进销存系统实训课件

主讲：汤燕

* 学习线路解析
* Html和Css回顾
* 重点面试题解析
* 模拟面试实践

# 本周安排

1. 还没有交需求分析的同学抓紧交需求分析。
2. 本周五收起实训日志，抓紧完成。（本周实训日志还是按照上课的写，教了什么就写什么，如果什么都没教，就写项目辅导，仅仅辅导情况几乎不可能）。
3. 本周五收项目代码。
4. 项目代码（周五检查）、实训日志、实训需求分析不齐、不全、考勤缺很多都可能影响实训成绩，甚至不及格，没有学分。
5. 本周开始进入技术综合复习阶段。

# 复习阶段

1. 如果面试题有问题单独问，发现公共问题我公开讲。
2. 不定时的选择重点、难点技术重点解析。
3. 模拟面试。
4. 非主观题分析。
5. 我们私下给整个班级定一个小目标。（就业和考研、考公人数、就业最高和平均薪资、技术项目质量和数量等各方面都领跑其他专业）。

要学会学习的方法，独立解决问题，逐渐减少对老师的依赖。老师逐渐过渡到只提供思路的阶段上。

前言

随着互联网的快速发展，研发类的人才越来越稀缺，优秀的研发工程师更是凤毛麟角，企业对人才的争夺也陷入白热化，同时招聘和面试的方式也花样百出，目的就是甄选更加优秀的人才。Web前端工程师是互联网公司必招的岗位，同时也是竞争最为激烈的岗位之一。

# 学习线路

## 高级前端开发工程师学习路线

高级前端到底要学些什么?按照什么样的顺序学？

Html 4.01 =>css2=>html5=>css3（canvas、svg主要用于游戏开发或者绘制报表，不是必须掌握的）=>制作一个企业网站项目（单纯的前端，不包含js效果）=>javascript基础=>jquery框架=>能制作网上商城项目（包含轮播、下拉菜单、页签效果、左侧菜单、商品放大镜等，考虑兼容性）=》bootstrap框架=》nodejs=>面向对象=> http协议=》ajax/json=>微信小程序开发=>angularjs/react/vuejs=>移动端混合开发webapp制作(MUI)=》模块化开发=》GIT工具=》前端工程化、自动化和性能调优。=》准备面试题、简历、作品=》模拟面试=》投简历=》面试=》上班。

## 我们的优势

除了学习了上面这些前端知识外，还学习了linux 操作系统、C语言、java语言、python语言、mysql关系型数据库。

## 我们的劣（lie）势

知识掌握比较肤浅、基础知识不够牢固，不够深入，缺乏工作经验，看起来还不够成熟，自信还不够。

目前大家这个阶段来之不易，尽管同学现阶段大家的水平跟高级前端还有一定的距离，但是万事万物都有本身的发展规律，有些知识能速成，有的过程还是要慢慢经历，慢慢成长。不要眼高手低，要脚踏实地。不要简单的东西看不上眼，觉得没必要做，复杂的东西又下不了手，不会做。

## 面试重点概念理解

自我介绍肯定会问。（外企还要用英文）。

标签、样式、兼容性。这些是最基本的要求。

浏览器内核。

什么是BFC ？

元素居中和自适应布局。

Js如何实现继承（难点）这是面向对象部分的。

模块化开发肯定会问。

Vue框架肯定会问。

闭包原型链肯定会问。

GIT工具必须会。

Node+mongodb+npm

前端自动化。（AMD/CMD/CommonJS模块化开发+项目打包grunt/gulp/webpack+css预编译处理less/sass）。

响应式布局 rem/em/媒体查询技术/bootstrap

Ecmascript/TypeScript相关概念。

TypeScript是JavaScript类型的超集，它可以编译成纯JavaScript。

TypeScript可以在任何浏览器、任何计算机和任何操作系统上运行，并且是开源的。

Ecma国际: 欧洲计算机制造商协会（European Computer Manufacturers Association）。

ECMAScript是一种由Ecma（欧洲计算机制造商协会,英文名称是European Computer Manufacturers Association）制定的一个标准，这个标准在万维网上广泛应用。

JavaScript语言就是按照这个标准做出来的，JavaScript是ECMAScript的具体实现。

2009年发布了ECMAScript第五版，2015年发布了第六版 ECMAScript6.

ECMAScript6是JavaScript语言的下一代标准.

TypeScript是一种由微软开发的自由和开源的编程语言。

TypeScript 最大的特点就是类型化，因此才叫做TypeScript。比起弱类型的JavaScript，类型化的TypeScript显得更加容易维护。

遇到不会回答的问题如何随机应变。

高校教育和职业教育

Web前端目前是一个大前端行业。

以后有新技术怎么办？

学会学习的方法。

每天看50道题。

Html4.01阶段

文档声明 | 标签语法（小写、闭合、不能交叉嵌套） | 布局标签 （table div p span表单元素form input textarea ） | 多媒体标签video | 加粗 斜体 缩小 上划线 下划线 删除线 | 语义化 | 超链接 |

Html5阶段（为移动而生）：

在原有html4.01基础上淘汰了一些平常几乎不用的标签。

增加了一些更加具有语义化的标签。

具体增加了哪些标签呢？面试秘籍：p22页。

Css2和css3阶段

在 HTML 结构的基础上，通过浮动、定位、Flex 弹性盒子等技术进行布局，通过背景、圆角、阴影、渐变等装饰美化网页内容，结合转换（平移、缩放、旋转等2D/3D）、过渡、动画等高级特性增强用户体验，能够编写大型综合性网页。

css选择器 | 盒模型 | 文档流 | 浮动 | 定位 | 字体 | 背景 | 边框 | 伪类  | 圆角 | 阴影 | 过渡 | 透明  | Web字体。

# Javascript服务器端编程-nodejs

## Nodejs出现的背景

现在你打开的100个网页中，有99个都使用了JavaScript。

JavaScript以前只是作为一门前端脚本语言，一门语言的运行，是需要执行环境(浏览器)，就如同我们的Java代码，它需要运行在JVM（虚拟机）中。

JavaScript的执行环境是什么呢？ 是我们的客户端浏览器（IE,FF,Chrorme....）。

Google在2008年年末推出了Chrome浏览器和崭新的JavaScript引擎V8。这是一个为了更快的Web体验而专门制作的更快的JavaScript引擎，V8让Web应用大大提速了。

2009年8月，IT业界制定了CommonJS标准，用于规范化服务器端JavaScript脚本语言。同年，美国程序员Ryan Dahl（瑞恩达尔）决定使用JavaScript做一些更加有意义的事情,给予JavaScript在开发中所处的新定位，使用JavaScript作为**后端(服务器)编程语言**来做开发。

使用Google Chrome开源浏览器**JavaScript V8引擎**为基础搭建了一个新的JavaScript执行环境。并且在V8引擎中基础上添加了**服务器编程语言应该有的功能，如文件系统，模块，包，操作系统API，网络通信等，并且给这个新的平台取名为NodeJs。**



Ryan Dahl（瑞恩达尔）推出了第一个遵循CommonJS标准的服务器端JavaScript脚本语言开发框架——Node.js。

Node.js的一个最重要的特性是通过单线程实现异步处理环境。通过事件环与非阻塞型I/O机制实现服务器端的异步处理。

NodeJs = JavaScript执行环境+JavaScript扩展的功能。

简单理解NodeJs就是一个可以让JavaScript脱离浏览器还能执行的平台,并且这个平台对JavaScript功能进行了增强。

NodeJs(官方)

英文：<https://nodejs.org/en/>

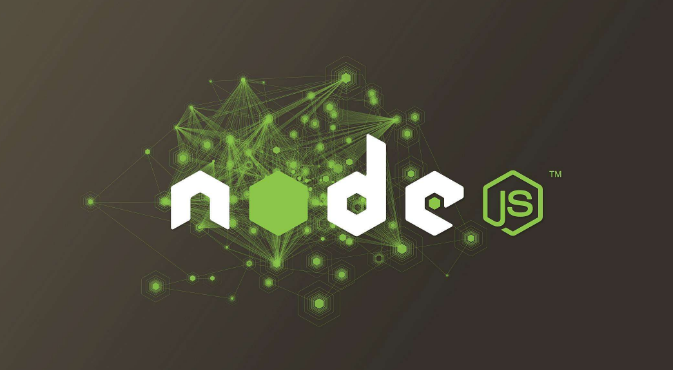
中文：<http://nodejs.cn/>

中文文档：[http://nodeapi.ucdok.com/#/api/](http://nodeapi.ucdok.com/" \l "/api/)

菜鸟文档：<http://www.runoob.com/nodejs/nodejs-tutorial.html>



**Node.js 是一个基于** [Chrome V8](https://developers.google.com/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](https://www.npmjs.com/)**，是全球最大的开源库生态系统。**



NodeJs ：**一个可以运行JavaScript的平台，基于Chrome JavaScript V8引擎，并且对JavaScript语言进行了增强，使JavaScript具有服务器语言开发的能力**（操作文件，读取系统信息，网络等。。。）；

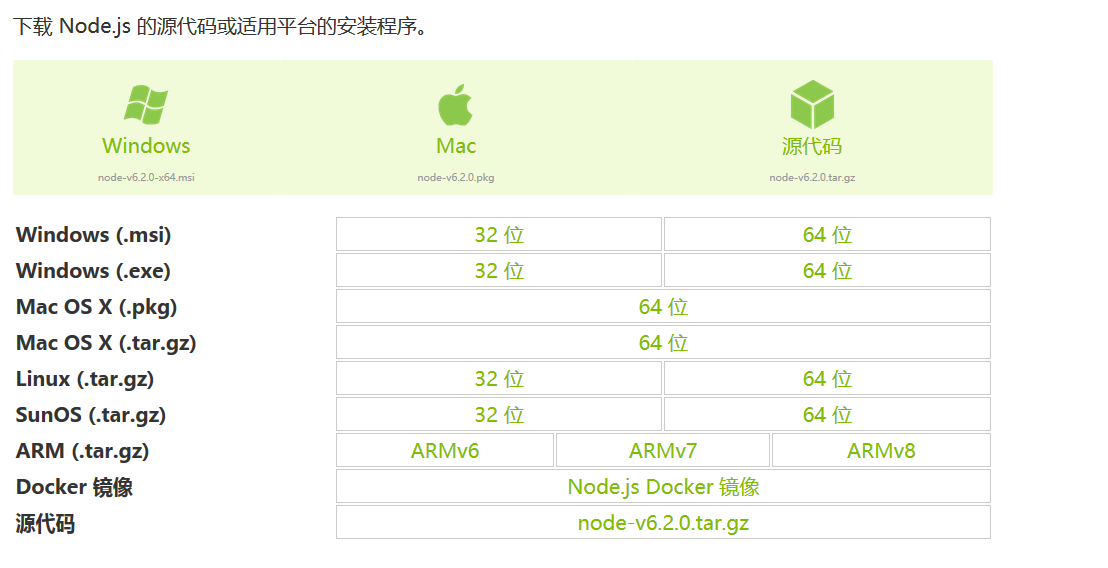
## 下载NodeJs

点击“下载”（**Download**）下载你的电脑操作系统对应的NodeJs版本；

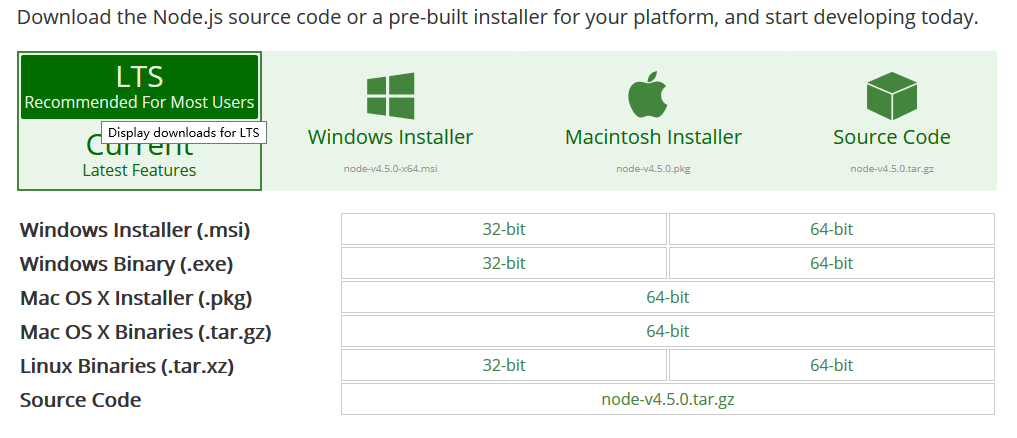
Windows 系统选择.msi; 目前已被都是64位操作系统,极少数是32位；

Mac 系统选择.pkg;

中文地址：<http://nodejs.cn/download/>



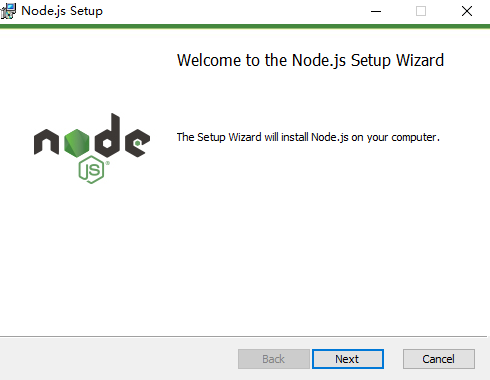
英文地址：<https://nodejs.org/en/download/>



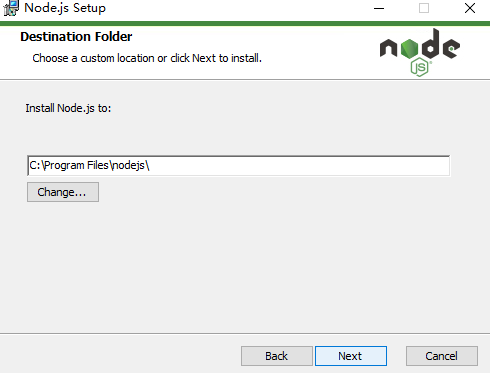
## 安装NodeJs

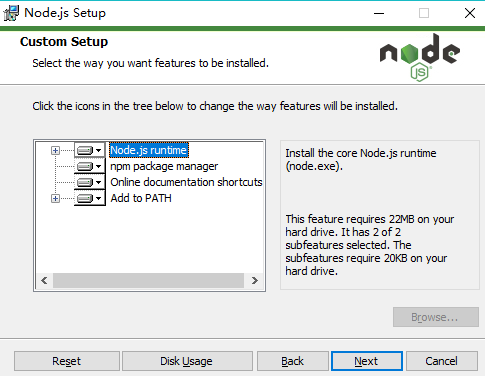
Windows系统下，选择和系统版本匹配的.msi后缀的安装文件。Mac OS X系统

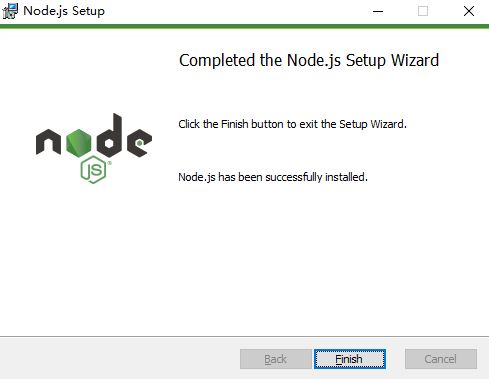
下，选择.pkg后缀的安装文件。默认安装即可。











## 常用dos命令介绍

dir 是英文单词directory(目录）的缩写，主要用来显示一个目录下的文件和子目录。

md是英文make directory（创建目录）的缩写

rd 是英文remove directory--删除目录

cd 是英文change directory(改变目录）的缩写

del del是英文delete（删除）的缩写 删除文件

echo 文本内容 > a.txt 重定向输出，此时创建文本文件a.txt;  
echo 文本内容 >>a.txt 向a.txt文件中追加信息;

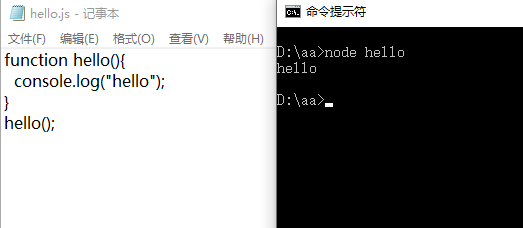
d: 切换到d盘

cd aaa 进入aaa目录

cd .. 进入上级目录

cd \. 回到根目录

## 在node环境下执行一段js代码



# 模块化开发





随着技术的进步，如今前端的代码数量和代码逻辑复杂度都大大超越了以前。

在开发一个复杂的应用程序的时候，我们需要把各个功能拆分、封装到不同的文件，在需要的时候引用该文件。没人会写一个几万行代码的文件，这样在可读性、复用性和维护性上都很差，几乎所有的编程语言都有自己的模块组织方式，比如Java中的包、C#中的程序集等，node.js使用模块和包来组织，其机制实现参照了CommonJS标准。

NodeJs采用模块方式，管理和组织代码，NodeJs的所有的功能都存在每个模块中。

## 什么是模块

最早的时候，所有Javascript代码都写在一个文件里面，只要加载这一个文件就够了。后来，代码越来越多，一个文件不够了，必须分成多个文件，依次加载。下面的网页代码，相信很多人都见过。

<script src="1.js"></script>  
　　<script src="2.js"></script>  
　　<script src="3.js"></script>  
　　<script src="4.js"></script>  
　　<script src="5.js"></script>  
　　<script src="6.js"></script>

这段代码依次加载多个js文件。

## 传统写法的缺点

传统写法有很大的缺点。

首先、加载的时候，浏览器会停止网页渲染，加载文件越多，网页失去响应的时间就会越长；

其次、由于js文件之间存在依赖关系，因此必须严格保证加载顺序（比如上例的1.js要在2.js的前面），依赖性最大的模块一定要放到最后加载，当依赖关系很复杂的时候，代码的编写和维护都会变得困难。

第三、如果在1.js里面定义了一个变量aa,在b.js里面又定义了一个变量aa,变量之间就会冲突。

## 模块化开发解决的问题

（1）实现js文件的异步加载，避免网页失去响应。

（2）管理模块之间的依赖性，便于代码的编写和维护。

（3）避免污染全局变量。

有了模块，我们就可以更方便地使用别人的代码，想要什么功能，就加载什么模块。

但是，这样做有一个前提，**那就是大家必须以同样的方式编写模块**，否则你有你的写法，我有我的写法，岂不是乱了套。

## 三种模块化开发的规范

目前，通行的Javascript模块规范共有三种：AMD \CMD\CommonJS。

### AMD

AMD, (Asynchronous Module Definition=异步模块定义)，这种规范是异步的加载模块，requirejs应用了这一规范,适合客户端浏览器环境。

require.js的诞生，就是为了解决这两个问题：

（1）实现js文件的异步加载，避免网页失去响应；

（2）管理模块之间的依赖性，便于代码的编写和维护。

官网下载地址：<https://requirejs.org/docs/download.html>



下载后，假定把它放在js子目录下面，就可以加载了。

　　<script src="js/require.js"></script>

有人可能会想到，加载这个文件，也可能造成网页失去响应。解决办法有两个，一个是把它放在网页底部加载，另一个是写成下面这样：

　　<script src="js/require.js" defer async="true" ></script>

async属性表明这个文件需要异步加载，避免网页失去响应。IE不支持这个属性，只支持defer，所以把defer也写上。

加载require.js以后，下一步就要加载我们自己的代码了。假定我们自己的代码文件是main.js，也放在js目录下面。那么，只需要写成下面这样就行了：

　　<script src="js/require.js" data-main="js/main"></script>

data-main属性的作用是，指定网页程序的主模块。在上例中，就是js目录下面的main.js，这个文件会第一个被require.js加载。由于require.js默认的文件后缀名是js，所以可以把main.js简写成main。

学习参考资料：

<http://www.ruanyifeng.com/blog/2012/11/require_js.html>

### CMD

CMD, (Common Module Definition), 是国产框架seajs遵循的规范，作者名叫 玉伯。

Seajs被很多网友吐槽，文档不全，路径问题等。

CMD推崇依赖就近，AMD推崇依赖前置。

比较AMD和CMD

|  |
| --- |
| //AMD  define(['./a','./b'], function (a, b) {  //依赖一开始就写好  a.test();  b.test();  });    //CMD  define(function (requie, exports, module) {  //依赖可以就近书写  var a = require('./a');  a.test();  //软依赖  if (status) {  var b = requie('./b');  b.test();  }  }); |

### CommonJs

**CommonJs采用的是同步加载文件方式，只适用于服务端。是服务器端模块化开发的规范。**

在NodeJs平台下进行模块化开发就要遵循**CommonJs规范。**

自己写一个模块

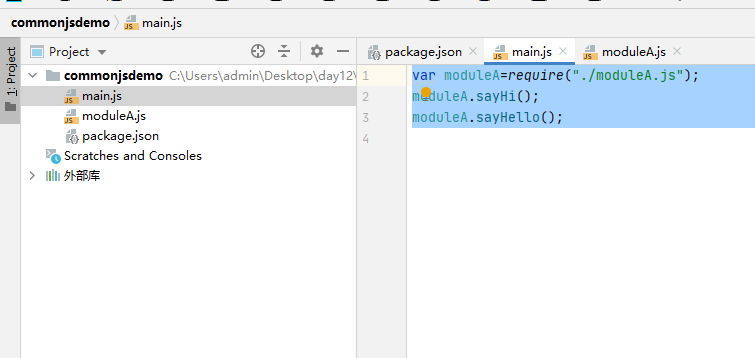
一个JS文件就是一个模块，所以定义一个模块，只需要新建一个JS文件即可。

开发的时候把代码写在很多个模块中，进行模块化开发，形成了很多个js文件。项目开发好了要上线，上线又要求减少http请求次数，需要把这些文件合并成为一个或者少数几个js文件，这个过程就需要项目打包。

打包要做的工作就是合并js,合并css,变量替换、加密js文件等。

### Js在服务器端的模块化开发

一个JS文件就是一个模块，所以定义一个模块，只需要新建一个JS文件即可。



|  |
| --- |
| 定义一个模块  function sayHi(){  console.log("hi"); }  function sayHello(){  console.log("hello"); } *//非常重要，导出模块功能（供外部使用）* exports.sayHi = sayHi; exports.sayHello= sayHello;  调用  var moduleA=*require*("./moduleA.js"); moduleA.sayHi(); moduleA.sayHello(); |

|  |
| --- |
| exports.hello1 = function () {  console.log('Hello World1!');  };  exports.hello2 = function () {  console.log('Hello World2!');  };  exports.hello3 = function () {  console.log('Hello World3!');  };  var aa={  hello1: function () {  console.log('Hello World1!');  },  hello2: function () {  console.log('Hello World1!');  },  hello3: function () {  console.log('Hello World1!');  }  }  正确写法：  module.exports=***aa***;  错误写法：  exports=***aa***; |

### 自己搭建一个http服务器

#### 响应普通文本

|  |
| --- |
| var http = require("http"); *//创建http服务器端，并传入回调函数，在浏览器发送请求到达执行。浏览器客户端每一次请求服务器都会执行一次传入的函数。* var server = http.createServer(function(req,res){  if(req.url.toString()=='/favicon.ico')  {  res.write("");  res.end();  return;  }  *//req:请求对象，包含用户提交的数据和客户端的信息；  //res:响应对象，服务端响应客户端数据，并且可以是动态的；* res.write("hello world!!!");*//服务端响应内容* res.end();*//服务端结束响应* }); *//监听网络端口* server.listen(3000);*//网络端口：计算机中安装了很多软件，而大部分软件需要使用到网络，为了划分网络通道，所以就需要端口。* |

#### 响应网页内容

|  |
| --- |
| *//fs文件模块* var fs = require("fs"); *//url模块，处理地址* var url = require("url"); *//http模块提供http网络交互功能* var http = require("http"); *//创建http服务器端，并传入回调函数，在浏览器发送请求到达执行。浏览器客户端每一次请求服务器都会执行一次传入的函数。* var server = http.createServer(function(req,res){  *//获取url中请求文件的地址* var path = url.parse(req.url).pathname;  *// "/" 为首页* if(path=="/"){  fs.readFile("index.html",function(err,data){  *//HTTP响应网页内容* res.write(data);  res.end();  });  } }); *//监听网络端口* server.listen(3000); |

### 设置响应头信息

纯文本

res.writeHead(200,{*"Content-Type":"text/plain;charset=utf-8"* });

res.write(**"<h1>纯文本</h1>"**);

**res.end();**

HTML

res.writeHead(200,{**"Content-Type"**:**"text/html"**});

res.write(**"<h1>html</h1>"**);

res.end();

JSON

res.writeHead(200,{**"Content-Type"**:**"text/json"**});

res.write(‘{“name”:”二狗”,”msg”:”你变了”}’);

res.end();

图片

res.writeHead(200,{**"Content-Type"**:**"image/png"**});

fs.readFile(“xxx.png”,(err,data){

res.write(data);

res.end();

});

设置Content-Type的作用：告诉浏览器服务端返回的数据是什么格式；

# 勾选和解释重点面试题

第一章和第二章

# 模拟面试

非技术类面试注意问题

每天模拟5个人。（专门抽调不考研和不考公务员的同学）

大家准备好5分钟的自我介绍和面试题。