

## Windows下安装nvm以及node

[http://blog.csdn.net/qq\\_36423639/article/details/70230571](http://blog.csdn.net/qq_36423639/article/details/70230571)

## Mac本下安装nvm安装并维护多个node.js版本

### 1、项目地址

<https://github.com/creationix/nvm/blob/master/README.md>

### 2、配置加速镜像

```
export NVM_NODEJS_ORG_MIRROR=https://npm.taobao.org/mirrors/node
```

查看当前node的所有版本

```
nvm ls-remote
```

下载对应的版本

```
nvm install 8.9.4
```

指定要使用哪个node版本

```
nvm use 8.0.0
```

## Mac下安装nvm以及node

先说一下nvm,node,npm之间的区别吧。

1.nvm的官方叫法：nodejs版本管理工具。

<1>nvm相当于是家长，一个家长可以管理多个孩子。 <2>也就是说:一个nvm可以管理很多node版本和npm版本。

## 2.nodejs

<1>在项目开发时的所需要的代码库

## 3.npm

<1>在安装的nodejs的时候，npm也会跟着一起安装，它是包管理工具。

<2>npm管理nodejs中的第三方插件

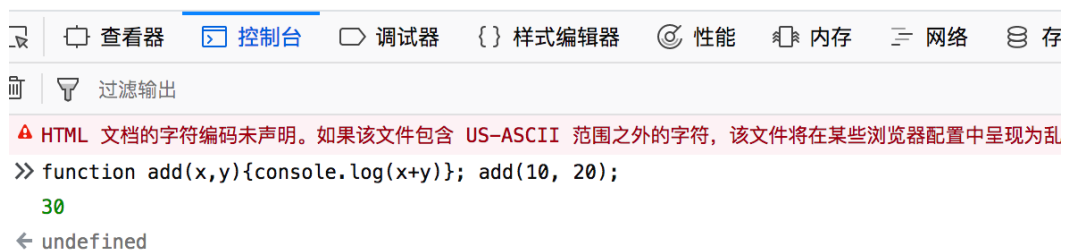
**【注】 nvm、nodejs、npm的关系： nvm是爸爸，管理nodejs和npm这一对双胞胎兄弟。 npm是哥哥， npm哥哥可以管理node弟弟的东西。**

=====

(1) OK，到这一步我们就将node配置好了，我们可以在终端中编写node的代码。

```
tianyufei:~ tianyufei$ node
[> function add(x,y){console.log(x+y)}; add(10,20);
30
undefined
> █
```

(2) 同时我们可以在控制台进行输入。



The screenshot shows a web browser's developer console with the 'Console' tab selected. The console displays the following output:

```
>> function add(x,y){console.log(x+y)}; add(10, 20);
30
← undefined
```

At the top of the console, there is a warning message: "HTML 文档的字符编码未声明。如果该文件包含 US-ASCII 范围之外的字符，该文件将在某些浏览器配置中呈现为乱码" (HTML document character encoding not declared. If the file contains characters outside the US-ASCII range, the file will be displayed as garbled in some browser configurations).

**【注】**说明我们浏览器的控制台和node都可以解析我们的JS文件。但是node和我们的控制台还是有区别的。

控制台输入命令

process

1、查看当前node中进程。

2、可以看到控制台区域没有这个权限。



【注】当前在node中没有浏览器端的DOM、BOM这样的对象。

## 下面我们要使用node来编译程序。

1、将当前目录打开使用 atom打开。

```
atom ./
```

2、我们编写js代码，在命令行中运行node程序

```
node index.js
```

【注】是否可以进行一遍写代码，一遍进行编译呢。我们需要安装两个组件。

3、我们需要安装nodemon，实时的侦测文件的变化

```
npm install nodemon -g
```

【注】我们会发现下载很慢，这个时候我们就需要去配置国内淘宝的镜像。

<https://npm.taobao.org/>

后续部分就可以通过文档中的步骤进行安装cnpm。那么安装成功以后，我们就可以通过cnpm安装组件了。cnpm的安装过程和npm命令相同。

```
cnpm install nodemon -g
```

4、安装成功以后，我们在命令行中执行代码，就可以实现实时监听的效果。

```
nodemon index.js
```

5、我们使用不同版本的node去运行index.js  
<1>指定具体的node版本去运行index.js

```
nvm run v8.0.0 index.js
```

<2>我们还可以去代码里去创建一个配置文件，这个文件叫做 .nvmrc

```
tianyuwei:hello tianyuwei$ node -v
v8.9.4
tianyuwei:hello tianyuwei$ nvm use ← 使用配置文件
Found '/Users/tianyuwei/Desktop/青岛 1705班 /2019node备课 /hello/.nvmrc' with version <8.0.0>

npm update check failed
Try running with sudo or get access
to the local update config store via
sudo chown -R $USER:$(id -gn $USER) /Users/tianyuwei/.config

Now using node v8.0.0 (npm v5.0.0)
tianyuwei:hello tianyuwei$ node -v ← 可以看到node版本已经被修改
v8.0.0
tianyuwei:hello tianyuwei$ node index.js ← 运行代码
hello world
40
tianyuwei:hello tianyuwei$
```

## 模块/包与CommonJS

如果大家想要查看commonJS的具体规范，可以访问commonJS的官方网站。  
<http://www.commonjs.org/>

### Node.js的模块

1. 内置的Node.js的模块
2. 第三方的Node.js模块(别人基于Nodejs开发的模块)
3. 自定义的Node.js模块

#### (1) 内置的模块

```
/*  
    a: 引入os模块, 内置模块不需要给出路径, 直接引入  
    关于os模块的具体内容, 可以查看node官方文档  
  
*/  
  
const os = require("os");  
  
//    b: 通过os模块显示主机名  
  
console.log(os.hostname());  
  
/*  
    c: 总结使用内置模块的流程  
        (1) 查看node官方文档  
        (2) 通过require引入内置模块  
        (3) 使用内置模块编写代码  
*/
```

(2) 第三方nodejs模块的使用，可以在下述这个网站上下载。

<https://www.npmjs.com/>

### 使用流程

1. 你有某一个特定的需求，你去搜索是否有这个需求对应的功能插件。
2. 可以使用nodejs的工具，将这个包安装到我们的项目中。
3. 在我们的项目中引入该插件，去查看该插件对应的文档，进行开发。

【注】使用自定义插件，我们可以在package.json中进行配置。

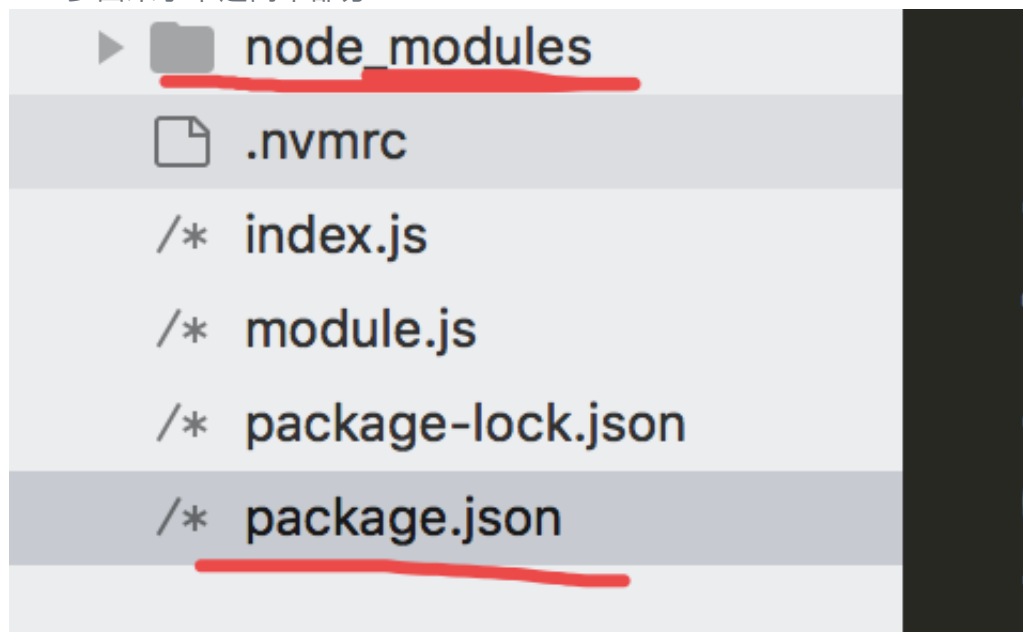
<1>npm init 初始化 创建package.json文件

```
npm init
```

<2>安装对应的插件(request下载数据模块 --save-dev是将路径配置保存在package.json文件中)

```
npm install request --save-dev
```

<3>多出来了下述两个部分



<4>可以在js文件中引入该模块编写代码

```

/*
    【注】require引入模块，会先去内置的模块中查找，
    如果没有会去node_modules中去进行查找。
*/
// 用来下载数据的模块
const request = require("request");

request({
  url: "http://api.douban.com/v2/movie/top250",
  json: true
}, (error, response, body) => {
  console.log(body);
})

```

### (3)自定义模块

<1>创建一个js文件(greeting)，我们可以将实现某个特定功能的函数，定义成一个模块。

```

const hello = () => {
  console.log("hello ~");
}

// 暴露接口被外部所使用，使用commonJS规范
module.exports.hello = hello;

```

<2>引入模块，注意路径前面加 ./

```

/*
    【注】这里引入自定义模块，需要注意引入路径。
    在这里就算在当前文件夹下，需要在前面加 ./。
*/
const hello = require("./src/greeting.js");
hello.hello();

```

<3>关于模块路径加载顺序，详解一下：

```
/*
    模块加载系统: commonjs 规范
    require("模块")
*/
require("./demo.js");

/*
    模块加载机制路径:
        绝对路径
        相对路径
*/
require("demo.js");
// 这种写法将会去加载node中的内置核心模块, 或者是node_modules中的模块。

/*
    1、首先按照加载的模块的文件名进行查找
    2、如果没有找到会在文件名后面添加.js进行查找
*/
```