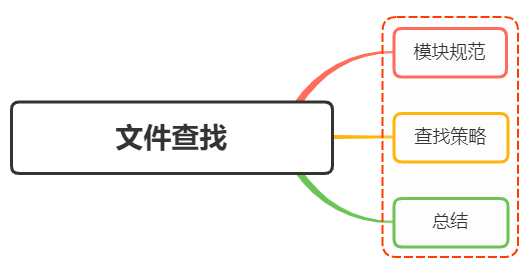
# 面试官：说说 Node 文件查找的优先级以及 Require 方法的文件查找策略?



## 一、模块规范

NodeJS对CommonJS进行了支持和实现，让我们在开发node的过程中可以方便的进行模块化开发：

* 在Node中每一个js文件都是一个单独的模块
* 模块中包括CommonJS规范的核心变量：exports、module.exports、require
* 通过上述变量进行模块化开发

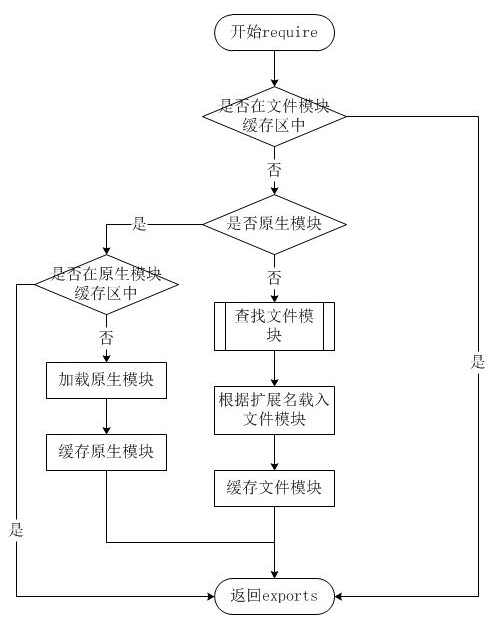
而模块化的核心是导出与导入，在Node中通过exports与module.exports负责对模块中的内容进行导出，通过require函数导入其他模块（自定义模块、系统模块、第三方库模块）中的内容

## 二、查找策略

require方法接收一下几种参数的传递：

* 原生模块：http、fs、path等
* 相对路径的文件模块：./mod或../mod
* 绝对路径的文件模块：/pathtomodule/mod
* 目录作为模块：./dirname
* 非原生模块的文件模块：mod

require参数较为简单，但是内部的加载却是十分复杂的，其加载优先级也各自不同，如下图：



从上图可以看见，文件模块存在缓存区，寻找模块路径的时候都会优先从缓存中加载已经存在的模块

### 原生模块

而像原生模块这些，通过require方法在解析文件名之后，优先检查模块是否在原生模块列表中，如果在则从原生模块中加载

### 绝对路径、相对路径

如果require绝对路径的文件，则直接查找对应的路径，速度最快

相对路径的模块则相对于当前调用require的文件去查找

如果按确切的文件名没有找到模块，则 NodeJs 会尝试带上 .js、.json或 .node拓展名再加载

### 目录作为模块

默认情况是根据根目录中package.json文件的main来指定目录模块，如：

{ "name" : "some-library",  
 "main" : "main.js" }

如果这是在./some-library node\_modules目录中，则 require('./some-library') 会试图加载 ./some-library/main.js

如果目录里没有 package.json文件，或者 main入口不存在或无法解析，则会试图加载目录下的 index.js 或 index.node 文件

### 非原生模块

在每个文件中都存在module.paths，表示模块的搜索路径，require就是根据其来寻找文件

在window下输出如下：

[ 'c:\\nodejs\\node\_modules',  
'c:\\node\_modules' ]

可以看出module path的生成规则为：从当前文件目录开始查找node\_modules目录；然后依次进入父目录，查找父目录下的node\_modules目录，依次迭代，直到根目录下的node\_modules目录

当都找不到的时候，则会从系统NODE\_PATH环境变量查找

#### 举个例子：

如果在/home/ry/projects/foo.js文件里调用了 require('bar.js')，则 Node.js 会按以下顺序查找：

* /home/ry/projects/node\_modules/bar.js
* /home/ry/node\_modules/bar.js
* /home/node\_modules/bar.js
* /node\_modules/bar.js

这使得程序本地化它们的依赖，避免它们产生冲突

## 三、总结

通过上面模块的文件查找策略之后，总结下文件查找的优先级：

* 缓存的模块优先级最高
* 如果是内置模块，则直接返回，优先级仅次缓存的模块
* 如果是绝对路径 / 开头，则从根目录找
* 如果是相对路径 ./开头，则从当前require文件相对位置找
* 如果文件没有携带后缀，先从js、json、node按顺序查找
* 如果是目录，则根据 package.json的main属性值决定目录下入口文件，默认情况为 index.js
* 如果文件为第三方模块，则会引入 node\_modules 文件，如果不在当前仓库文件中，则自动从上级递归查找，直到根目录

## 参考文献

* http://nodejs.cn/api/modules.html#modules\_file\_modules
* https://blog.csdn.net/qq\_36801250/article/details/106352686
* https://www.cnblogs.com/samve/p/10805908.html