@11ty/recursive-copy

A temporary fork of `timkendrick/recursive-copy` to satisfy https://github.com/11ty/eleventy/issues/3299 as Eleventy slowly moves to use Node native API `fs.cp`.

- v4.x requires Node 18+
- v3.x requires Node 0.10+
- v2.x requires Node 0.10+

Simple, flexible file copy utility

Features

- Recursively copy whole directory hierarchies
- Choose which files are copied by passing a filter function, regular expression or glob
- Rename files dynamically, including changing the output path
- Transform file contents using streams
- Choose whether to overwrite existing files
- Choose whether to copy system files
- Filters out junk files by default
- Emits start, finish and error events for each file that is processed
- Optional promise-based interface

Examples

Node-style callback interface

```
var copy = require('@11ty/recursive-
        copy');

copy('src', 'dest', function(error,
        results) {
    if (error) {
        console.error('Copy failed: ' +
        error);
    } else {
        console.info('Copied ' +
        results.length + ' files');
    }
});
```

Promise interface

```
var copy = require('@11ty/recursive-
        copy');

copy('src', 'dest')
    .then(function(results) {
        console.info('Copied ' +
        results.length + ' files');
    })
    .catch(function(error) {
        console.error('Copy failed: ' +
        error);
    });
```

ES2015+ usage

```
import copy from '@11ty/recursive-copy';

try {
    const results = await copy('src',
        'dest');
    console.info('Copied ' +
        results.length + ' files');
} catch (error) {
    console.error('Copy failed: ' +
        error);
}
```

Advanced options

```
var copy = require('@11ty/recursive-
        copy');

var path = require('path');
var through = require('through2');

var options = {
    overwrite: true,
    expand: true,
    dot: true,
    junk: true,
    filter: [
        '**/*',
        '!.htpasswd'
    ],
    rename: function(filePath) {
        return filePath + '.orig';
    },
    transform: function(src, dest,
        stats) {
        if (path.extname(src) !==
        '.txt') { return null; }
        return through(function(chunk,
        enc, done)  {
            var output =
        chunk.toString().toUpperCase();
            done(null, output);
        });
    }
};

copy('src', 'dest', options)
    .on(copy.events.COPY_FILE_START,
        function(copyOperation) {
        console.info('Copying file ' +
        copyOperation.src + '...');
    })
    .on(copy.events.COPY_FILE_COMPLETE,
        function(copyOperation) {
        console.info('Copied to ' +
        copyOperation.dest);
    })
    .on(copy.events.ERROR,
        function(error, copyOperation) {
        console.error('Unable to copy '
        + copyOperation.dest);
    })
    .then(function(results) {
        console.info(results.length + '
        file(s) copied');
    })
    .catch(function(error) {
```

CopyOperation

Property	Type	Description
`src`	`string`	Source path of the relevant file/folder/symlink
`dest`	`string`	Destination path of the relevant file/folder/symlink
`stats`	`fs.Stats`	Stats for the relevant file/folder/symlink

Event	Handler signature
`copy.events.ERROR`	`function(error`
`copy.events.COMPLETE`	`function(Array`
`copy.events.CREATE_DIRECTORY_START`	`function(CopyO`
`copy.events.CREATE_DIRECTORY_ERROR`	`function(error`
`copy.events.CREATE_DIRECTORY_COMPLETE`	`function(CopyO`
`copy.events.CREATE_SYMLINK_START`	`function(CopyO`
`copy.events.CREATE_SYMLINK_ERROR`	`function(error`
`copy.events.CREATE_SYMLINK_COMPLETE`	`function(CopyO`
`copy.events.COPY_FILE_START`	`function(CopyO`
`copy.events.COPY_FILE_ERROR`	`function(error`
`copy.events.COPY_FILE_COMPLETE`	`function(CopyO`

…where the types referred to in the handler signature are as follows:

`ErrorInfo`

Property	Type	Description
`src`	`string`	Source path of the file/folder/symlink that failed to copy
`dest`	`string`	Destination path of the file/folder/symlink that failed to copy

```
        return console.error('Copy
        failed: ' + error);
    });
```

Usage

`copy(src, dest, [options], [callback])`

Recursively copy files and folders from `src` to `dest`

Arguments:

Name	Type	Required	Default	Descripti
`src`	`string`	Yes	N/A	Source fi path
`dest`	`string`	Yes	N/A	Destinati folder pat
`options.overwrite`	`boolean`	No	`false`	Whether overwrite destinatio
`options.expand`	`boolean`	No	`false`	Whether symbolic
`options.dot`	`boolean`	No	`false`	Whether files begi with a `.`
`options.junk`	`boolean`	No	`false`	Whether OS junk f

Name	Type	Required	Default	Description
				(e.g. .DS_Sto Thumbs.db)
options.filter	function, RegExp, string, array	No	null	Filter function / regular expression / glc that determines which files to c (uses maximatc
options.rename	function	No	null	Function that m source paths to destination path
options.transform	function	No	null	Function that returns a transfo stream used to modify file contents
options.results	boolean	No	true	Whether to retu an array of copy results
options.concurrency	number	No	255	Maximum num of simultaneous copy operations
options.debug	boolean	No	false	Whether to log debug informat
callback	function	No	null	Callback, invok on success/failu

Returns:

Promise<Array> Promise, fulfilled with array of copy results:

```
[
    {
```

```
        "src": "/path/to/src",
        "dest": "/path/to/dest",
        "stats": <Stats>
    },
    {
        "src": "/path/to/src/file.txt",
        "dest": "/path/to/dest/
file.txt",
        "stats": <Stats>
    },
    {
        "src": "/path/to/src/subfolder",
        "dest": "/path/to/dest/
subfolder",
        "stats": <Stats>
    },
    {
        "src": "/path/to/src/subfolder/
nested.txt",
        "dest": "/path/to/dest/
subfolder/nested.txt",
        "stats": <Stats>
    }
]
```

Events

The value returned by the copy function implements the EventEmitter interface, and emits the following events: