

The Node.js logo, consisting of the word "NodeJs" in a light blue, sans-serif font. The letters have a subtle 3D effect with a darker blue shadow. The background is a dark blue-grey with diagonal stripes of red and light grey.

# NodeJs

# 目录

COMPANY

01 NodeJs介绍

02 下载和安装nodejs

03 NodeJs一些示例

# 第一部分-引入部分

## Node.js介绍

## 一.(1)先明白啥是服务器

服务器是计算机的一种，它比普通计算机运行更快、负载更高、价格更贵。

服务器作为电子设备，其内部的结构十分的复杂，但与普通的计算机内部结构相差不大，如：cpu、硬盘、内存，系统、系统总线等。

## 一.(2)为什么要学node.js?

我们到目前为止只了解了一小撮前端内容——写写页面，写写交互..但如果想提供24小时不关机、时刻获取数据、返回数据等对数据的服务，前端是实现不了的，它缺少了中间的权限认证——通过服务器的中间处s理对数据库进行访问。但是我们不知道服务端怎么做的、接口怎么做的、数据怎么处理的，我们该怎么入手服务端这一板块的知识呢？

**Node.js作为一个工具，帮助我们接触服务端**

# 一.(3)什么是node.js ?

看一下这个Hello World程序 ( 先不用看懂每一句意思 )

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

## 一.(3)什么是node.js ?

node.js不是一个语言  
不是一个库  
不是一个框架

According to official explanation:

Node.js® is a JavaScript runtime built on Chrome's V8 JavaScript engine.

简单来说，它是一个javascript运行时环境  
——再直白点，**node.js可以解析、执行js代码**

## 一.(3)什么是node.js ?

Node.js 是一个基于 Chrome V8 引擎的 JavaScript 运行环境。Node.js 使用了一个事件驱动、非阻塞式 I/O 的模型，使其轻量又高效。

Node.js 的包管理器 npm，是全球最大的开源库生态系统。  
<https://www.npmjs.com/>  
(可选了解，不过有兴趣深入nodejs学习的应该尽量多了解)



# 一.(3)什么是node.js ?

## Node.js 教程



简单的说 Node.js 就是运行在服务端的 JavaScript。

Node.js 是一个基于Chrome JavaScript 运行时建立的一个平台。

Node.js是一个事件驱动I/O服务端JavaScript环境，基于Google的V8引擎，V8引擎执行Javascript的速度非常快，性能非常好。

## 谁适合阅读本教程？

如果你是一个前端程序员，你不懂得像PHP、Python或Ruby等动态编程语言，然后你想创建自己的服务，那么Node.js是一个非常好的选择。

Node.js 是运行在服务端的 JavaScript，如果你熟悉Javascript，那么你将会很容易的学会Node.js。

当然，如果你是后端程序员，想部署一些高性能的服务，那么学习Node.js也是一个非常好的选择。

## 一.(4)node.js的重要性

以往 , javascript只能由浏览器解析并执行

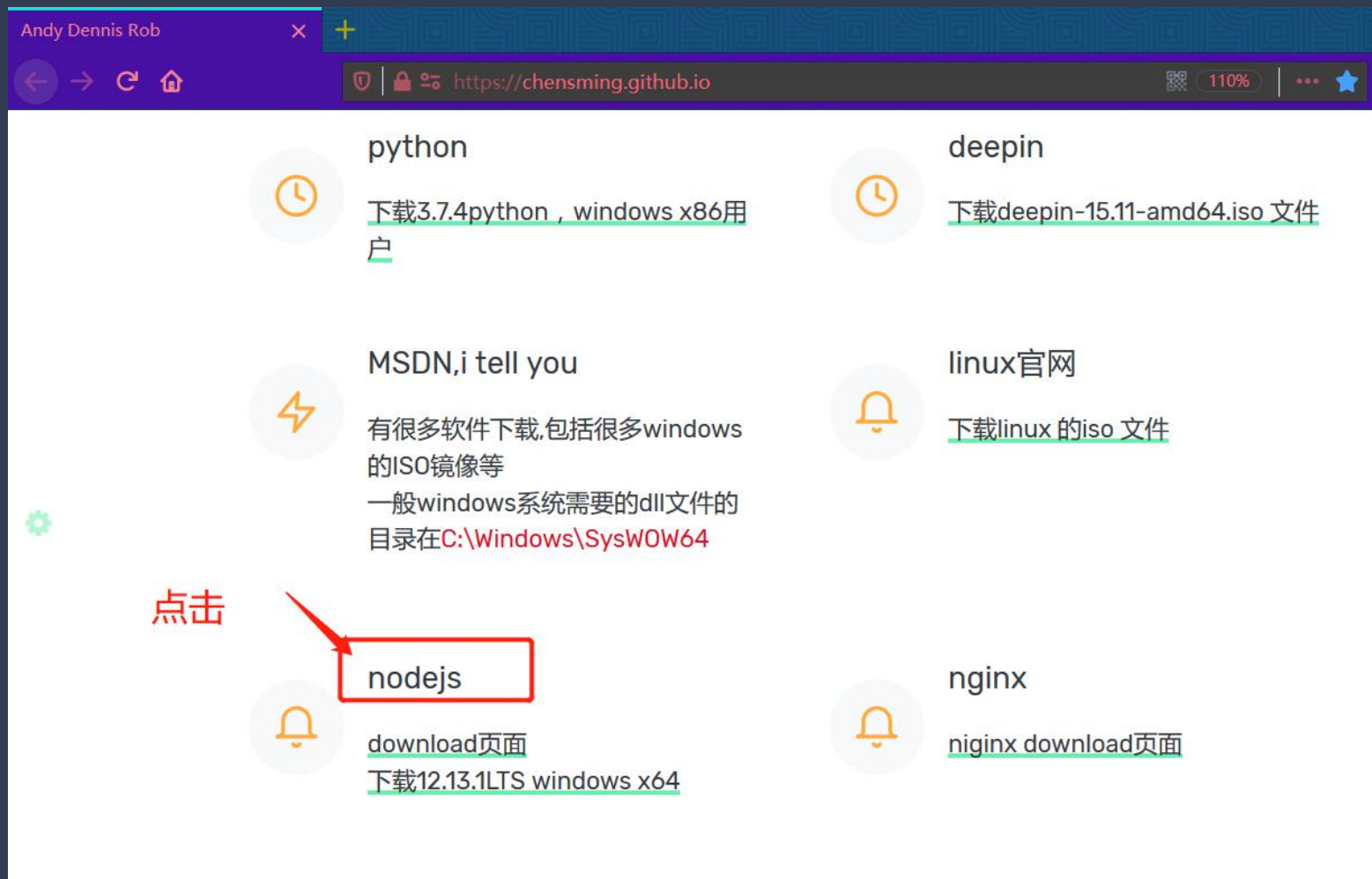
现在 , node.js使得这个流程完全脱离浏览器来进行

# 第二部分

## 下载与安装

### nodejs

# 1.可以去我的主页那里



## 2.也可以直接

nodejs.org

org—organization  
也有些其他常见的域名  
如.io...

[HOME](#)[ABOUT](#)[DOWNLOADS](#)[DOCS](#)[GET INVOLVED](#)[SECURITY](#)[NEWS](#)[FOUNDATION](#)

Node.js® is a JavaScript runtime built on [Chrome's V8 JavaScript engine](#).

## Download for Windows (x64)

**12.13.1 LTS**

Recommended For Most Users

**13.2.0 Current**

Latest Features

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

[Other Downloads](#) | [Changelog](#) | [API Docs](#)

Or have a look at the [Long Term Support \(LTS\) schedule](#).

Sign up for [Node.js Everywhere](#), the official Node.js Monthly Newsletter.

 **LINUX FOUNDATION** [COLLABORATIVE PROJECTS](#)

[Report Node.js issue](#) | [Report website issue](#) | [Get Help](#)

© Node.js Foundation. All Rights Reserved. Portions of this site originally © Joyent.

Node.js is a trademark of Joyent, Inc. and is used with its permission. Please review the [Trademark Guidelines](#) of the Node.js Foundation.

Linux Foundation is a registered trademark of The Linux Foundation.

LTS 是long time support的意思，就是长期可用。

因此建议下载



也可以去我的主页那

<https://chensming.github.io/>

download页面就是到nodejs官网下载页面

下面那个点击直接下载windows x64  
12.13.1LTS版本的nodejs




# download页面

Andy Dennis Rob

Node.js

Download | Node.js

https://nodejs.org/en/download/



HOME | ABOUT | DOWNLOADS | DOCS | GET INVOLVED | SECURITY | NEWS | FOUNDATION


## Downloads


Latest LTS Version: 12.13.1 (includes npm 6.12.1)


Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS  
Recommended For Most Users

Current  
Latest Features

  
Windows Installer  
node-v12.13.1-x64.msi

  
macOS Installer  
node-v12.13.1.pkg

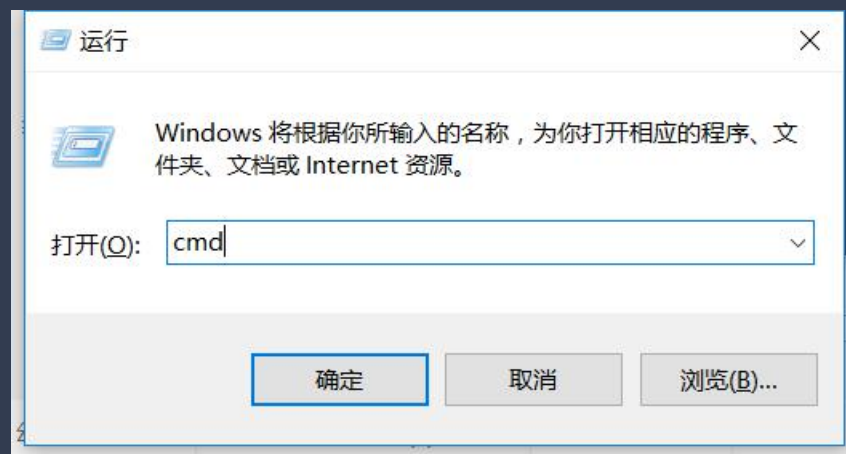
  
Source Code  
node-v12.13.1.tar.gz

Windows Installer (.msi)  
Windows Binary (.zip)  
macOS Installer (.pkg)  
macOS Binary (.tar.gz)  
Linux Binaries (x64)  
Linux Binaries (ARM)  
Source Code

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v12.13.1.tar.gz	

下载完成后点击安装包，跟着安装向导一步步就好了。

安装结束后按win+R，打开运行窗口，输入cmd，回车





在弹出的黑框框（可以改变背景色，比如我选的是绿色的）输入node -v，会输出它的版本号。（我的是久一点的版本）

下面检验环境变量有没有装好

```
C:\Windows\system32\cmd.exe
Microsoft Windows [版本 10.0.15063]
(c) 2017 Microsoft Corporation。保留所有权利。

C:\Users\Dell>node -v
v10.16.2

C:\Users\Dell>
```

## 启动

新建文件  
打开文件夹...  
添加工作区文件夹...

## 最近

BinaryTree D:\Desktop\vscodeProject\DSEChenhy  
DSESingleDocsChenhy D:\Desktop\vscodeProject  
FinalExamReview D:\Desktop\vscodeProject  
Theory D:\Desktop  
backup D:\Desktop\vscodeProject\vscodeproject  
更多... (Ctrl+R)

## 帮助

快捷键速查表(可打印)  
入门视频

## 自定义

### 工具和语言

安装对 JavaScript, Python, PHP, Azure, Docker 和 更多 的支持

### 设置和按键绑定

安装 Vim, Sublime, Atom 和 其他 的设置和快捷键

### 颜色主题

使编辑器和代码呈现你喜欢的外观

## 学习

### 查找并运行所有命令

使用命令面板快速访问和搜索命令 (Ctrl+Shift+P)

### 界面概览

调试控制台 终端

PowerShell  
(C) 2016 Microsoft Corporation. 保留所有权利。

sktop\vscodeProject\nodejs> node -v

sktop\vscodeProject\nodejs> []

1: powershell



3.输入node -v

# 第三部分

NodeJs

一些示例

## 1.全局变量

这里介绍一个函数 一个定时器的功能

```
setTimeout(()=>{  
  console.log("2 seconds have passed");  
},2000)
```

## 运行结果

```
PS D:\Desktop\vscodeProject\nodejs> cd partOne  
PS D:\Desktop\vscodeProject\nodejs\partOne> node app  
2 seconds have passed
```

## 1.全局变量

### ES6写法

```
setTimeout(()=>{  
  console.log("2 seconds have passed");  
},2000)
```

与下面这个效果相同

```
setTimeout(function(){  
  console.log("2 seconds have passed");  
},2000)
```

## 1.全局变量

ctrl + c 结束

//2.setInterval会不停的执行，可以按ctrl+c停止

```
var time = 0;  
setInterval(function () {  
    time += 2;  
    console.log(time+" seconds have passed");  
}, 2000);
```

运行结果

```
PS D:\Desktop\vscodeProject\nodejs\partOne> node app  
2 seconds have passed  
4 seconds have passed  
6 seconds have passed  
8 seconds have passed  
10 seconds have passed
```

## 2.函数

```
function sayHi(){  
    console.log('Hi');  
}  
  
sayHi();
```

## 运行结果

```
PS D:\Desktop\vscodeProject\nodejs\partOne> cd ..  
PS D:\Desktop\vscodeProject\nodejs> cd partTwo  
PS D:\Desktop\vscodeProject\nodejs\partTwo> node app  
Hi
```

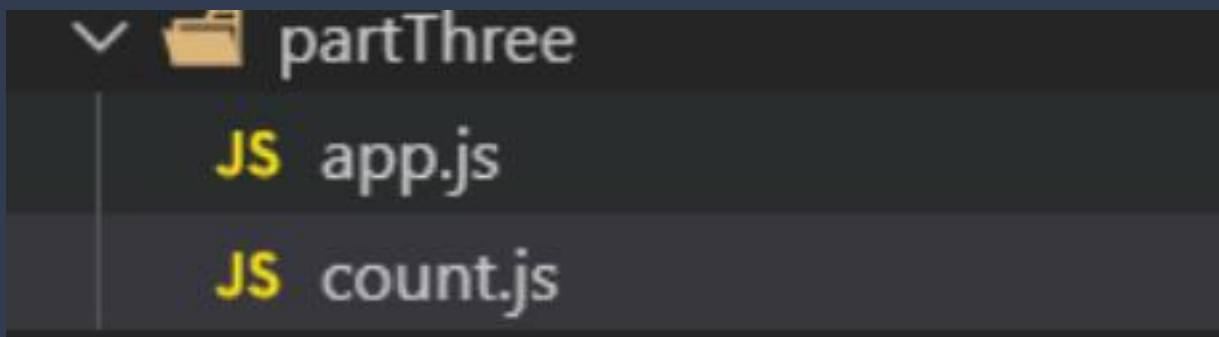
### 3.模块

就是把你写好的一个操作打包好

提供给别人去用

但你要提供一个接口（供别人使用的方法）

这里用到两个文件 文件夹的目录如图





JS count.js X

partThree > JS count.js > adder

```
1  var counter = function (arr) {
2    |   return "There are " + arr.length + " elements in the array";
3  }
4
5  //ES6
6  var adder = function(a,b){
7    |   //`为字母键上面的1与! 键左边的键
8    |   return `the sum of the 2 numbers is ${a+b}`;
9    | }
10
11 var pi = 3.1415926;
12 /*
13 module.exports.counter = counter;
14 module.exports.adder = adder;
15 module.exports.pi = pi;
16 */
17 module.exports = {
18   |   counter : counter,
19   |   adder : adder,
20   |   pi:pi
21 }
22
```

## app.js文件中

JS count.js

JS app.js

×

partThree > JS app.js > ...

```
1  /*
2  partTree: 模块
3  */
4
5
6  var stuff = require('./count');
7
8  console.log(stuff.counter(['ruby', 'nodejs', 'react']));
9  console.log(stuff.adder(3,2));
10 console.log(stuff.pi);
```

### 3.模块

这里有个 函数重载 的用法

其实你们不必想的很复杂，就是nodejs很聪明，  
可以通过你传入的参数个数决定调用哪个函数

运行结果

```
PS D:\Desktop\vscodeProject\nodejs\partThree> node app
There are 3 elements in the array
the sum of the 2 numbers is 5
3.1415926
```

## 4.事件

myEmitter监听事件

```
var events = require('events');

var myEmitter = new events.EventEmitter();
myEmitter.on('someEvent',function(message){
    console.log(message);
})

myEmitter.emit('someEvent','the event was emitted');
```

## 5.fs读写文件 同步和异步

### 同步方式读文件

```
partFive > JS app.js > ...
```

```
1  // 1.  
2  var fs = require('fs');  
3  //readFileSync()为fs的同步方法  
4  var readMe= fs.readFileSync("readMe.txt","utf8");  
5  console.log(readMe);  
6
```

## 同步方式写文件

```
//2.  
var fs = require('fs');  
  
var readMe = fs.readFileSync("readMe.txt","utf8");  
  
fs.writeFileSync("writeMe.txt",readMe);
```

这里从readMe.txt读文件写到writeMe.txt文件里去

```
JS app.js  readMe.txt ×  
partFive > readMe.txt  
1 you read me.
```

```
JS app.js  writeMe.txt ×  
partFive > writeMe.txt  
1 you read me.
```

这里说一个异步方法写函数的例子，详细方法见附件的源代码partFive部分

```
//异步
var fs = require('fs');
var readMe = fs.readFile("readMe.txt", "utf8", function(err, data){
    if (err) return console.error(err);
    console.log(data);
});
```

err : 错误信息，逻辑值为真，  
所以如果程序执行错误的话，会在控制台输出错误信息

data : 是前面执行完后传入到回调函数的信息

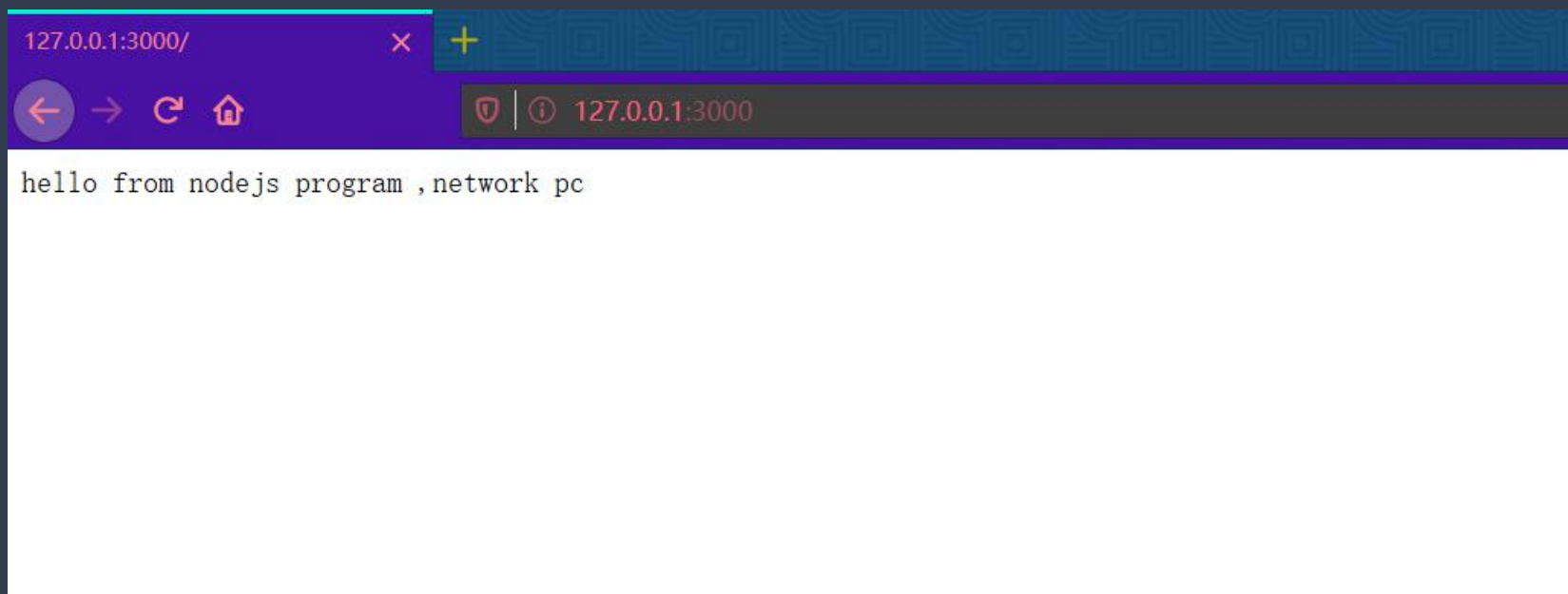


## 8.搭建服务器 text/plain类型的输出

```
5  var http= require('http');
6
7  var server = http.createServer(function(request,response){
8      console.log('Request received');
9      response.writeHead(200,{ 'Content-Type': 'text/plain' });
10     // response.write("hello from out application");
11     // response.end();
12     //以上两行等同下一行no
13
14     response.end("hello from nodejs program ,network pc ");
15 })
16
17 server.listen(3000, '127.0.0.1');
18 //server.listen(4000);
19 console.log('Server started on localhost port 3000');
20
```



node app.js后，在浏览器输入127.0.0.1:3000



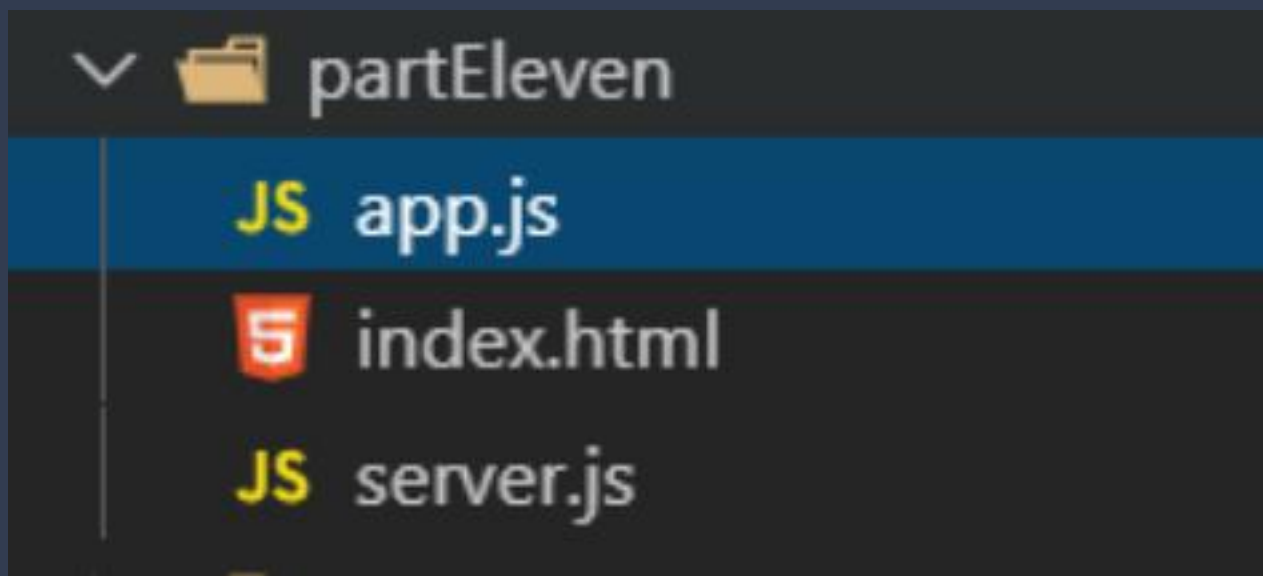
每刷新一次，控制台都会有一次  
Request received的信息输出

```
PS D:\Desktop\vscodeProject\nodejs\partEight> node app
Server started on localhost port 3000
Request received
Request received
Request received
Request received
```

```
7 var server = http.createServer(function(request,response){
8   console.log('Request received');
```

## 11.搭建服务器 提供html的响应功能

这里我们会让index.html能够响应并返回给客户端



JS app.js

JS server.js X

partEleven &gt; JS server.js &gt; ...

```
3  */
4
5  var http = require('http');
6  var fs = require('fs');
7
8  function startServer() {
9
10     var onRequest = function (request, response) {
11         console.log('Request received');
12         response.writeHead(200, {
13             'Content-Type': 'text/html'
14         });
15         var myReadStream = fs.createReadStream(__dirname + '/index.html', 'utf8');
16         myReadStream.pipe(response);
17     }
18
19     var server = http.createServer(onRequest);
20
21     server.listen(3000, '127.0.0.1');
22     //server.listen(4000);
23     console.log('Server started on localhost port 3000');
24 }
25
26 exports.startServer = startServer;
```

引入刚刚写好的server文件,并开启服务器

这里顺带提一下，在控制台关闭服务器按ctrl + c

```
JS app.js ×  
partEleven > JS app.js > ...  
1  var server = require('./server');  
2  
3  server.startServer();
```

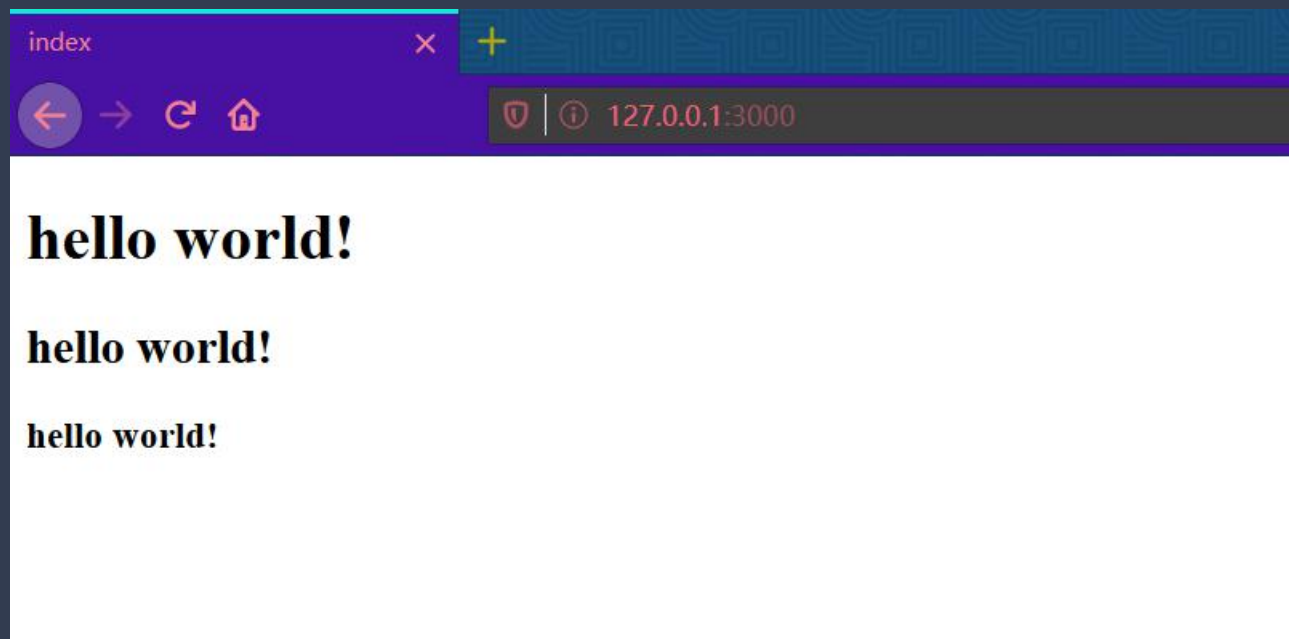
写好一个html文档，叫index.html（默认主页）

index.html ×

partEleven > index.html > ...

```
1  <!DOCTYPE html>
2  <html lang="en">
3
4  <head>
5      <meta charset="UTF-8">
6      <meta name="viewport" content="width=device-width, initial-scale=1.0">
7      <meta http-equiv="X-UA-Compatible" content="ie=edge">
8      <title>index</title>
9  </head>
10
11 <body>
12     <h1>hello world!</h1>
13     <h2>hello world!</h2>
14     <h3>hello world!</h3>
15 </body>
16
17 </html>
```

node app.js后，在浏览器输入127.0.0.1:3000



每刷新一次，控制台都会有一次  
Request received的信息输出

```
PS D:\Desktop\vscodeProject\nodejs> cd partEleven
PS D:\Desktop\vscodeProject\nodejs\partEleven> node app
Server started on localhost port 3000
Request received
Request received
PS D:\Desktop\vscodeProject\nodejs\partEleven> |
```

不讲的内容，课后可自行看看

6.创建和删除目录

7.流和管道

9.web服务器 json格式传送

10.与第11差不多，只不过11以模块形式（更推荐11写的那种方式）

12.路由(运行多个页面，推荐看一下)

13.分成更细的模块

## 13.分成更细的模块

15\*[npm](#)

16\*[npm](#)

15,16不建议看这个程序，不过推荐的链接是可以的。



# 加油！

经过这一节

你们已经

踏进后端的世界