```
['foo.js',
        'bar.js'].filter(matcher); //
        ['foo.js']
```

Changelog

See release notes page on GitHub

- **v3.0:** Removed `startIndex` and `endIndex` arguments. Node 8.x-only.
- **v2.0:** micromatch moves away from minimatch-parity and inline with Bash. This includes handling backslashes differently (see https://github.com/micromatch/micromatch#backslashes for more information).
- **v1.2:** anymatch uses micromatch for glob pattern matching. Issues with glob pattern matching should be reported directly to the micromatch issue tracker.

License

ISC

anymatch build unknown

Javascript module to match a string against a regular expression, glob, string, or function that takes the string as an argument and returns a truthy or falsy value. The matcher can also be an array of any or all of these. Useful for allowing a very flexible user-defined config to define things like file paths.

Note: This module has Bash-parity, please be aware that Windows-style backslashes are not supported as separators. See https://github.com/micromatch/micromatch#backslashes for more information.

Usage

anymatch(matchers, testString, [returnIndex], [options])

matchers: (*Array|String|RegExp|Function*) String to be directly matched, string with glob patterns, regular expression test, function that takes the testString as an argument and returns a truthy value if it should be matched, or an array of any number and mix of these types.

testString: (*String|Array*) The string to test against the matchers. If passed as an array, the first element of the array will

be used as the `testString` for non-function matchers, while the entire array will be applied as the arguments for function matchers.

- **options**: (*Object* [optional]_) Any of the picomatch options.
  - **returnIndex**: (*Boolean [optional]*) If true, return the array index of the first matcher that that testString matched, or -1 if no match, instead of a boolean result.

```
const anymatch = require('anymatch');

const matchers = [ 'path/to/file.js',
        'path/anyjs/**/*.js', /
        foo.js$/, string =>
        string.includes('bar') &&
        string.length > 10 ] ;

anymatch(matchers, 'path/to/
        file.js'); // true
anymatch(matchers, 'path/anyjs/
        baz.js'); // true
anymatch(matchers, 'path/to/
        foo.js'); // true
anymatch(matchers, 'path/to/
        bar.js'); // true
anymatch(matchers, 'bar.js'); // false

// returnIndex = true
anymatch(matchers, 'foo.js',
        {returnIndex: true}); // 2
anymatch(matchers, 'path/anyjs/foo.js',
        {returnIndex: true}); // 1

// any picomatc
```



```
// using globs to match directories and
        their children
anymatch('node_modules',
        'node_modules'); // true
anymatch('node_modules', 'node_modules/
        somelib/index.js'); // false
anymatch('node_modules/**',
        'node_modules/somelib/
        index.js'); // true
anymatch('node_modules/**', '/absolute/
        path/to/node_modules/somelib/
        index.js'); // false
anymatch('**/node_modules/**', '/
        absolute/path/to/node_modules/
        somelib/index.js'); // true

const matcher = anymatch(matchers);
['foo.js',
        'bar.js'].filter(matcher);  //
        [ 'foo.js' ]
anymatch master* ❱
```

anymatch(matchers)

You can also pass in only your matcher(s) to get a curried function that has already been bound to the provided matching criteria. This can be used as an `Array#filter` callback.

```
var matcher = anymatch(matchers);

matcher('path/to/file.js'); // true
matcher('path/anyjs/baz.js', true); // 1
```