# **[node-glob学习](https://www.cnblogs.com/liulangmao/p/4552339.html)**

node的glob模块允许你使用 \*等符号, 来写一个glob规则,像在shell里一样,获取匹配对应规则的文件.

这个glob工具基于javascript.它使用了 minimatch 库来进行匹配

用法:

**首先下载glob包:**

npm install glob

**调用格式:**

[IMG_257](https://www.cnblogs.com/liulangmao/p/javascript:void(0);)

var glob = require("glob")

// options 是可选的

glob("\*\*/\*.js", options, function (er, files) {

// files 是匹配到的文件的数组.

// 如果 `nonull` 选项被设置为true, 而且没有找到任何文件,那么files就是glob规则本身,而不是空数组

// er是当寻找的过程中遇的错误

})

[IMG_258](https://www.cnblogs.com/liulangmao/p/javascript:void(0);)

"globs" 就是模型,比如当你在命令行里输入 ls \*.js,  又或者是你在 .gitignore 文件里写的 bulid/\* 这些.

在解析路径模型的时候, 大括号里用多个逗号隔开的内容会被展开, 里面的部分也可以包含"/" ,比如  a{/b/c, bcd}  会被展开成 a/b/c 和 abcd

**路径中的某一段可以使用下面的这些字符表示,他们各自都有很炫的作用:**

* 1. \* : 匹配该路径段中0个或多个任意字符:

//\*:匹配路径中某部分:0个或多个字符

glob("js/\*.js",function (er, files) {

console.log(files)

})

获取js目录下的所有js文件.(不包括以'.'开头的文件)

* 2. ? : 匹配该路径段中1个任意字符:

//?:匹配路径中某部分:1个字符

glob("js/?.js",function (er, files) {

console.log(files)

})

获取js目录下所有名字只有1个字的js.

* 3. [...] : 匹配该路径段中在指定范围内字符:  
  注意不能组合,只能是其中一个字符

//[]:匹配路径中某部分:指定的范围

glob("js/a[0-3].js",function (er, files) {

console.log(files)

})

获取js目录下a开头,第二个字符为0-3之间(包括0和3)的js(a03.js不能被匹配到)

* 4. \*(pattern|pattern|pattern) : 匹配括号中多个模型的0个或多个或任意个的组合  
  注意|前后不能有空格

//\*(pattern|pattern|pattern): 匹配路径中的某部分: 多个模型中的0个或多个.

//除了三个模型本身,如果是组合也可以,比如ab.js,但是仅仅包含某个模型是不行的,比如a4.js.

glob("js/\*(a|a1|b).js",function (er, files) {

console.log(files)

})

获取js目录下a.js,a1.js,b.js,或者a,a1,b这几个字符的组合的js,比如ab.js

* 5. !(pattern|pattern|pattern) : 匹配不包含任何模型  
  需要注意: !(pattern|pattern|pattern)  不等于 !(\*(pattern|pattern|pattern))

//!(pattern|pattern|pattern): 匹配路径中的某部分: 不包含任何模型.

//带有a或者b的,都排除.需要注意的是,它并非是\*(a|b)的取反

glob("js/!(a|b).js",function (er, files) {

console.log(files)

})

获取js目录下名字中不包含a,也不包含b的所有文件.

6. ?(pattern|pattern|pattern) : 匹配多个模型中的0个或任意1个.  
它和 4 的区别是,不可以组合.必须完全匹配

//?(pattern|pattern|pattern): 匹配路径中的某部分: 多个模型中的0个或1个.

//精确匹配模型,不可以组合.

glob("js/?(a|a2|b).js",function (er, files) {

console.log(files)

})

获取js目录下a.js,a2.js,b.js

* 7. +(pattern|pattern|pattern) : 匹配多个模型中的1个或多个.  
  它和 4 的区别是,必须有一个,为空不匹配

//+(pattern|pattern|pattern): 匹配路径中的某部分: 多个模型中的1个或多个.

//可以是任意一个模型,也可以是他们的组合,比如ab.js

glob("js/+(a|a1|b).js",function (er, files) {

console.log(files)

})

获取js目录下a.js,a1.js,b.js,或者a,a1,b这几个字符的组合的js,比如ab.js

* 8. @(pattern|pat\*|pat?erN) : 匹配多个模型中的任意1个.

//@(pattern|pattern|pattern): 匹配路径中的某部分: 多个模型中的1个.

//精确匹配模型,不可以组合.和?的区别就是不可以为空.必须要是其中的一个.

glob("js/@(a|a1|b).js",function (er, files) {

console.log(files)

})

和 6 的区别是不匹配为空的情况

* 9. \*\* : 和 1 一样,可以匹配任何内容,但\*\*不仅匹配路径中的某一段,而且可以匹配 'a/b/c' 这样带有'/'的内容,所以,它还可以匹配子文件夹下的文件.

//\*\*: 不是一个单独的路径中的某部分,而是可以带有'/',所以所有当前文件夹和子文件夹下都进行匹配

glob("\*\*/@(a|a1|b).js",function (er, files) {

console.log(files)

})

获取当前目录所有文件夹及子文件夹下的a.js,a1.js,b.js  
还有一种方式是设置 matchBase 属性为 true ,同样可以起到在当前路径下搜索所有子文件夹的效果:

//matchBase: 设置为true以后,在当前目录下所有的文件夹和子文件夹里寻找匹配的文件

glob("@(a|a1|b).js",{matchBase:true},function (er, files) {

console.log(files)

})

**没有获取到任何匹配文件:**

当glob没有获取到任何匹配的文件是,并不会像shell里那样返回模型本身,files参数返回的是一个空数组,如果需要让files返回的是模型本身,需要设置 nonull 属性为 true

//nonull: 设置为true以后,如果没有找到匹配的文件,不返回空字符串,而是返回原始glob语句

glob("@(c|d|e).js",{nonull:true},function (er, files) {

console.log(files)

})

**同步获取匹配文件列表:**

前面讲到的都是异步的方法,传入一个回调,当获取到匹配的文件的时候执行回调.如果需要同步的获取文件列表,可以这样做:

var files = glob.sync(pattern, [options])

**Glob类:**

通过实例化一个glob.Glob类,可以获得一个glob对象:

var Glob = require("glob").Globvar mg = new Glob(pattern, options, cb)

实例化的时候传入的参数和glob(pattern,options,cb)是一样的.

它能够得到一个返回值,这个返回值是一个EventEmitter.

如果在选项中设置 sync 属性为 true, 表示同步获取.不可以传入cb回调. 要获取匹配结果,可以通过 g.found 来获取:

var globInstance = new glob.Glob("@(a|a1|b).js",{nonull:true,matchBase:true,sync:true});

console.log(globInstance.found);

**事件:**

* end :  end事件会在文件匹配结束,找出所有匹配结果的时候触发,它接受的参数就是找到的文件的数组
* match :  match事件会在每次匹配到一个文件的时候触发,它接受的参数就是匹配到的文件
* error :  error事件会在匹配遇到错误的时候触发.接受的参数就是错误信息
* abort :  当实例调用了.abort()方法时,abort事件被触发

**方法:**

* pause 暂停匹配搜索
* resume 继续匹配搜索
* abort 永远停止匹配搜索,不能继续

[IMG_259](https://www.cnblogs.com/liulangmao/p/javascript:void(0);)

var globInstance = new glob.Glob("js/@(a|a1|b).js",{nonull:true});

globInstance.on('match',function(file){

console.log(file)

});

globInstance.on('end',function(files){

console.log(files)

});

globInstance.on('abort',function(){

console.log('abort')

});

globInstance.pause();

globInstance.resume();

globInstance.abort();

[IMG_260](https://www.cnblogs.com/liulangmao/p/javascript:void(0);)

**属性:**

* minimatch glob所使用的minimatch对象.
* options 传递给函数的options选项.
* aborted 调用过abort()函数后它的值就是true.
* cache
* statCache
* symlinks
* realpathCache

**options选项:**

options用于配置模型匹配时候的匹配方式. 所有可以被传入到minimatch里的参数也都可以被传入到glob,另外node-glob还自己添加了一些配置项.

所有的选项如果没有特殊说明,默认值都是false

所有的选项也都适用于Glob类.

* cwd
* root
* dot
* nomount
* mark
* nosort
* stat
* silent
* strict
* cache
* statCache
* symlinks
* sync
* nounique
* nonull
* debug
* nobrace
* noglobstar
* noext
* nocase
* matchBase
* nonull
* nodir
* ignore
* follow
* realpath
* nonegate
* nocomment

原谅我这么多选项实在不想一个一个翻译过来了.......................毕竟大多数也用不到,如果有特殊需求,再查看文档吧............

参考原文: [https://github.com/isaacs/node-glob](https://github.com/isaacs/node-glob" \t "https://www.cnblogs.com/liulangmao/p/_blank)