only behaviour in the standard library.

Regular expression find/replace

Find:

```
fs.moveSync\((.+?), (.+?)(, .+?)?\)
```

Replace:

```
fs.renameSync($1, $2)
```

Copying a file

fs-extra

```
fs.copySync(sourcePath, destinationPath)
```

fs

```
fs.copyFileSync(sourcePath, destinationPath)
```

Copying a folder recursively

Sadly, there's no simple way to do this using the standard `fs` module as it only provides a `copyFile()` method and does not have

the equivalent of `fs-extra`'s `copy()` method.

Ensuring that a directory exists

fs-extra

```
fs.ensureDirSync('/some/directory/structure')
```

fs

```
function ensureDirSync (directory) {
  if (!fs.existsSync(directory)) {
    fs.mkdirSync(directory, { recursive: true })
  }
}

ensureDirSync('/some/directory/structure')
```

Note: `fs-extra` also supports an option that is either an integer or `{mode: <integer>}`. If you need that functionality, use the `fs.chmodSync()` method after creating the directory.

Writing a file

fs-extra

```
fs.outputFileSync(…)
```

fs

```
fs.writeFileSync(…)
```

Note that with `fs-extra`, if the parent directory doesn't exist, it is created.

Bonus: promises and async/await

To use the asynchronous methods in Node's `fs` module with `async`/`await`, import the `fs/promises` module instead. Note that the latter does not include the synchronous methods or stream methods.

e.g., To remove a directory recursively and not fail if it doesn't exist:

```
import fsPromises from 'fs/promises'

fsPromises.rm('/some/directory', {recursive: true, force: true})
```

About the author

I'm an activist, designer, and developer. I'm one-half of Small Technology Foundation, a tiny and independent two-person not-for-profit based in Ireland. We advocate for and build small technology to protect personhood and democracy in the digital network age.

To support my work, become a patron of our foundation.

I'm available for public speaking and media inquiries.