# 模块

## cmd终端的使用

输入node -v 并回车，查看node版本。按ctrl+c即可退出nodejs

输入cls即可清屏。退出终端（命令提示行窗口） exit

切换盘符： d:

查看所有目录：dir

进入根目录 : cd\

返回上一级目录: cd..两个点

进入目录: cd 目录名

当前目录： cd . 一个点

创建一个文件夹：md hello ； 删除一个文件夹： rd hello ;

## 为什么使用模块？

在计算机程序的开发过程中，随着程序代码越写越多，在一个文件里代码就会越来越长，越来越不容易维护。

为了编写可维护的代码，我们把很多函数分组，分别放到不同的文件里，这样，每个文件包含的代码就相对较少，很多编程语言都采用这种组织代码的方式。

使用模块最大的好处是大大提高了代码的可维护性。当一个模块编写完毕，就可以被其他地方引用。我们在编写程序的时候，也经常引用其他模块，包括Node内置的模块和来自第三方的模块。

## Node.js基于什么规范？这个规范有什么特点？

node.js模块基于CommonJS规范

在 Node 环境中，一个.js文件就称之为一个模块（module），有自己的作用域，在一个文件里面定义的变量、函数、类，都是私有的，对其他文件不可见，而 Node 应用由模块组成，采用 CommonJS 模块规范（CommonJS 定义的模块分为: 模块标识(module)、模块定义(exports) 、模块引用(require)）

CommonJS规范规定，每个模块内部，module变量代表当前模块。这个变量是一个对象，它的exports属性（即module.exports）是对外的接口。加载某个模块，其实是加载该模块的module.exports属性（注意模块的路径描述）

CommonJS 模块特点：  
1、所有代码都运行在模块作用域，不会污染全局作用域  
2、模块可以多次加载，但是只会在第一次加载时运行一次，然后运行结果就被缓存了，以后再加载，就直接读取缓存结果。要想让模块再次运行，必须清除缓存  
3、模块加载的顺序，按照其在代码中出现的顺序

浏览器不兼容CommonJS的根本原因，正是在于缺少四个Node.js环境的变量：module，exports，global，require

node模块分为内置的node.js模块，第三方的node.js模块，自定义的node.js模块

## 什么是module对象，它有哪些属性和方法

### module对象的属性

module对象，代表当前模块。它有以下属性：

module.id 模块的识别符，通常是带有绝对路径的模块文件名  
module.filename 模块的文件名，带有绝对路径  
module.loaded 返回一个布尔值，表示模块是否已经完成加载  
module.parent 返回一个对象，表示调用该模块的模块  
module.children 返回一个数组，表示该模块要用到的其他模块  
module.exports 表示模块对外输出的值

### require方法

require 方法指向当前模块的module.require命令。require()方法用来引入外部模块文件

使用require 方法引入模块后，该函数会返回一个对象，这个对象代表的是引入的对象。

关于路径加载规则：

require("./b.js") 表示从当前目录找

require("/b.js") 表示从根目录找

require("b.js") node在使用模块名字来引入模块时，它会首先在当前目录的node\_modules中寻找是否含有该模块；如果有则直接使用，如果没有则去上一级目录的node\_modules中寻找；如果有则直接使用，如果没有则再去上一级目录寻找，直到找到为止；直到找到磁盘的根目录，如果依然没有，则报错

### exports属性

module.exports属性表示当前模块对外输出的接口，其他文件加载该模块，实际上就是读取module.exports。

在Node中，一个模块的中的变量和函数在其他模块中无法访问，只能在当前模块里面使用。对外输出东西，必须加给exports，向外部暴露属性或方法

而为了方便，Node为每个模块提供一个exports变量，指向module.exports，正因为如此不能直接对exports变量赋值，因为这样会等于切断了exports与module.exports的联系。

既可以使用exports导出，也可以使用module.exports导出

console.log(exports ); *// {}*

console.log(module.exports ); *// {}*

console.log(module.exports === exports); *// true*

### 加载机制

CommonJS模块的加载机制是，输入的是被输出的值的拷贝，即一旦输出一个值，模块内部的变化就影响不到这个值，例如：

b.js

var num = 3;

function addNum() {

    num++;

}

module.exports = {

    exnum: num,

    exAddNum: addNum

};

a.js

var obj = require('./b.js');

console.log(obj);

console.log(obj.exnum);  *// 3*

obj.exAddNum();

console.log(obj.exnum); *// 3*

# NPM

## 什么是NPM

NPM（NodeJS Package Manager）其实是Node.js的包管理工具

作用：1，统一下载途径。2，自动下载依赖的东西

NPM随Node一起安装的，终端输入 npm -v 回车验证版本。

npm里面有很多命令，终端 输入 npm 直接回车可见

如：adduser ,login ,install ,uninstall

npm不仅可以帮助我们下载包，还可以提交自己的包

## NPM官网

NPM 网站： http://www.npmjs.com 全球最大的开源库生态系统

中文文档：<https://www.npmjs.cn/>

可以从npm官网找到我们所需要的东西：如 搜索 mysql

### 注册

我们可以通过官网自己注册一个npm的账号

### 登录

cmd登录：npm login

输入：用户名,密码,邮箱

回车即可登录

## NPM镜像

使用 nrm 切换镜像源

下载 nrm：npm install -g nrm

查看可切换的镜像源： nrm ls (\*表示正在使用的镜像源)

\* npm -------- https://registry.npmjs.org/

yarn ------- https://registry.yarnpkg.com/

cnpm ------- http://r.cnpmjs.org/

taobao ----- https://registry.npm.taobao.org/

nj --------- https://registry.nodejitsu.com/

npmMirror -- https://skimdb.npmjs.com/registry/

edunpm ----- http://registry.enpmjs.org/

切换淘宝镜像源：nrm use taobao

# 包

## 包和模块的区别

### 解释1

模块可以是任何一个文件或目录 只要能被node.js通过require加载即可  
包是一个文件或目录 必须有一个package.json文件来描述，里面有多个模块  
npm 上所有代码都是一个包 但是只有能通过require来加载的叫做模块  
一个a.js文件就可以是一个模块  
把a.js和用来描述它的package.json文件放在同一个文件夹下那么就可以称作一个包  
包含package.json文件的文件夹名字不要包含大写字母和中文  
模块 + package.json = 包

小结：一个module是不是package就看其是否拥有package.json文件，拥有了，它就也是一个package了。

包描述文件  
包描述文件用于表达非代码相关的信息，它是一个JSON格式的文件 – package.json，位于包的根目录下，是包的重要组成部分。

## 创建包

一个包，可以是为了实现某个小功能而开发的一个包；也可以是一个项目：如一个网站APP，一个博客，一个CMS系统等。

创建包之前必须先初始化包信息  
执行命令：npm init

初始化包的信息，该命令为你的项目创建一个 package.json 文件。我们下载的每个包里面也都会有个package.json文件。

package name： 包的名字，如 cq-sum-package

包的名字一定不能和NPM系统中现有的包重名！否则发布包时会失败。包的名称和版本就是你项目package.json里的name和version哦！还有一点要注意的是npm对包名的限制：不能有大写字母/空格/下划线!

version： 包的版本，默认：1.0.0  
description: 包的描述：如：这是个测试包，不要下哦  
entry point： 包的入口文件（主文件，入口地址），默认：index.js  
test command: 包的测试命令,可以暂时空着  
git repository: git的仓库地址，可以从npm官网直接点到git上去，可以暂时空着  
keywords: 关键词，就是别人怎么能搜到你，如 testaixue  
author: 作者名字  
license: 包的协议，就是你的包遵循什么样的开源协议，默认：ISC

回车，项目文件夹里面多了个 package.json，它就是在顶层目录的包描述文件，用来描述当前包的情况。

对于 entry point: (index.js) 参数，键入你所希望的名称，这是当前应用的入口文件；如果你希望采用默认的 index.js 文件名，只需按 “回车” 键即可。除 entry point: (index.js) 参数外，其他参数你可以直接按 “回车” 键接受默认设置即可。

2．设置入口文件

既然我们定义的入口文件是index.js，那么我们就要在顶层目录中创建一个index.js，在index.js里面输入以下代码：

exports.sum = function () {

    var total = 0;

    for (let i = 0; i < *arguments*.length; i++) {

        const element = *arguments*[i];

        total += element;

    }

    return total;

}

## 发布包

发布包命令：npm publish

发布后，全世界的人都可以下载你的包

更新自己的包，需要再次发布：

回到自己开发的包的所在目录，在index.js里面增加如下代码：

exports.division = function (a, b) {

    return a / b;

}

cd 到要发布的包所在文件夹 ， 然后执行：npm publish ，重新发布，但是会发布失败，因为不能发布相同的版本，只需要把版本号改下即可发布。新发布的版本会覆盖掉以前的版本。

## 安装包

执行命令：npm install cq-sum-package

npm install 可以写成： npm i

你会发现node\_modules文件夹里面多了个 cq-sum-package 文件夹

这个文件夹其实就是你的当初的整个项目文件夹，你得项目文件都在里面。

执行命令：npm install mysql

cd到项目文件夹（注意：你cd到哪个文件夹，npm就会把包安装到哪个文件夹）

会在项目文件夹里面创建 一个node\_modules文件夹，且把mysql数据库及其所依赖的文件都下载到node\_modules文件夹里面。

测试一下：

const mysql = require('mysql');  
console.log(mysql);

const math = require("math")

*// 可以看到提供了哪些方法给我们用，其实现原理可以到包的math.js文件里面去看*

console.log(math);

console.log( math.add(123,456) ); *//两数相加*

console.log( math.sum([123,456,444,666]) ); *// 求和*

### 全局安装

全局安装的方式：npm install -g <package-name>

当你试图安装如命令行工具，例如 grunt CLI的时候，使用全局安装

### 本项目内安装

本地安装的方式：npm install <package-name>

当你需要通过require('XXX')的方式引入包中的模块的时候，使用本地安装

### 依赖

本地安装的时候，将依赖包信息写入package.json中

在团队协作中，一个常见的情景是他人从github上clone你的项目，然后通过npm install自动安装必要的依赖包，（刚从github上clone下来是没有node\_modules的，需要安装）

那么根据什么信息自动安装依赖包呢？就是你的package.json中的dependencies和devDepencies。所以，在本地安装的同时，将依赖包的信息（要求的名称和版本）写入package.json中是很重要的！

npm install <package-name> 安装好后写入package.json的dependencies中（生产环境依赖）

npm install <package-name> --save-dev 安装好后写入package.json的devDepencies中（开发环境依赖）

## 使用包

在项目文件夹里面新建一个js，输入以下代码：

const cqsum = require('cq-sum-package');

console.log( cqsum.sum(111,222,333) );

## 更新包

cd到项目文件夹，输入 npm update cq-sum-package 即可更新包到最新版本

如果更新无效，可以npm uninstall cq-sum-package。然后 npm install cq-sum-package 重新安装。

## 卸载包

全局卸载包

npm uninstall -g <package>

局部卸载包

npm uninstall <package>

如：npm uninstall mysql

## 撤销发布包

需要登录状态

命令：npm unpublish [<@scope>/]<pkg>[@<version>]

注意：如果报权限方面的错，加上--force

npm --force unpublish cq-sum-package

例如：npm unpublish xxxx@1.0.0

根据规范，只有在发包的72小时内才允许撤销发布的包。超过了72小时了，请联系npm维护者，等他们查了没有任何依赖之后会帮你下架的