escalade

> *A tiny (183B to 210B) and fast utility to ascend parent directories*

With escalade, you can scale parent directories until you've found what you're looking for.
Given an input file or directory, `escalade` will continue executing your callback function until either:

1. the callback returns a truthy value
2. `escalade` has reached the system root directory (eg, `/`)

> ***Important:***
> *Please note that `escalade` only deals with direct ancestry – it will not dive into parents' sibling directories.*

Notice: As of v3.1.0, `escalade` now includes Deno support! Please see Deno Usage below.

Install

```
$ npm install --save escalade
```

Modes

There are two "versions" of `escalade` available:

"async"

> ***Node.js:*** *>= 8.x*
> ***Size (gzip):*** *210 bytes*
> ***Availability:*** *CommonJS, ES Module*

This is the primary/default mode. It makes use of `async` / `await` and `util.promisify`.

"sync"

> ***Node.js:*** *>= 6.x*
> ***Size (gzip):*** *183 bytes*
> ***Availability:*** *CommonJS, ES Module*

This is the opt-in mode, ideal for scenarios where `async` usage cannot be supported.

Usage

Example Structure

```
/Users/lukeed
  └── oss
    ├── license
    └── escalade
      ├── package.json
      └── test
```

```
└── fixtures
  ├── index.js
  └── foobar
    └── demo.js
```

Example Usage

```
//~> demo.js
import { join } from 'path';
import escalade from 'escalade';

const input = join(__dirname, 'demo.js');
// or: const input = __dirname;

const pkg = await escalade(input, (dir, names) => {
  console.log('~> dir:', dir);
  console.log('~> names:', names);
  console.log('---');

  if (names.includes('package.json')) {
    // will be resolved into absolute
    return 'package.json';
  }
});

//~> dir: /Users/lukeed/oss/escalade/test/fixtures/foobar
//~> names: ['demo.js']
//---
//~> dir: /Users/lukeed/oss/escalade/test/fixtures
//~> names: ['index.js', 'foobar']
//---
//~> dir: /Users/lukeed/oss/escalade/test
//~> names: ['fixtures']
//---
//~> dir: /Users/lukeed/oss/escalade
//~> names: ['package.json', 'test']
//---

console.log(pkg);
//=> /Users/lukeed/oss/escalade/package.json

// Now search for "missing123.txt"
// (Assume it doesn't exist anywhere!)
const missing = await escalade(input, (dir, names) => {
  console.log('~> dir:', dir);
  return names.includes('missing123.txt') && 'missing123.txt';
});

//~> dir: /Users/lukeed/oss/escalade/test/fixtures/foobar
//~> dir: /Users/lukeed/oss/escalade/test/fixtures
//~> dir: /Users/lukeed/oss/escalade/test
//~> dir: /Users/lukeed/oss/escalade
```

```
//~> dir: /Users/lukeed/oss
//~> dir: /Users/lukeed
//~> dir: /Users
//~> dir: /

console.log(missing);
//=> undefined
```

> ***Note:*** *To run the above example with "sync" mode, import from* `escalade/sync` *and remove the* `await` *keyword.*

API

escalade(input, callback)

Returns: `string|void` or `Promise<string|void>`

When your `callback` locates a file, `escalade` will resolve/return with an absolute path.
If your `callback` was never satisfied, then `escalade` will resolve/return with nothing (undefined).

> ***Important:***
> *The* `sync` *and* `async` *versions share the same API.*
> *The* ***only*** *difference is that* `sync` *is not Promise-based.*

input

Type: `string`

The path from which to start ascending.

This may be a file or a directory path.
However, when `input` is a file, `escalade` will begin with its parent directory.

> ***Important:*** *Unless given an absolute path,* `input` *will be resolved from* `process.cwd()` *location.*

callback

Type: `Function`

The callback to execute for each ancestry level. It always is given two arguments:

1. `dir` - an absolute path of the current parent directory
2. `names` - a list (`string[]`) of contents *relative to* the `dir` parent

> ***Note:*** *The* `names` *list can contain names of files and directories.*

When your callback returns a *falsey* value, then `escalade` will continue with `dir` 's parent directory, re-invoking your callback with new argument values.

When your callback returns a string, then `escalade` stops iteration immediately.
If the string is an absolute path, then it's left as is. Otherwise, the string is resolved into an absolute path *from* the `dir` that housed the satisfying condition.

> ***Important:*** *Your* `callback` *can be a* `Promise/AsyncFunction` *when using the "async" version of* `escalade` .

Benchmarks

> *Running on Node.js v10.13.0*

```
# Load Time
  find-up          3.891ms
  escalade         0.485ms
  escalade/sync    0.309ms

# Levels: 6 (target = "foo.txt"):
  find-up           x 24,856 ops/sec ±6.46% (55 runs sampled)
  escalade          x 73,084 ops/sec ±4.23% (73 runs sampled)
  find-up.sync      x  3,663 ops/sec ±1.12% (83 runs sampled)
  escalade/sync     x  9,360 ops/sec ±0.62% (88 runs sampled)

# Levels: 12 (target = "package.json"):
  find-up           x 29,300 ops/sec ±10.68% (70 runs sampled)
  escalade          x 73,685 ops/sec ± 5.66% (66 runs sampled)
  find-up.sync      x  1,707 ops/sec ± 0.58% (91 runs sampled)
  escalade/sync     x  4,667 ops/sec ± 0.68% (94 runs sampled)

# Levels: 18 (target = "missing123.txt"):
  find-up           x 21,818 ops/sec ±17.37% (14 runs sampled)
  escalade          x 67,101 ops/sec ±21.60% (20 runs sampled)
  find-up.sync      x  1,037 ops/sec ± 2.86% (88 runs sampled)
  escalade/sync     x  1,248 ops/sec ± 0.50% (93 runs sampled)
```

Deno

As of v3.1.0, `escalade` is available on the Deno registry.

Please note that the API is identical and that there are still two modes from which to choose:

```ts
// Choose "async" mode
import escalade from 'https://deno.land/escalade/async.ts';

// Choose "sync" mode
import escalade from 'https://deno.land/escalade/sync.ts';
```

> ***Important:*** *The `allow-read` permission is required!*

Related

- premove - A tiny (247B) utility to remove items recursively
- totalist - A tiny (195B to 224B) utility to recursively list all (total) files in a directory
- mk-dirs - A tiny (420B) utility to make a directory and its parents, recursively

License

MIT © Luke Edwards