Question

utils.ts

Do you have one of the following...

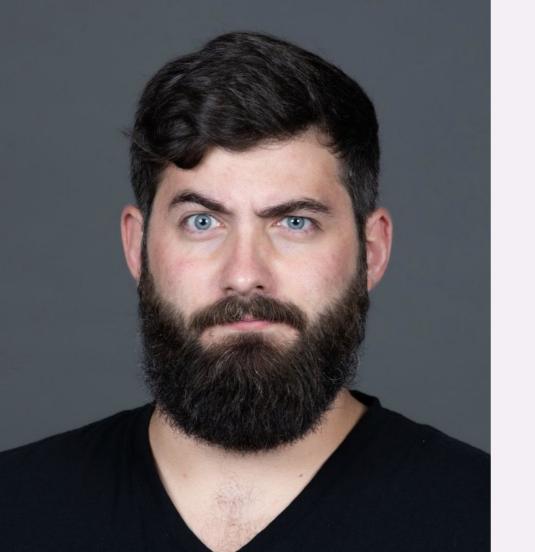
```
utils/
extras/
utils.js
```

And are they copied from different projects?

Then you might have a problem

From Zero To Hero

Building and Shipping Your First JavaScript Library



Hello!

Mike Hartington

Director of DevRel @ ionic

@mhartington@mhartington.io

What Is A JS Library?

Spoiler, it's complicated

According to npm:

A package is a file or directory that is described by a package.json file. A package must contain a **package.json** file in order to be published to the npm registry.

Very Basic

As bare bones as it can be

```
1 ./my-lib
2 |-- package.json
3 |-- index.js
```

Package.json

```
1 {
2    "name": "my-lib",
3    "version": "1.0.0",
4    "main": "index.js",
5 }
```

Let's make our own package!

If it's that simple

Why Bother?

Code isolation

Sharing across projects

Versioned separate from everything else

To the Terminal!

main

Summary

File Structure

```
index.js
index.js
node_modules/
package-lock.json
package.json
safe.js
```

Requiring Dependencies

```
const os = require('os');
const path = require('path');
const stream = require('stream');

const fs = require('fs-extra');

const statSafe = require('./safe').stat;
const readdirSafe = require('./safe').readdir;
```

Exporting Functions

```
const statSafe = require('./safe').stat;
const readdirSafe = require('./safe').readdir;
async function fileToString(filePath) {...}

module.exports = {
   statSafe,
   readdirSafe,
   fileToString,
};
```

Exposing to Node

```
1 {
2    "name": "mh-file-utils",
3    "version": "0.0.1",
4    "main": "index.js",
5    "dependencies": {
6        "fs-extra": "^11.1.1"
7    }
8 }
```

Congrats

You just published an npm package 🎉

But it's 2023

The CJS-ESM Problem

What is CJS?

CommonJS: Package format from 2009

Uses requires and module exports

Mainly only used in Node

What is ESM?

ES Modules: Package format from 2015

Uses import and export

Browser, Node, and beyond!

Both ESM and CJS are supported in Node*

Up to package authors to configure

It can be...confusing

To the Terminal!

esm-port

Summary

File Structure

```
index.mjs
node_modules
package-lock.json
package.json
safe.mjs
```

Requiring Dependencies

```
import { readFile } from 'fs/promises'
import 'fs-extra/esm';

import { stat as statSafe, readdir as readdirSafe } from './safe.mjs';
```

Exporting Functions

```
1 export default {
2  statSafe,
3  readdirSafe,
4  fileToString,
5 };
```

Exposing to Node

```
4 "type": "module",
```

Congrats

It's about to get more complicated 🙃

Main Vs Module Vs Exports

main

"main": "./index.cjs"

CJS was exposing the main entry

Only supports one entry point

Not deprecated, but specialized

module

"module": "./index.mjs"

Older ESM-older entry point

Only one ESM entry point

Not recommended (expect for TypeScript)

exports

Object for defining multiple entry points

Mostly replaces older module

Supports both ESM & CJS

To the Terminal!

esm-cjs

Summary

Use exports when possible

ESM should be the default (it's 2023)

Dual Packages should be avoided

Adding A Build Step

Build Process

Only if you need it

Simplifies what you ship

Can make dev more complex

tsup

esbuild based build system

JavaScript, TypeScript, and more

lt's not webpack 🎉

To the Terminal!

esm-better

Alternatives

unbuild (unified not "un")

Robust ecosystem

Used by vite (internally), and others



Automation

Making releasing this easy

Conventional Commits

Meaningful commits to drive versioning

chore() , feat(scope) , BREAKING

Makes your team write good messages

Semantic Release

Fully automated release process

Commit-drive versioning

Connect with CI to handle release process

semantic-release-cli

To the Terminal!

automation

Summary

Init semantic-release

```
npx semantic-release-cli setup
? What is your npm registry? 'https://registry.npmjs.org/'
? What is your npm username? 'mhartington'
? What is your npm password? '[hidden]'
? Provide a GitHub Personal Access Token? ''
? What CI are you using? 'Github Actions'
```

Add a workflow

```
name: CI
on:
  push:
   branches:
     stable
jobs:
  test:
 runs-on: ubuntu-latest
    steps:
     - uses: actions/checkout@v3
with:
       fetch-depth: 0
     - uses: actions/setup-node@v3
       with:
       node-version: 18
     - run: npm ci
     - run: npm build
     - name: Release
       env:
         NPM_TOKEN: ${{ secrets.NPM_TOKEN }}
        run: npm run semantic-release
```

Have meaningful commits

```
git commit -m 'feat(scope): add a new feature'
git commit -m 'chore(): update formatting'
git commit -m 'fix(scope): return correct error code'
```

Congrats, you've shipped your first Library!

Wrapping up

You do not need to do all of this

A simple package json + index js is also valid

JS Ecosystem is confusing 🙃

But you got this

mhartington/first-js-lib-demo

esm-port

esm-cjs

esm-better

automation

Questions?

</html>