@npmcli/installed-package-contents

Get the list of files installed in a package in node_modules, including bundled dependencies.

This is useful if you want to remove a package node from the tree *without* removing its child nodes, for example to extract a new version of the dependency into place safely.

It's sort of the reflection of npm-packlist, but for listing out the *installed* files rather than the files that *will* be installed. This is of course a much simpler operation, because we don't have to handle ignore files or package.json `files` lists.

USAGE

```
// programmatic usage
const pkgContents = require('@npmcli/installed-package-contents')

pkgContents({ path: 'node_modules/foo', depth: 1 }).then(files => {
  // files is an array of items that need to be passed to
  // rimraf or moved out of the way to make the folder empty
  // if foo bundled dependencies, those will be included.
  // It will not traverse into child directories, because we set
  // depth:1 in the options.
  // If the folder doesn't exist, this returns an empty array.
})

pkgContents({ path: 'node_modules/foo', depth: Infinity }).then(files => {
  // setting depth:Infinity tells it to keep walking forever
  // until it hits something that isn't a directory, so we'll
  // just get the list of all files, but not their containing
  // directories.
})
```

As a CLI:

```
$ installed-package-contents node_modules/bundle-some -d1
node_modules/.bin/some
node_modules/bundle-some/package.json
node_modules/bundle-some/node_modules/@scope/baz
node_modules/bundle-some/node_modules/.bin/foo
node_modules/bundle-some/node_modules/foo
```

CLI options:

```
Usage:
  installed-package-contents <path> [-d<n> --depth=<n>]
```

```
Lists the files installed for a package specified by <path>.

Options:
  -d<n> --depth=<n>   Provide a numeric value ("Infinity" is allowed)
                      to specify how deep in the file tree to traverse.
                      Default=1
  -h --help           Show this usage information
```

OPTIONS

- `depth` Number, default `1`. How deep to traverse through folders to get contents. Typically you'd want to set this to either `1` (to get the surface files and folders) or `Infinity` (to get all files), but any other positive number is supported as well. If set to `0` or a negative number, returns the path provided and (if it is a package) its set of linked bins.
- `path` Required. Path to the package in `node_modules` where traversal should begin.

RETURN VALUE

A Promise that resolves to an array of fully-resolved files and folders matching the criteria. This includes all bundled dependencies in `node_modules`, and any linked executables in `node_modules/.bin` that the package caused to be installed.

An empty or missing package folder will return an empty array. Empty directories *within* package contents are listed, even if the `depth` argument would cause them to be traversed into.

CAVEAT

If using this module to generate a list of files that should be recursively removed to clear away the package, note that this will leave empty directories behind in certain cases:

- If all child packages are bundled dependencies, then the `node_modules` folder will remain.
- If all child packages within a given scope were bundled dependencies, then the `node_modules/@scope` folder will remain.
- If all linked bin scripts were removed, then an empty `node_modules/.bin` folder will remain.

In the interest of speed and algorithmic complexity, this module does *not* do a subsequent readdir to see if it would remove all directory entries, though it would be easier to look at if it returned `node_modules` or `.bin` in that case rather than the contents. However, if the intent is to pass these arguments to `rimraf`, it hardly makes sense to do *two* `readdir` calls just so that we can have the luxury of having to make a third.

Since the primary use case is to delete a package's contents so that they can be re-filled with a new version of that package, this caveat does not pose a problem. Empty directories are already ignored by both npm and git.