参考：<http://www.cnblogs.com/imwtr/p/4350282.html>

[**Node.js起步 -- （1）**](http://www.cnblogs.com/imwtr/p/4350282.html)

**先来简单介绍nodeJS**

　　我们知道JavaScript是运行在浏览器中的，浏览器为它提供了一个上下文(context)，从而让JavaScript得以解析执行。

nodeJS其实可以这么理解，它是另外一种上下文，它允许在后端(脱离浏览器环境）解析执行JavaScript代码。

因为它使用了Google的V8虚拟机（Google的Chrome浏览器使用的上下文环境），所以代码可以同理被执行。

nodeJS除了可以在浏览器端显示结果之外，还可以直接在控制台输出。

**nodeJS的特点**

　　nodeJS主要有非阻塞、基于事件驱动、单进程这几个特点。NodeJS的作者说，他创造NodeJS的目的是为了实现高性能Web服务器，他首先看重的是事件机制和异步IO模型的优越性，而不是JS它更适用于轻量、可伸缩，适于实时数据交互应用。nodeJS内置了很多对象，比如http，fs等，我们可以用它搭建我们的HTTP服务器，在服务器上进行开发。

**安装**

　　对于我自己来说，很少进linux去干活，所以目前只有Windows端的安装方式。Linux的安装方式自己网上搜一搜吧..我贴一下别人的

[复制代码](javascript:void(0);)

先安装依赖包

　sudo apt-get install g++ curl libssl-dev apache2-utils

　　sudo apt-get install git-core

then

git clone git://github.com/joyent/node.git

　　cd node

　　./configure

　　make

　　sudo make install

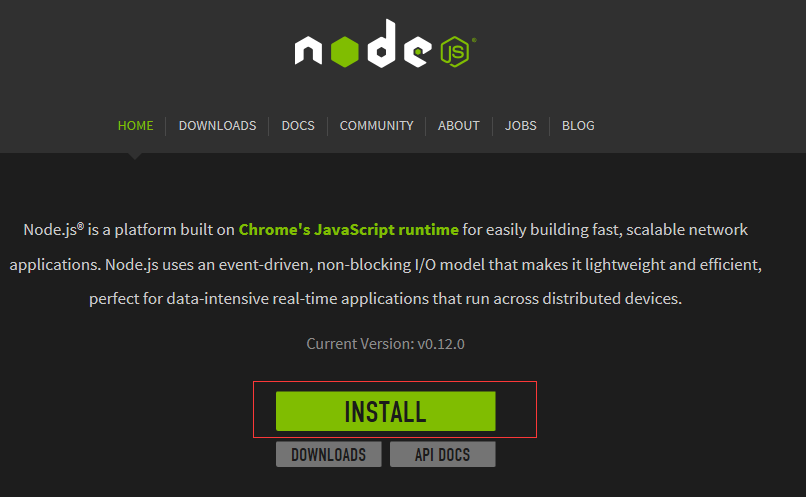
[复制代码](javascript:void(0);)

听说要注意Python版本的问题，建议2.6+的版本，低版本会出现Node安装错误，

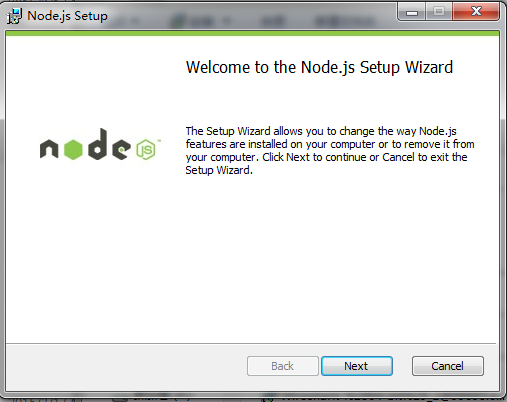
查询Python版本可在终端中输入：pyhton -v

**Windows下**

安装很简单，上[官网](https://nodejs.org/)，把最新版本的msi文件下载下来，直接安装~可以修改路径。

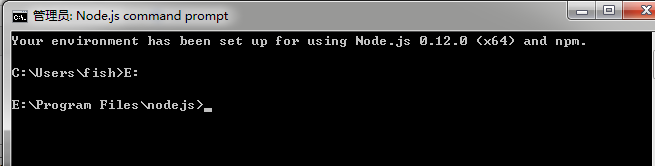
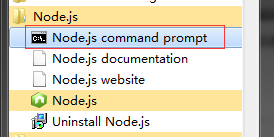


一路下去无阻碍



　　安装成功之后最新版本的默认已经帮你放进系统路径里边去了，不必管了。

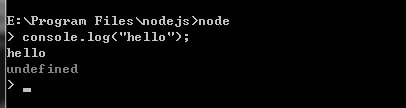
之后可以到开始菜单里边找到nodeJS打开其命令行进入，默认是进入C盘的，一般还是切换到安装路径所在盘，比如我的是E盘



**看nodeJS是否安装成功，查看其版本号  node -v**

　　然后简单地测试一下，先输入node命令，然后直接控制台输入console.log("hello");

因为console.log方法实际上是有返回值的，这里没有给他返回，所以第二个就是undefined



**再来测试一下，这次我们要看见  Hello World ！**

在安装路径下（比如我是E:\Program Files\nodejs）新建一个main.js 文件，输入

[复制代码](javascript:void(0);)

var http = require('http');

　　http.createServer(function (req, res) {

　　 res.writeHead(200, {'Content-Type': 'text/plain'});

　　 res.end('Hello World\n');

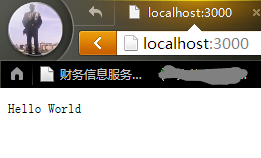
　　}).listen(3000, "127.0.0.1");

　　console.log('Server running at http://127.0.0.1:3000/');

[复制代码](javascript:void(0);)

**然后在里边输入 node main.js 执行**（一般node里边退出就是按一次或两次 Ctrl+C ) ，这时服务器就已经搭建好了~  
浏览器中输入其中的地址：http://localhost:3000

我们的Hello World 出来了~



**然后先来解释一下这小段代码。**

　　var http = require('http');上面说到了，http是nodeJS里边内置的对象模块，我们使用require这种方式把它引进来（对象模块一般就是这么引入的，我们也可以自己建立自己的模块，然后将它们引入，之后再提到）。

然后http现在这个对象模块有一个方法是，http.createServer(),这个方法创建一个服务之后再监听一个地址：http.createServer().listen(port,ip)

从而搭建了一个服务器。

　　createServer()里边有一个匿名函数，主要就是用来处理相关信息了。req是请求request，res是回复response。req一般用于请求阶段的解析处理等，比如常见的get中的地址栏url字符的处理。res一般用于收到请求后相应的操作，比如写响应头响应体渲染页面等等。

在这里，res.writeHead(200, {'Content-Type': 'text/plain'});就相当于写了个响应头， res.end('Hello World\n');相当于写了个响应体。

 　　nodeJS提供了很多内置对象方法，比如http,fs,EventEmitter,url等等

这些东西的使用建议参考[官方文档](https://nodejs.org/api/)

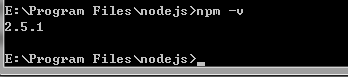
或者taobao建立的平台CNodejs ,[去里边逛逛吧](https://cnodejs.org/api)

还有这个[7天学会nodeJS](http://nqdeng.github.io/7-days-nodejs/)

**基本的HTTP服务器搭建就是这样了，下面介绍node强大的工具npm(node packages manager,包管理工具)。**

它主要用于管理我们的包，下载安装包或者卸载包等等。

旧版本的node需要自个去下载安装，不过新版本的已经集成了。所以直接在这里输入npm -v 查看它的版本吧~



npm安装方式一般说来有两种，全局或者本地

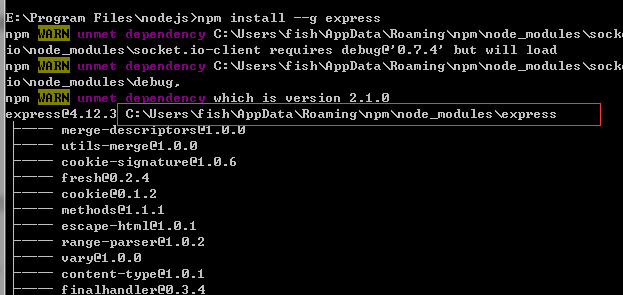
好那我们就先来体验下，装个express框架吧。

（express.js是nodejs的一个MVC开发框架，并且支持jade，ejs等多种模板)

我们可以使用两种方式安装：

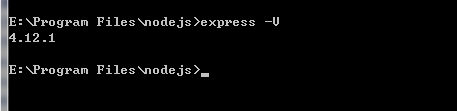
全局安装：npm install --g express  .(我发现--g或者-g皆可)这种方式会把包安装到全局路径里边，一般是C盘的npm/node\_modules/下边

哇哦....

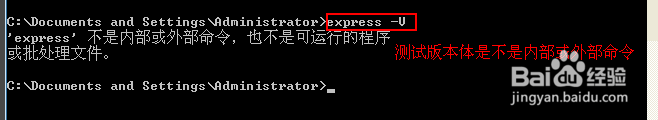


不要在意那些错误..

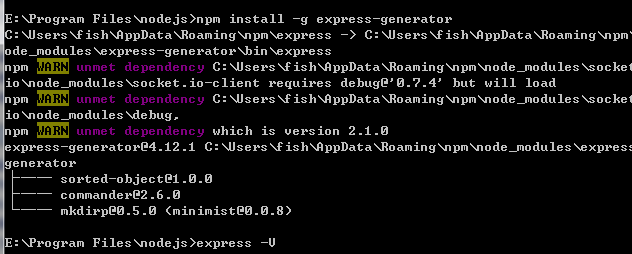
接下来用我想express -V(大写的)查看它的版本吧



有些朋友可能会遇到错误，这里贴个某[网友的](http://jingyan.baidu.com/article/922554468a3466851648f419.html)的~



解决方式如其所说：新版本将命令工具分家了，所以先另外再安装它npm install -g express-generator



我说过了，不要在意上面那些细节..O\_O

好了，现在我们来试试express ？

还是那个main.js 我们把它改成这样：

[复制代码](javascript:void(0);)

var express = require("express");

//var app = express.createServer(); // 在新版本已经不兼容了，所以直接用下面那个

var app = express(); // 或者直接 var app= require("express")();

var port = 3000;

var ip = "127.0.0.1";

app.listen(port,ip);// ip 这个参数可不写，默认本地

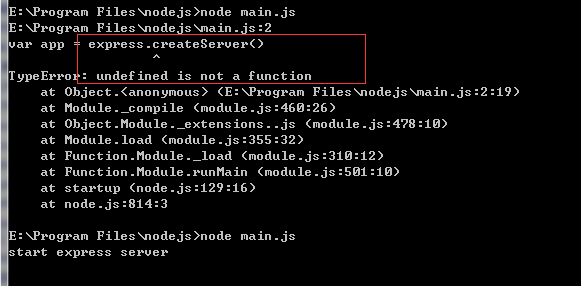
app.get('/',function(req,res){

res.send("Hello World\n");

});

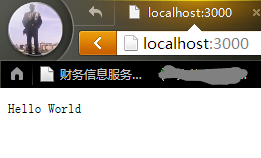
console.log("start express server\n");

[复制代码](javascript:void(0);)

如上所示，新版本可能会报错没那个方法  


express封装了http，所以可以直接使用它定义的方法了

**然后浏览器输入http://localhost:3000/  又是**



然后再来简单介绍上面的代码。其余都不必说了，**说一下app.get() 其实这就是一个路由处理。**

好那我们来试试路由的处理~

[复制代码](javascript:void(0);)

var express = require("express");

//var app = express.createServer(); // 在新版本已经不兼容了，所以直接用下面那个

var app = express(); // 或者直接 var app= require("express")();

var port = 3000;

var ip = "127.0.0.1";

app.listen(port,ip);// ip 这个参数可不写，默认本地

app.get('/',function(req,res){

res.send("Hello World\n");

});

app.get('/one',function(req,res){ // 或者“/one/

res.send("one\n");

});

app.get('/user',function(req,res){ //或者 "/user/"

res.send("user\n");

});

app.get('/user/:id',function(req,res){ //或者 "/user/"

res.send("user: "+req.params.id);

});

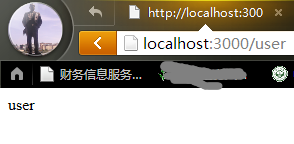
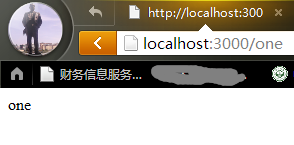
console.log("start express server\n");

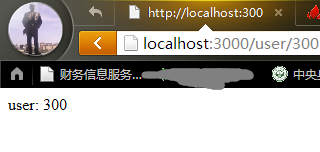
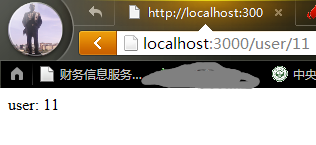
[复制代码](javascript:void(0);)

**如上所示，get的第一个参数就是路径地址，第二个参数就是回调函数。**

你使用不同的地址路径，就可以进行不同的操作了。

比如浏览器地址的不同可路由：



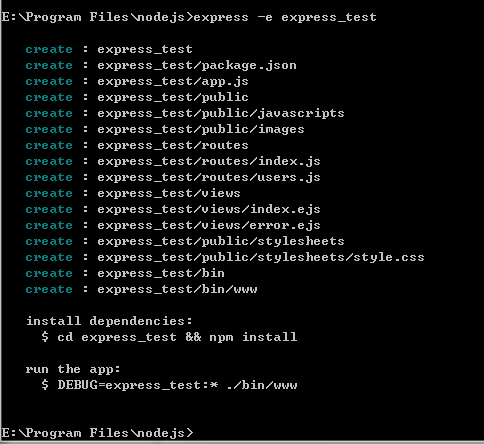


**上头那个user/11中的11就被当做user的参数id值。我们可以通过req.params这个全局方法来获取相应的参数值 req.params.id**

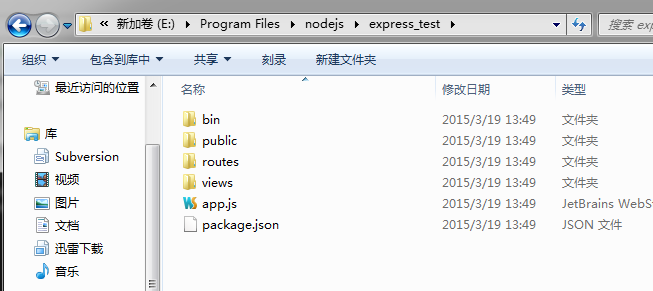
**对于express，其实有一种更普遍的方法，那就是直接用它来创建一个项目：**

比如，我要创建项目express\_test 就输入

**express -e express\_test**



好了，项目创建成功，看一下它的结构吧~



项目创建成功之后，生成四个文件夹，主文件app.js与配置信息文件packeage.json

**bin是项目的启动文件**，配置以什么方式启动项目，默认 npm start

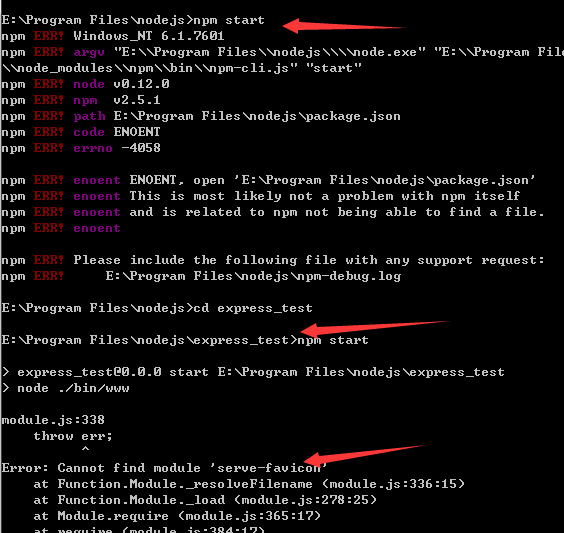
**public是项目的静态文件**，放置js css img等文件

**routes是项目的路由信息文件**,控制地址路由

**views是视图文件**，放置模板文件ejs或jade等（其实就相当于html形式文件啦~)

**express这样的MVC框架模式，是一个Web项目的基本构成**。

更细的部分先不考虑，我们先来简单地启动吧~



看到没有，第一步出错了，因为它不知道是什么项目，**所以要先进入你想要启动的项目。cd express\_test**

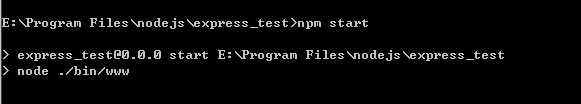
然后又出错了是怎么回事？

看到了吧，它找不到所需模块，**所以一般我们创建新项目的时候也要先给他安装相应的模块**。

**我们直接使用npm install 就行了**，它会自动检测package.json文件下载安装相应的模块

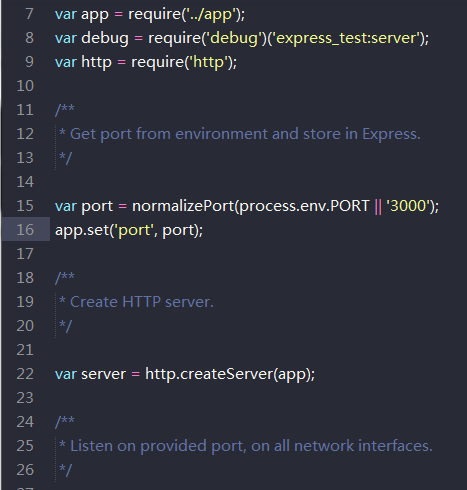


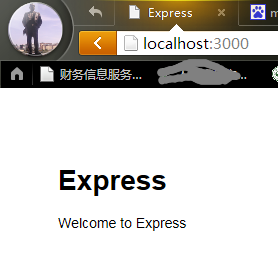
**好吧，那就启动吧~**



ok，启动成功，浏览器输入地址查看一下~

**默认端口是3000，可以到bin里边的www文件查看，或者自己修改吧~**





哇哦~成功了！简单的express项目就是这样子了..

express模块就先简单地介绍到这来。更多的express后面会继续讲一讲~

**最后随手谈一下上面提到的require相关的知识**

编写稍大一点的程序时一般都会将代码模块化。在NodeJS中，一般将代码合理拆分到不同的JS文件中，每一个文件就是一个模块，而文件路径就是模块名。

在编写每个模块时，都有require、exports、module三个预先定义好的变量可供使用。

**require函数用于在当前模块中加载和使用别的模块，传入一个模块名，返回一个模块导出对象。**

模块名可使用相对路径（以./开头），或者是绝对路径（以/或C:之类的盘符开头），模块名中的.js扩展名可以省略。

举个例子：

var express = require("express");

var func1 = require("./func1");// 是与当前main.js同路径下的func1.js模块

var func2 = require("../nodejs/func2");// func2.js则位于与main.js父文件同路径下的

**exports对象是当前模块的导出对象，用于导出模块公有方法和属性。**

别的模块通过require函数使用当前模块时得到的就是当前模块的exports对象。

比如上面提到的func1.js

我们可以这样写代码然后导出来：

[复制代码](javascript:void(0);)

exports.func1 = function(){

console.log("This is func1");

};

/\*或者这样

function func1(){

console.log("This is func1");

}

exports.func1 = func1;

\*/

[复制代码](javascript:void(0);)

　　module对象可以访问到当前模块的一些相关信息，但最多的用途是替换当前模块的导出对象。

例如模块导出对象默认是一个普通对象，如果想改成一个函数的话，可以使用以下方式。

module.exports = function () {

console.log('Hello World!');

};

这样一来，模块默认导出对象被替换为一个函数。我们在main.js里边就可以直接 var app = require("./func1")(); 操作了

还可以导出一个对象的形式：

[复制代码](javascript:void(0);)

//在func1.js里边这样

module.exports = function(name, age) {

this.name = name;

this.age = age;

this.about = function() {

console.log(this.name +' is '+ this.age +' years old');

};

};

//然后在main.js里边这样

var Func1 = require('./func1.js');

var r = new Func1('xiaoming', 12);

r.about(); // xiaoming is 12 years old