# node.js

# 一， 预备知识

## 1，web概念，B/S，C/S构架？

**什么是Web ?**

Web就是网页，是www（world wide web）的简称。

**Internet?**

因特网， 它泛指互联网，专指因特网。

**什么是互联网？**

任何设备只要能通信，就是互联网。

**什么是因(英)特网？**

我们平常所说的互联网专指因特网。

**Web，互联网，因特网的区别？**

互联网包含因特网，因特网包含Web。

互联提供很多服务，Web只是这么多服务中的一个。

**互联网都提供了哪些服务？**

* 1. 远程登录服务(Telnet)
* 2. 文件传输服务(FTP)
* 3. 电子邮件服务(E-Mail)
* 4. 网络新闻服务(Usenet)
* 5. 名址服务(Finger、Whois、X.500、Netfind)
* 6. 文档查询索引服务(Archie、WAIS)
* 7. 信息浏览服务(Gopher、WWW)
* 8. 其它信息服务(Talk、IRC、MUD)

**理解Web?**

1，网页是谁提供的？ 服务器（server）

2，网页在哪里看？ 浏览器(browser)

**B/S架构：**

像上面的一样，网页由**S**erver提供，**B**rowser提供浏览器的功能。这就是B/S架构。

**C/S架构：**

S还是指**S**erver， 服务器， C是指**C**lient， 比如QQ， 杀毒软件，各种游戏客户端....

**B/S架构的有缺点：**

优点：不需要下载安装，更新更深方便。

缺点：所有业务处理都要在服务器上处理，对服务器造成了很大压力。

**C/S架构的有缺点：**

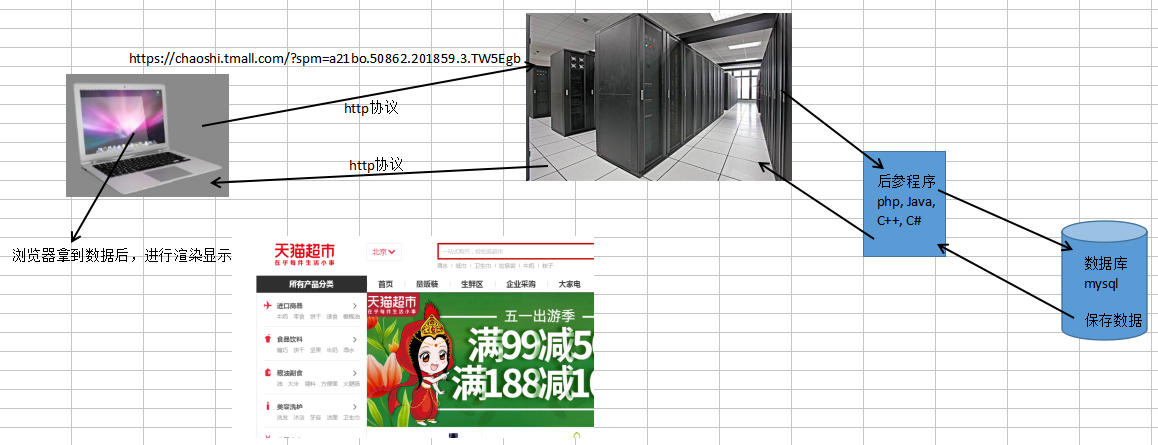
优点：使用非常方便，性能好一点。

缺点：需要下载，安装，更新维护不方便。

## 2，“上网”的本质？

服务器也是一台电脑，我们自己的电脑上面装了QQ，暴风影音，... 目的是为了方便我们使用，服务器上面不是装这些软件，人家装一些其它的提供服务的软件。服务器的系统和我们用的电脑的系统也不一样。

以上淘宝网为例，输入www.taobao.com，当按下回车，发生了什么故事：



对于上面的流程：

**服务器**：系统，提供Web服务的软件，PHP提供的程序服务，提供数据服务

linux apache php程序 mysql

**node.js:** window node.js node.js mysql或mongdb

## 3，新语法之严格模式

除了前面我们所学的正常模式年，ECMAScrpt5又添加一个严格模式，说的直白一点，就是让我们的JS程序在更加严格的条件上运行。

**设立严格模式的目的：**

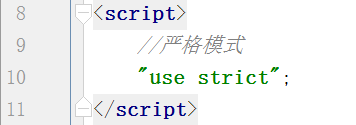
* - 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
* - 消除代码运行的一些不安全之处，保证代码运行的安全；
* - 提高编译器效率，增加运行速度；
* - 为未来新版本的Javascript做好铺垫。

"严格模式"体现了Javascript更合理、更安全、更严谨的发展方向，包括IE 10在内的主流浏览器，都已经支持它，许多大项目已经开始全面拥抱它。

另一方面，同样的代码，在"严格模式"中，可能会有不一样的运行结果；一些在"正常模式"下可以运行的语句，在"严格模式"下将不能运行。掌握这些内容，有助于更细致深入地理解Javascript，让你变成一个更好的程序员。

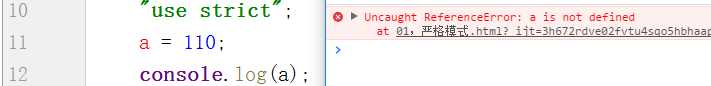
**如何使用严格模式：**

use strict;

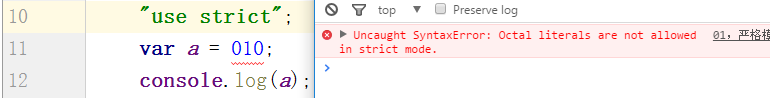


**在严格模式下，我们前面所学的语法都有哪些改变？**

1，变量必须先定义再使用，不能使用隐式的全局变量

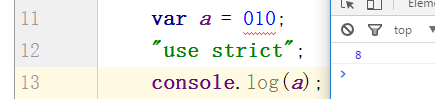


2，不能使用八进制

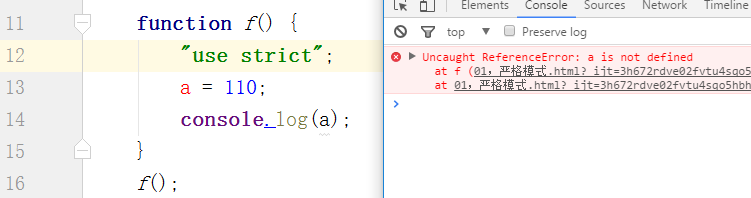


**注意点：**

1，如果你要使用严格模式，必须将use strict写在第一行。如果不写在第一行，那么就相当于没有起用严格模式：



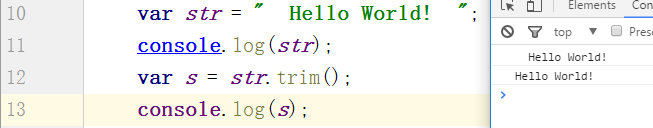
2，在一个函数中，也能起用严格模式：在函数中，也需要写在第一行。



参考文章：<http://www.ruanyifeng.com/blog/2013/01/javascript_strict_mode.html>

## 4，新语法之trim()

trim是ES5提供的一个新方法，用来去除前面和后面的空格：



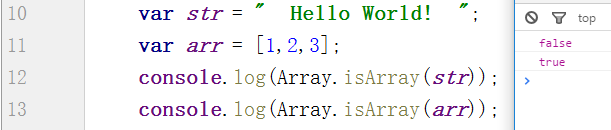
## 5，新语法之数组新增方法

**ES5开始就新增的一些方法**

* Array.isArray()
* forEach()
* map()
* filter()
* some()
* every()

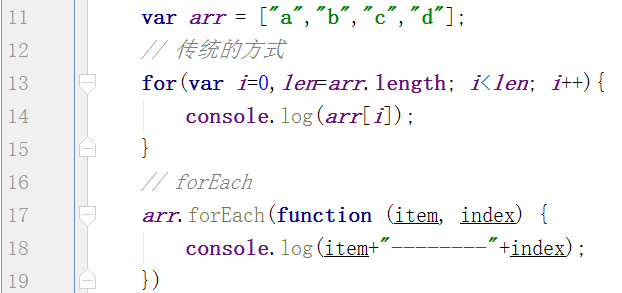
### （1）Array.isArray()

是Array构造器的静态方法，Array进行调用。



### （2）forEach()

**用来遍历数组的。**



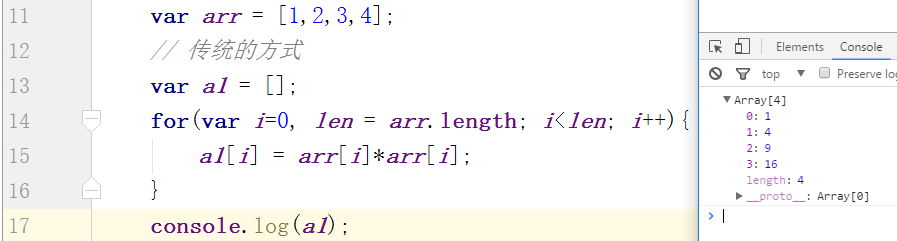
### （3）map()

地图，映射

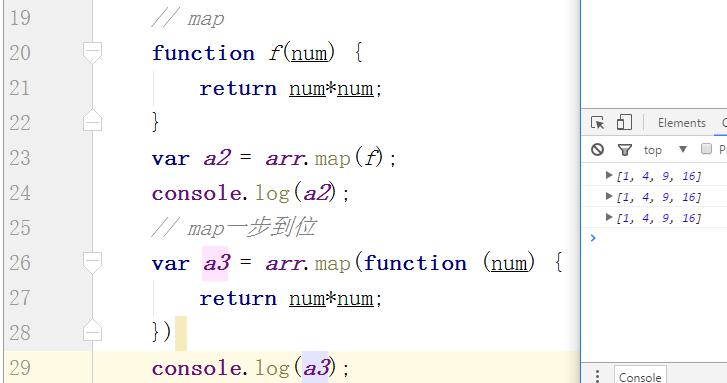
map方法返回一个由原数组中的每个元素调用一个指定的方法后的返回值组成一个新数组。

需求：对于一个数组arr=[1,2,3]，求里面每个元素的平方，放到一个新的数组里面

**传统方式实现：**



**map:**

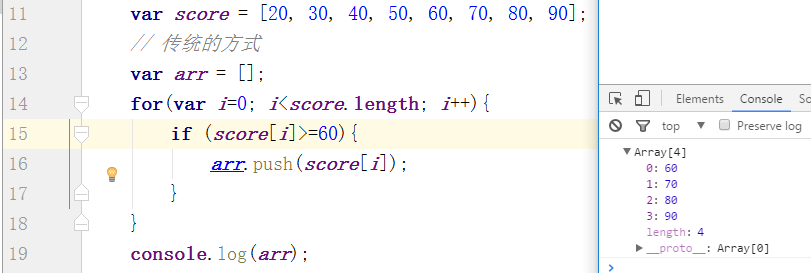


### filter()

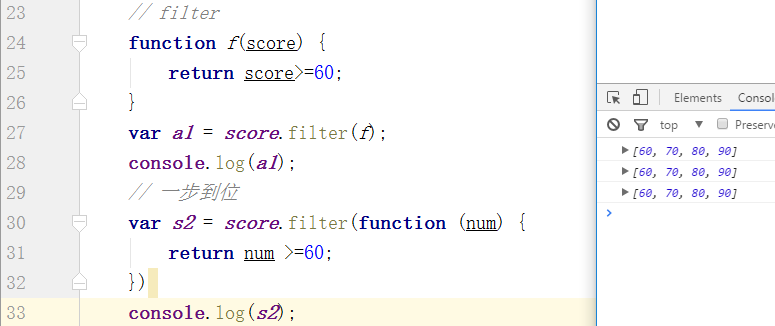
filter， 对于一个数组，运行某个规则，过滤掉某些元素，返回剩余的元素，剩余的元素组成一个新数组 。

需求：对于一个数组：var score = [30,50,60,70]; 把及格的分数找出来

**传统方式：**

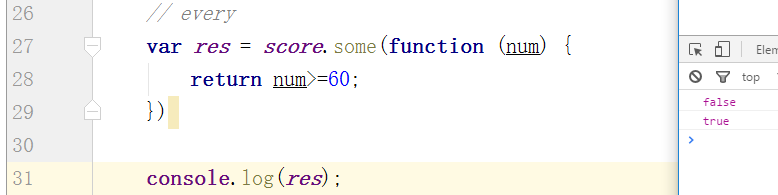


**filter:**



### （5）some

some是一些的意思 类似于 ||



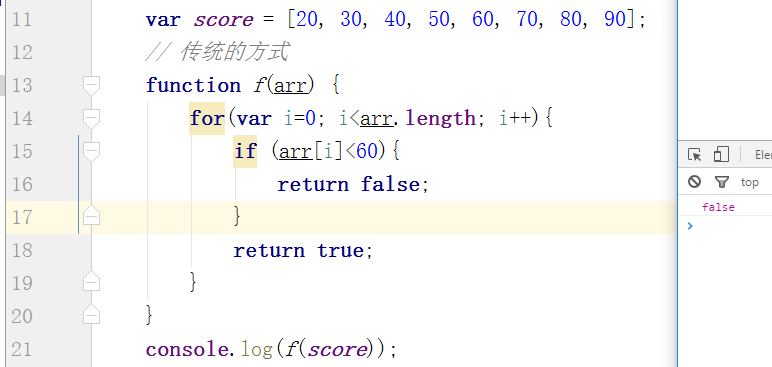
### （6）every

every是所以的意思

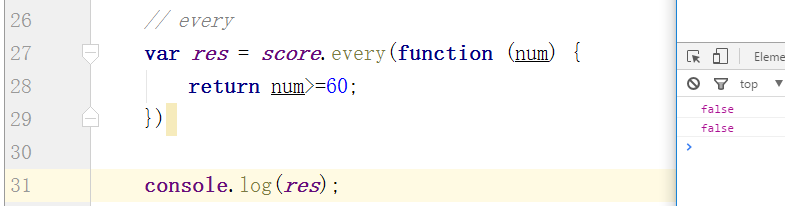
判断数组中每一个元素是否满足某个规则。 类似 &&

需求：看一个数组中的分数是否都及格了

传统方式：



every:



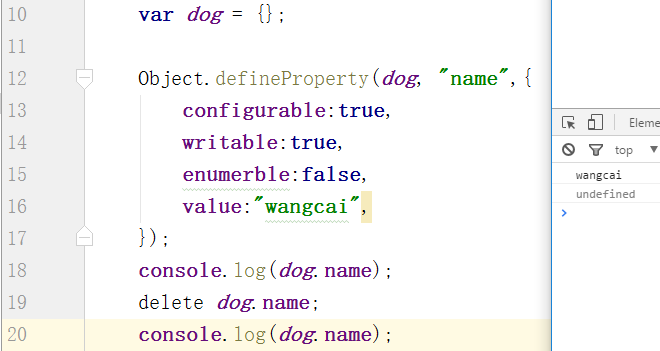
## 6，新语法之对象新增方法

从ES5开始，也新增了一个方法，如下：

* 继承相关方法：create()、getPrototypeOf()
* 属性相关方法：defineProperty()、defineProperties()、getOwnPropertyDescriptor()、getOwnPropertyNames()、keys()
* 防篡改方法：preventExtensions()、isExtensible()、seal()、isSealed()、freeze()、isFrozen()

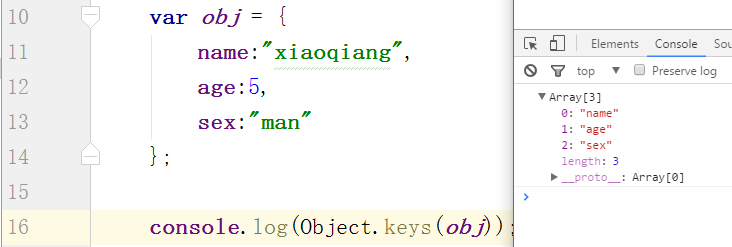
所有的这些方法都是Object的**静态**的方法，使用**Object.方法名**来调用。

### （1）defineProperty()



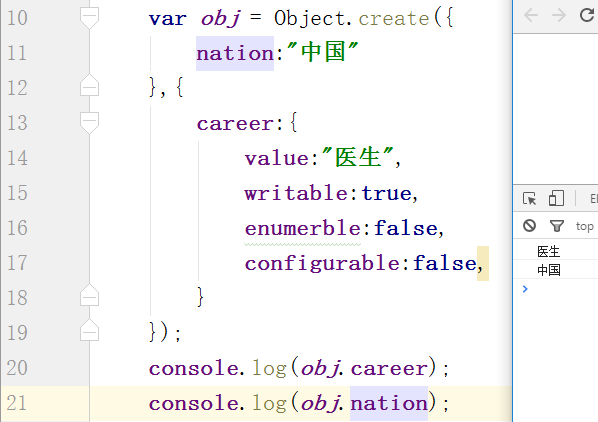
### keys()

获取对象所有的属性名称，形成一个数组



### （3）create()

用来实现继承的



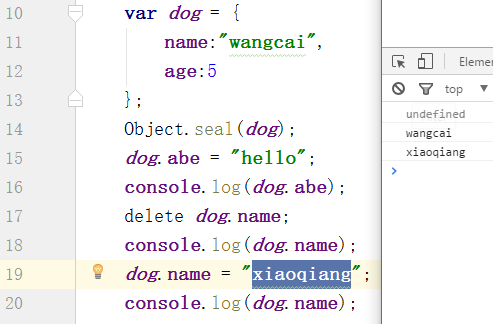
### freeze()

防篡改 不允许新增，不允许删除，不允许修改



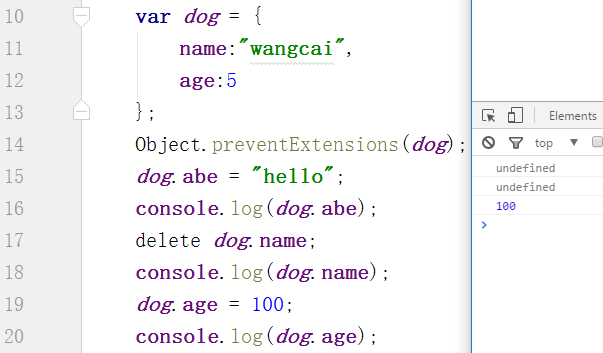
### （5）seal()

不允许新增，也不允许删除，但是允许修改



### preventExtensions()

不允许新增，允许删除，允许修改



## 7，新语法之let和const

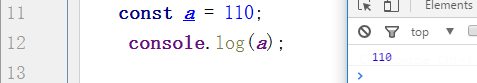
我们前面声明一变量都使用var, 使用var声明变量，有哪些特点：

* 1，声明提升
* 2，不能被删除
* 3，变量的值可以修改
* 4，可以先声明，不赋值，后面再赋值
* 5，使用var也可以重复声明

在JS是没有块级作用域，只有全局作用域和函数作用域。

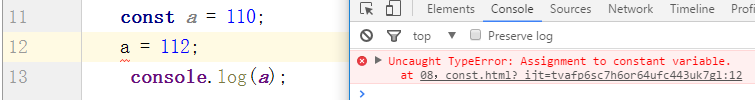
如果说让一门编程语言非常强大，那么只有变量是不能满足的。所以在ES6中， 又提供了cont let

### （1）const 用来表示定义一个常量。

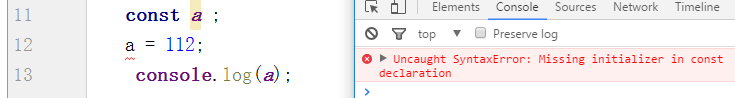


**const定义的常量都有哪些特点：**

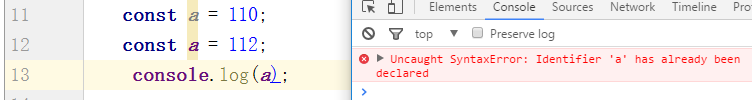
1，一经定义，不能修改



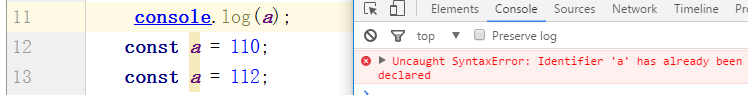
2，常量必须在声明是初始化



3，常量不能重复声明



4，const变量没有提升功能



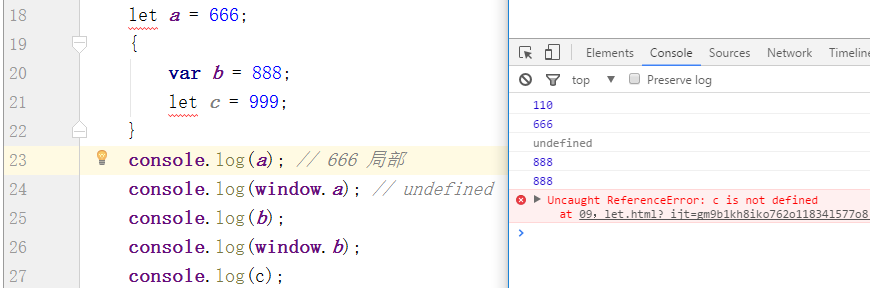
**写代码时，什么时候用var, 什么时候用const?**

1，如果你需要一个会变的量，就用var

2，如果你需要一个不变的量，就用const

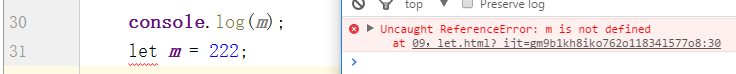
### Let

表示可以声明一个局部的变量，实际上它就是为JS提供一个块级作用域。

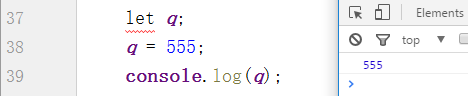


**let声明的局部变量特点：**

1，只在局部中起作用，局部：**函数**， **{}**

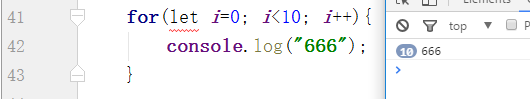
2，没有提升的功能

3，不能重复声明

4，可以先声明，不赋值，后面再赋值

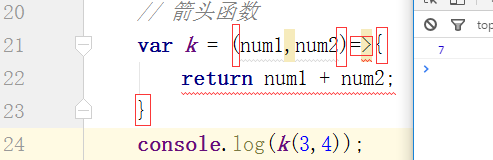
**什么时候作用Let?**

目前，在for循环中使用比较多：



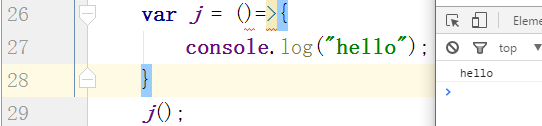
## 8，新语法之箭头函数

写法：

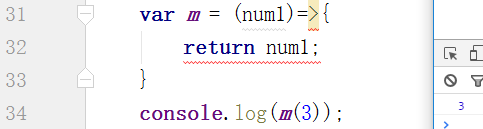


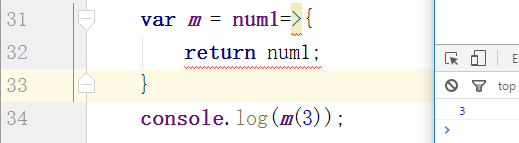
**根据参数的不同，还有几中写法：**

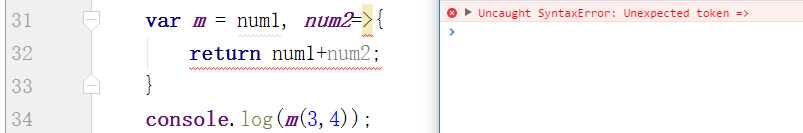
**1， ()=>{...} // 0个参数时**



**2， x=>{...} // 1个参数**

一个参数时的写法：

当一个参数时，可以把()省略不写：

3，多个参数时，不能省略（）

**新语法多数是ES6的内容，原来为什么不讲？**

1，兼容性问题，ES6在客户端JS中兼容性不好。

2，因为我们讲nodejs在针对服务器，服务器不存在兼容性问题。

## 9，JS特性之JS都有哪些特性？

JS是一门运行在浏览器端的脚本语言。

**特性：**

1，弱类型语言

2，解释性

## 10，JS特性之一切皆对象

证明：基本数据类型是对象，再证明一下引用数据类型也是对象 ==> 一切皆对象

**JS中对象分三类：**

1，内置对象

2，宿主对象

3，自定义对象

JS是一门**基于对象**的编程语言，不是一门面向对象的编程语言，面向对象语言有三大特征，封装，继承，多态。JS是模拟的这三大特征。

## 11，JS特性之单线程

**什么是线程？**

在一个软件运行过程中，真正干活是线程，一个进程对应了很多线程。

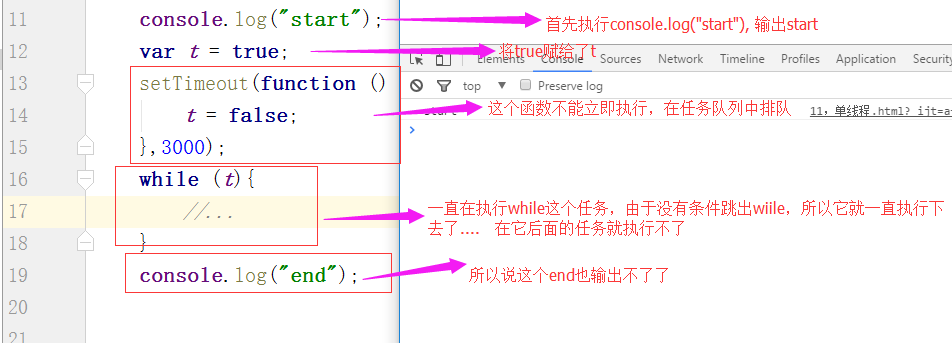
**什么是进程？**

进程（Process）是计算机中的程序关于某数据集合上的一次运行活动，是系统进行资源分配和调度的基本单位，是操作系统结构的基础。

**什么是单线程？**

如果有多个任务，任务只能一个一个的完成。

代码举例如下：



**单线程的优缺点：**

优点：实现起来简单，任务一个一个。

缺点：只要有一个任务耗时较长，后面的任务必须排队等待，会拖延整个程序的执行。

## 12，JS特性之同步和异步

* 同步 Synchronous
* 异步 Asynchronous

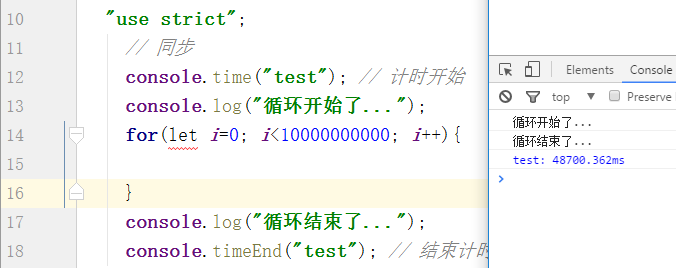
同步： 打电话， 同步操作会导致阻塞，会消耗更多的时间，能够立即得到结果

异步： 发短信，异步操作不会导致阻塞，比较节约时间，但是不能立即得到结果

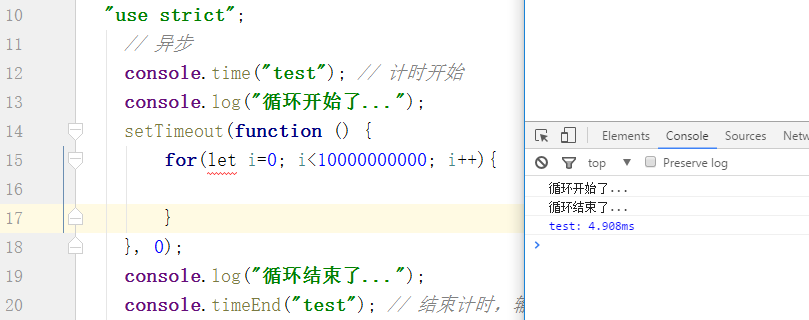
同步模式中，后一个任务需要等待前一个任务结束才能执行，程序的执行顺序与任务的排列顺序一致。

异步模式中，每一个任务有一个或多个回调函数，前一个任务结束后，不是执行后一个任务，而是调用回调函数，后一个任务则是不等前一个任务结束才执行，所以程序的执行顺序与任务的排列顺序不一致。

用代码演示同步：



用代码演示异步：



其中，setTimeout(f, 0)不是说立即执行代码，而是将f函数放到任务队列中。然后继续执行后续的代码，后续的任务执行完，才去队列中找下一个任务，所以才会执行for循环。需要注意的是，使用异步后，并不是说原来的循环用的时间从40多秒，缩短到了4ms。

**在JS中如何实现异步：**

* 1，回调函数
* 2，事件机制

## 13，JS特性之回调函数

**生活中例子：**

在大学中，你去宿舍找同学。

第一种模式：每隔几分钟，去看一下 --- 轮询

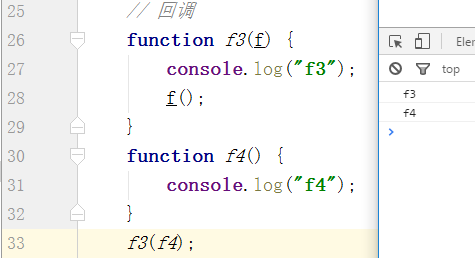
第二种模式：拜托别人，他回来后，叫我 --- 回调

**在JS中回调也是一种思想。**

**表现**：将一个函数作为另一个函数的参数， 匿名函数



另一种体现：



## 14，JS特性之事件机制

事件可以改变代码的执行顺序：



事件也可以实现异步。

# 二，Node.js快速入门

## 1，再说一下客户端JavaScript

**1，什么是客户端JS？**

JavaScript是一门运行在浏览器端的脚本语言。对应到WEB标准，它是提供行为的，也就是行为层。行为层就是提供动态交互的。

2，**客户端JS它的运行环境是什么？**

浏览器

3，**客户端JS可以做什么事？**

1，特效

2，网页游戏

3，组织页面结构

**4，客户端JS不能干什么？**

1，不能进行文件操作，出于安全考虑

2，不能操作数据库

3，不能与操作系统打交道

.....

所以说，目前我们学习的JS能力有限。

**5，JS只能在浏览器中运行吗？**

不是

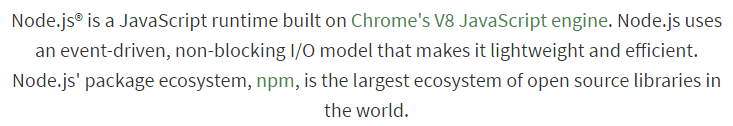
Flash, 它是基于JS

**Node.js**

## 2，什么是Node.js

官网：<https://nodejs.org/en/>

官网对Nodejs的介绍：



**提炼：**

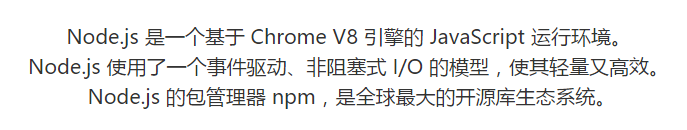
1，nodejs是基于V8引擎

2，事件驱动

3，非阻塞I/O

4，npm

中文网：<http://nodejs.cn/>



**大白话：**

Node.js其实就是一个执行环境。不是一门编程语言。在node.js这个执行环境中，可以运行我们的JavaScript, 还可以使用JS进行编程。

对比：

浏览器也是一个执行环境，我们的JS代码也可以跑在浏览器中。

**ECMAScript是一个语言的标准，在不同的平台（执行环境），有不同的实现，目前三种：**

1，客户端的JS，代码跑在浏览器这个执行环境上（平台）的

2，Flash软件中的actionScript

3，代码跑在Nodejs这个执行环境上面的JavaScript

再次说明：nodejs不是一门语言，而是执行环境，或平台

## 3，Node.js可以做什么？

PHP，JSP， ASP, 传统3P是进行服务端编程。 Nodejs也是一样的，NodeJS有自己独特的优势，它可以做：

1，网站/WEB应用

2，命令行工具（grunt, bower）

传统的PHP， JSP， ASP能做的，nodejs几乎都可以做。

## 4，学习Node.js的必要性

1，潮流

2，后面学习管理工具也是基于node

3，是移动端开发的基础

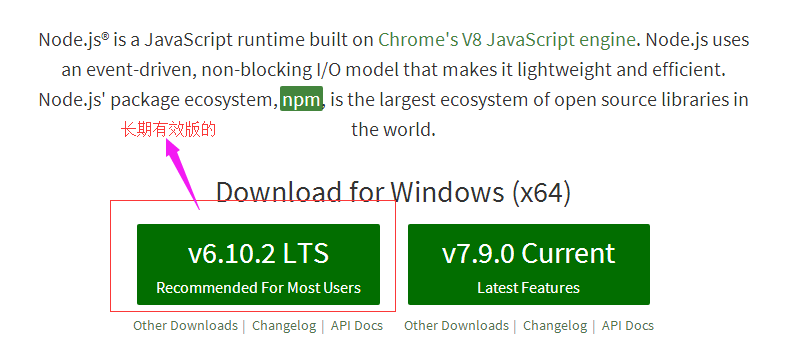
## 5，Node.js的特点？

* 单线程
* 异步式I/O
* 事件驱动机制
* 跨平台

## 6，Node.js下载与安装

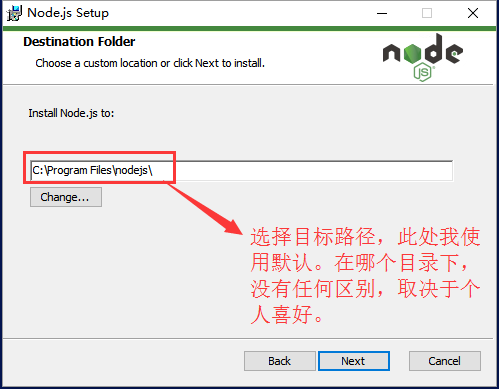
**下载**：<https://nodejs.org/en/>

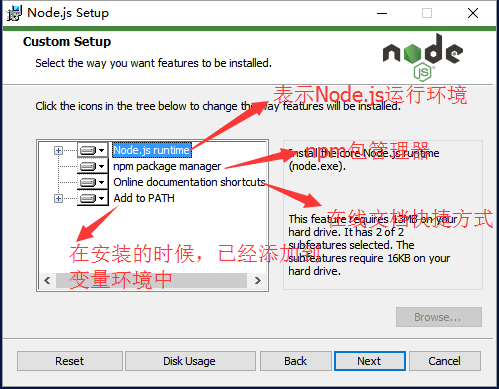
**下载长期有效版（三年）的：**



**安装时，和安装QQ差不多，傻瓜式安装。**

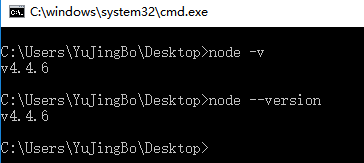
**安装过程中的问题：**





**测试是否安装成功：**

使用node -v 或node --version查看版本：



**注意的问题：**

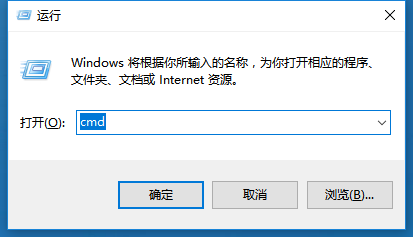
cmd窗口有缓存功能，安装之后，需要重启cmd窗口才可以生效。

## 7，Node.js初体验

当我们把nodejs安装完毕后，就可以写js代码了。

打开一个写代码的地方，cmd窗口：

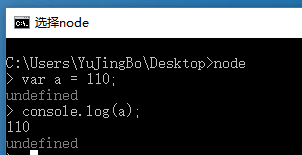
如何调出cmd窗口：win + r 然后输入cmd:



还有一种方式：

按住键盘上的shift，然后鼠标右键，再然后，在点击在此处打开命令窗口。

有了写代码的地方，我们就可以写代码了：



上面这个写代码的环境，有一个术语，REPL：

R：read 读取

E：eval 执行

P：print 输出

L：loop 循环整个过程

### 输出Hello, World!

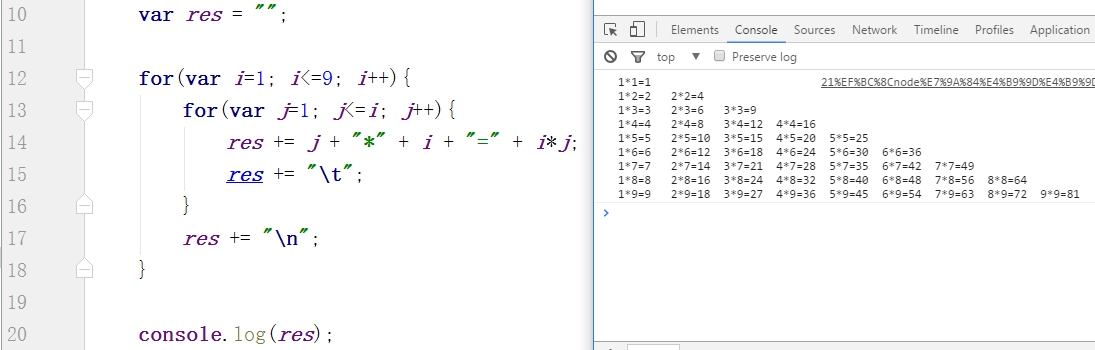


在上面的REPL环境，我们前面所学习的任何JS代码都可以写，但是不能写DOM，BOM中的代码。

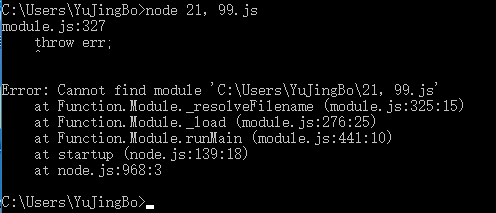
按两个Ctrl + C就可以退出REPL环境。

### （2）99乘法表

在浏览器这个执行环境中的99乘法表：



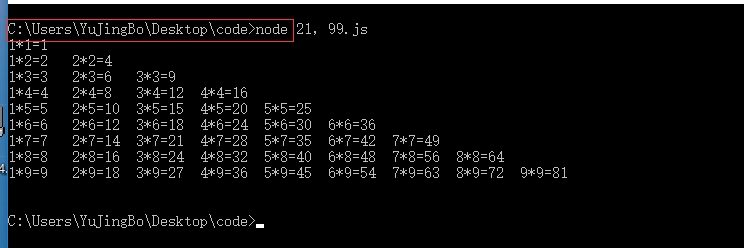
当使用node执行时，出现如下问题：



解决：



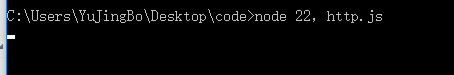
好的办法：



### 创建http服务

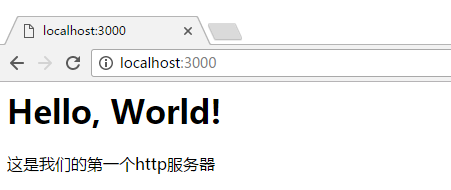
第一步：写服务器端的代码：



第二步：在REPL环境中运行我们的代码：

第三步：访问我们的服务器：

说明：此时我们的电脑充当了服务器和客户端双重角色。在我们的电脑上有一个特殊的域名，localhost, 我们的服务器，一直在监听3000端口，所以访问时，加需要加上端口



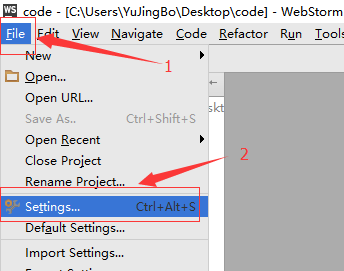
## 8，搭建开发环境

### （1），安装

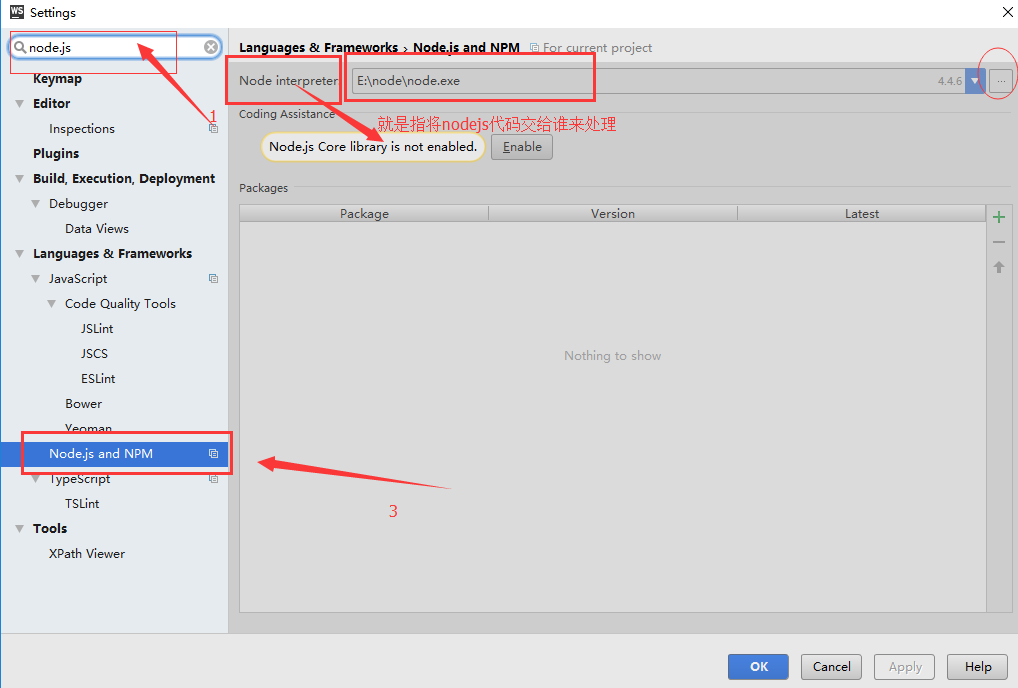
安装不说了。

### （2），设置

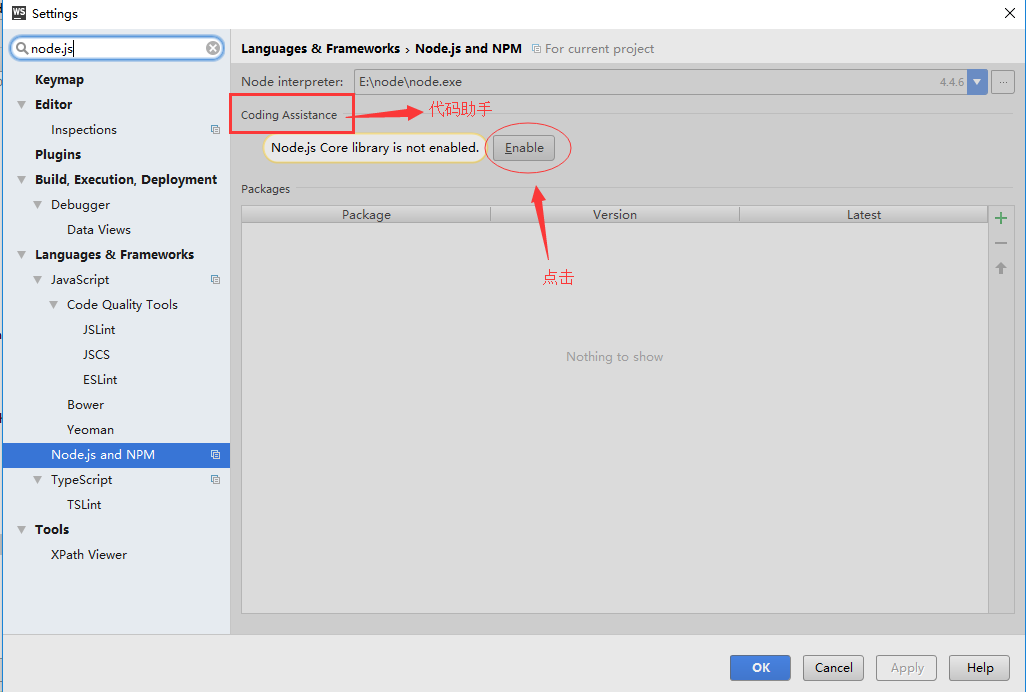
第一步：



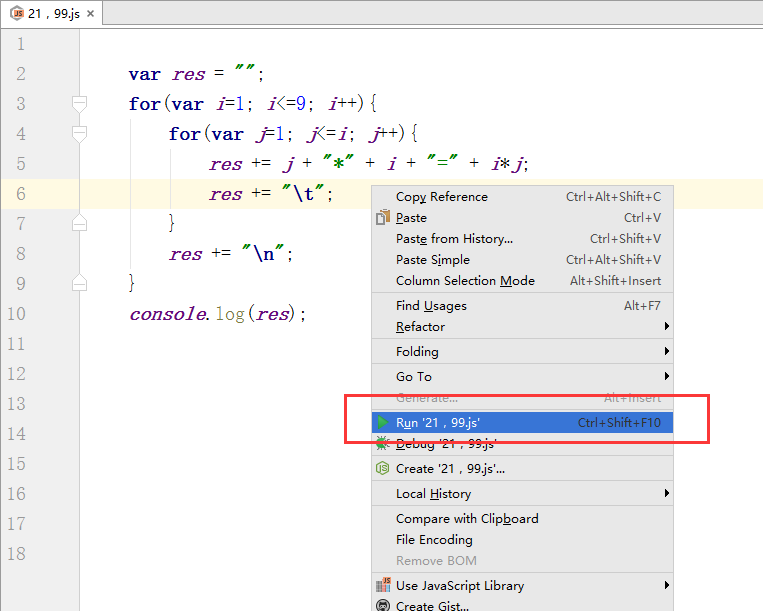
第二步：

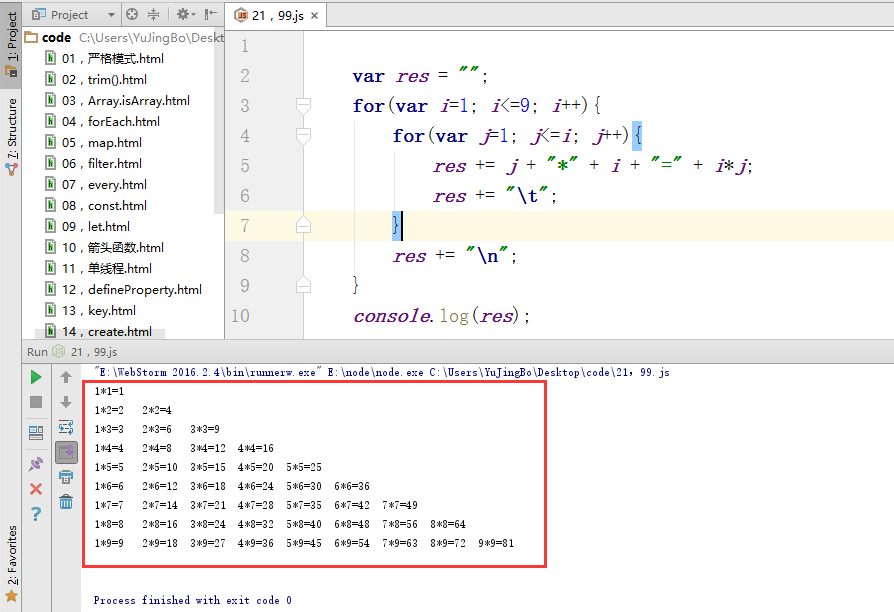


3，智能提示



### ，写代码，运行代码





**复习：**

**1，什么是node.js?**

node.js是让JavaScript运行在服务器的一个执行环境（平台）。

**2，什么是V8引擎？**

V8引擎本身是Chrome浏览器的JS解释部分，但是有个人，叫Ryan Dahl，鬼才般地把这个V8引擎搬到了服务器上，用于做服务器软件，V8引擎底层是使用C++开发的，nodejs是基于V8引擎，使得Nodejs的性能也非常高。

**3，nodejs有什么特点？**

**单线程：**

在Java, php, .net等服务器语言中，它会为每一个客户端连接创建一个新的线程，每个线程会消耗2M左右的内存，也就是说，理论上，一个8GB的内存的服务器，可以同时连接最多4000个左右。如果想支持更多的用户，就需要增加服务器的数量，这样，服务器的硬件成本就增加了。nodejs不会为每个用户创建一个新的线程，就使用一个线程，当有用户连接时，就触发一个内部事件，通过非阻塞I/O，事件驱动，可以让nodejs实现宏观上的并行。

**好处**：可以让操作系统不再创建线程，节约时间。

**坏处**：当一个用户造成了线程的崩溃，整个服务器都崩毁了，其它人也崩溃。

**非阻塞I/O:**

如，我们去连接数据库，需要一段时间，在传统的单线程处理机制中，整个线程都暂时停下来，等待数据库返回结果，才能执行后面的代码，**用专业的话说，I/O操作阻塞了代码的执行，降低了程序的执行效率。**但是nodejs是非阻塞I/O,因此在执行上面的数据连接代码之后，将立即执行后面的代码，将数据库的返回结果放到回调函数中，这样就提高了程序的执行效率。在阻塞模式下，一个线程只能处理一项任务，如果想要执行多个任务，就是使用多线程，在非阻塞模式下，一个线程一直在不停执行操作，这个线程的利用率一下是100%，它的思想是：与其多个人干活，有一部分人闲着，不如让一个人玩命地干活......

**事件驱动：**

在nodejs中，同一时刻，只能执行一个事件回调函数，但是在执行一个事件回调函数的中涂，可以去执行其它事件，然后再继续执行原来的事件的回调函数，这个机制也有一个术语，叫“事件环”。事件队列.....

# 三，模块机制

**问题：把代码不写在一个文件，有什么好处？**

１，修改代码方便，利于维护，利于团队开发。

## 1，什么是模块

从广义上来说，一个模块就是一个文件，前面学习html文件，css文件，js文件，json文件，txt文件..... 模块也是nodejs的基本组成部分。

**为什么nodejs引入模块机制？**

当我们去实现一个复杂的系统时，我们不能将所有的代码写在一个文件中，我们一般都会分项目的结构，为了可重用，可维护。如果需要在nodejs中组织我们的代码，nodejs也为我们提供了模块机制，当我们需要使用某个模块时，就引入就行了。为了可重用，可维护....

**在nodejs中常见的模块类型有哪些？**

* 1，js代码，后缀 .js
* 2，json代码，后缀 .json
* 3，c/C++代码， 后缀 .dll

## 2，Node.js常见的模块

**有两大类：**

1，系统模块 如：http模块，fs（filesystem）模块......

nodejs提供好的模块

2，用户模块

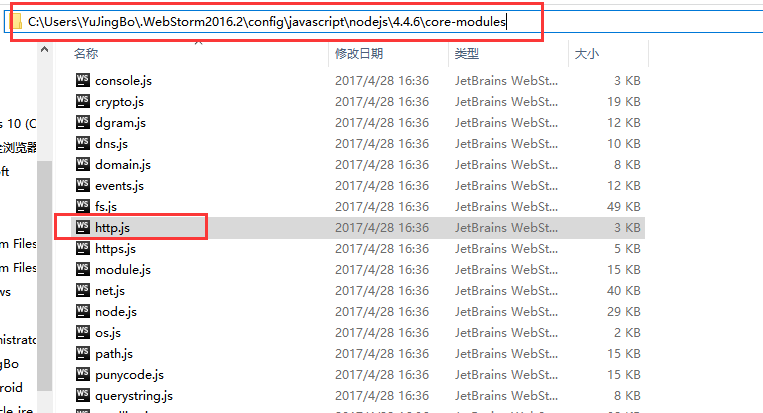
a, 第三方模块, 别人开发好的模块，我们也可以用。如express

b，自定义模块，需要我们自己编写的模块。

**系统模块和用户模块的区别：**

系统模块是强编译好的模块，通常以二进制文件存在，模块引入时，可以直接加入到内存中。

**系统模块在计算机中的存放位置：**



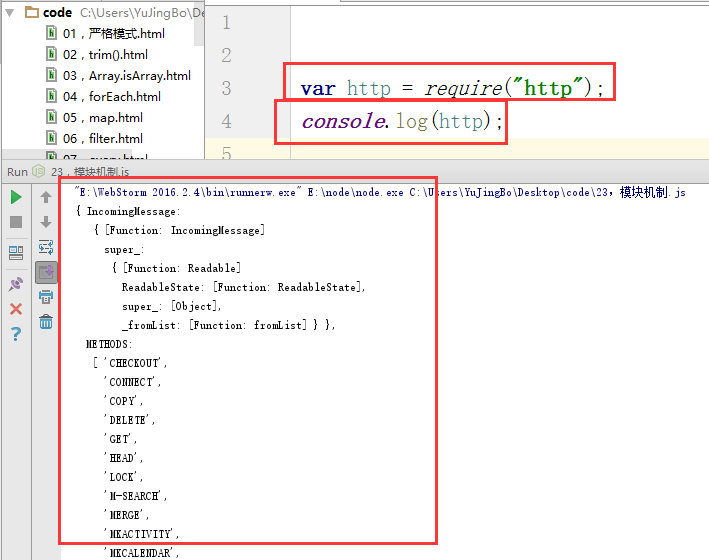
**用户模块：**

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

## 3，加载模块

require函数 用来引入模块的。

### （1）加载系统模块

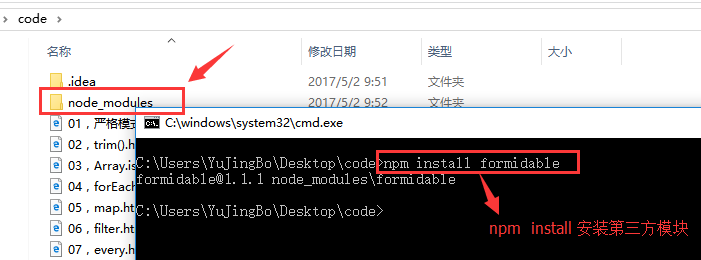


针对不同类型的模块，引入时，是有区别：

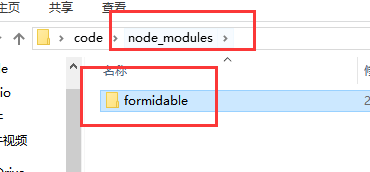
* 系统模块：require(“模块名”)
* 第三方模块：require(“模块名”)
* 自定义模块：require(“./模块名”) 或 require(“./目录/模块名”)

### （2）加载第三方模块

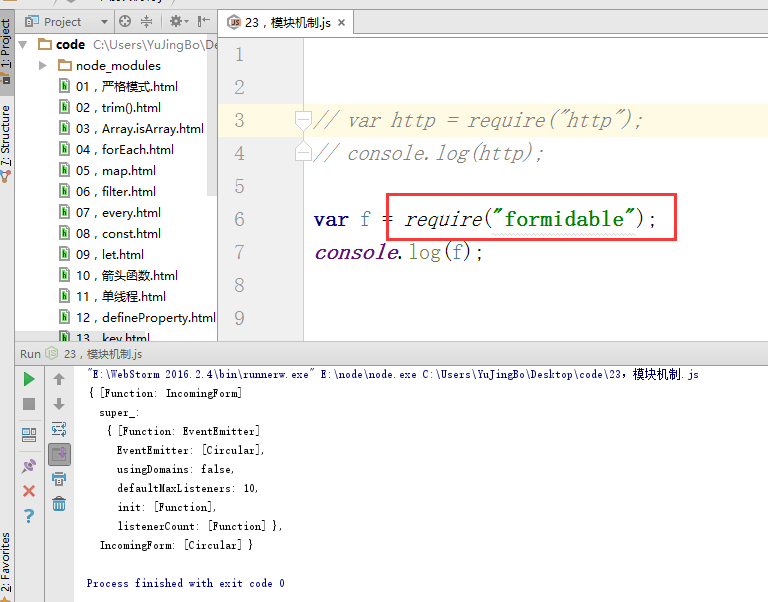
**引入第三方模块举例：**



我们安装的第三方模块就会存在node\_modules中

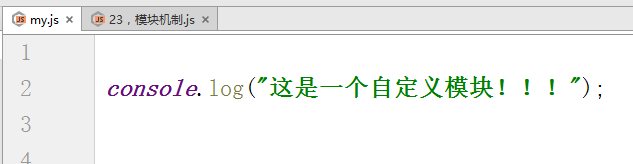


在代码中引入：

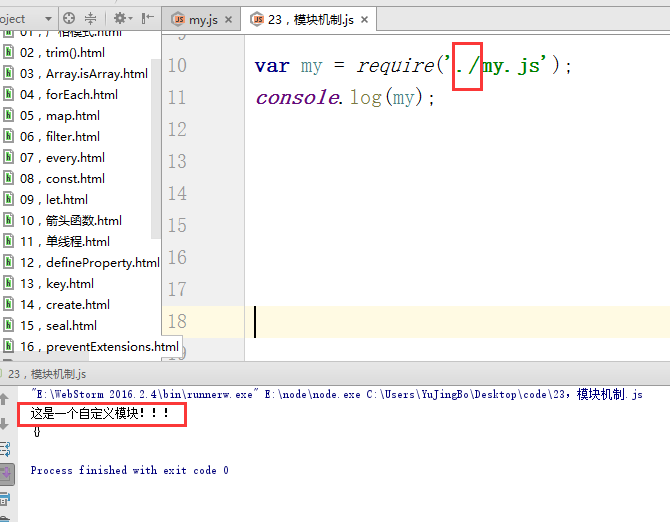


### 加载自定义模块

对于自定义模块，我们首先需要写一个模块：



加载自定义模块，代码如下： 一定要把路径写对了！



**可以加载哪些文件呢？**

1，js文件

2，json文件

3，node文件

如果当多个不同类型的文件重名时，它有一个加载的规则：

目录 ---- > .js ----> .json ----- .node 如果一直找不到，就抛出错误。

对于系统模块和第三方模块它们一般都不写后缀，所以后面，我们在自定义模块开发中，也不写后缀：

## 4，自定义模块

系统模块和第三方模块，直接拿来用就可以了，重点是自定义模块。

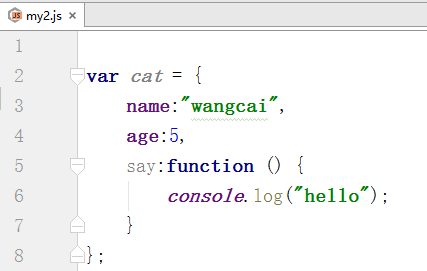
当我们写好一个自定义模块时，只需要使用require引入就可以用了。

**自定义模块也有一个使用流程：**

创建模块 --- > 导出模块 ---- > 引入模块 ------ > 使用模块

**创建模块：**

很简单，就是创建文件，写代码， 如下：



**导出模块：**



对于导出模块这个步骤，还有一个方式：exports



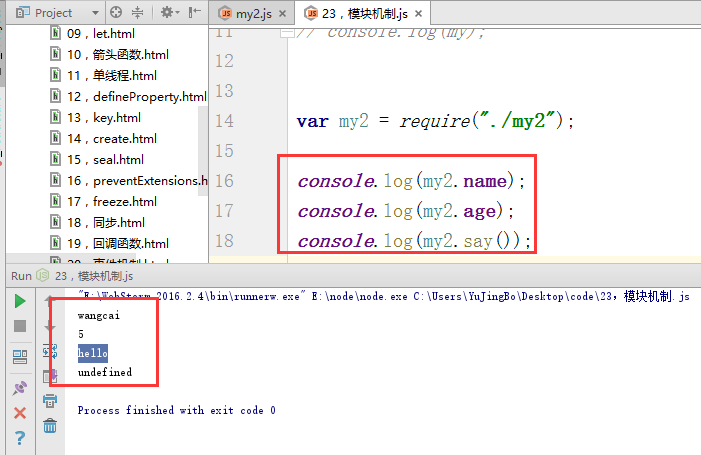
**区别：**

当导出一个类名（对象）时,使用module.exports，当导出一个变量或函数时，使用exports

**引入模块：**



**使用模块：**



# 四，文件系统

也是nodejs中的一个模块，fs 。

在nodejs有两大非常重要的模块：

* fs模块
* http模块

在编程语言中，文件是指文件和目录。也就是说目录也是文件。

## 1，什么是文件系统

fs 是filesystem的缩写，文件系统

通常所说的文件系统， 主要是指对文件系统的操作，增删改查，CRUD操作：

* C：create 增
* R：read 读
* U：updata 改
* D：delete 删

学习时，先学习文件的操作，再学习目录的操作。

## 2，常见的文件操作

对于文件操作，通常有两种方式，同步和异步。

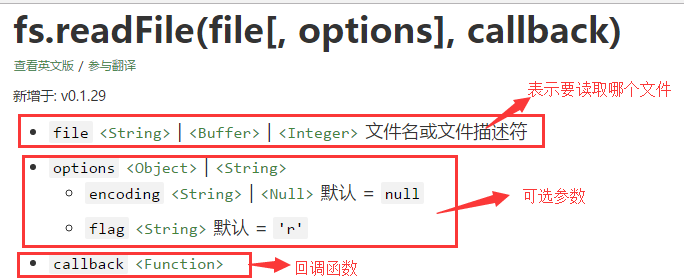


### ，读取文件

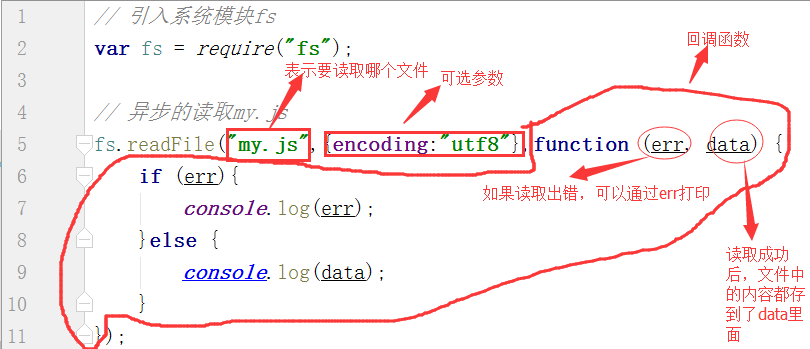
也是有两个版本：



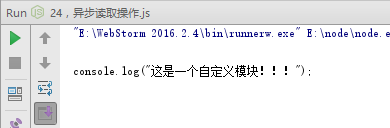
**异步读取：**



代码演示：



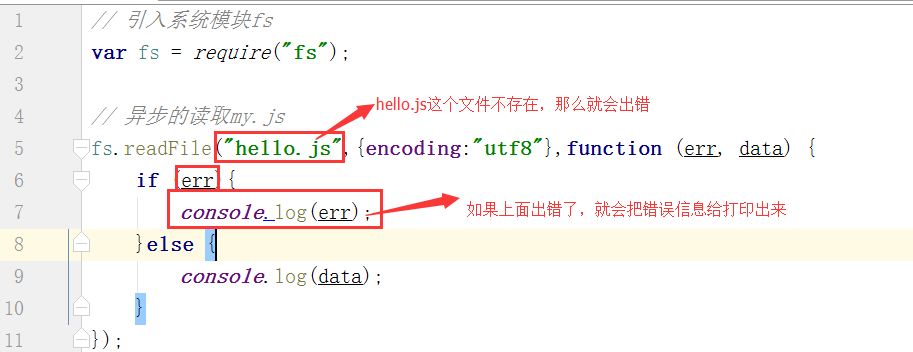
结果：



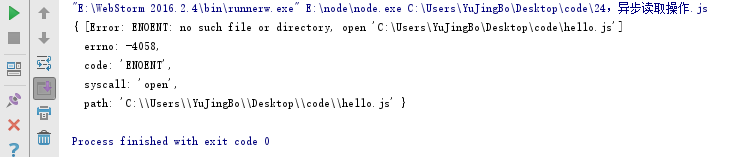
注意的问题：如果我们没有指定{encoding:"utf8"}，最终我们得到的就是一个buffer。

对于异步操作，是通过回调函数实现的，第一个参数是err,如果出错了，错误的信息都保存在err中。

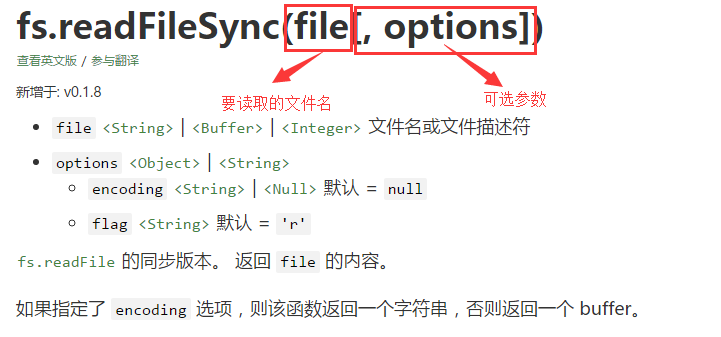
如果出错了：



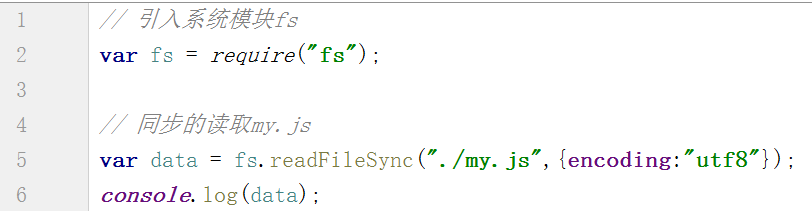
结果如下：



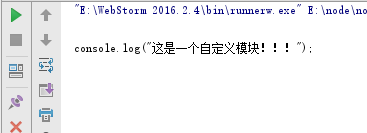
**同步读取：**



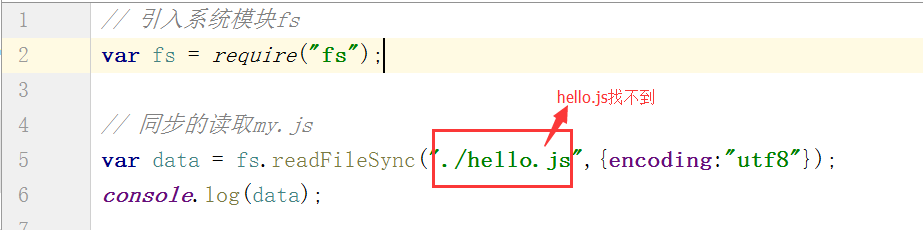
代码演示：



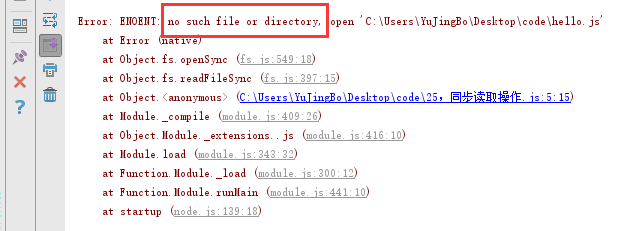
结果：



如果出错了：



结果：



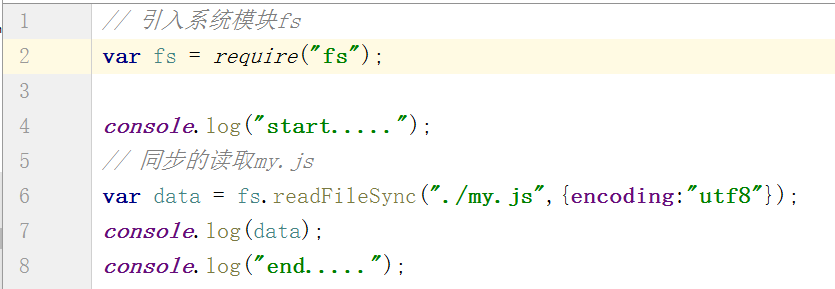
小结：

1，所有的文件操作都有两个版本，同步的和异步的。

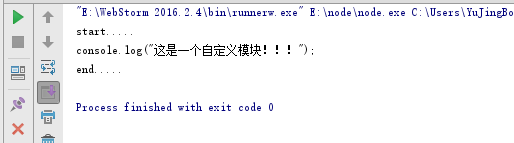
2，对于同步的，返回值就是文件中的内容，对于异步的，文件中的内容在回调函数的参数里面。

**关于执行顺序的问题：**

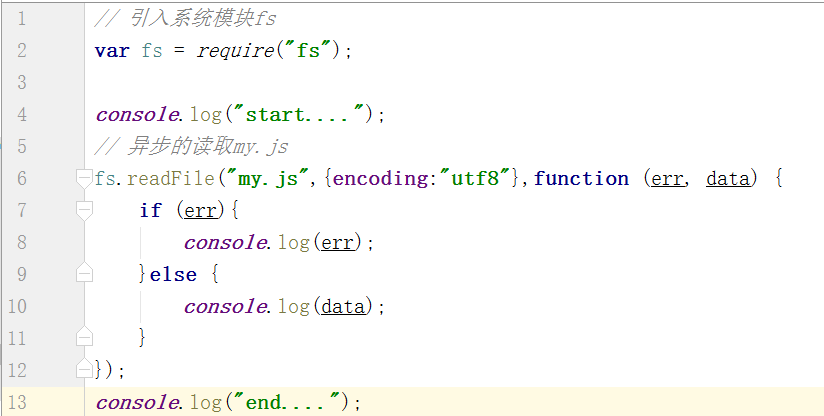
对于同步的执行顺序：



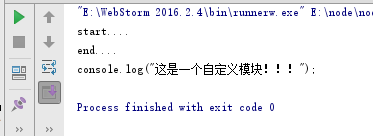
结果：



对于异步的执行顺序：



结果：

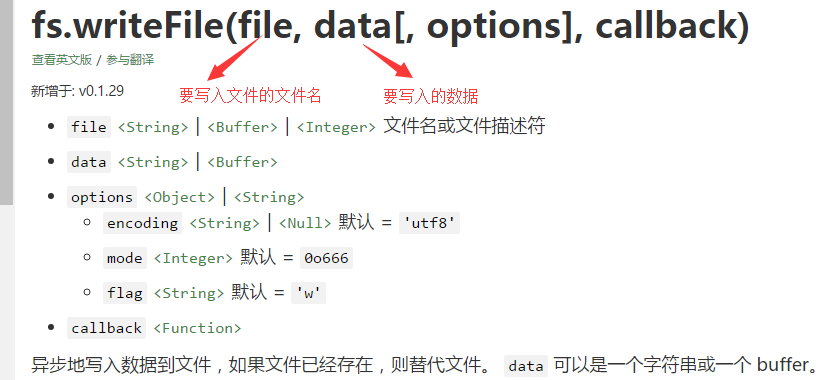


### ，写文件操作

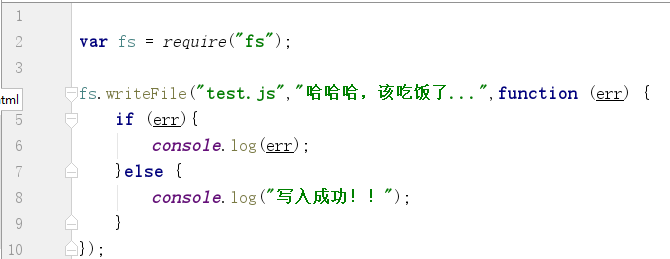
a，覆盖式的写文件

同读取文件操作一样，写文件是fs.writeFile()和fs.writFileSync()

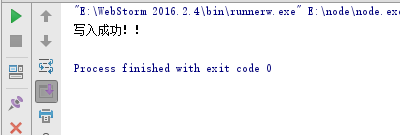
**异步操作：**

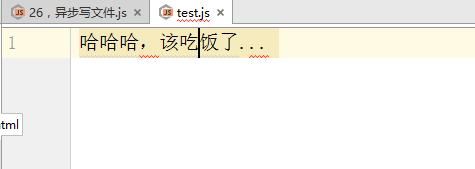


代码演示：

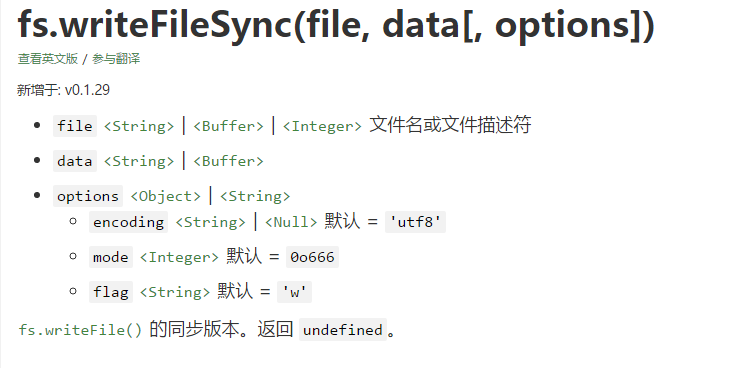


结果：

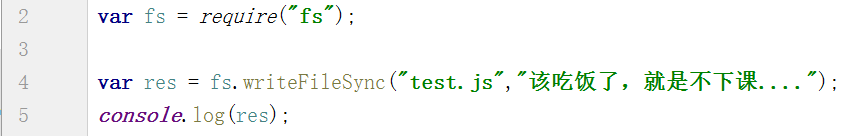


在test.js中就有了写入的内容：  
 

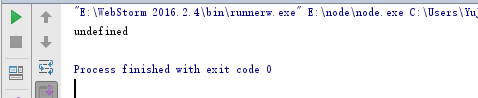
**同步操作：**



代码演示：

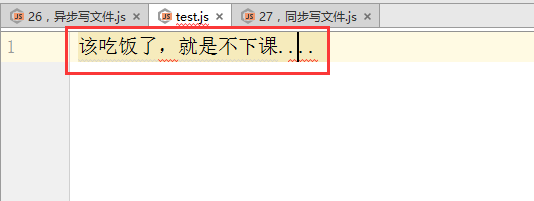


结果：



这里面的undefined表示没有错误，或是写入成功了。

在test.js就有了刚才写入的内容了，它会覆盖之前写入的内容：



b，追加式的写文件

append追加的意思， 就是当我们往文件里面写入内容时，写入的内容会追加到原来的内容后面。也有同步和异步的两个版本：

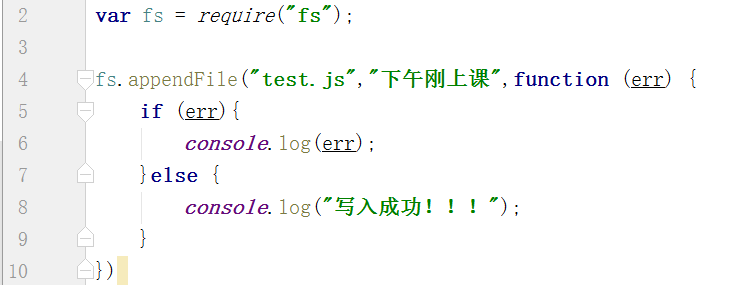


**异步操作：**

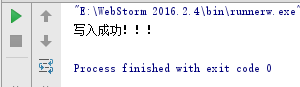
fs.appendFile();



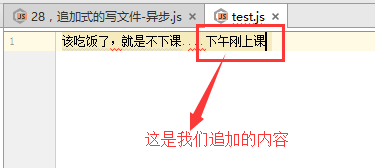
代码演示：



结果：



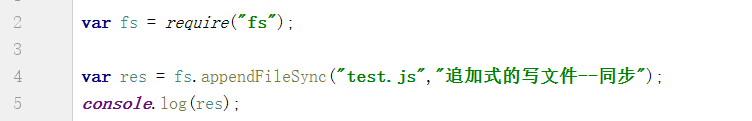
在test.js中就追加了我们上面写的内容：



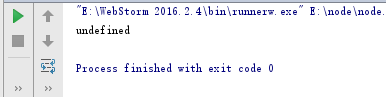
**同步操作：**



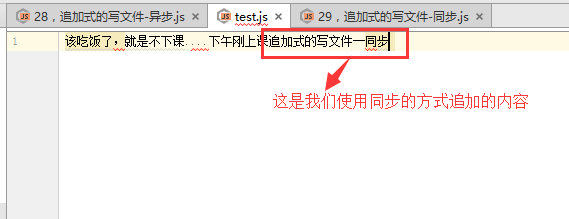
代码演示：



结果：



在test.js中就有了我们上面追加的内容了：



**小结：**

1，对于写文件，有两种方式，一个是追加式的，一个是覆盖式的，它们均有同步的版本也有异步的版本。

2，在写文件时，如果文件不存在，那么它将新建一个文件

### ，获取文件信息

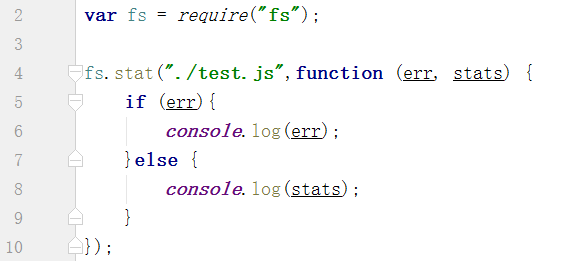
使用到的API：



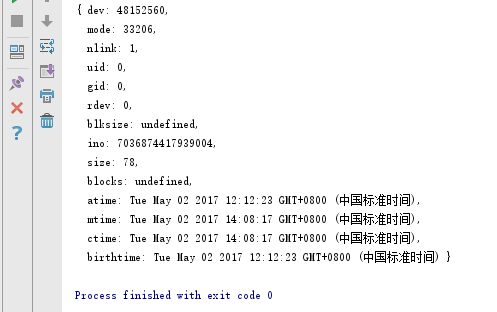
异步的操作：



代码演示：

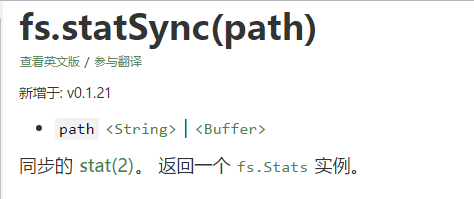


结果：

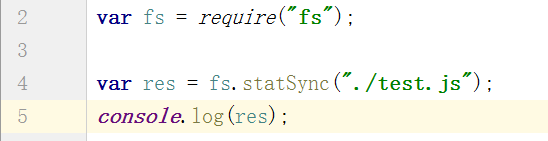


注意：上面的gid, uid, mode, ino.... 这些信息在linux或unix操作系统下才可能有用。

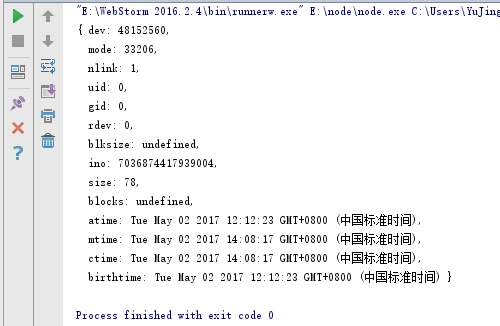
**同步的操作：**



代码演示：



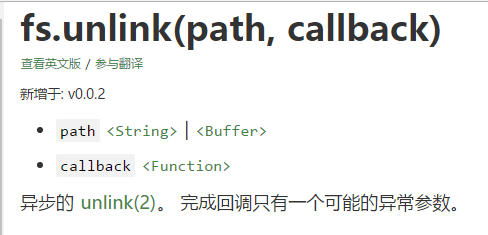
结果：



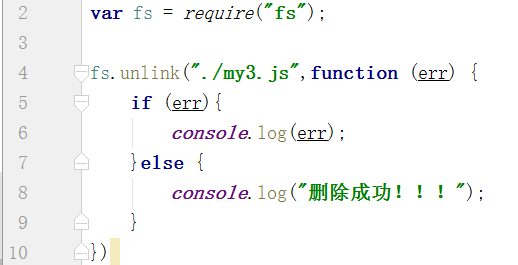
### ，删除文件



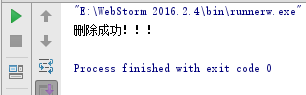
**异步的：**



代码演示：



结果：

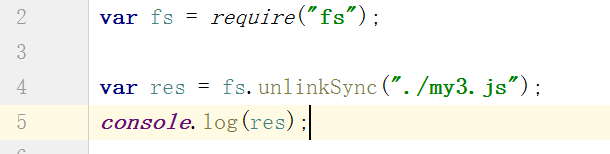


最终，my3.js被删除了，在我们的项目中就不存在了。

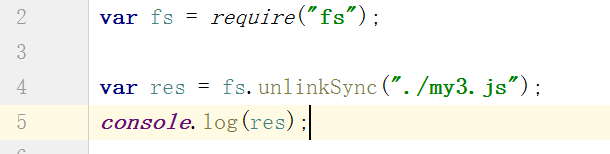
**同步的：**



代码演示：



结果：



最终，my3.js被删除了，在我们的项目中就不存在了。

## 3，常见的目录操作

### ，创建目录

mk make 制造 在编程中，可以理解成创建。

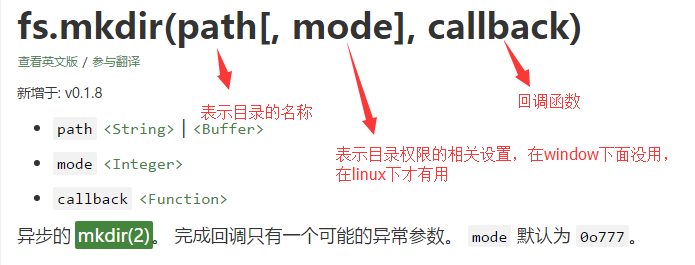
directory 目录

创建目录 ---> mkdir

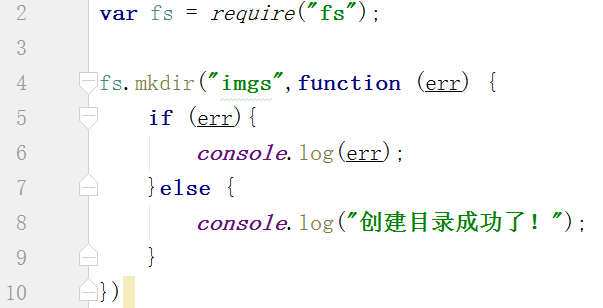
对应的API如下：



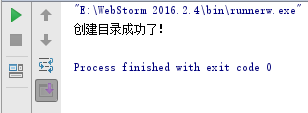
**异步的：**



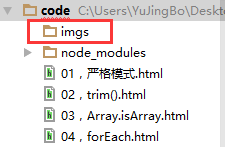
代码演示如下：



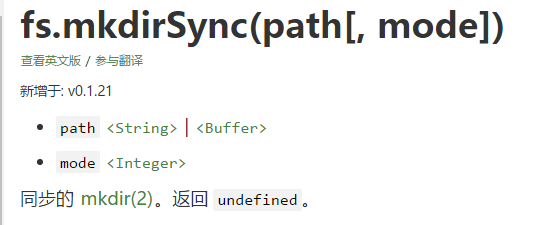
结果：



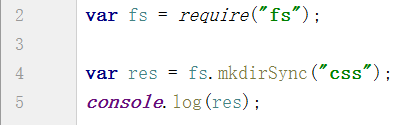
最终在我们项目中就多了一个imgs的目录，也就是文件夹。如下：



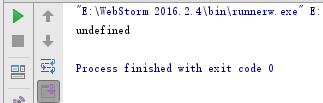
**同步的：**



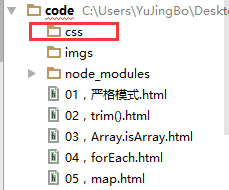
代码演示如下：



结果：



当执行完上面的代码后，在我们项目中，就有了一个css目录：



小知识 ：判断一个目录是否存在？

对应的API如下：



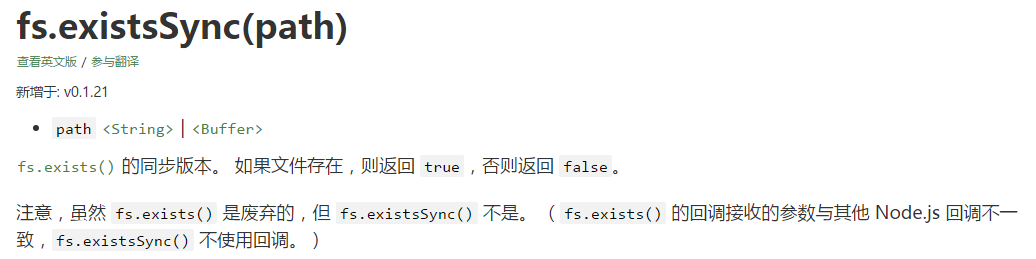
异步的：



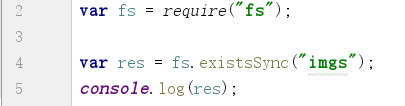
代码演示：



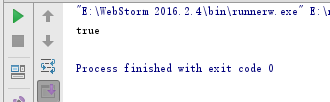
同步的：



代码演示：

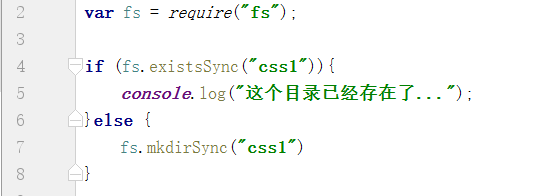


结果：



如果目录存在，就返回true， 如果不存在，就返回false.

现在，当我们想去创建一个目录，就需要去考虑目录是否存在，代码如下：



### ，删除目录

delete remove

rm -- > remove 删除

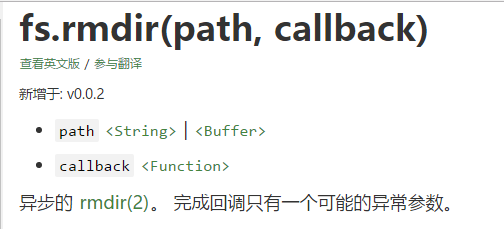
directory 目录

删除目录 --> rmdir

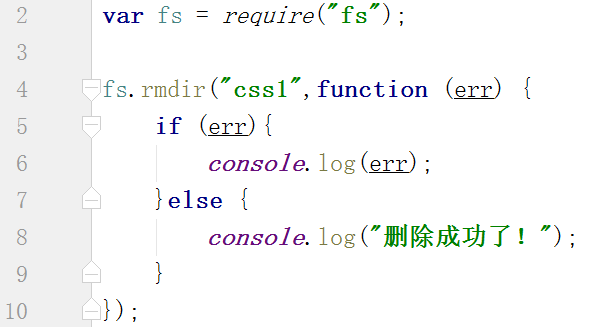
对应的API：



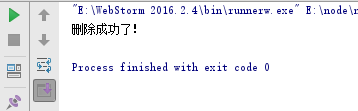
**异步操作：**



代码演示：



结果：

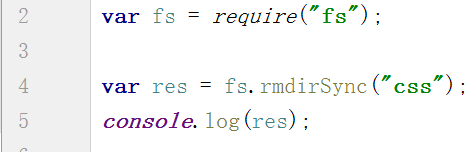


在我们项目中，就没有了css1这个目录了。

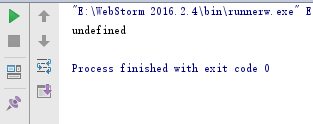
**同步操作：**



代码演示如下：



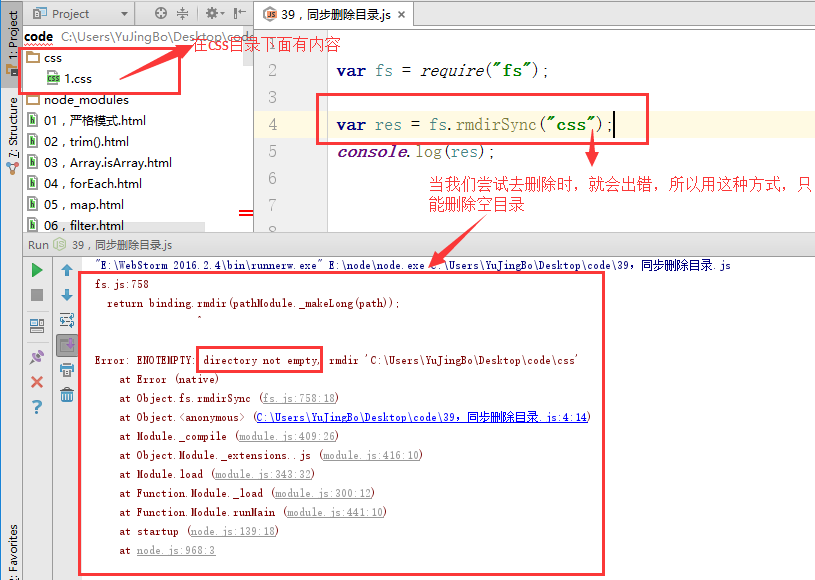
结果：



在我们项目中，就没有了css这个目录了。

注意点：

我们现在只能删除空的目录，不能删除带内容的目录：



### ，读取目录

read dir

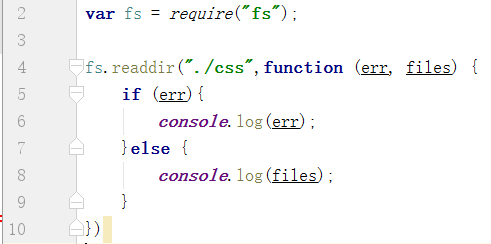
读取目录 readdir



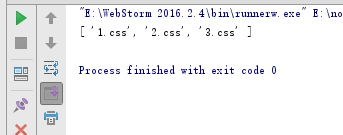
异步的：



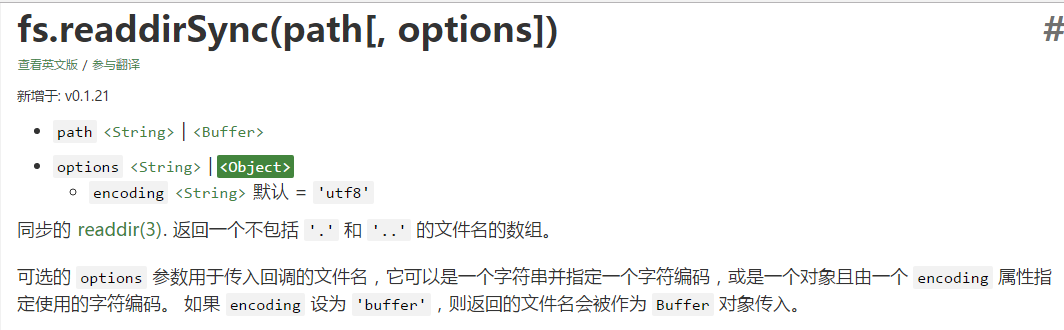
代码演示如下：



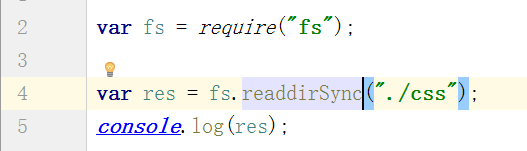
结果：



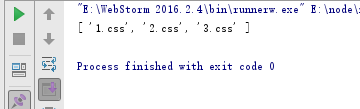
**同步的：**



代码演示如下：

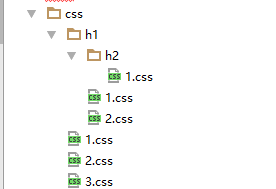


结果：



### ，递归删除目录

在一个目录下面，还有其它的目录或文件，在二级目录下面，还有其它的三级目录....



现在，要删除css目录，怎么删除？

分析：

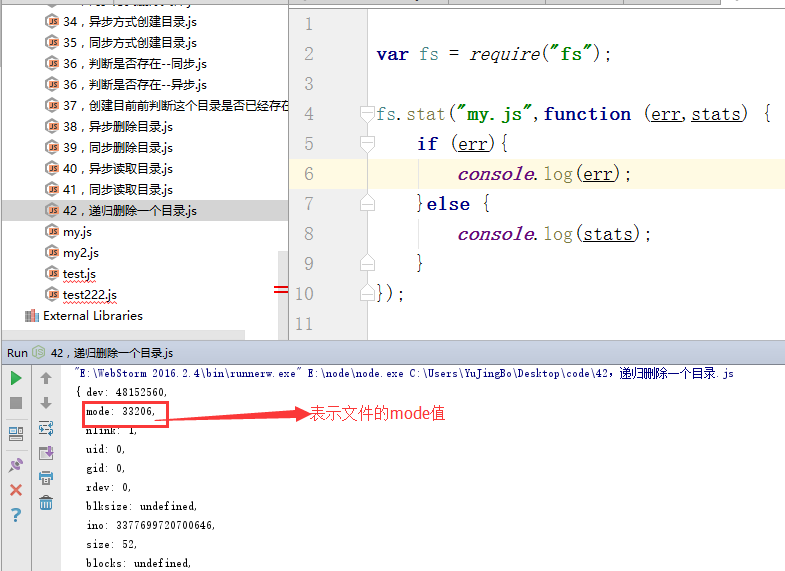
1，肯定要用到unlink 还 rmdir

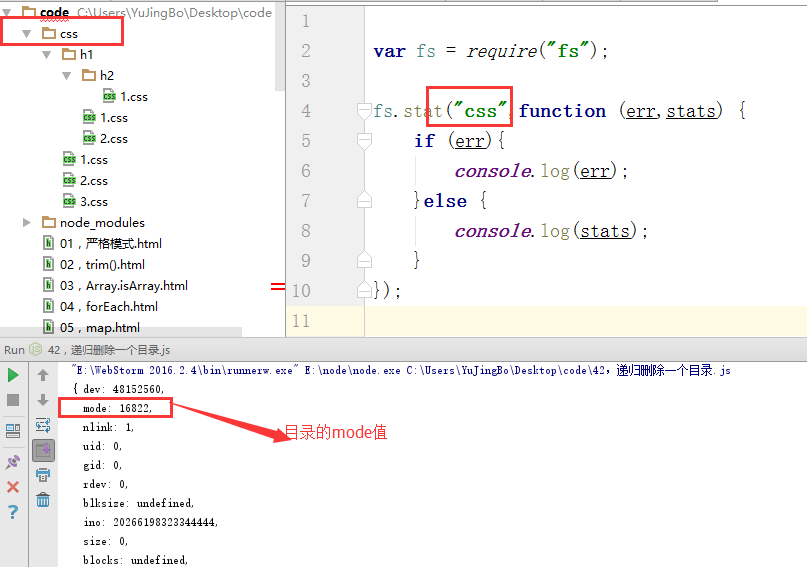
2，肯定还在判断出它是一个文件还是一个目录

3，如何去判断它是一个文件，还是一个目录 stat

先看一个stat：

文件的mode值和目录的mode值是不一样的：





有了这两个值，就可以判断出它是一个文件还是一个目录了。

有了上面的判断出的结果，如果是文件，就用unlink, 如果是目录，就用rmdir。

封装一个函数，给这个函数起一个名字：remove();

代码如下：



需要调用：



## 4，综合练习

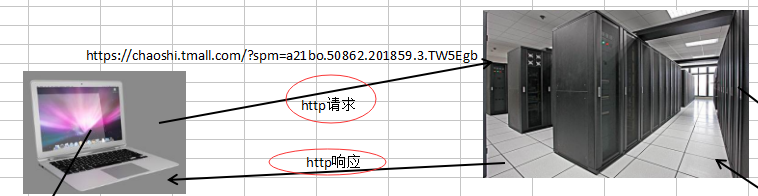
# 五，使用node.js进行Web开发

### 1，什么是HTTP

**H**yper**T**ext **T**ansfer **P**rotocol 超文本传输协议

http协议是服务器传输超文本到浏览器的传输协议，也是一个标准（规范），用来规定，服务器和客户端是如何进行信息传输的。

http包含http请求和http响应：

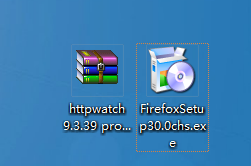


如果查看细节：

（1）在浏览器中有一个开发者工具，network选项：

（2）专业一点的：httpwatch 在ie，火狐中可以使用。

火狐浏览器，httpwatch,下载版本如下：



对httpwatch地简单介绍：



### （1）wampServer软件介绍

这个软件包是用来搭建php开发环境的，我们用来就是为了提供出一个服务器环境。

WampServer就是Windows Apache Mysql PHP集成安装环境，即在window下的apache、php和mysql的[服务器软件](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8%E8%BD%AF%E4%BB%B6" \t "http://baike.baidu.com/_blank)。

这个软件也是傻瓜式安装。

当这个软件安装完后，就相当于，你有了一个服务器。此时你的电脑即充当客户端也充当了服务器。

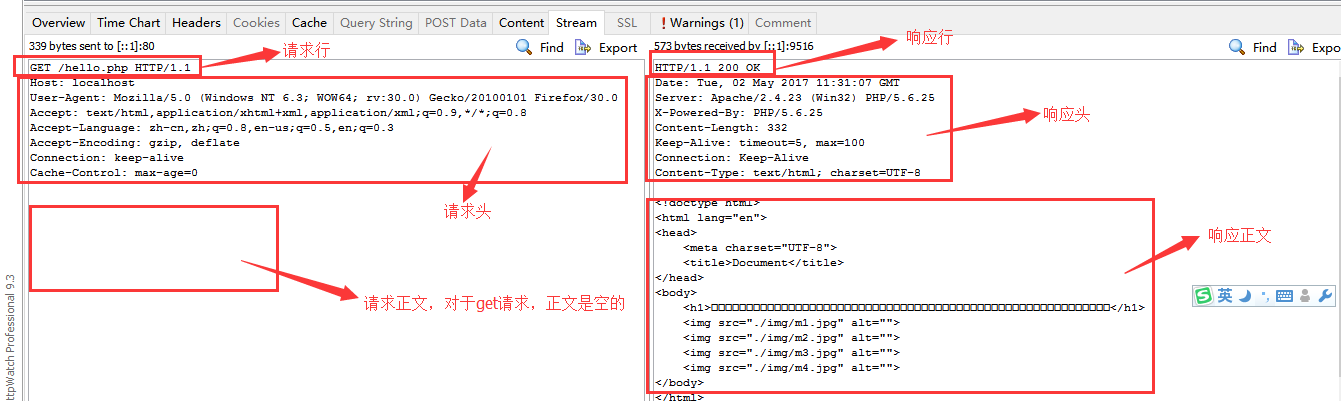
当我们访问百度服务器时，www.baidu.com， 当我们想访问我们的本地服务器，有一个特殊的域名，叫：localhost

### （2）http请求

http请求由三部分组成：

* 请求行
* 请求头
* 请求正文

通过httpwatch可以看到请求行，请求头和请求正文：



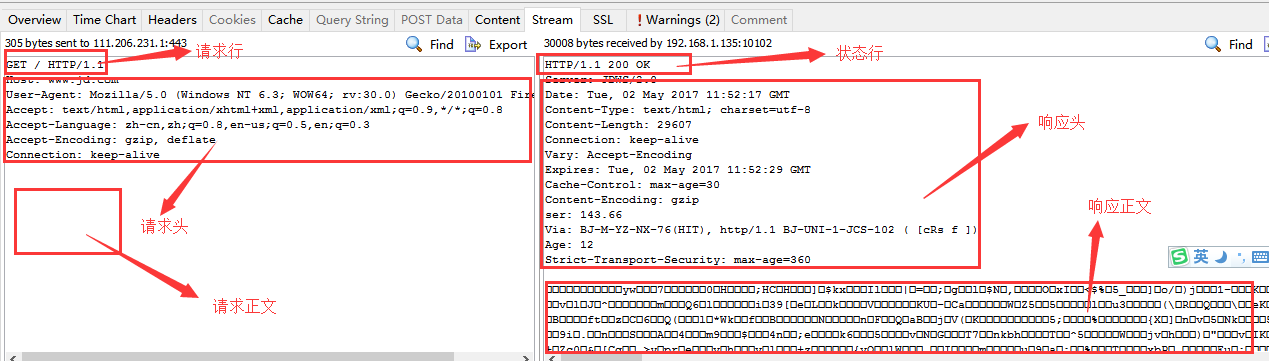
post请求：



### （3）http响应

http响应也分三部分组成：

* 状态行
* 响应头
* 响应正文

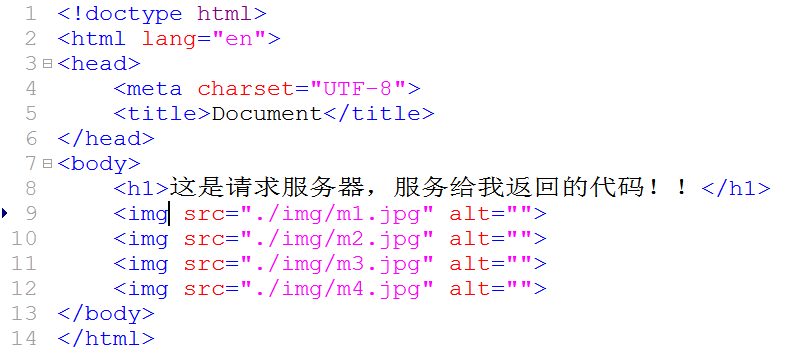


**问题**：在css中，通常会使用大图，小图合成大图，有的地方也叫它为精灵图，或叫它雪碧图，你使用精灵图有什么好处？

答：在web中，每一个文件都是资源，每一个资源都对应了一次请求和响应了，如果有10张图片，就需要请求10次，每一次都是需要花费时间的。当把10张小图，合并成一个大图时，这个资源所对应的请求就只有一次，从而可以节省时间，提高性能。

### （4）通过httpwatch分析本地服务器的请求和响应

在服务端有如下代码： hello.php



除了上面的代码，还有一些图片，在上面的代码中用到了这些图片：



当我们在浏览器中输入localhost/hello.php时，这时就会去请求服务器：



### 2，Server对象

**复习** ： http模型

http由两部分：http请求和http响应



**请求：**

1，请求行

2，请求头（消息报头）

3，请求正文

**响应：**

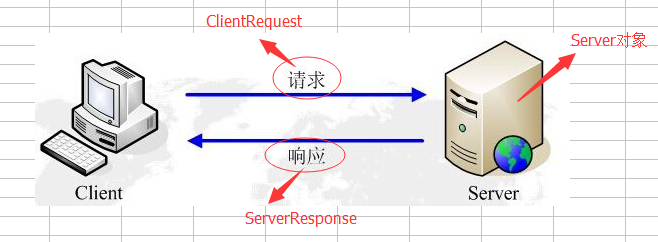
1，状态行

2，响应头（消息报头）

3，响应正文

对于上面的http模型，在nodejs中都是通过对象来描述的，服务器也是一个对象，请求也是一个对象，响应也是一个对象。

在http模型中，常见的三个对象所处的位置：



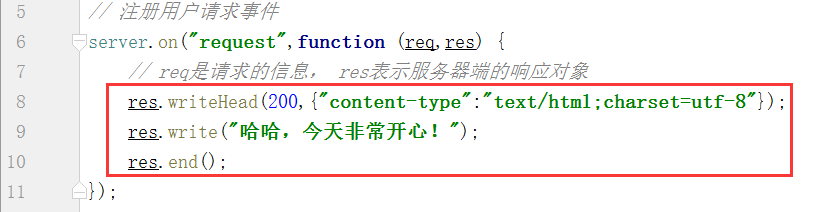
我们先去学习Server对象：

**第一步，如何去创建一个Server对象？**

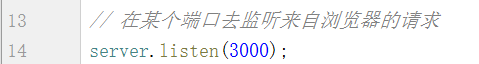
答：通过http.createServer(); 就可以创建一个Server对象。



**第二步，注册用户请求事件**



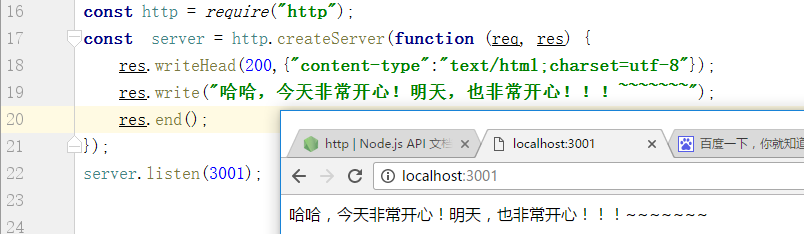
**第三步，在某个端口监听来自浏览器的请求**



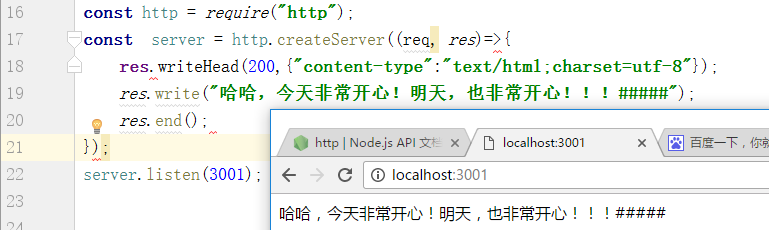
具体代码如下：



上面写的代码是常规的方式，目的就为了让你理解它，现在我们用下面的方式写：



使用es6的写法：



官方的写法：

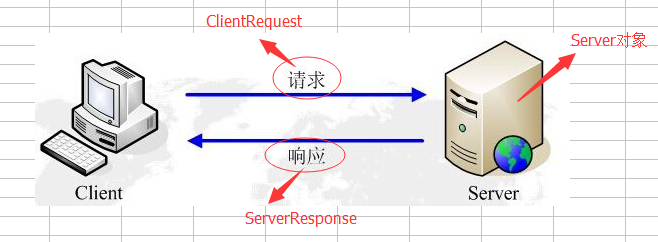
先说一个字符串解析变量：





### 3，ServerResponse对象

再看一个http模型：



**1，这个对象怎么创建出来？**

它和Server对象不一样，Server对象需要通过http.CreateServer创建，ServerRespoonse这个对象不需要手动创建，它会自动创建，作为request监听器的第二个参数：



**2，ServerResponse对象都有哪些方法？**

* 1，writeHead 把一个响应头写入请求
* 2，write 把数据写入响应正文
* 3，end 表示完成响应，同时我们也可以把数据写在end中
* 4，setHeader 设置特定的http头信息
* 5，getHeader 获取指定的http头信息

**3，回顾一下，http响应的三个构成部分？**

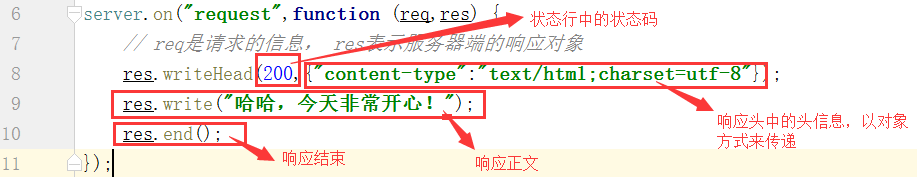
1，状态行

2，响应头

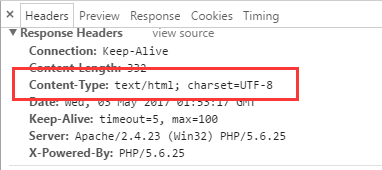
3，响应正文

**4，write表示写入响应的正文，end表示响应结束，同时可以顺便写入正文，writehead，getHeader， setHeader它们是用来设置状态行和响应头的。**

**5，解释前面写过的代码：**

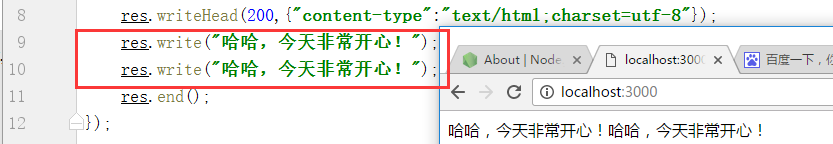


**6，看一下，谷歌浏览器中的响应的部分：**

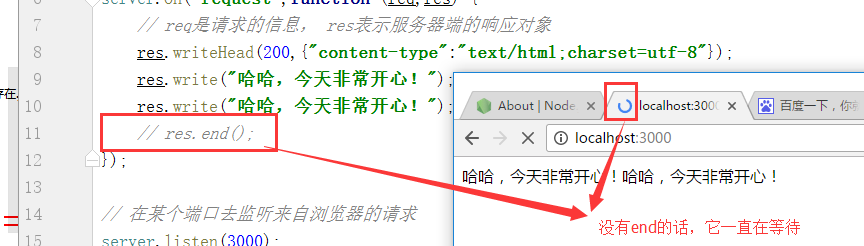


**7，使用ServerResponse注意的几个问题：**

a, write方法可以写多个：



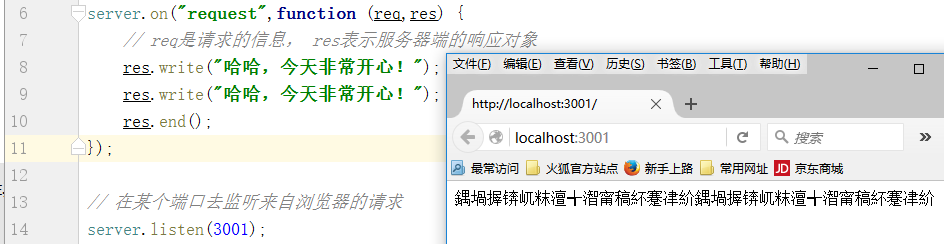
b，end方法不能没有：



c，writehead不能重复写：

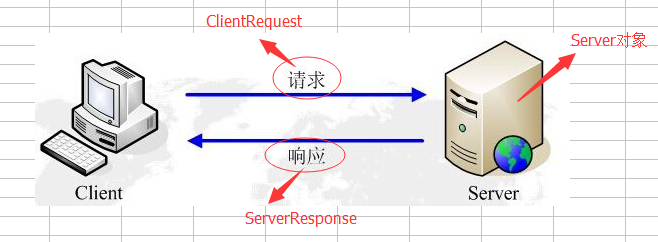


d，writeHead不能省略

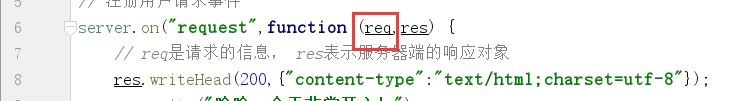


### 4，IncommingMessage对象

http模型如下：



**1，对于下面的代码，req是什么意思 ？**

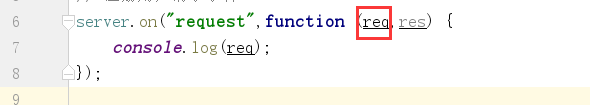


req是指IncommingMesage:



**2，IncommingMessage是如何创建的？**

它和Server对象不一样，Server对象需要通过http.CreateServer创建，IncommingMessage这个对象不需要手动创建，它会自动创建，作为request监听器的第一个参数：



**3，IncommingMessage有哪些方法，属性？**

* httpVersion http的版本
* headers 头信息
* rawHeaders 原始头信息
* url 请求的url字符串
* method 请求的方式
* statusCode 响应的状态码

### 5，URL路由

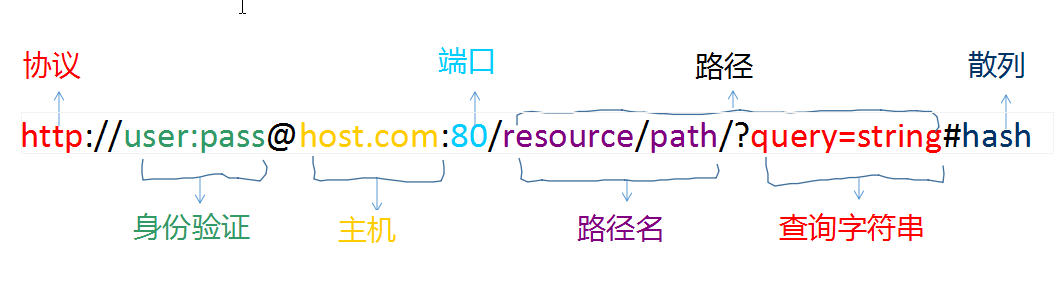
### （1）URL介绍

URL： Uniform Resource Location 统一资源定位符

URL用来表示一个资源（html页面，音乐，图片，视频.....）在互联网上的一个地址，我们可以通过URL访问到这个资源。

在html中，href属性，src属性都可以对应到一个资源。

URL具有唯一性，它的唯一性，是如何实现的？

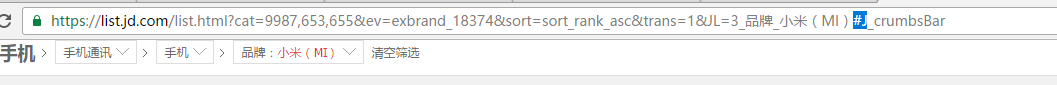


协议：通信协议，常见的有http，https, ftp等

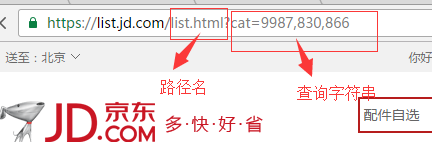
身份验证：对于http,https协议，身份验证，忽略不计。

主机：[www.baidu.com](http://www.baidu.com) localhost 192.168.0.110

端口：对于http服务，一般是80端口

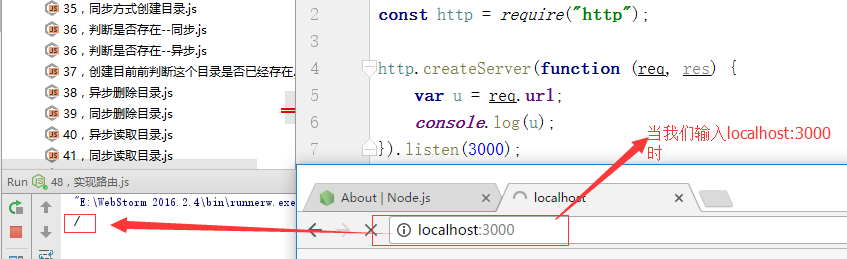


举例：



### 实现路由

利用req对象的url属性：



当我们输入：localhost:3000/user 结果是： /user

当我们输入：localhost:3000/login 结果是： /login

当我们输入：localhost:3000/vip 结果是： /vip

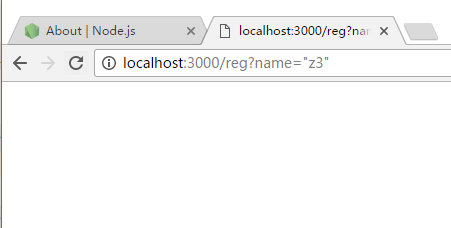
需求：当我们输入localhost:3000/user， 在页面出现**用户管理页面**，当我们输入localhost:3000/login， 在页面**出登录页面**...

实现如下：



当用户输入不同的url时，它会出现不同的页面。

如果说，当你输入这了个url： localhost:3000/user/?name=”z3”,结果就是一个空白页面，如下：

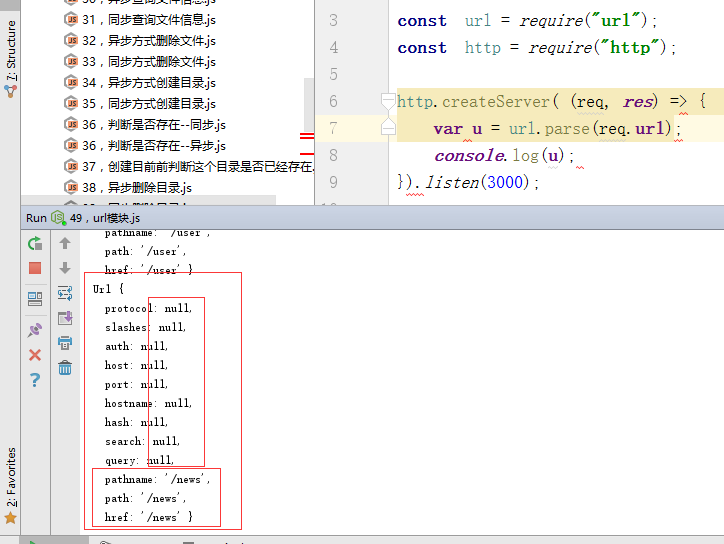


当我们想对查询字符串解析时，那么我们需要进一步的解析我们的url。在nodejs中专门又提供了一个模块，这个模块专门用来解析url， 这个模块就叫url模块。

### url模块

如果使用这个模块，也需要引入。

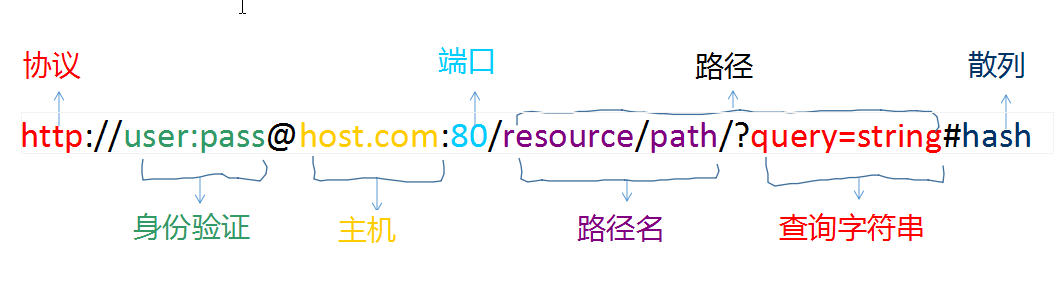
对于url模块使用如下：



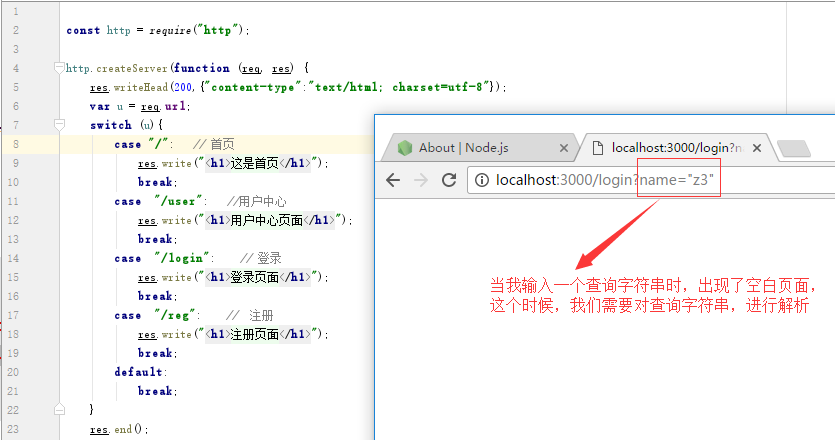
上面的结果中，没有协议，也没端口，假如我们需要这些信息，怎么办？



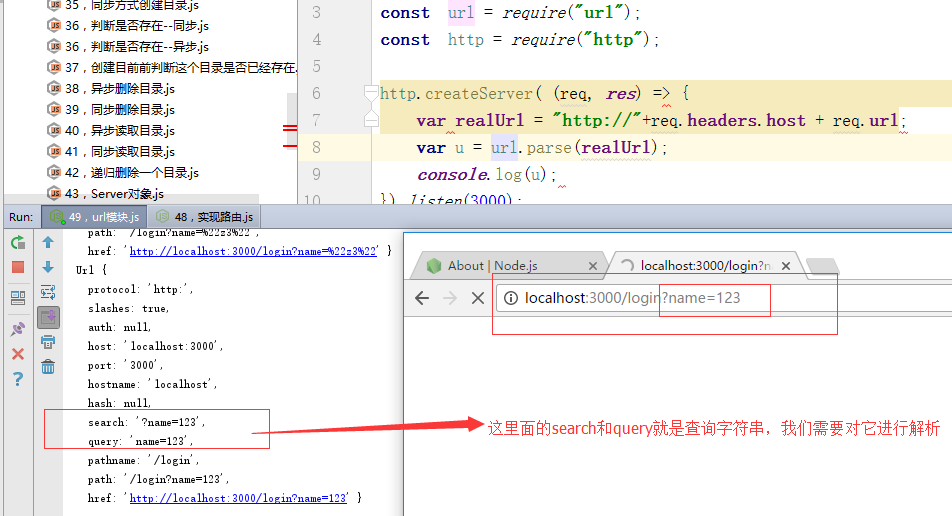
### 6，解析查询字符串



**什么时候需要对查询字符串解析？**



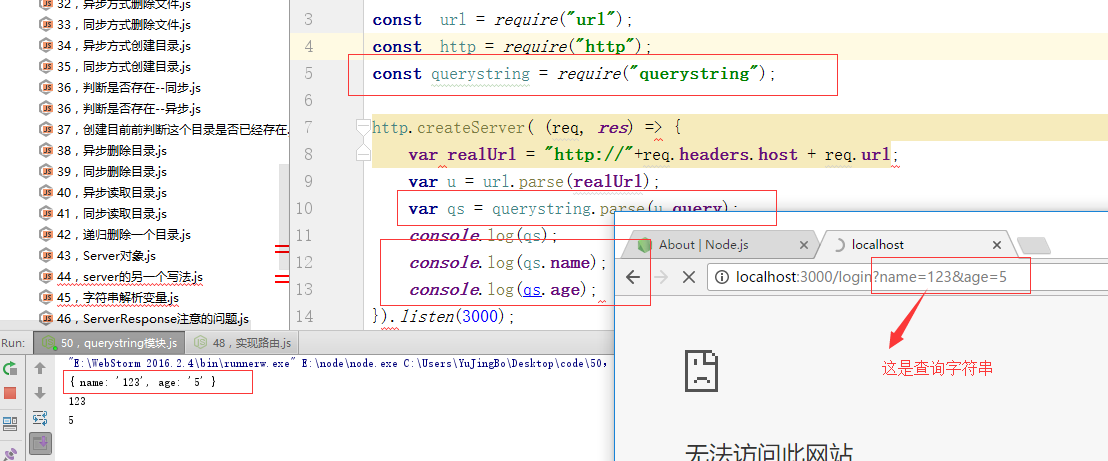
得到查询字符串：



query name=123&age=5进行解析：

通过原始我方式，我们可以达到需求。那么原始的方式，比较复杂，nodejs提供了一个模块：querystring模块。

使用如下：



### 7，载入静态页面

前面的学习，我们写过很多页面，我们也可以将这些页面放到**空间**中，然后通过域名进行访问。

在node中，是没有Web容器，我们需要自己写代码来载入静态页面。

写代码实现载入静态页面：

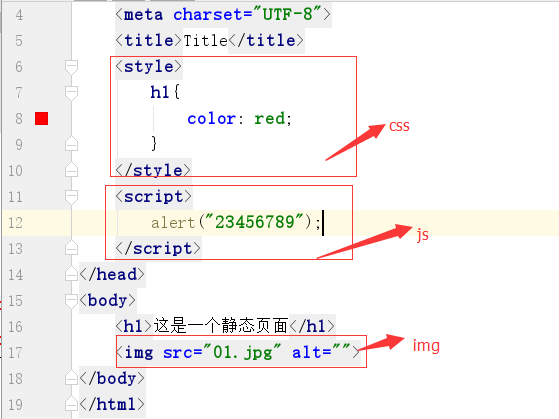
先准备静态页面：



再写一个js文件，来载入上面的静态页面：



向静态页面中加入js, css，img .....

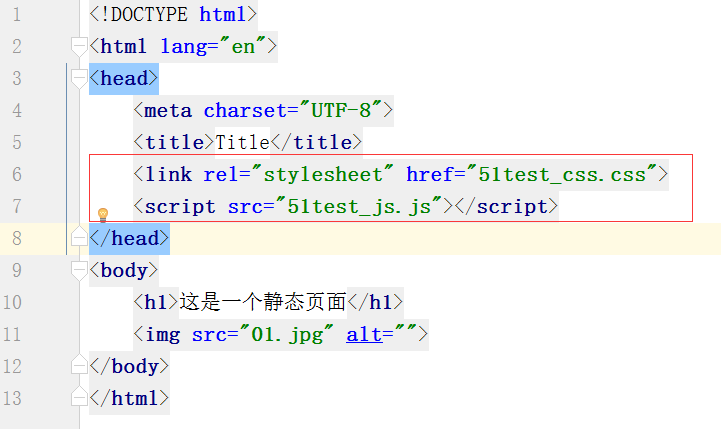


加入后，再去加载静态页面：



为什么？ 对于img， 我们并没有读取...

再进一步看一下，对于js, css使用外部引入：



再去载入静态页面：



**如何解决上面的问题：**

1，如何获取请求信息

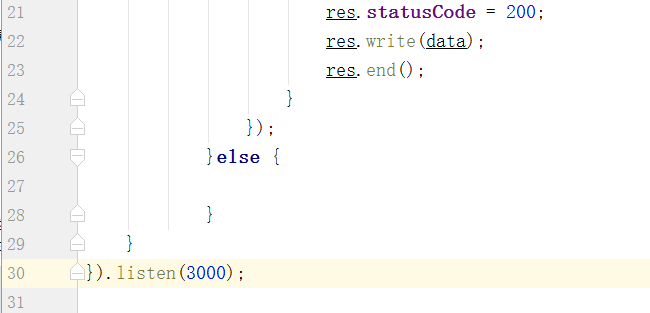
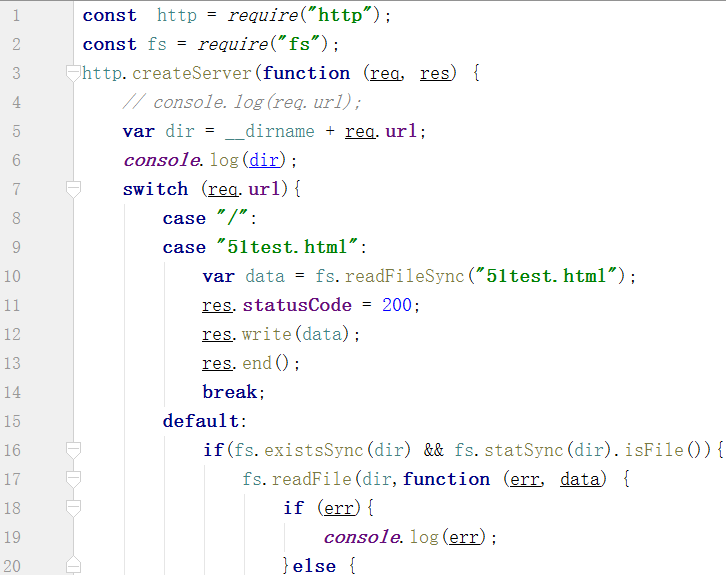
2，需要得到资源的位置



最终实现：



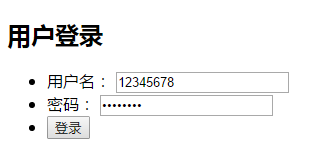
还有一个问题：通常我们访问一个网站是直接通过一个域名，在域名后面是不加内容，通过下面的代码可以处理：



### 8，post请求及响应

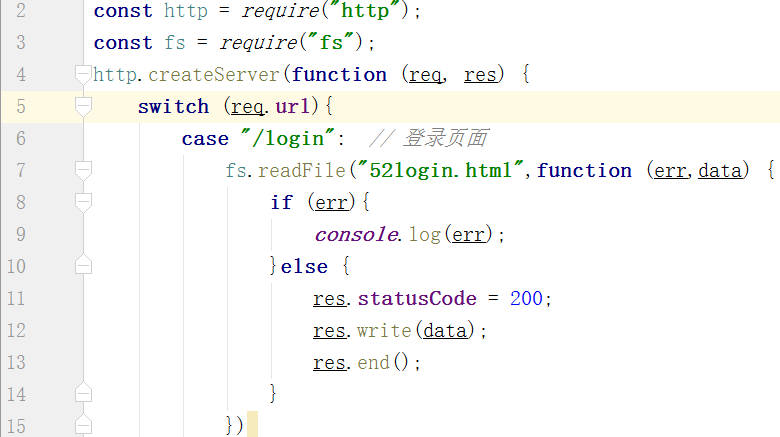
上面说的内容是载入静态的页面，除了静态的页面，还有动态的页面，post请求就是动态页面中的一种。

**需求：有一个表单，输入用户名和密码，当我点击提交按钮时，需要服务器作出处理。**

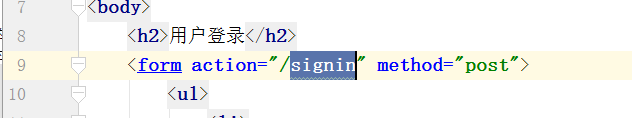


**如何实现：**

1，载入静态页面

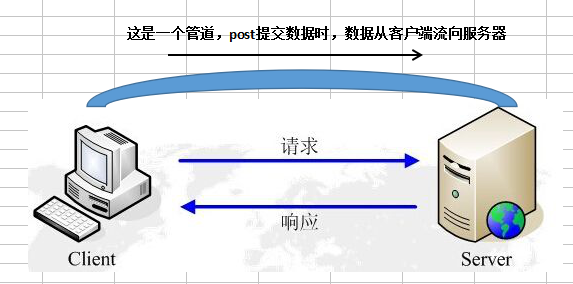


2，在表单中，我们是将数据提交给了sigin来处理：

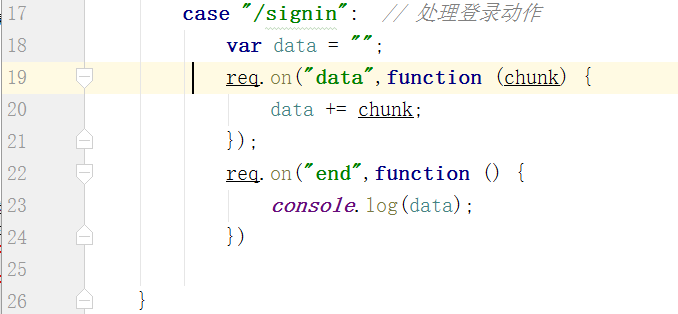


3，在singin中，进行登录动作处理：

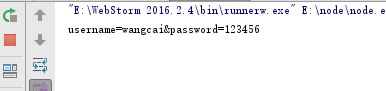
知识点，对于post请求，数据可以想象成，像水一样，通过管道流向服务器。



对于这个请求，我们可以通过req中有两个事件， data, end 我们可以得到数据:

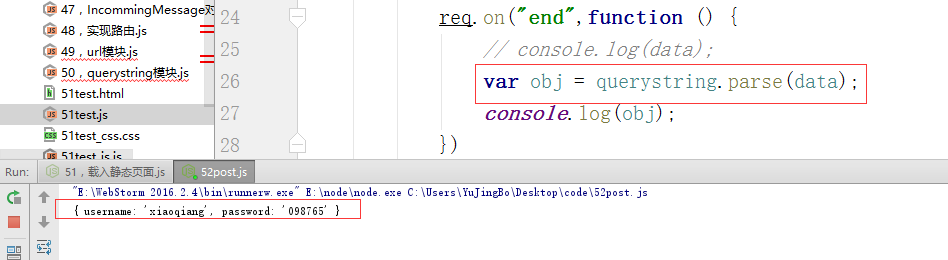


最后，我们就可以拿到用户输入的用户名和密码：



再思考一个问题，上面我们虽然拿到了用户名和密码，但是不太好用，怎么把它转化成一个对象？

使用querystring模块！ 使用如下：

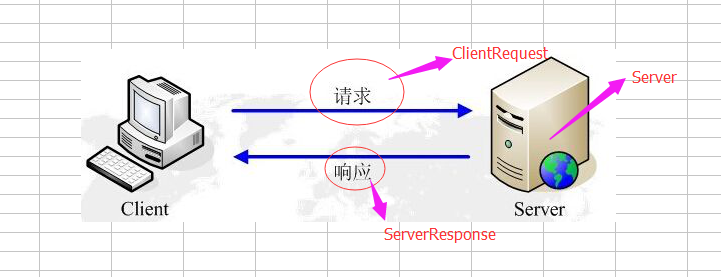


通过上面的操作，我们就可以方便地使用用户名和密码了：



### 9，ClientRequest对象

再回顾一下http模型：



对于ClientRequest对象，一般用的不多，为什么？因为平常我们输入一个网址，点击一个超链接等等，其实就已经发出了一个请求。但是它也有自己的用处，比如写爬虫。

**1，如何创建这个ClientRequest对象？**



**2，这个对象中都有哪些属性和方法？**

write: 把一个正文写入请求

host：请求的服务器域名或IP地址，默认是localhost

**3，如何去发送一个请求？ 通过end()可以发出一个请求。**

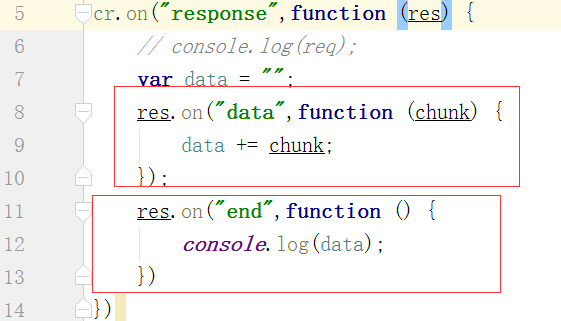


**4，上面的请求已经发送了，怎么得到服务器的响应呢？**

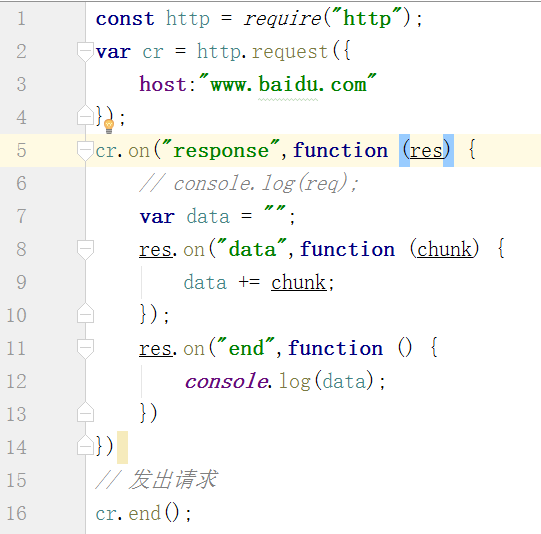
通过response事件，注册response事件如下：



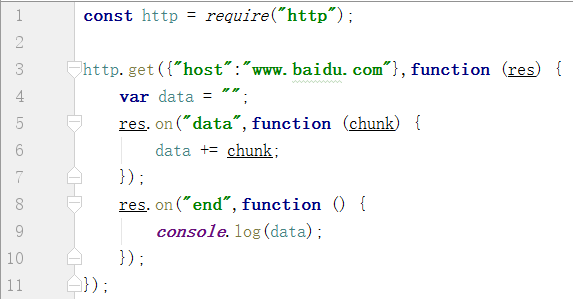
**5，在req中有两个事件，data, end ?**



**6，最终代码：**



**7，上面的写法是为了让大家更好理解，通常会按照下面的写法：**



## 10，爬虫抓取网络图片

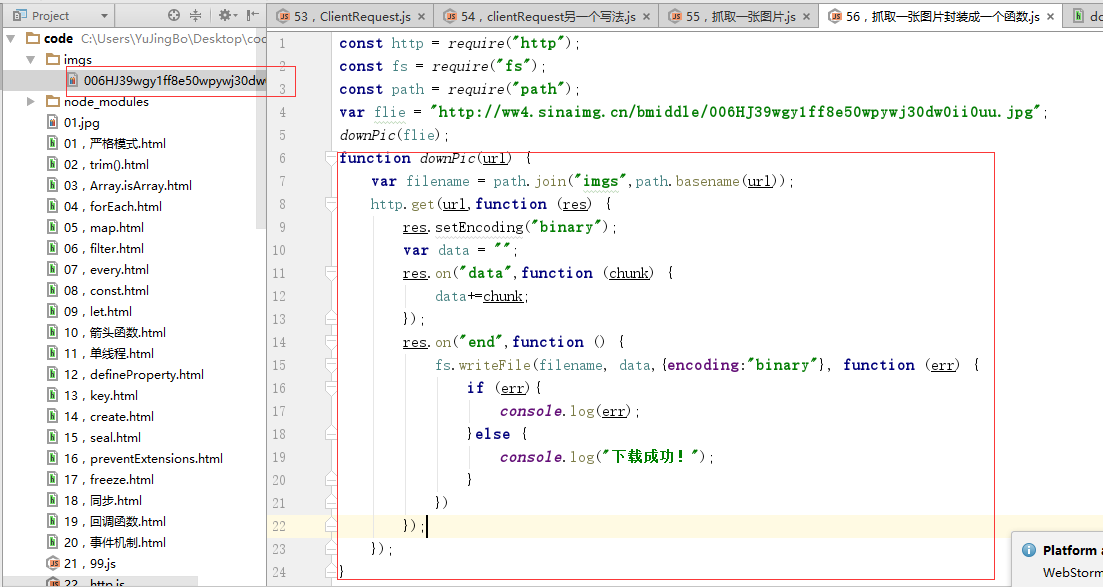
需求： 抓取这个网站上的图片<http://www.dbmeinv.com/>

### 给一张图片的地址，把它抓取下来



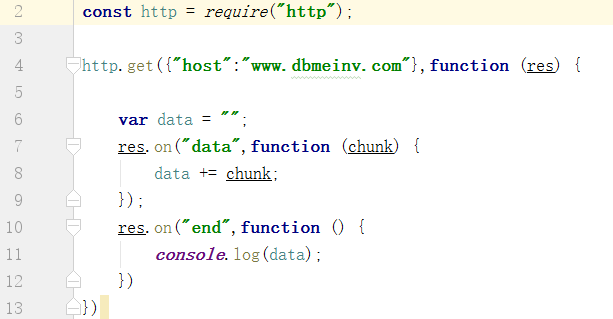
### 给一个网址，把这个网址中的图片都抓取下来

1，把上面的代码封装成一个函数：

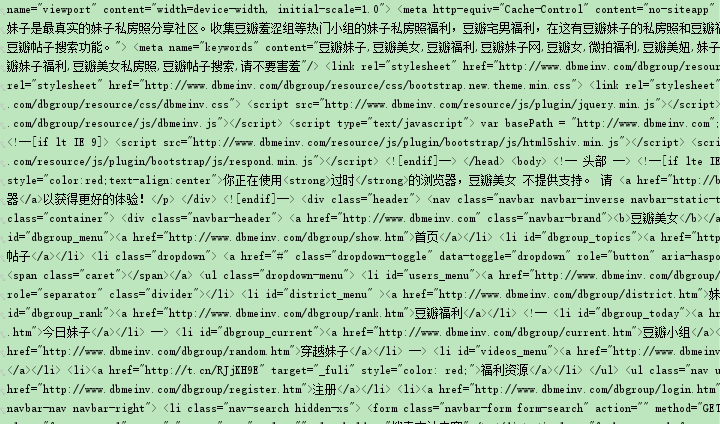


**2，抓取整个网站上面的图片**

第一步，拿到这个网站的源代码



代码如下：



第二步， 把上面代码中所有的src提取出来

对于这人字符串，如何匹配： ‘src="http://ww4.sinaimg.cn/bmiddle/006HJ39wgy1ff8dt4w8u4j30dw0mbae4.jpg" />’



# 六，使用nodejs开发留言板

## 1，npm介绍

npm : node package manger node包管理器

**模块**：一个文件就可以看作是一个模块。

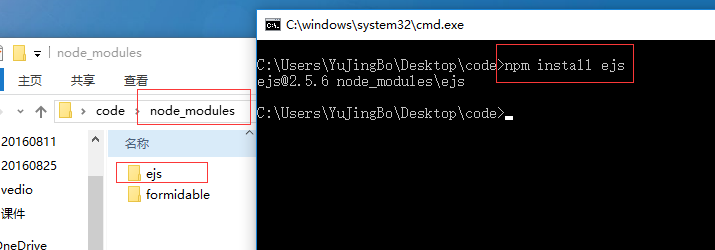
**包**：在项目开发中，一个模块有时候不足以完成某个功能，这个时候需要多个模块共同完成，这多个模块以一定的规范组织在一起，就形成了一个文件夹，也就是包。

npm在我们安装node的时候，它就给安装好了。

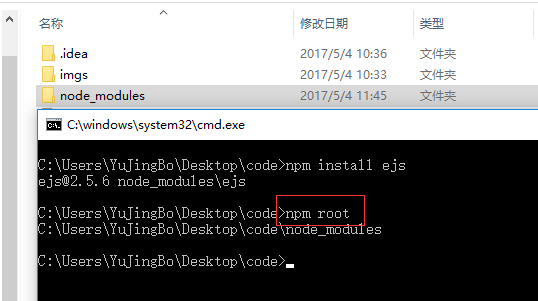
**常用的命令：**

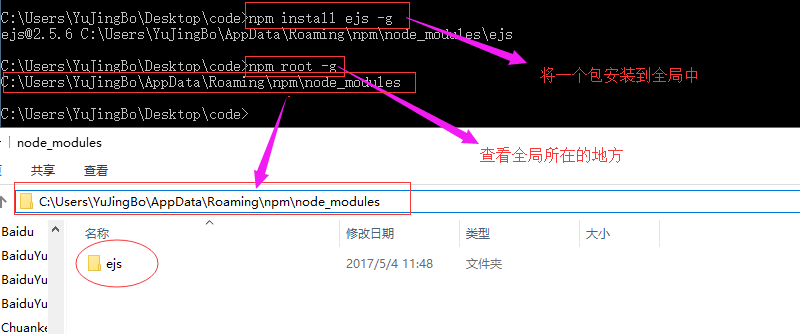
* npm help 帮助
* npm install 包名 本地安装
* npm install 包名 -g 全局安装
* npm uninstall 包名 删除包
* npm update 包名 更新包
* npm root 查看当前包的安装路径
* npm root -g 查看全局包的安装路径

**代码演示安装ejs:**



**代码演示包的安装路径:**





## 2，ejs模板

有前面我们写的页面中，里面的数据都是假的，通常页面中的数据是从服务中拿到的，数据拿到后，需要将数据放到页面中。

ejs模板就是将数据与html模板分离开。

**对于一个完整页面：**

1，数据部分，通常是json

2，模板页面，在模板页面中，可以通过特定的语法，把数据加上去。

**对于特定的语法，不多，有三个：**

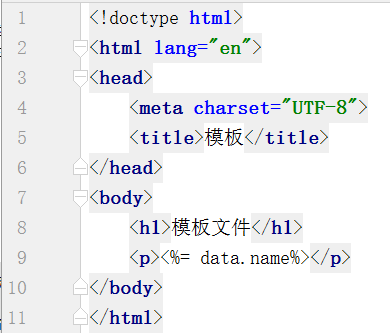
* 执行语句： <% %>
* 显示语句：<%= mes%> 用来显示mes变量
* include语句： <% include 模板文件名 %>

**如何将模板与数据结合在一起？**

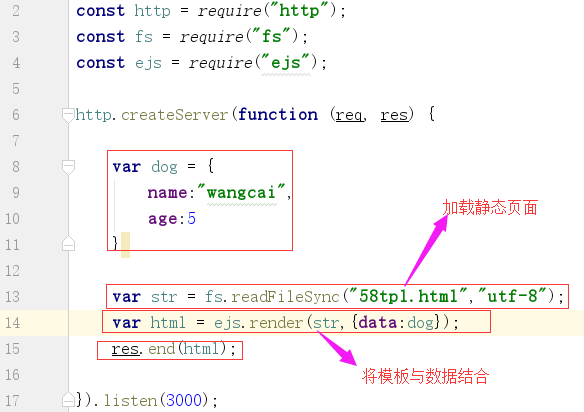
* render, 针对数据
* renderFile, 针对文件

使用代码如下：

模板文件：



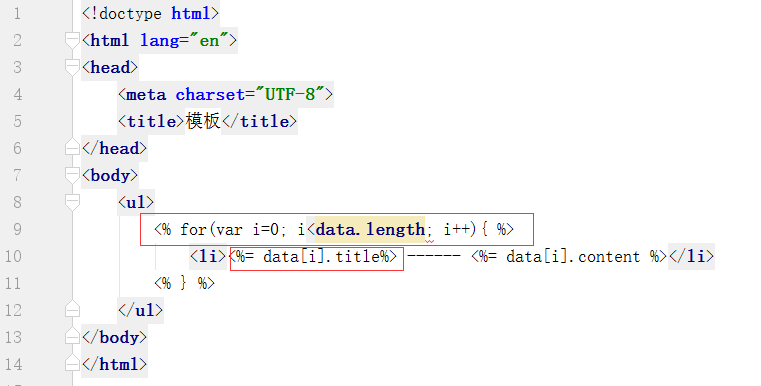
数据文件：



模板文件和数据文件结合后，在模板中，就可以使用数据了，不再使用假数据。

**再来看一个案例：**

**html模板：**



处理：



## 3，需求分析

有两个功能：

1，发表留言

2，显示留言

**保存数据？**

按理说，留言都要保存到数据库中，目前我们没有学习数据库，怎么办？

我们将数据保存到文件中。

**模板技术？**

ejs ....

**创建服务器，载入静态文件，路由....**

## 4，功能实现

看代码！