

# glob-parent

[NPM version](https://www.npmjs.com/package/glob-parent) [Downloads](https://www.npmjs.com/package/glob-parent) [Azure Pipelines Build Status](https://dev.azure.com/gulpjs/gulp/_build/latest?definitionId=2&branchName=master) [Travis Build Status](https://travis-ci.org/gulpjs/glob-parent) [AppVeyor Build Status](https://ci.appveyor.com/project/gulpjs/glob-parent) [Coveralls Status](https://coveralls.io/r/gulpjs/glob-parent) [Gitter chat](https://gitter.im/gulpjs/gulp)

Extract the non-magic parent path from a glob string.

## Usage

var globParent = require('glob-parent');

globParent('path/to/\*.js'); // 'path/to'

globParent('/root/path/to/\*.js'); // '/root/path/to'

globParent('/\*.js'); // '/'

globParent('\*.js'); // '.'

globParent('\*\*/\*.js'); // '.'

globParent('path/{to,from}'); // 'path'

globParent('path/!(to|from)'); // 'path'

globParent('path/?(to|from)'); // 'path'

globParent('path/+(to|from)'); // 'path'

globParent('path/\*(to|from)'); // 'path'

globParent('path/@(to|from)'); // 'path'

globParent('path/\*\*/\*'); // 'path'

// if provided a non-glob path, returns the nearest dir

globParent('path/foo/bar.js'); // 'path/foo'

globParent('path/foo/'); // 'path/foo'

globParent('path/foo'); // 'path' (see issue #3 for details)

## API

### globParent(maybeGlobString, [options])

Takes a string and returns the part of the path before the glob begins. Be aware of Escaping rules and Limitations below.

#### options

{

// Disables the automatic conversion of slashes for Windows

flipBackslashes: true

}

## Escaping

The following characters have special significance in glob patterns and must be escaped if you want them to be treated as regular path characters:

* ? (question mark) unless used as a path segment alone
* \* (asterisk)
* | (pipe)
* ( (opening parenthesis)
* ) (closing parenthesis)
* { (opening curly brace)
* } (closing curly brace)
* [ (opening bracket)
* ] (closing bracket)

**Example**

globParent('foo/[bar]/') // 'foo'

globParent('foo/\\[bar]/') // 'foo/[bar]'

## Limitations

### Braces & Brackets

This library attempts a quick and imperfect method of determining which path parts have glob magic without fully parsing/lexing the pattern. There are some advanced use cases that can trip it up, such as nested braces where the outer pair is escaped and the inner one contains a path separator. If you find yourself in the unlikely circumstance of being affected by this or need to ensure higher-fidelity glob handling in your library, it is recommended that you pre-process your input with [expand-braces](https://github.com/jonschlinkert/expand-braces) and/or [expand-brackets](https://github.com/jonschlinkert/expand-brackets).

### Windows

Backslashes are not valid path separators for globs. If a path with backslashes is provided anyway, for simple cases, glob-parent will replace the path separator for you and return the non-glob parent path (now with forward-slashes, which are still valid as Windows path separators).

This cannot be used in conjunction with escape characters.

// BAD

globParent('C:\\Program Files \\(x86\\)\\\*.ext') // 'C:/Program Files /(x86/)'

// GOOD

globParent('C:/Program Files\\(x86\\)/\*.ext') // 'C:/Program Files (x86)'

If you are using escape characters for a pattern without path parts (i.e. relative to cwd), prefix with ./ to avoid confusing glob-parent.

// BAD

globParent('foo \\[bar]') // 'foo '

globParent('foo \\[bar]\*') // 'foo '

// GOOD

globParent('./foo \\[bar]') // 'foo [bar]'

globParent('./foo \\[bar]\*') // '.'

## License

ISC