← Node.js Stream Node.js 函数 →

Node.js模块系统

为了让Node.js的文件可以相互调用，Node.js提供了一个简单的模块系统。

模块是Node.js 应用程序的基本组成部分，文件和模块是一一对应的。换言之，一个 Node.js 文件就是一个模块，这个文件可能是JavaScript 代码、JSON 或者编译过的 C/C++ 扩展。

创建模块

在 Node.js 中，创建一个模块非常简单，如下我们创建一个 'main.js' 文件，代码如下:

```
var hello = require('./hello');
hello.world();
```

以上实例中，代码 require('./hello') 引入了当前目录下的hello.js文件（./ 为当前目录，node.js默认后缀为js）。

Node.js 提供了exports 和 require 两个对象，其中 exports 是模块公开的接口，require 用于从外部获取一个模块的接口，即所获取模块的 exports 对象。

接下来我们就来创建hello.js文件，代码如下：

```
exports.world = function() {
  console.log('Hello World');
}
```

在以上示例中，hello.js 通过 exports 对象把 world 作为模块的访问接口，在 main.js 中通过 require('./hello') 加载这个模块，然后就可以直接访 问 hello.js 中 exports 对象的成员函数了。

有时候我们只是想把一个对象封装到模块中，格式如下：

```
module.exports = function() {
  // ...
}
```

例如:

```
//hello.js
function Hello() {
        var name;
        this.setName = function(thyName) {
                name = thyName;
```

```
        };
        this.sayHello = function() {
                console.log('Hello ' + name);
        };
};
module.exports = Hello;
```

这样就可以直接获得这个对象了：

```
//main.js
var Hello = require('./hello');
hello = new Hello();
hello.setName('BYVoid');
hello.sayHello();
```

模块接口的唯一变化是使用 module.exports = Hello 代替了exports.world = function() {}。 在外部引用该模块时，其接口对象就是要输出的 Hello 对象本身，而不是原先的 exports。

服务端的模块放在哪里

也许你已经注意到，我们已经在代码中使用了模块了。像这样：

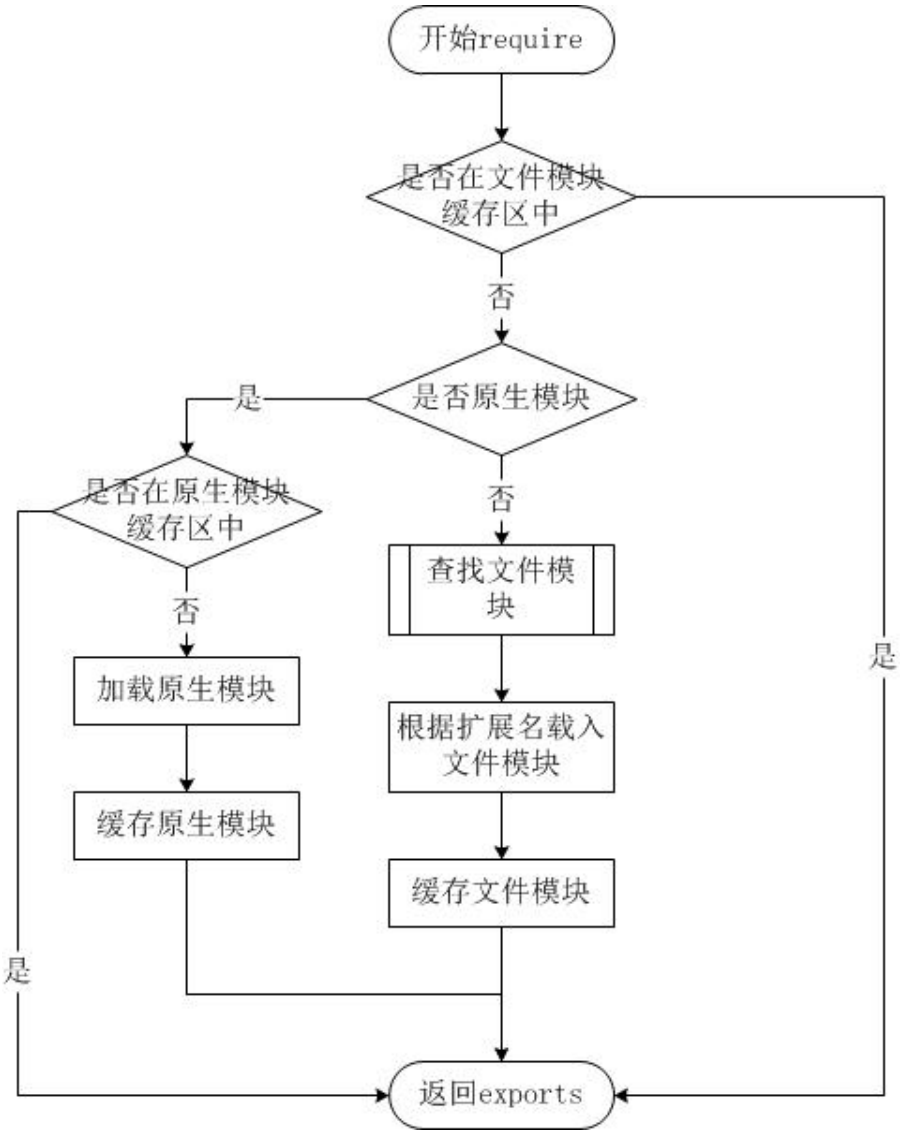
```
var http = require("http");

...

http.createServer(...);
```

Node.js中自带了一个叫做"http"的模块，我们在我们的代码中请求它并把返回值赋给一个本地变量。

这把我们的本地变量变成了一个拥有所有 http 模块所提供的公共方法的对象。

Node.js 的 require方法中的文件查找策略如下：

由于Node.js中存在4类模块（原生模块和3种文件模块），尽管require方法极其简单，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。如下图所示：



从文件模块缓存中加载

尽管原生模块与文件模块的优先级不同，但是都不会优先于从文件模块的缓存中加载已经存在的模块。

从原生模块加载

原生模块的优先级仅次于文件模块缓存的优先级。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。以http模块为例，尽管在目录下存在一个http/http.js/http.node/http.json文件，require("http")都不会从这些文件中加载，而是从原生模块中加载。

原生模块也有一个缓存区，同样也是优先从缓存区加载。如果缓存区没有被加载过，则调用原生模块的加载方式进行加载和执行。

从文件加载

当文件模块缓存中不存在，而且不是原生模块的时候，Node.js会解析require方法传入的参数，并从文件系统中加载实际的文件，加载过程中的包装和编译细节在前一节中已经介绍过，这里我们将详细描述查找文件模块的过程，其中，也有一些细节值得知晓。

require方法接受以下几种参数的传递：

- http、fs、path等，原生模块。
- ./mod或../mod，相对路径的文件模块。
- /pathtomodule/mod，绝对路径的文件模块。

mod，非原生模块的文件模块。

← Node.js Stream

Node.js 函数 →