CommonJs和 Nodejs模块、自定义模

块

**目录**

[一、什么是CommonJs？ 1](#_bookmark0)

[二、Nodejs 中的模块化 1](#_bookmark1)

[三、npm init 生成package.json 3](#_bookmark2)

**一、什么是 CommonJs？**

JavaScript 是一个强大面向对象语言，它有很多快速高效的解释器。然而， JavaScript 标准定义的 API 是为了构建基于浏览器的应用程序。并没有制定一个用于更广泛的应用程序的标准库。CommonJS 规范的提出,主要是为了弥补当前 JavaScript 没有标准库的缺陷。它的终极目标就是：提供一个类似 Python，Ruby 和 Java 语言的标准库,而不只是让 JavaScript 停留在小脚本程序的阶段。用 CommonJS API 编写出的应用，不仅可以利用 JavaScript 开发客户端应用，**而且还可以编写以下应用**。

* 服务器端 JavaScript 应用程序。（nodejs）
* 命令行工具。
* 桌面图形界面应用程序。

**CommonJS 就是模块化的标准，nodejs 就是 CommonJS（模块化）的实现。**

# 二、Nodejs 中的模块化

Node 应用由模块组成，采用 CommonJS 模块规范。

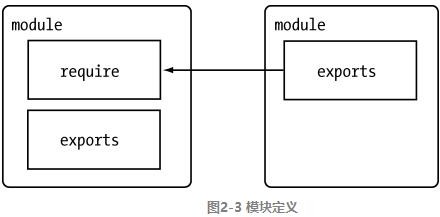
## 在Node 中，模块分为两类:

一类是 Node 提供的模块,称为**核心模块**；另一类是用户编写的模块，称为**文件模块**。

* + - 核心模块部分在 Node 源代码的编译过程中，编译进了二进制执行文件。在 Node 进程启动时，部分核心模块就被直接加载进内存中，所以这部分核心模块引入时，文件定位和编译执行这两个步骤可以省略掉，并且在路径分析中优先判断，所以它的加载速度是最快的。 **如：HTTP 模块 、URL 模块、Fs 模块都是 nodejs 内置的核心模块，可以直接引入使用。**
    - 文件模块则是在运行时动态加载，需要完整的路径分析、文件定位、编译执行过程、 速度相比核心模块稍微慢一些，但是用的非常多。**这些模块需要我们自己定义。接下来我们看一下 nodejs 中的自定义模块。**

## CommonJS（Nodejs）中自定义模块的规定：

1. 我们可以把公共的功能**抽离成为一个单独的 js 文件作为**一个模块，默认情况下面这个模块里面的方法或者属性，外面是没法访问的。如果要让外部可以访问模块里面的方法或者属性，就必须在模块里面通过 exports 或者 module.exports 暴露属性或者方法。
2. 在需要使用这些模块的文件中，通过 require 的方式引入这个模块。这个时候就可以使用模块里面暴露的属性和方法。



* 1. **定义使用模块：**

*// 定义一个tools.js的模块*

*//模块定义*

**var *tools*** = {

sayHello: **function**() { **return 'hello NodeJS'**;

},

add: **function**(x, y) { **return** x + y;

}

};

*// 模块接口的暴露*

*// module.exports = tools;* ***exports***.**sayHello** = ***tools***.sayHello; ***exports***.**add** = ***tools***.add;

**var *http*** = *require*(**'http'**);

*// 引入自定义的tools.js模块*

**var tools**= *require*(**'./tools'**)**;**

**tools.sayHello(); //使用模块**

# 三、npm init 生成 package.json

npm init --yes