maximatch

Extends multimatch() with support for filter functions and regular expressions

Install

```
$ npm install --save maximatch
```

Usage

```
var maximatch = require('maximatch');

maximatch(['unicorn', 'cake',
        'rainbows'], ['*', '!cake']);
//=> ['unicorn', 'rainbows']

maximatch(['unicorn', 'cake',
        'rainbows'], function(path) {
        return path.length > 4; });
//=> ['unicorn', 'rainbows']

maximatch(['unicorn', 'cake',
        'rainbows'], /^[^k]+$/);
```

```
//=> ['unicorn', 'rainbows']

maximatch(['unicorn', 'cake',
        'rainbows'], [function(path) {
        return path.charAt(0) ===
        'u'; }, /w/]);
//=> ['unicorn', 'rainbows']
```

See the tests for more usage examples and expected matches.

API

Same as `minimatch.match()` except for `pattern` also accepting a filter function, a regular expression, or an array that can contain globs, filter functions and regular expressions.

```
var results = maximatch(paths,
        patterns);
```

The return value is an array of matching paths.

How multiple patterns work

Positive patterns (e.g. `foo` or `*`) add to the results, while negative patterns (e.g. `!foo`) subtract from the results.

Therefore a lone negation (e.g. `['!foo']`) will never match anything – use `['*', '!foo']` instead.

Globbing patterns

Just a quick overview.

- `*` matches any number of characters, but not `/`
- `?` matches a single character, but not `/`
- `**` matches any number of characters, including `/`, as long as it's the only thing in a path part
- `{}` allows for a comma-separated list of "or" expressions
- `!` at the beginning of a pattern will negate the match

Related

See globby if you need to match against the filesystem instead of a list.

License