

require机制主要部分实行流程

require的文件加载和依赖管理确实非常好用,下面简单分析其机制原理



代码实现分析

'use strict'

//自己实现require主要的部分

function \$require(id) {

const fs = require('fs');

const path = require('path');

//缓存

\$require.**cache**=\$require.**cache**|| {}

//step1.先找到文件(加载文件的完整路径)

const filename = path.join(__dirname, id);

if(\$require.**cache**[filename]){

return \$require.**cache**[filename].exports;

}

//step2.读取文件脚本内容

let code=fs.readFileSync(filename, 'utf8');

//step3.执行代码,所要执行的代码, 需要营造一个私人的空间

let module= {**id**:filename,**exports**:{}};

let exports=module.**exports**; *// exports指向module.exports*

const dirname=path.dirname(filename);

code='(function(\$require, module, exports, __dirname, __filename){ '+code+' })(\$require, module, exports, dirname, filename)';

eval(code);

//step4.缓存

\$require.**cache**[filename] =module;

//step5.返回值

return module.**exports**;

}

require扩展名

require加载文件时可以省略扩展名

比如: require('./module');

加载优先级:

- 1.加载的目录中存在module.js文件, 则最优先加载js文件
- 2.加载的目录中存在module.json文件, 如果不存在js文件, 此时优先加载json文件解析
- 3.加载的目录中存在module.node文件,如果js、json都不存在, 则此时优先加载预编译好c++模块
- 4.如果上述的文件都不存在, module是文件夹, 则
 - a)优先加载该目录下package.json中main指向的文件 (一般是./module/default.js)
 - b)不存在package配置文件, 则就会直接加载index.js文件

require加载文件规则

如果参数字符串不以“./”或者“/”开头，则表示加载的是一个默认提供的核心模块（位于Node的系统的安装目录）

`require('fs');` => 加载核心模块中的文件系统模块

或者从当前目录向上搜索node_modules目录中的文件

`require('my_module');` => 各级node_modules文件夹中搜索my_module.js文件。

作者：沈林生

链接：<https://www.jianshu.com/p/c4503e3f2829>

来源：简书

著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。