**前**

**端**

**面**

**试**

**宝**

**典**

# 

[1](#_Toc26545)

[第一部分（基础知识部分） 19](#_Toc10902)

[1.1 基础部分 19](#_Toc18698)

[1.1.1 Html4/5常见问题 19](#_Toc26486)

[1、HTML5 为什么只需要写 <!DOCTYPE HTML>？ 19](#_Toc20806)

[2、html5有哪些新特性、HTML5新标签的浏览器兼容问题如何处理? 19](#_Toc27982)

[3、简述一下你对HTML语义化的理解。 20](#_Toc21179)

[4、XHTML与HTML有什么区别？ 20](#_Toc25618)

[5、Doctype作用? 严格模式与混杂模式如何区分？它们有何意义? 21](#_Toc24738)

[6、介绍一下你对浏览器内核的理解？常见的浏览器内核有哪些？ 21](#_Toc14567)

[7、对WEB标准以及W3C的理解与认识? 22](#_Toc26414)

[8、Iframe的作用？ 22](#_Toc5192)

[1.1.2 Css2/3 23](#_Toc3484)

[1、CSS3新特性有哪些？ 23](#_Toc26218)

[2、描述一下CSS中的渐进增强，优雅降级之间的区别？ 24](#_Toc31120)

[3、对BFC规范的理解有哪些？ 26](#_Toc25500)

[4、什么是CSS reset是什么？normalize.css 是什么？ 27](#_Toc1650)

[5、你遇到过的兼容问题有哪些？如何解决的？ 27](#_Toc22703)

[6、清除浮动有几种方式? 28](#_Toc22704)

[7、要你出一套适应不同分辨率，不同终端的前端实现方案你有什么思路？ 30](#_Toc16306)

[8、移动WEB和响应式有什么分别？ 31](#_Toc13325)

[9、px em rem 取用选择依据？ 31](#_Toc32648)

[10、css权重如何计算？ 32](#_Toc30051)

[11、水平垂直居中的方式有几种？（至少理解两种） 32](#_Toc21787)

[12、css sprites，如何使用？ 35](#_Toc21382)

[1.1.3 Js部分 35](#_Toc6972)

[1、说说你对this的理解？ 35](#_Toc26664)

[2、介绍一下 JS 有哪些内置对象。 35](#_Toc3695)

[3、表单验证传输的什么数据？明文还是暗文==加密？如何加密？是每一次传输数据，都是加密之后才传输吗？ 35](#_Toc18666)

[4、如何实现跨域？ 36](#_Toc21559)

[5、什么是闭包？ 38](#_Toc31240)

[6、什么是原型链？ 38](#_Toc2460)

[7、实现继承的方法有什么？ 38](#_Toc3898)

[8、请列举字符串操作的方法。 41](#_Toc31487)

[9、 AJAX请求数据步骤是什么？传输的数据是用的get还是post？ 41](#_Toc13504)

[10、Javascript内置的常用对象有哪些？并列举该对象常用的方法？（每种记忆三到五个） 43](#_Toc30589)

[11、JS的数据类型有哪些？ 49](#_Toc27062)

[12、javaScript的Dom节点操作创建、插入、删除、复制、查找 49](#_Toc1130)

[13、请说说事件委托机制？这样做有什么好处？ 50](#_Toc15906)

[14、谈谈你对jquery的理解 51](#_Toc6575)

[15、call和apply的区别 52](#_Toc14939)

[16、原生JS的window.onload与Jquery的$(document).ready(function () {})，$(function () {})有什么不同？ 52](#_Toc5647)

[17、在JS的计时器运行原理是怎样的，为什么可以触发计时效果? 计时器是多线程吗? 53](#_Toc697)

[18、JavaScript中的作用域、预解析与变量声明提升？ 53](#_Toc21152)

[19、javascript的typeof返回哪些数据类型 55](#_Toc8619)

[20、简述列举文档对象模型DOM里document的常用的查找访问节点的方法并做简单说明 55](#_Toc14074)

[21、简述创建函数的几种方式 56](#_Toc29612)

[22、Javascript创建对象的几种方式？ 56](#_Toc31720)

[23、js延迟加载的方式有哪些？ 62](#_Toc19372)

[24、 ECMAScript对象的继承结构 67](#_Toc30884)

[25、什么是事件冒泡/捕获？ 68](#_Toc23524)

[26、如何阻止事件冒泡和默认事件? 68](#_Toc30429)

[27、面向对象和类的区别？ 70](#_Toc3635)

[28、简述for in 循环的特点及使用场景？ 70](#_Toc11298)

[29、例举强制类型转换和隐式类型转换？ 71](#_Toc4209)

[30、split() join() 的区别 71](#_Toc11105)

[31、数组方法pop() push() unshift() shift() 72](#_Toc15458)

[32、怎样判断一个JavaScript变量是array还是obiect？  72](#_Toc30620)

[33、解释jsonp的原理，以及为什么不是真正的ajax 76](#_Toc42)

[34、JavaScript的事件流模型都有什么，以及怎么阻止他们？ 76](#_Toc26879)

[35、Javascript 中的垃圾回收机制 77](#_Toc21360)

[36、请解释一下浏览器的同源策略 77](#_Toc24577)

[37、事件绑定和普通事件有什么区别 78](#_Toc26260)

[38、怎么判断一个变量是否null/undefined 78](#_Toc24119)

[39、”==”和“===”的不同 79](#_Toc25768)

[40、Javascript中callee和caller的作用？ 79](#_Toc6905)

[1.1.4 es6部分（必会） 81](#_Toc9734)

[1、 新增声明命令let和const 81](#_Toc6677)

[2、 模板字符串（Template String） 82](#_Toc24757)

[3、 函数的扩展 82](#_Toc15643)

[4、 对象的扩展 83](#_Toc3338)

[5、 for...of  循环 84](#_Toc19465)

[6、 import和export 85](#_Toc10268)

[7、 Promise对象 86](#_Toc6241)

[8、 解构赋值 87](#_Toc2821)

[9、 set数据结构 88](#_Toc13810)

[10、 Spread Operator 展开运算符(...) 89](#_Toc1290)

[1.1.5程序题（JS程序） 90](#_Toc24280)

[1、写出代码对下列数组去重并从大到小排列{5,2,3,6,8,6,5,4,7,1,9} 90](#_Toc6171)

[2、用JavaScript实现冒泡排序。数据为23、45、18、37、92、13、24 91](#_Toc1148)

[3、用js实现随机选取10–100之间的10个数字，存入一个数组，并排序。 91](#_Toc29264)

[4、已知数组var stringArray = [“This”, “is”, “Baidu”, “Campus”]，Alert出”This is Baidu Campus”。 92](#_Toc23964)

[5、已知有字符串foo=”get-element-by-id”,写一个function将其转化成驼峰表示法”getElementById”。 92](#_Toc21581)

[6、下面这段JS输出什么，并简述为什么？ 92](#_Toc428)

[7、有这样一个URL：http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e，请写一段JS程序提取URL中的各个GET参数(参数名和参数个数不确定)，将其按key-value形式返回到一个json结构中，如{a:’1′, b:’2′, c:”, d:’xxx’, e:undefined}。 94](#_Toc21420)

[8、var numberArray = [3,6,2,4,1,5]; （考察基础API） 94](#_Toc25604)

[9、输出今天的日期，以YYYY-MM-DD的方式，比如今天是2014年9月26日，则输出2014-09-26 95](#_Toc8676)

[10、将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的{$id}替换成10，{$nny （使用正则表达式） 95](#_Toc21945)

[11、请写出下面输出的值 95](#_Toc842)

[12、如何垂直居中一个浮动元素？ 96](#_Toc11034)

[13、如何垂直居中一个img? 96](#_Toc16147)

[14、以下js的运行结果是什么，为什么？ 97](#_Toc18974)

[15、看下列代码，将会输出什么?(变量声明提升) 97](#_Toc22666)

[16、把两个数组合并，并删除第二个元素。 97](#_Toc20814)

[17、写一个function，清除字符串前后的空格。（兼容所有浏览器） 98](#_Toc8920)

[18、Javascript中, 以下哪条语句一定会产生运行错误？答案( B ) 98](#_Toc17133)

[19、以下两个变量a和b，a+b的哪个结果是NaN？ 答案( C ) 98](#_Toc3076)

[20、var a=10; b=20; c=4; ++b+c+a++ 以下哪个结果是正确的答案( B ) 98](#_Toc15979)

[21、写出程序运行的结果？ 98](#_Toc8483)

[22、阅读以下代码，请分析出结果： 99](#_Toc25190)

[23、截取字符串abcdefg的efg 99](#_Toc7370)

[24、判断一个字符串中出现次数最多的字符，统计这个次数 99](#_Toc969)

[25、将数字 12345678 转化成 RMB形式 如： 12,345,678  100](#_Toc6714)

[26、加减运算 100](#_Toc2760)

[27、计算字符串字节数： 100](#_Toc6892)

[28、看下列代码,输出什么？ 101](#_Toc3269)

[1.2 前端性能优化 101](#_Toc14684)

[1、你如何对网站的文件和资源进行优化？ 101](#_Toc12903)

[2、一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。 102](#_Toc31554)

[3、图项目中图片处理相关的优化，项目中用到的优化方案，图片大小达到多少的时候选择处理？ 103](#_Toc13166)

[1.3计算机相关术语 103](#_Toc1700)

[1、关于计算机相关术语的介绍 103](#_Toc15871)

[2、http超文本传输协议 104](#_Toc22629)

[3、TCP协议 107](#_Toc3457)

[4、计算机网络的分层体系结构 109](#_Toc20529)

[5、计算机存储器相关知识 110](#_Toc21832)

[6、浏览器 111](#_Toc17663)

[7、服务器 114](#_Toc1432)

[8、线程与进程的区别 115](#_Toc11908)

[9、经典编程算法 118](#_Toc26058)

[10、经典排序算法 119](#_Toc4933)

[11、黑盒、白盒、灰盒测试 119](#_Toc5278)

[12、二叉排序树 121](#_Toc10281)

[1.4 git使用 121](#_Toc18419)

[1、关于git（简介） 121](#_Toc13761)

[2、 git常用命令总结 122](#_Toc12166)

[3、 git命令详解请参考以下链接 123](#_Toc14490)

[第二部分（框架部分） 123](#_Toc14951)

[2.1、流行框架前端热潮知识 123](#_Toc7509)

[2.1.1、 前端框架与类库的区别。 123](#_Toc15676)

[1、总括前端框架与类库的区别 123](#_Toc15968)

[2、jquery和javascript的区别？ 124](#_Toc10135)

[3、谈谈小程序的双向绑定和vue的异同？ 125](#_Toc14461)

[2.1.2、VUE 125](#_Toc10460)

[1、 VUE的适用场景 125](#_Toc5243)

[2、 VUE是什么 126](#_Toc27029)

[3、 Vue父子组件间传值 126](#_Toc30676)

[4、 React 和 Vue 的区别 126](#_Toc25635)

[5、 Vue.js核心思想： 127](#_Toc23300)

[6、 什么是MVVM： 128](#_Toc39)

[7、 MVVM和MVC的区别： 129](#_Toc31706)

[8、 MVVM和jquery的区别： 129](#_Toc28195)

[9、 Vue的优点是什么： 129](#_Toc13073)

[10、 Vue组件之间的传值： 130](#_Toc27930)

[11、 Vue路由之间跳转有哪些： 130](#_Toc20468)

[12、 vue-cli怎样使用自定义组件? 遇到过哪些问题： 130](#_Toc12858)

[13、 vue如何实现按需加载配合webpack设置： 131](#_Toc29145)

[14、 vuex是什么？怎么使用？哪种功能场景使用它？ 132](#_Toc10968)

[15、 vuex有哪几种属性？ 132](#_Toc22196)

[16、 不用Vuex会带来什么问题？ 133](#_Toc21316)

[17、  v-show和v-if指令的共同点和不同点？ 133](#_Toc3865)

[18、  如何让CSS只在当前组件中起作用？ 133](#_Toc28989)

[19、  <keep-alive></keep-alive>的作用是什么? 134](#_Toc19291)

[20、 Vue中引入组件的步骤? 134](#_Toc14096)

[21、 指令v-el的作用是什么? 134](#_Toc21174)

[22、 在Vue中使用插件的步骤? 134](#_Toc15527)

[23、 请列举出3个Vue中常用的生命周期钩子函数?（举两例） 135](#_Toc31900)

[24、 active-class是哪个组件的属性？ 135](#_Toc26129)

[25、 怎么定义vue-router的动态路由以及如何获取传过来的动态参数？ 135](#_Toc1443)

[26、 vue-router有哪几种导航钩子？ 135](#_Toc19276)

[27、 vue生命周期？ 136](#_Toc9379)

[28、 什么是vue生命周期？ 137](#_Toc4950)

[29、 vue生命周期的作用是什么？ 137](#_Toc10757)

[30、 第一次页面加载会触发哪几个钩子？ 137](#_Toc1233)

[31、 DOM 渲染在 哪个周期中就已经完成？ 137](#_Toc598)

[32、 简单描述每个周期具体适合哪些场景？ 137](#_Toc24764)

[33、 说出至少4种vue当中的指令和它的用法？ 138](#_Toc32405)

[34、 vue-loader是什么？使用它的用途有哪些？ 138](#_Toc24)

[35、 scss是什么？ 138](#_Toc6107)

[36、 scss在vue-cli中的安装使用步骤是？ 138](#_Toc13155)

[37、 scss有哪几大特性？ 139](#_Toc7001)

[38、 v-for为什么使用key？ 139](#_Toc12890)

[39、 为什么避免 v-if 和 v-for 用在一起？ 139](#_Toc11358)

[40、 VNode是什么？ 140](#_Toc25897)

[41、 虚拟 DOM是什么？ 140](#_Toc23156)

[42、 vue中利用索引修改数组的时候，页面会跟着同步吗？ 140](#_Toc7940)

[43、 vue首屏加载过慢如何解决？ 140](#_Toc28557)

[44、 路由中如何去除url上的‘#’？ 140](#_Toc32645)

[45、 vue中的单项数据流和双向数据绑定是什意思？ 141](#_Toc12628)

[46、 vue中双向数据绑定的原理是什么？ 141](#_Toc29554)

[47、 为什么vue组件中的data必须是函数？ 141](#_Toc28778)

[48、 你知道webpack中babel、plugin、loader都有什么作用吗？ 141](#_Toc4537)

[49、 keep-alive的作用？ 142](#_Toc32404)

[50、 $route和router的区别？ 142](#_Toc14680)

[2.1.3、React 142](#_Toc27136)

[1、谈谈你对 React 的了解？ 142](#_Toc18017)

[2、什么是 JSX？为什么浏览器无法读取 JSX？（必会） 143](#_Toc29348)

[3、shouldComponentUpdate 是做什么的？（必会） 143](#_Toc17495)

[4、React 性能优化是哪个周期函数？（必会） 144](#_Toc15291)

[5、React 中 keys 的作用是什么？（必会） 144](#_Toc69)

[6、React 中 refs 的作用是什么？（必会） 144](#_Toc8695)

[7、请列举 React 中定义组件的方法？（必会） 144](#_Toc24736)

[8、调用 setState 之后发生了什么？（必会） 145](#_Toc9306)

[9、你怎么理解 redux 的 state 的？（必会） 145](#_Toc22843)

[10、除了在构造函数中绑定 this，还有其它方式吗？（必会） 145](#_Toc31966)

[11、(在构造函数中)调用 super(props)的目的是什么？（必会） 146](#_Toc17362)

[12、简述 flux 思想？（必会） 146](#_Toc12112)

[13、事件在 React 中的处理方式？（必会） 146](#_Toc27128)

[14、列出 Redux 的核心方法？（必会） 147](#_Toc18441)

[15、(组件的)状态(state)和属性(props)之间有何不同？（必会） 147](#_Toc13165)

[16、何为受控组件(controlledcomponent) ？（必会） 147](#_Toc4458)

[17、何为高阶组件(higherordercomponent) ？（必会） 148](#_Toc17445)

[18、React 中组件如何进行数据传值？（必会） 148](#_Toc32330)

[19、解释 Reducer 的作用（必会） 149](#_Toc28553)

[20、redux 有什么缺点（必会） 149](#_Toc14515)

[21、了解 redux 么，说一下 redux（必会） 149](#_Toc32669)

[22、Redux 的工作流程 150](#_Toc13768)

[23、vue 和 React 的区别（必会） 150](#_Toc5013)

[24、React 生命周期函数有哪些？（必会） 151](#_Toc5239)

[25、运行阶段生命周期调用顺序？（必会） 151](#_Toc12646)

[26、React 中 component 和 pureComponent 区别是什么？（必会） 152](#_Toc28809)

[27、什么是无状态组件，与有状态组件的区别？（必会） 152](#_Toc21312)

[28、调用 render 时，DOM 一定会更新吗，为什么？（必会） 153](#_Toc25831)

[29、在哪些生命周期中可以修改组件的 state？（必会） 153](#_Toc29689)

[30、connect()前两个参数是什么？（必会） 154](#_Toc26206)

[31、React-router 的原理（高薪常问） 154](#_Toc11600)

[32、React 的 diff 原理（高薪常问） 155](#_Toc18250)

[33、为什么建议传递给 setState 的参数是一个 callback 而不是一个对 156](#_Toc4447)

[象（高薪常问） 156](#_Toc32193)

[34、redux 中间件原理（高薪常问） 156](#_Toc24745)

[35、React 性能优化的方案（高薪常问） 156](#_Toc26618)

[36、为什么虚拟 DOM 会提高性能?说下他的原理（高薪常问） 157](#_Toc2392)

[37、setState 何时同步何时异步？（高薪常问） 157](#_Toc26882)

[2.1.4、微信小程序 158](#_Toc422)

[1、微信小程序主要目录和文件的作用？ 158](#_Toc18430)

[2、wxml与标准的html的异同？ 159](#_Toc24220)

[3、WXSS和CSS的异同？ 159](#_Toc20100)

[4、怎么封装微信小程序的数据请求的？ 159](#_Toc20852)

[5、小程序页面间有哪些传递数据的方法？ 160](#_Toc18141)

[6、谈谈小程序的生命周期函数？ 160](#_Toc18790)

[7、 简述微信小程序原理？ 161](#_Toc16681)

[8、 路由谈谈原生小程序开发、wepy、mpvue的对比 161](#_Toc9400)

[9、小程序的双向绑定和vue有何不同？ 162](#_Toc32436)

[10、哪些方法可以提高微信小程序的应用速度？ 162](#_Toc29491)

[11、微信小程序的优劣势？ 163](#_Toc7435)

[12、如何实现下拉刷新 163](#_Toc7544)

[2.1.5、NodeJS 163](#_Toc9863)

[1、NodeJS的适用场景 163](#_Toc8135)

[2、NodeJS是什么 164](#_Toc7277)

[3、NodeJS非阻塞I/O模型执行流程 166](#_Toc15416)

[4、node核心内置类库(事件，流，文件，网络等) 166](#_Toc28373)

[5、nodejs中流(stream)的理解 168](#_Toc26482)

[2.1.6、less/sass 预处理 169](#_Toc14577)

[1、 为什么要使用less/sass 预处理器（优点、缺点） 169](#_Toc3171)

[2、 less/sass区别 170](#_Toc3183)

[1、原生app,WEBAPP,混合app的差异  170](#_Toc26257)

[2、 ionic流行开发框架（混合APP） 176](#_Toc11531)

[2.1.7、前端工具介绍或使用方法 177](#_Toc28653)

[1、 前端工程化工具 177](#_Toc25994)

[2、 Webpack 178](#_Toc1623)

[3、 svn和git的区别 179](#_Toc10677)

[4、 Git/GitHub 180](#_Toc20023)

[2.2、如何表述项目 181](#_Toc5394)

[2.2.1、项目描述方式 181](#_Toc17102)

[1、项目功能模块分析 181](#_Toc5220)

[2、 项目如何开展分配 181](#_Toc10803)

[2.2.2、项目口述、面述话术模板 181](#_Toc14401)

[1、基于项目功能模块分析 181](#_Toc4632)

[2、从项目如何开展分配分析 182](#_Toc24891)

[第三部分（非技术部分） 183](#_Toc25544)

[3.1、人事面试 183](#_Toc14961)

[1、请你自我介绍一下你自己？ 183](#_Toc20731)

[2、你觉得你个性上最大的优点是什么？ 184](#_Toc23935)

[3、 说说你最大的缺点？ 184](#_Toc10360)

[4、你对加班的看法？ 185](#_Toc4910)

[5、你对薪资的要求？ 185](#_Toc22685)

[6、你的职业规划？ 186](#_Toc32332)

[7、你还有什么问题要问吗？ 186](#_Toc22336)

[8、如果通过这次面试我们单位录用了你，但工作一段时间却发现你根本不适合这个职位，你怎么办？ 187](#_Toc10896)

[9、在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？ 187](#_Toc8587)

[10、在完如果你的工作出现失误，给本公司造成经济损失，你认为该怎么办？ 187](#_Toc1926)

[11、谈谈你对跳槽的看法？ 188](#_Toc23487)

[12、工作中你难以和同事、上司相处，你该怎么办？ 188](#_Toc22471)

[13、你对于我们公司了解多少？ 189](#_Toc6144)

[14、请说出你选择这份工作的动机？ 189](#_Toc7953)

[15、你最擅长的技术方向是什么？ 189](#_Toc17563)

[16、你能为我们公司带来什么呢？ 189](#_Toc11768)

[17、最能概括你自己的三个词是什么？ 190](#_Toc1381)

[18、为什么要离职? 190](#_Toc8261)

[19、对工作的期望与目标何在？ 190](#_Toc6433)

[20、就你申请的这个职位，你认为你还欠缺什么？ 191](#_Toc2888)

[21、你通常如何处理別人的批评？ 191](#_Toc13076)

[22、怎样对待自己的失败？ 192](#_Toc17308)

[23、什么会让你有成就感？ 192](#_Toc12843)

[24、你为什么愿意到我们公司来工作？ 192](#_Toc7734)

[25、你和别人发生过争执吗？你是怎样解决的？ 192](#_Toc10849)

[26、对这项工作，你有哪些可预见的困难？ 193](#_Toc28334)

[27、如果我录用你，你将怎样开展工作？ 193](#_Toc6727)

[28、在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？与上级意见不一是，你将怎么办？ 194](#_Toc21805)

[29、你工作经验欠缺，如何能胜任这项工作？ 194](#_Toc13787)

[30、您在前一家公司的离职原因是什么？ 195](#_Toc16480)

[31、为了做好你工作份外之事，你该怎样获得他人的支持和帮助？ 196](#_Toc14613)

[32、果你在这次面试中没有被录用，你怎么打算？ 196](#_Toc27737)

[33、谈谈你过去做过的成功案例？(工作中遇到什么问题) 197](#_Toc1999)

[34、如何安排自己的时间？会不会排斥加班？ 197](#_Toc11608)

[35、这个职务的期许？ 197](#_Toc19953)

[36、什么选择我们这家公司？ 198](#_Toc3515)

[37、谈谈如何适应办公室工作的新环境？ 198](#_Toc10766)

[38、工作中学习到了些什么？ 198](#_Toc16564)

[39、除了本公司外，还应聘了哪些公司？ 199](#_Toc32210)

[40、何时可以到职？ 199](#_Toc14200)

# 第一部分（基础知识部分）

## 1.1 基础部分

### 1.1.1 Html4/5常见问题

#### 1、HTML5 为什么只需要写 <!DOCTYPE HTML>？

答、（1）HTML5不基于SGML，因此不需要对DTD进行引用，但是需要DOCTYPE来规范浏览器的行为（让浏览器按照它们应该的方式来运行）；（2）HTML4.01基于SGML，所以需要对DTD进行引用，才能让浏览器知道该文档所使用的文档类型。

#### 2、html5有哪些新特性、HTML5新标签的浏览器兼容问题如何处理?

答、html5新增了以下的几大类元素：

内容元素：article、footer、header、nav、section。

表单控件：calendar、date、time、email、url、search。

控件元素：webworker, websockt, Geolocation。

移出的元素有下列这些：

显现层元素：basefont，big，center，font, s，strike，tt，u。

性能较差元素：frame，frameset，noframes。

HTML5已形成了最终的标准，概括来讲，它主要是关于图像，位置，存储，多任务等功能的增加。

新增的元素有绘画 canvas ，用于媒介回放的 video 和 audio 元素，本地离线存储 localStorage 长期存储数据，浏览器关闭后数据不丢失，而sessionStorage的数据在浏览器关闭后自动删除。

新的技术：canvas,svg,webworker, websocket, Geolocation

#### 3、简述一下你对HTML语义化的理解。

答：（1）HTML语义化让页面的内容结构化，结构更清晰，便于对浏览器、搜索引擎解析；

（2）即使在没有样式CSS的情况下也能以一种文档格式显示，并且是容易阅读的；

（3）搜索引擎的爬虫也依赖于HTML标记来确定上下文和各个关键字的权重，有利于SEO；

（4）使阅读源代码的人更容易将网站分块，便于阅读、维护和理解。

#### 4、XHTML与HTML有什么区别？

答、xhtml:1.所有的标记都必须要有一个相应的结束标记

2.所有标签的元素和属性的名字都必须使用小写

3.所有的XML标记都必须合理嵌套

4.所有的属性必须用引号""括起来

5.把所有<和&特殊符号用编码表示

6.给所有属性赋一个值=""

7.不要在注释内容中使“--”=""

8.图片必须有说明文字="" <="" code=""/>

#### 5、Doctype作用? 严格模式与混杂模式如何区分？它们有何意义?

（1）<!DOCTYPE> 声明位于文档中的最前面，处于 <html> 标签之前。告知浏览器以何种模式来渲染文档。

（2）严格模式的排版和 JS 运作模式是以该浏览器支持的最高标准运行。

（3）在混杂模式中，页面以宽松的向后兼容的方式显示。模拟老式浏览器的行为以防止站点无法工作。

（4）DOCTYPE不存在或格式不正确会导致文档以混杂模式呈现。

#### 6、介绍一下你对浏览器内核的理解？常见的浏览器内核有哪些？

答、要或者说核心的部分是“Rendering Engine”，可大概译为“渲染引擎”，不过我们一般习惯将之称为“浏览器内核”。负责对网页语法的解释（如标准通用标记语言下的一个应用HTML、JavaScript）并渲染（显示）网页。 所以，通常所谓的浏览器内核也就是浏览器所采用的渲染引擎，渲染引擎决定了浏览器如何显示网页的内容以及页面的格式信息。不同的浏览器内核对网页编写语法的解释也有不同，因此同一网页在不同的内核的浏览器里的渲染（显示）效果也可能不同，这也是网页编写者需要在不同内核的浏览器中测试网页显示效果的原因。

常见浏览器内核

Trident内核：IE,MaxThon,TT,The World,360,搜狗浏览器等。[又称MSHTML]

Gecko内核：Netscape6及以上版本，FF,MozillaSuite/SeaMonkey等。

Presto内核：Opera7及以上。 [Opera内核原为：Presto，现为、Blink;]

Webkit内核：Safari,Chrome等。 [ Chrome的：Blink（WebKit的分支）]

EdgeHTML内核：Microsoft Edge。 [此内核其实是从MSHTML fork而来，删掉了几乎所有的IE私有特性]

#### 7、对WEB标准以及W3C的理解与认识?

1. 标签闭合、标签小写、不乱嵌套、提高搜索机器人搜索几率、使用外链css和 js 脚本、结构行为表现的分离，
2. 文件下载与页面速度更快、内容能被更多的用户所访问、内容能被更广泛的设备所访问、更少的代码和组件，
3. 容易维护、改版方便，不需要变动页面内容、提供打印版本而不需要复制内容、提高网站易用性。

#### 8、Iframe的作用？

答、用法：

Iframe是用来在网页中插入第三方页面，早期的页面使用iframe主要是用于导航栏这种很多页面都相同的部分，这样可以在切换页面的时候避免重复下载。

优点：便于修改，模块分离，像一些信息管理系统会用到。

但现在基本上不推荐使用。除非特殊需要，一般不推荐使用。

缺点：（1）iframe的创建比一般的DOM元素慢了1-2个数量级

（2）iframe标签会阻塞页面的加载，如果页面的onload事件不能及时触发，会让用户觉得网页加载很慢，用户体验不好.在Safari和Chrome中可以通过js动态设置iframe的src属性来避免阻塞.

（3）iframe对于SEO不友好，替代方案一般就是动态语言的Incude机制和ajax动态填充内容等.

### 1.1.2 Css2/3

#### 1、CSS3新特性有哪些？

答、1. 颜色：新增RGBA，HSLA模式

     2. 文字阴影（text-shadow）

    3.边框： 圆角（border-radius） 边框阴影： box-shadow

    4. 盒子模型：box-sizing

    5.背景：background-size 设置背景图片的尺寸 background-origin 设置背景图片的原点 background-clip 设置背景图片的裁切区域，以”，”分隔可以设置多背景，用于自适应布局

    6.渐变：linear-gradient、radial-gradient

    7 过渡：transition，可实现动画

    8. 自定义动画

    9. 在CSS3中唯一引入的伪元素是 ：selection.

    10. 媒体查询，多栏布局

    11. border-image

    12.2D转换：transform：translate(x，y) rotate(x，y) skew(x，y) scale(x，y)

    13. 3D转换

#### 2、描述一下CSS中的渐进增强，优雅降级之间的区别？

答、优雅降级和渐进增强印象中是随着css3流出来的一个概念。由于低级浏览器不支持css3，但css3的效果又太优秀不忍放弃，所以在高级浏览中使用css3而低级浏览器只保证最基本的功能。咋一看两个概念差不多，都是在关注不同浏览器下的不同体验，关键的区别是他们所侧重的内容，以及这种不同造成的工作流程的差异。

    举个例子：

a{

    display：block；

    width：200px；

    height： 100px；

    background：aquamarine；

    /\*我就是要用这个新css属性\*/

     transition：all 1s ease 0s；

   /\*可是发现了一些低版本浏览器不支持怎么吧\*/

    /\*往下兼容\*/

    -webkit-transition：all 1s ease 0s；

    -moz-transition：all 1s ease 0s；

    -o-transition： all 1s ease 0s；

   /\*那么通常这样考虑的和这样的侧重点出发的css就是优雅降级\*/

}

a：hover{

    height：200px；

}

/  \*那如果我们的产品要求我们要重低版本的浏览器兼容开始\*/

  a{

 /\*优先考虑低版本的\*/

    -webkit-transition：all 1s ease 0s；

    -moz-transition：all 1s ease 0s；

    -o-transition： all 1s ease 0s；

    /\*高版本的就肯定是渐进渐强\*/

   transition：all 1s ease 0s；

}

“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。

“渐进增强”观点则认为应关注于内容本身。

#### 3、对BFC规范的理解有哪些？

答、 1）定义：

BFC(Block formatting context)直译为"块级格式化上下文"。它是一个独立的渲染区域，只有Block-level box参与， 它规定了内部的Block-level Box如何布局，并且与这个区域外部毫不相干。

布局规则：A. 内部的Box会在垂直方向，一个接一个地放置。

B. Box垂直方向的距离由margin决定。属于同一个BFC的两个相邻Box的margin会发生重叠。

C. 每个元素的margin box的左边， 与包含块border box的左边相接触(对于从左往右的格式化，否则相反)。即使存在浮动也是如此。

D. BFC的区域不会与float box重叠。

E. BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素。反之也如此。

F. 计算BFC的高度时，浮动元素也参与计算。

3）哪些元素会生成BFC：

     A. 根元素

     B. float属性不为none

     C. position为absolute或fixed

     D. display为inline-block， table-cell， table-caption， flex， inline-flex

     F. overflow不为visible

#### 4、什么是CSS reset是什么？normalize.css 是什么？

Reset重置浏览器的css默认属性 ，浏览器的品种不同，样式不同，然后重置，让他们统一。（暴力的，强制的）

* Normalize.css是css reset的替代方案，保护有用的浏览器默认样式而不是完全去掉它们（温和的，根据浏览器特性的）
* 一般化的样式：为大部分HTML元素提供
* 修复浏览器自身的bug并保证各浏览器的一致性
* 优化CSS可用性：用一些小技巧
* 解释代码：用注释和详细的文档来。

#### 5、你遇到过的兼容问题有哪些？如何解决的？

1、png24位的图片在iE6浏览器上出现背景，解决方案是做成PNG8.

2、浏览器默认的margin和padding不同。解决方案是加一个全局的{margin：0;padding：0;}来统一。

3、 IE6双边距bug：块属性标签float后，又有横行的margin情况下，在ie6显示margin比设置的大。浮动ie产生的双倍距离 #itcast{ float：left; width：10px; margin：0 0 0 100px;} 这种情况之下IE会产生20px的距离，解决方案是在float的标签样式控制中加入 \_display：inline;将其转化为行内属性。( “\_” 这个符号只有ie6会识别)

4、 IE下,可以使用获取常规属性的方法来获取自定义属性,

也可以使用getAttribute()获取自定义属性;

Firefox下,只能使用getAttribute()获取自定义属性。

解决方法：统一通过getAttribute()获取自定义属性。

5、 IE下,even对象有x,y属性,但是没有pageX,pageY属性;

Firefox下,event对象有pageX,pageY属性,但是没有x,y属性。

解决方法：（条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。

6、 Chrome 中文界面下默认会将小于 12px 的文本强制按照 12px 显示,

可通过加入 CSS 属性 -webkit-text-size-adjust：none; 解决。

7、超链接访问过后hover样式就不出现了 被点击访问过的超链接样式不在具有hover和active了。解决方法是：改变CSS属性的排列顺序：L-V-H-A 、 a：link {} a：visited {} a：hover {} a：active {}

#### 6、清除浮动有几种方式?

答：1、父级div定义 height

原理：父级div手动定义height，就解决了父级div无法自动获取到高度的问题。 简单、代码少、容易掌握 ，但只适合高度固定的布局.

2、结尾处加空div标签 clear：both

原理：在浮动元素的后面添加一个空div兄弟元素，利用css提高的clear：both清除浮动，让父级div能自动获取到高度 ，如果页面浮动布局多，就要增加很多空div，让人感觉很不好 .

3、父级div定义 伪类：after 和 zoom

/\*清除浮动代码\*/

.clearfix：after{

content：""；

display：block；

visibility：hidden；

height：0；

line-height：0；

clear：both；

}

.clearfix{zoom：1}

原理：IE8以上和非IE浏览器才支持：after，原理和方法2有点类似，zoom(IE专有属性)可解决ie6，ie7浮动问题 ，推荐使用，建议定义公共类，以减少CSS代码。

4、父级div定义 overflow：hidden

超出盒子部分会被隐藏，不推荐使用.

5. 双伪元素法：

.clearfix：before，.clearfix：after {

content： ""；

display： block；

clear：both；

}

.clearfix {

zoom： 1；}

#### 7、要你出一套适应不同分辨率，不同终端的前端实现方案你有什么思路？

答、流式布局：

使用非固定像素来定义网页内容，也就是百分比布局，通过盒子的宽度设置成百分比来根据屏幕的宽度来进行伸缩，不受固定像素的限制，内容向两侧填充。这样的布局方式，就是移动web开发使用的常用布局方式。这样的布局可以适配移动端不同的分辨率设备。

响应式开发：

那么Ethan Marcotte在2010年5月份提出的一个概念，简而言之，就是一个网站能够兼容多个终端。越来越多的设计师也采用了这种设计。

CSS3中的Media Query（媒介查询），通过查询screen的宽度来指定某个宽度区间的网页布局。

超小屏幕（移动设备） 768px以下

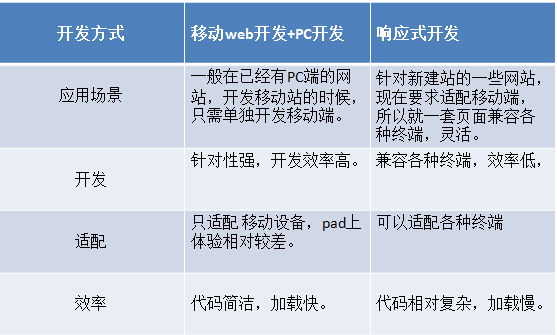
小屏设备 768px-992px

中等屏幕 992px-1200px

宽屏设备 1200px以上

由于响应式开发显得繁琐些，一般使用第三方响应式框架来完成，比如bootstrap来完成一部分工作，当然也可以自己写响应式。

#### 8、移动WEB和响应式有什么分别？



#### 9、px em rem 取用选择依据？

答、 1）px像素（Pixel）。绝对单位。像素px是相对于显示器屏幕分辨率而言的，是一个虚拟长度单位，是计算机系统的数字化图像长度单位，如果px要换算成物理长度，需要指定精度DPI。

2）em是相对长度单位，相对于当前对象内文本的字体尺寸。如当前对行内文本的字体尺寸未被人为设置，则相对于浏览器的默认字体尺寸。它会继承父级元素的字体大小，因此并不是一个固定的值。

3）rem是[CSS3](http://www.html5cn.org/portal.php?mod=list&catid=16" \t "http://www.mamicode.com/_blank)新增的一个相对单位（root em，根em），使用rem为元素设定字体大小时，仍然是相对大小，但相对的只是HTML根元素。

4）区别：IE无法调整那些使用px作为单位的字体大小，而em和rem可以缩放，rem相对的只是HTML根元素。这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。目前，除了IE8及更早版本外，所有浏览器均已支持rem。

#### 10、css权重如何计算？

页面显示样式的优先级取决于其“特殊性”’，特殊性越高，就显示最高的，当特殊性相等时，显示后者

特殊性表述为4个部分：0,0,0,0

一个选择器的特殊性如下:

* 对于选择器是#id的属性值,特殊性值为：0,1,0,0
* 对于属性选择器，class或伪类，特殊性值为：0,0,1,0
* 对于标签选择器或伪元素，特殊性值为：0,0,0,1
* 通配符‘\*’特殊性值为：0,0,0,0
* 内联样式特殊性值为：1,0,0,0

#### 11、水平垂直居中的方式有几种？（至少理解两种）

* 1.absolute + transform：绝对定位加+转换
* <div class="parent">
* <div class="child">Demo</div>
* </div>
* <style>
* .parent {
* position：relative;
* }
* .child {
* position： absolute;
* left： 50%;
* top：50%;
* transform：translate(-50%, -50%);
* }
* </style>

2.inline-block + text-align + table-cell + vertical-align（单元格方式）

<div class="parent">

<div class="child">Demo</div>

</div>

<style>

.parent {

text-align：center;

display：table-cell;

vertical-align：middle;

}

.child {

display： inline-block;

}

</style>

3.flex + justify-content + align-items（弹性模型）

<div class="parent">

<div class="child">Demo</div>

</div>

<style>

.parent {

display： flex;

justify-content： center; /\* 水平居中 \*/

align-items：center; /\*垂直居中\*/

}

<style>

#### 12、css sprites，如何使用？

答、Css 精灵图，把一堆小的图片整合到一张大的图片（png）上，减轻服务器对图片的请求数量。再利用css的“background-image”、“background-repeat”、“background-position”的组合进行背景定位

### 1.1.3 Js部分

#### 1、说说你对this的理解？

答、this是一个关键字，它代表函数运行时，自动生成的一个内部对象，只能在函数内部使用。

1.作为纯粹的函数调用 this指向全局对象

2.作为对象的方法调用 this指向调用对象

3.作为构造函数被调用 this指向新的对象（new会改变this的指向）

4.apply调用 this指向apply方法的第一个参数

#### 2、介绍一下 JS 有哪些内置对象。

Object 是 JavaScript 中所有对象的父对象

数据封装类对象：Object、Array、Boolean、Number、String

其他对象：Function、Argument、Math、Date、RegExp、Error

#### 3、表单验证传输的什么数据？明文还是暗文==加密？如何加密？是每一次传输数据，都是加密之后才传输吗？

答、概述：

GET是从服务器上请求数据，POST是发送数据到服务器。事实上，GET方法是把数据参数队列（query string）加到一个URL上，值和表单是一一对应的。比如说，name=John。在队列里，值和表单用一个&符号分开，空格用+号替换，特殊的符号转换成十六进制的代码。因为这一队列在URL里边，这样队列的参数就能看得到，可以被记录下来，或更改。通常GET方法还限制字符的大小（大概是 256字节 ）。

事实上POST方法可以没有时间限制的传递数据到服务器，用户在浏览器端是看不到这一过程的，所以POST方法比较适合用于发送一个保密的（比如信用 卡号）或者比较大量的数据到服务器。

区别：

Post是允许传输大量数据的方法，而Get方法会将所要传输的数据附在网址后面，然后一起送达服务器，因此传送的数据量就会受到限制，但是执行效率却比Post方法好。

总结：

1、get方式的安全性较Post方式要差些，包含机密信息的话，建议用Post数据提交方式；

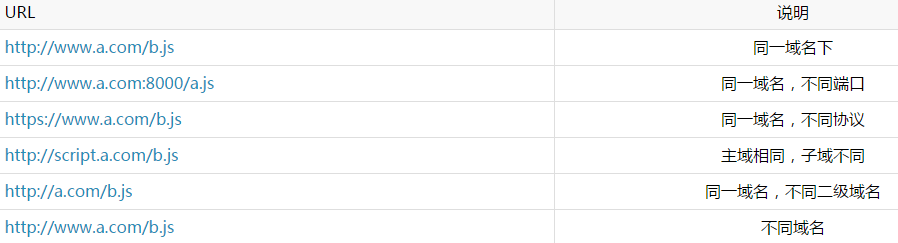
2、在做数据查询时，建议用Get方式；而在做数据添加、修改或删除时，建议用Post方式；

所以：

表达如果是向服务器传输数据(如帐号密码等)都是加密数据(post)， 如果只是单单想要从服务器获得数据或者传输的数据并不重要， 可以直接使用明文方式传输( get )

#### 4、如何实现跨域？

解答：



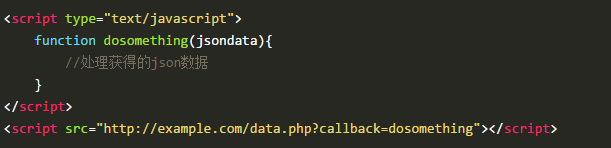
对于端口和协议的不同，只能通过后台来解决。我们要解决的是域名不同的问题。

1.下面是用php进行的设置，“\*”号表示允许任何域向我们的服务端提交请求、header{"Access-Control-Allow-Origin： \*"}

2.JSONP(JSON with Padding 填充式JSON 或参数式JSON)

在js中，我们虽然不能直接用XMLHttpRequest请求不同域上的数据，但是在页面上引入不同域上的js脚本文件却是可以的，jsonp正是利用这个特性来实现的。

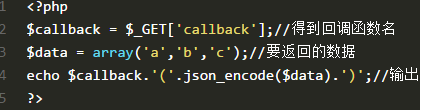
JSONP由两部分组成：回调函数和数据。回调函数是当响应到来时应该在页面中调用的函数，而数据就是传入回调函数中的JSON数据。



首先第一个script标签定义了一个处理数据的函数；

然后第二个script标签载入一个js文件，http：//example.com/data.php 是数据所在地址，但是因为是当做js来引入的，所以http：//example.com/data.php 返回的必须是一个能执行的js文件；

最后js文件载入成功后会执行我们在url参数中指定的函数，并且会把我们需要的json数据作为参数传入。所以php应该是这样的：



**JSONP的优缺点**

**优点：**

它的兼容性更好，在更加古老的浏览器中都可以运行，不需要XMLHttpRequest或ActiveX的支持；

能够直接访问响应文本，支持在浏览器与服务器之间双向通信

**缺点：**

JSONP 是从其他域中加载代码执行。如果其他域不安全，很可能会在响应中夹带一些恶意代码，而此时除了完全放弃JSONP 调用之外，没有办法追究。因此在使用不是你自己运维的Web 服务时，一定得保证它安全可靠。

它只支持GET请求而不支持POST等其它类型的HTTP请求；它只支持跨域HTTP请求这种情况，不能解决不同域的两个页面之间如何进行JavaScript调用的问题

#### 5、什么是闭包？

答、简单的说，作用域是针对变量的，比如我们创建一个函数a1，函数里面又包了一个子函数a2。此时就存在三个作用域：

全局作用域、a1作用域、a2作用域；即全局作用域包含了a1的作用域，a1的作用域包含了a2的作用域。

当a1在查找变量的时候会先从自身的作用域区查找，找不到再到上一级a2的作用域查找，如果还没找到就到全局作用域区查找，这样就形成了一个作用域链。

理解闭包首先要理解，js垃圾回收机制，也就是当一个函数被执行完后，其作用域会被收回，如果形成了闭包，执行完后其作用域就不会被收回。

如果某个函数被他的父函数之外的一个变量引用，就会形成闭包。

闭包的作用，就是保存自己私有的变量，通过提供的接口（方法）给外部使用，但外部不能直接访问该变量。

#### 6、什么是原型链？

答、Javascript是面向对象的，每个实例对象都有一个\_\_proto\_属性，该属性指向它原型对象，这个实例对象的构造函数有一个原型属性prototype，与实例的\_\_proto\_\_属性指向同一个对象。当一个对象在查找一个属性的时，自身没有就会根据\_\_proto\_\_ 向它的原型进行查找，如果都没有，则向它的原型的原型继续查找，直到查到Object.prototype.\_proto\_为null，这样也就形成了原型链。

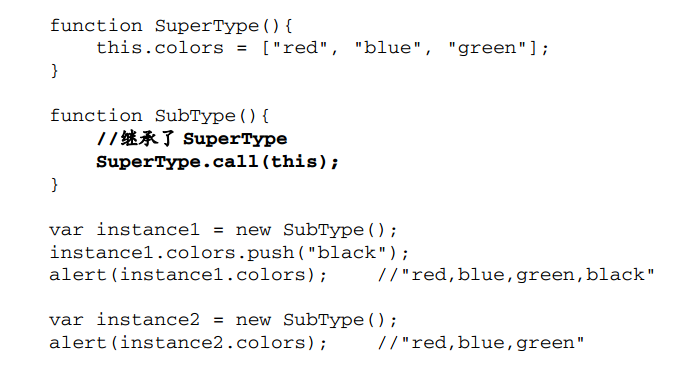
#### 7、实现继承的方法有什么？

答、

（1）借用构造函数。也叫伪造对象或经典继承。

思路：在子类构造函数的内部调用超类型构造函数。可以通过使用apply()和call()方法在新创建的对象上执行构造函数。

缺点：方法都在构造函数中定义，函数的复用就无从谈起。在超类型的原型中定义的方法，对子类而言也是不可见的，结果所有的类型都只能使用构造函数模式。



（2）组合继承。也叫伪经典继承。指的是将原型链和借用构造函数的技术组合到一起，从而发挥二者之长。

思路：使用原型链实现对原型属性属性和方法的继承，通过借用构造函数来实现实例属性的继承。

优点：既通过在原型上定义方法实现了函数复用，又能保证每一个实例都有它自己的数组。

组合继承避免了原型链和借用构造函数的缺陷，融合了他们的优点，成为JavaScript中常用的继承模式。

（3）原型链继承。

思路：借助原型可以基于已有的对象创建对象，同时还不必因此创建自定义类型。

在object()函数内部，先创建一个临时的构造函数，然后将传入的对象作为这个构造函数的原型，最后返回了这个临时类型的一个新实例。

Function object(o){

Function F(){}；

F.prototype=o；

Return new F()；

}

（4）寄生式继承。

思路：创建一个仅用于封装继承过程的函数，该函数在内部以某种方式来增强对象，最后再像真的是它做了所有的工作一样返回对象。

Function createAonter(original){

Var clone=object(original)；//通过调用函数创建一个新对象

Clone.sayHi=function(){ //以某种方式来增强这个对象

Alert(“hi”)；

}

Return clone； //返回这个对象

}

缺点：使用寄生式继承来为对象添加函数，会由于不能做到函数复用二降低效率，这一点和构造函数模式类似。

（5）寄生组合式继承。是JavaScript最常用的继承模式。

思路：通过借用构造函数来继承属性，通过原型链的混成形式来继承方法。

本质上，就是使用寄生式继承来继承超类型的原型，然后再将结果指定给子类型的原型。

开发人员普遍认为寄生组合式继承时引用类型最理想的继承范式。

extend（）方法才用了这样的方式。

#### 8、请列举字符串操作的方法。

charCodeAt方法返回一个整数，代表指定位置字符的Unicode编码；

charAt方法返回指定索引位置处的字符。如果超出有效范围的索引值返回空字符串；

slice方法返回字符串的片段；

substring方法返回位于String对象中指定位置的子字符串。

substr方法返回一个从指定位置开始的指定长度的子字符串。

indexOf方法返回String对象内第一次出现子字符串位置。如果没有找到子字符串，则返回-1；

lastIndexOf方法返回String对象中字符串最后出现的位置。如果没有匹配到子字符串，则返回-1；

search方法返回与正则表达式查找内容匹配的第一个字符串的位置。

concat方法返回字符串值，该值包含了两个或多个提供的字符串的连接；

split将一个字符串分割为子字符串，然后将结果作为字符串数组返回；

#### 9、 AJAX请求数据步骤是什么？传输的数据是用的get还是post？

答：

var xhr；

xhr = new XMLHttpRequest()； //创建一个异步对象

xhr.open("Get"， "test.ashx"， true)；//Get方式括号中的三个参数分别为：1.发送请求的方式 2.y要请求的页面 3.是否异步

//xhr.open("post"，"test.ashx"，true)；

//xhr.setRequestHeader("Content-Type"， "application/x-www-form-urlencoded")； Post方式发送数据

//这个回调函数主要用来检测服务器是否把数据返回给异步对象

xhr.setRequestHeader("If-Modified-Since"，"0")；//设置浏览器不使用缓存

xhr.onreadystatechange = function () {

if (xhr.readystate == 4) {

//readyState属性指出了XMLHttpRequest对象在发送/接收数据过程中所处的几个状态。XMLHttpRequest对象会经历5种不同的状态。

//0、未初始化。对象已经创建，但还未初始化，即还没调用open方法；

//1、已打开。对象已经创建并初始化，但还未调用send方法；

//2、已发送。已经调用send 方法，但该对象正在等待状态码和头的返回；

//3、正在接收。已经接收了部分数据，但还不能使用该对象的属性和方法，因为状态和响应头不完整；

//4、已加载。所有数据接收完毕

if(xhr.status==200){ //检测服务器返回的响应报文的状态码是否为200

alert(xhr.responseText)；//服务器返回的Response数据

//解析服务器返回的json格式的数据

var s=xhr.responseText；

var json=eval("("+s+")")；

alert(jason.data)；

}

}；

}；

xhr.send(null)；//异步对象发送请求

//xhr.send("txtName=roger&txtPwd=123")； 以post方式发送数据

ajax中get和post方式请求数据都是明文的。

#### 10、Javascript内置的常用对象有哪些？并列举该对象常用的方法？（每种记忆三到五个）

**Arguments 函数参数集合**

arguments[ ] 函数参数的数组

Arguments 一个函数的参数和其他属性

Arguments.callee 当前正在运行的函数

Arguments.length 传递给函数的参数的个数

**Array 数组**

length属性 动态获取数组长度

join() 将一个数组转成字符串。返回一个字符串。

reverse() 将数组中各元素颠倒顺序

delete运算符 只能删除数组元素的值，而所占空间还在，总长度没变(arr.length)。

shift() 删除数组中第一个元素，返回删除的那个值，并将长度减1。

pop() 删除数组中最后一个元素，返回删除的那个值，并将长度减1。

unshift() 往数组前面添加一个或多个数组元素，长度要改变。arrObj.unshift(“a” ， “b，“c”)

push() 往数组结尾添加一个或多个数组元素，长度要改变。arrObj.push(“a” ， “b” ， “c”)

concat( ) 连接数组

slice( ) 返回数组的一部分

sort( ) 对数组元素进行排序

splice( ) 插入、删除或替换数组的元素

toLocaleString( ) 把数组转换成局部字符串

toString( ) 将数组转换成一个字符串

**Boolean 布尔对象**

Boolean.toString( ) 将布尔值转换成字符串

Boolean.valueOf( ) Boolean对象的布尔值

**Date 日期时间**

创建Date对象的方法

（1）创建当前(现在)日期对象的实例，不带任何参数

var today = new Date()；

（2）创建指定时间戳的日期对象实例，参数是时间戳。

时间戳：是指某一个时间距离1970年1月1日0时0分0秒，过去了多少毫秒值(1秒=1000毫秒)。

var timer = new Date(10000)； //时间是 1970年1月1日0时0分10秒

（3）指定一个字符串的日期时间信息，参数是一个日期时间字符串

var timer = new Date(“2015/5/25 10：00：00”)；

（4）指定多个数值参数

var timer = new Date(2015+100，4，25，10，20，0)； //顺序为：年、月、日、时、分、秒，年、月、日是必须的。

方法：

Date.getDate( ) 返回一个月中的某一天

Date.getDay( ) 返回一周中的某一天

Date.getFullYear( ) 返回Date对象的年份字段

Date.getHours( ) 返回Date对象的小时字段

Date.getMilliseconds( ) 返回Date对象的毫秒字段

Date.getMinutes( ) 返回Date对象的分钟字段

Date.getMonth( ) 返回Date对象的月份字段

Date.getSeconds( ) 返回Date对象的秒字段

Date.getTime( ) 返回Date对象的毫秒表示

Date.getTimezoneOffset( ) 判断与GMT的时间差

Date.getUTCDate( ) 返回该天是一个月的哪一天(世界时)

Date.getUTCDay( ) 返回该天是星期几(世界时)

Date.getUTCFullYear( ) 返回年份(世界时)

Date.getUTCHours( ) 返回Date对象的小时字段(世界时)

Date.getUTCMilliseconds( ) 返回Date对象的毫秒字段(世界时)

Date.getUTCMinutes( ) 返回Date对象的分钟字段(世界时)

Date.getUTCMonth( ) 返回Date对象的月份(世界时)

Date.getUTCSeconds( ) 返回Date对象的秒字段(世界时)

Date.getYear( ) 返回Date对象的年份字段(世界时)

Date.parse( ) 解析日期/时间字符串

Date.setDate( ) 设置一个月的某一天

Date.setFullYear( ) 设置年份，也可以设置月份和天

Date.setHours( ) 设置Date对象的小时字段、分钟字段、秒字段和毫秒字段

Date.setMilliseconds( ) 设置Date对象的毫秒字段

Date.setMinutes( ) 设置Date对象的分钟字段和秒字段

Date.setMonth( ) 设置Date对象的月份字段和天字段

Date.setSeconds( ) 设置Date对象的秒字段和毫秒字段

Date.setTime( ) 以毫秒设置Date对象

Date.setUTCDate( ) 设置一个月中的某一天(世界时)

Date.setUTCFullYear( ) 设置年份、月份和天(世界时)

Date.setUTCHours( ) 设置Date对象的小时字段、分钟字段、秒字段和毫秒字段(世界时)

Date.setUTCMilliseconds( ) 设置Date对象的毫秒字段(世界时)

Date.setUTCMinutes( ) 设置Date对象的分钟字段和秒字段(世界时)

Date.setUTCMonth( ) 设置Date对象的月份字段和天数字段(世界时)

Date.setUTCSeconds( ) 设置Date对象的秒字段和毫秒字段(世界时)

Date.setYear( ) 设置Date对象的年份字段

Date.toDateString( ) 返回Date对象日期部分作为字符串

Date.toGMTString( ) 将Date转换为世界时字符串

Date.toLocaleDateString( ) 回Date对象的日期部分作为本地已格式化的字符

串

Date.toLocaleString( ) 将Date转换为本地已格式化的字符串

Date.toLocaleTimeString( ) 返回Date对象的时间部分作为本地已格式化的字符串

Date.toString( ) 将Date转换为字符串

Date.toTimeString( ) 返回Date对象日期部分作为字符串

Date.toUTCString( ) 将Date转换为字符串(世界时)

Date.UTC( ) 将Date规范转换成毫秒数

Date.valueOf( ) 将Date转换成毫秒表示

**Error 异常对象**

Error.message 可以读取的错误消息

Error.name 错误的类型

Error.toString( ) 把Error对象转换成字符串

EvalError 在不正确使用eval()时抛出

SyntaxError 抛出该错误用来通知语法错误

RangeError 在数字超出合法范围时抛出

ReferenceError 在读取不存在的变量时抛出

TypeError 当一个值的类型错误时，抛出该异常

URIError 由URl的编码和解码方法抛出

Function 函数构造器

Function.apply( ) 将函数作为一个对象的方法调用

Function.arguments[] 传递给函数的参数

Function.call( ) 将函数作为对象的方法调用

Function.caller 调用当前函数的函数

Function.length 已声明的参数的个数

Function.prototype 对象类的原型

Function.toString( ) 把函数转换成字符串

**Math 数学对象**

Math对象是一个静态对象

Math.PI 圆周率。

Math.abs() 绝对值。

Math.ceil() 向上取整(整数加1，小数去掉)。

Math.floor() 向下取整(直接去掉小数)。

Math.round() 四舍五入。

Math.pow(x，y) 求x的y次方。

Math.sqrt() 求平方根。

Number 数值对象

Number.MAX\_VALUE 最大数值

Number.MIN\_VALUE 最小数值

Number.NaN 特殊的非数字值

Number.NEGATIVE\_INFINITY 负无穷大

Number.POSITIVE\_INFINITY 正无穷大

Number.toExponential( ) 用指数计数法格式化数字

Number.toFixed( ) 采用定点计数法格式化数字

Number.toLocaleString( ) 把数字转换成本地格式的字符串

Number.toPrecision( ) 格式化数字的有效位

Number.toString( ) 将—个数字转换成字符串

Number.valueOf( ) 返回原始数值

**Object 基础对象**

Object 含有所有JavaScript对象的特性的超类

Object.constructor 对象的构造函数

Object.hasOwnProperty( ) 检查属性是否被继承

Object.isPrototypeOf( ) 一个对象是否是另一个对象的原型

Object.propertyIsEnumerable( ) 是否可以通过for/in循环看到属性

Object.toLocaleString( ) 返回对象的本地字符串表示

Object.toString( ) 定义一个对象的字符串表示

Object.valueOf( ) 指定对象的原始值

**RegExp 正则表达式对象**

RegExp.exec( ) 通用的匹配模式

RegExp.global 正则表达式是否全局匹配

RegExp.ignoreCase 正则表达式是否区分大小写

RegExp.lastIndex 下次匹配的起始位置

RegExp.source 正则表达式的文本

RegExp.test( ) 检测一个字符串是否匹配某个模式

RegExp.toString( ) 把正则表达式转换成字符串

**String 字符串对象**

Length 获取字符串的长度。如：var len = strObj.length

toLowerCase() 将字符串中的字母转成全小写。如：strObj.toLowerCase()

toUpperCase() 将字符串中的字母转成全大写。如：strObj.toUpperCase()

charAt(index) 返回指定下标位置的一个字符。如果没有找到，则返回空字符串。

substr() 在原始字符串，返回一个子字符串

substring() 在原始字符串，返回一个子字符串。

区别：'''

“abcdefgh”.substring(2，3) = “c”

“abcdefgh”.substr(2，3) = “cde”

'''

split() 将一个字符串转成数组。

charCodeAt( ) 返回字符串中的第n个字符的代码

concat( ) 连接字符串

fromCharCode( ) 从字符编码创建—个字符串

indexOf( ) 返回一个子字符串在原始字符串中的索引值(查找顺序从左往右查找)。如果没有找到，则返回-1。

lastIndexOf( ) 从后向前检索一个字符串

localeCompare( ) 用本地特定的顺序来比较两个字符串

match( ) 找到一个或多个正则表达式的匹配

replace( ) 替换一个与正则表达式匹配的子串

search( ) 检索与正则表达式相匹配的子串

slice( ) 抽取一个子串

toLocaleLowerCase( ) 把字符串转换小写

toLocaleUpperCase( ) 将字符串转换成大写

toLowerCase( ) 将字符串转换成小写

toString( ) 返回字符串

toUpperCase( ) 将字符串转换成大写

valueOf( ) 返回字符串

#### 11、JS的数据类型有哪些？

简单数据类型：Undefined、Null、Boolean、Number和String。

复杂数据类型：Object

#### 12、javaScript的Dom节点操作创建、插入、删除、复制、查找

一、创建节点、追加节点

1. createElement（标签名）创建一个元素节点（具体的一个元素）。
2. createTextNode（节点文本内容）创建一个文本节点

3、createDocumentFragment()    //创建一个DOM片段

4、appendChild（节点）追加一个节点。

二、插入节点

1、appendChild（节点）也是一种插入节点的方式，还可以添加已经存在的元素，会将其元素从原来的位置移到新的位置。

2、insertBefore（a,b）是参照节点，意思是a节点会插入b节点的前面。

三、删除、移除节点

1、removeChild(节点) 删除一个节点，用于移除删除一个参数（节点）。其返回的被移除的节点，被移除的节点仍在文档中，只是文档中已没有其位置了。

四、复制节点

cloneNode() 方法，用于复制节点， 接受一个布尔值参数， true 表示深复制（复制节点及其所有子节点）， false 表示浅复制（复制节点本身，不复制子节点）

五、替换节点

1、replaceChild(插入的节点，被替换的节点) ，用于替换节点，接受两个参数，第一参数是要插入的节点，第二个是要被替换的节点。返回的是被替换的节点。

六、查找节点

1、getElementsByTagName()    //通过标签名称

2、 getElementsByName()    //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)

3、getElementById()    //通过元素Id，唯一性

#### 13、请说说事件委托机制？这样做有什么好处？

答、事件委托，就是某个事件本来该自己干的，但是自己不干，交给别人来干。就叫事件委托。打个比方：一个button对象，本来自己需要监控自身的点击事件，但是自己不来监控这个点击事件，让自己的父节点来监控自己的点击事件。

好处：

A，提高性能：列如，当有很多li同时需要注册事件的时候，如果使用传统方法来注册事件的话，需要给每一个li注册事件。然而如果使用委托事件的话，就只需要将事件委托给该一个元素即可。这样就能提高性能。

B，新添加的元素还会有之前的事件；

#### 14、谈谈你对jquery的理解

JQuery是继[prototype](http://baike.baidu.com/view/1217697.htm" \t "http://baike.baidu.com/_blank)之后又一个优秀的[Javascript](http://baike.baidu.com/view/16168.htm" \t "http://baike.baidu.com/_blank)库。它是轻量级的js库 ，它兼容[CSS3](http://baike.baidu.com/view/1713027.htm" \t "http://baike.baidu.com/_blank)，还兼容各种浏览器（[IE](http://baike.baidu.com/subview/703/10271308.htm" \t "http://baike.baidu.com/_blank) 6.0+， [FF](http://baike.baidu.com/subview/31049/5089479.htm" \t "http://baike.baidu.com/_blank)1.5+， [Safari](http://baike.baidu.com/subview/110484/5036395.htm" \t "http://baike.baidu.com/_blank) 2.0+， [Opera](http://baike.baidu.com/subview/10019/7098372.htm" \t "http://baike.baidu.com/_blank) 9.0+），jQuery2.0及后续版本将不再支持IE6/7/8浏览器。jQuery使用户能更方便地处理[HTML](http://baike.baidu.com/view/692.htm" \t "http://baike.baidu.com/_blank)（[标准通用标记语言](http://baike.baidu.com/view/5286041.htm" \t "http://baike.baidu.com/_blank)下的一个应用）、events、实现动画效果，并且方便地为网站提供[AJAX](http://baike.baidu.com/subview/1641/5762264.htm" \t "http://baike.baidu.com/_blank)交互。

jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的[插件](http://baike.baidu.com/view/18979.htm" \t "http://baike.baidu.com/_blank)可供选择。jQuery能够使用户的html页面保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需要定义id即可。

jQuery是一个兼容多浏览器的[javascript](http://baike.baidu.com/view/16168.htm" \t "http://baike.baidu.com/_blank)库，核心理念是write less，do more(写得更少，做得更多)。 jQuery是免费、开源的，使用[MIT](http://baike.baidu.com/subview/74918/8382747.htm" \t "http://baike.baidu.com/_blank)许可协议。jQuery的语法设计可以使开发更加便捷，例如操作[文档](http://baike.baidu.com/view/55621.htm" \t "http://baike.baidu.com/_blank)对象、选择[DOM](http://baike.baidu.com/view/14806.htm" \t "http://baike.baidu.com/_blank)元素、制作动画效果、事件处理、使用[Ajax](http://baike.baidu.com/view/1641.htm" \t "http://baike.baidu.com/_blank)以及其他功能。除此以外，jQuery提供API让开发者编写插件。其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

关于jQuery的内部封装原理

1、为了防止全局变量污染，把jQuery的代码写在一个自调函数中，

2、咱们平常使用的$实际上jQuery对外暴漏的一个工厂函数，

3、而构造函数在jQuery的内部叫init，并且这个构造函数还被添加

到了jQuery的原型中。当我们调用工厂函数的时候 返回的其实是

一个构造函数的实例

4、jQuery为了让第三方能够对其功能进行扩展，所以把工厂函数的原

型与构造函数的原型保持了一致。这样子对外暴漏工厂函数，即可

对原型进行扩展。

#### 15、call和apply的区别

答、 它们的共同之处：都“可以用来代替另一个对象调用一个方法，将一个函数的对象上下文从初始的上下文改变为由 thisObj 指定的新对象。”

它们的不同之处：

Apply：最多只能有两个参数——新this对象和一个数组 argArray。如果给该方法传递多个参数，则把参数都写进这个数组里面，当然，即使只有一个参数，也要写进数组里面。如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj， 并且无法被传递任何参数。

Call：则是直接的参数列表，主要用在js对象各方法互相调用的时候，使当前this实例指针保持一致，或在特殊情况下需要改变this指针。如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

更简单地说，apply和call功能一样，只是传入的参数列表形式不同如： func.call(func1，var1，var2，var3)对应的apply写法为：func.apply(func1，[var1，var2，var3])。

#### 16、原生JS的window.onload与Jquery的$(document).ready(function () {})，$(function () {})有什么不同？

答：

**1.执行时间**

window.onload必须等到页面内包括图片的所有元素加载完毕后才能执行。

$(document).ready()是DOM结构绘制完毕后就执行，不必等到加载完毕。

**2.编写个数不同**

window.onload不能同时编写多个，如果有多个window.onload方法，只会执行一个

$(document).ready()可以同时编写多个，并且都可以得到执行

**3.简化写法**

window.onload没有简化写法

$(document).ready(function(){})可以简写成$(function(){})；

#### 17、在JS的计时器运行原理是怎样的，为什么可以触发计时效果? 计时器是多线程吗?

答：

1. javascript引擎只有一个线程，强迫异步事件排队等待被执行。

2. setTimeout和setInterval本质上不同的地方是他们如何执行异步代码的。

3. 如果一个定时器正在执行的时候被阻塞了，那么它将会被推迟到下一个可能的执 行点，这既是使得延迟时间有可能会超过声明定时器时设置的值。

4. Interval如果有足够的时间来执行（大于制定的延迟），那么它将会无延迟的一个紧接着一个执行。

原理：

计时器通过设定一定的时间段（毫秒）来异步的执行一段代码。因为 Javascript 是一个单线程语言，计时器提供了一种绕过这种语言限制来执行代码的能力。

总结：

计时器是单线程的， 需要等待上一个执行完， 如果上一个没有执行完， 下一个需要延迟执行， 直到上一个执行完。

#### 18、JavaScript中的作用域、预解析与变量声明提升？

**作用域:**

就是变量的有效范围。

如何检测一个变量的作用域:在指定的区域内使用这个变量，如果不报错，说明这个变量的作用域包含此区域。

函数作用域:只有函数能够划分变量的作用域，这种作用域的规则就叫函数作用域。

如果在函数内访问一个变量，优先找局部变量和形参,如果没有找到，去定义该函数的环境中查找，直到全局为止。

在ES6之前，只有函数可以划分变量的作用域，所以在函数的外面无法访问函数内的变量。在ES6之前，没有块级作用域的概念，所以在代码块的外面可以访问代码块内的变量

块级作用域：凡是代码块就可以划分变量的作用域，这种作用域的规则就叫块级作用域。

块级作用域 函数作用域 词法作用域

之间的区别：

1、块级作用域和函数作用域描述的是，什么东西可以划分变量的作用域

2、词法作用域描述的是，变量的查找规则。

之间的关系：

（1）、 块级作用域 包含 函数作用域。

（2）、 词法作用域 与 块级作用域、函数作用域之间没有任何交集，

他们从两个角度描述了作用域的规则。

ES6之前js采用的是函数作用域+词法作用域，ES6 js采用的是块级作用域+词法作用域 。

**预解析：**

在代码整体执行之前，先解析一部分。

预解析之后，代码才会从上往下依次整体执行，但是预解析执行过的代码不会

重复执行。

js预解析干了什么事：js中预解析会把声明部分的代码预先执行。

声明相关的代码可以分为两部分：

1、 变量声明

通过var关键字定义的变量。

2、函数声明

通过function关键字声明的函数

预解析时如果遇到重复的变量声明，那么忽略。

预解析时如果遇到重复的函数声明，保留后面的函数。

预解析时如果遇到变量与函数重名的情况，保留函数。

**变量声明提升：**

使用var关键字定义的变量，被称为变量声明；

函数声明提升的特点是，在函数声明的前面，可以调用这个函数。

#### 19、javascript的typeof返回哪些数据类型

typeof一般判断基本数据类型。是一个操作符而不是函数，圆括号可有可无。

typeof 返回值有：string，number，boolean，undefined，object ，function，

基本数据类型：Boolean、Number、String、Undefined、Null

基本数据类型中数字，字符串，布尔类型返回其对类型

undefined返回undefined

九大内置构造函数及其他所有函数返回function；

其他所有复杂类型对象和null返回object

alert(typeof [1, 2]); //object

    alert(typeof 'leipeng'); //string

    var i = true;

    alert(typeof i); //boolean

    alert(typeof 1); //number

    var a;

    alert(typeof a); //undefined

    function a(){;};

    alert(typeof a) //function

#### 20、简述列举文档对象模型DOM里document的常用的查找访问节点的方法并做简单说明

Document.getElementById 根据元素id查找元素

Document.getElementByName 根据元素name查找元素

Document.getElementTagName 根据指定的元素名查找元素

#### 21、简述创建函数的几种方式

第一种（函数声明）：

function sum1(num1,num2){

   return num1+num2;

}

第二种（函数表达式）

var sum2 = function(num1,num2){

   return num1+num2;

}

第三种（函数对象方式）

var sum3 = new Function("num1","num2","return num1+num2");

#### 22、Javascript创建对象的几种方式？

1. **简单对象的创建** 使用对象**字面量**的方式{}

创建一个对象（最简单，好理解，推荐使用）

代码如下

var Cat  = {};//**JSON**

 Cat.name="kity";//添加属性并赋值

 Cat.age=2;

 Cat.sayHello=function(){

  alert("hello "+Cat.name+",今年"+Cat["age"]+"岁了");//可以使用“.”的方式访问属性，也可以使用HashMap的方式访问

 }

 Cat.sayHello();//调用对象的（方法）函数

**2.用function(函数)来模拟class**

**2.1 创建一个对象，相当于new一个类的实例(无参构造函数)**

代码如下

function Person(){

}

var personOne=new Person();//定义一个function，如果有new关键字去"实例化",那么该function可以看作是一个类

personOne.name="dylan";

personOne.hobby="coding";

personOne.work=function(){

alert(personOne.name+" is coding now...");

}

personOne.work();

**2.2 可以使用有参构造函数来实现，这样定义更方便，扩展性更强（推荐使用）**

代码如下

function Pet(name,age,hobby){

   this.name=name;//this作用域：当前对象

   this.age=age;

   this.hobby=hobby;

   this.eat=function(){

      alert("我叫"+this.name+",我喜欢"+this.hobby+",也是个吃货");

   }

}

var maidou =new Pet("麦兜",5,"睡觉");//实例化/创建对象

 maidou.eat();//调用eat方法(函数)

**3.使用工厂方式来创建（Object关键字）**

代码如下：

var wcDog =new Object();

 wcDog.name="旺财";

 wcDog.age=3;

 wcDog.work=function(){

   alert("我是"+wcDog.name+",汪汪汪......");

 }

 wcDog.work();

**4.使用原型对象的方式  prototype关键字**

代码如下：

function Dog(){

 }

 Dog.prototype.name="旺财";

 Dog.prototype.eat=function(){

 alert(this.name+"是个吃货");

 }

 var wangcai =new Dog();

 wangcai.eat();

**5.混合模式(原型和构造函数)**

代码如下：

function Car(name,price){

  this.name=name;

  this.price=price;

}

 Car.prototype.sell=function(){

   alert("我是"+this.name+"，我现在卖"+this.price+"万元");

  }

var camry =new Car("凯美瑞",27);

camry.sell();

**6.动态原型的方式(可以看作是混合模式的一种特例)**

代码如下：

function Car(name,price){

  this.name=name;

  this.price=price;

  if(typeof Car.sell=="undefined"){

   Car.prototype.sell=function(){

    alert("我是"+this.name+"，我现在卖"+this.price+"万元");

   }

 Car.sell=true;

  }

}

var camry =new Car("凯美瑞",27);

camry.sell();

以上几种，是javascript中最常用的创建对象的方式。

#### 23、js延迟加载的方式有哪些？

1. defer和async（异步加载）

当浏览器碰到 script 脚本的时候：

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

没有 defer 或 async，浏览器会立即加载并执行指定的脚本，“立即”指的是在渲染该 script 标签之下的文档元素之前，也就是说不等待后续载入的文档元素，读到就加载并执行。

1. <script async src="script.js"></script>

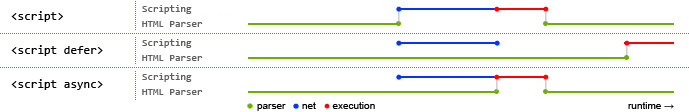
有 async，加载和渲染后续文档元素的过程将和 script.js 的加载与执行并行进行（异步）。

1. <script defer src="myscript.js"></script>

有 defer，加载后续文档元素的过程将和 script.js 的加载并行进行（异步），但是 script.js 的执行要在所有元素解析完成之后，DOMContentLoaded 事件触发之前完成。

从实用角度来说，**首先把所有脚本都丢到 </body> 之前是最佳实践，因为对于旧浏览器来说这是唯一的优化选择，此法可保证非脚本的其他一切元素能够以最快的速度得到加载和解析。**

**补充：**图示



蓝色线代表网络读取，红色线代表执行时间，这俩都是针对脚本的；绿色线代表 HTML 解析。

此图告诉我们以下几个要点：

1. *defer* 和 *async* 在网络读取（下载）这块儿是一样的，都是异步的（相较于 HTML 解析）
2. 它俩的差别在于脚本下载完之后何时执行，显然 *defer* 是最接近我们对于应用脚本加载和执行的要求的
3. 关于 ***defer***，此图未尽之处在于它是按照加载顺序执行脚本的，这一点要善加利用
4. ***async*** 则是一个乱序执行的主，反正对它来说脚本的加载和执行是紧紧挨着的，所以不管你声明的顺序如何，只要它加载完了就会立刻执行
5. 仔细想想，*async* 对于应用脚本的用处不大，因为它完全不考虑依赖（哪怕是最低级的顺序执行），不过它对于那些可以不依赖任何脚本或不被任何脚本依赖的脚本来说却是非常合适的，最典型的例子：Google Analytics
6. 动态创建DOM方式（创建script，插入到DOM中，加载完毕后callBack）

**创建script，插入到DOM中**，加载完毕后callBack，见代码

代码如下：（有兼容性封装）

function loadScript(url, callback){

var script = document.createElement\_x("script")

script.type = "text/javascript";

if (script.readyState){ //IE

script.onreadystatechange = function(){

if (script.readyState == "loaded" ||

script.readyState == "complete"){

script.onreadystatechange = null;

callback();

}

};

} else { //Others： Firefox, Safari, Chrome, and Opera

script.onload = function(){

callback();

};

}

script.src = url;

document.body.appendChild(script);

}

3. 按需异步载入js

默认情况javascript是同步加载的，也就是javascript的加载时阻塞的，后面的元素要等待javascript加载完毕后才能进行再加载，对于一些意义不是很大的javascript，如果放在页头会导致加载很慢的话，是会严重影响用户体验的。

**细节补充：**

**(1) defer，只支持IE**

defer属性的定义和用法（我摘自w3school网站）

defer 属性规定是否对脚本执行进行延迟，直到页面加载为止。

有的 javascript 脚本 document.write 方法来创建当前的文档内容，其他脚本就不一定是了。

如果您的脚本不会改变文档的内容，可将 defer 属性加入到 <script> 标签中，以便加快处理文档的速度。因为浏览器知道它将能够安全地读取文档的剩余部分而不用执行脚本，它将推迟对脚本的解释，直到文档已经显示给用户为止。

示例：

代码如下：

<script type="text/javascript" defer="defer">

alert(document.getElementById("p1").firstChild.nodeValue);

</script>

**(2) async：**

async的定义和用法(是HTML5的属性)

async 属性规定一旦脚本可用，则会异步执行。

示例：

代码如下：

<script type="text/javascript" src="demo\_async.js" async="async"></script>

注释：async 属性仅适用于外部脚本（只有在使用 src 属性时）。

注释：有多种执行外部脚本的方法：

•如果 async="async" 脚本相对于页面的其余部分异步地执行（当页面继续进行解析时，脚本将被执行）

•如果不使用 async 且 defer="defer" 脚本将在页面完成解析时执行

•如果既不使用 async 也不使用 defer 在浏览器继续解析页面之前，立即读取并执行脚本

#### ECMAScript对象的继承结构

继承的规律：

1 对象继承的终点是Object.prototype

2 所有函数默认的显示原型（即函数的prototype） 都继承 Object.prototype

3 谁的实例，这个实例就继承谁的prototype

3.1 所有的函数，都被看作是Function的实例，所以都继承 Function.prototype

3.2 所有的数组，都被看作是Array的实例，所以都继承 Array.prototype

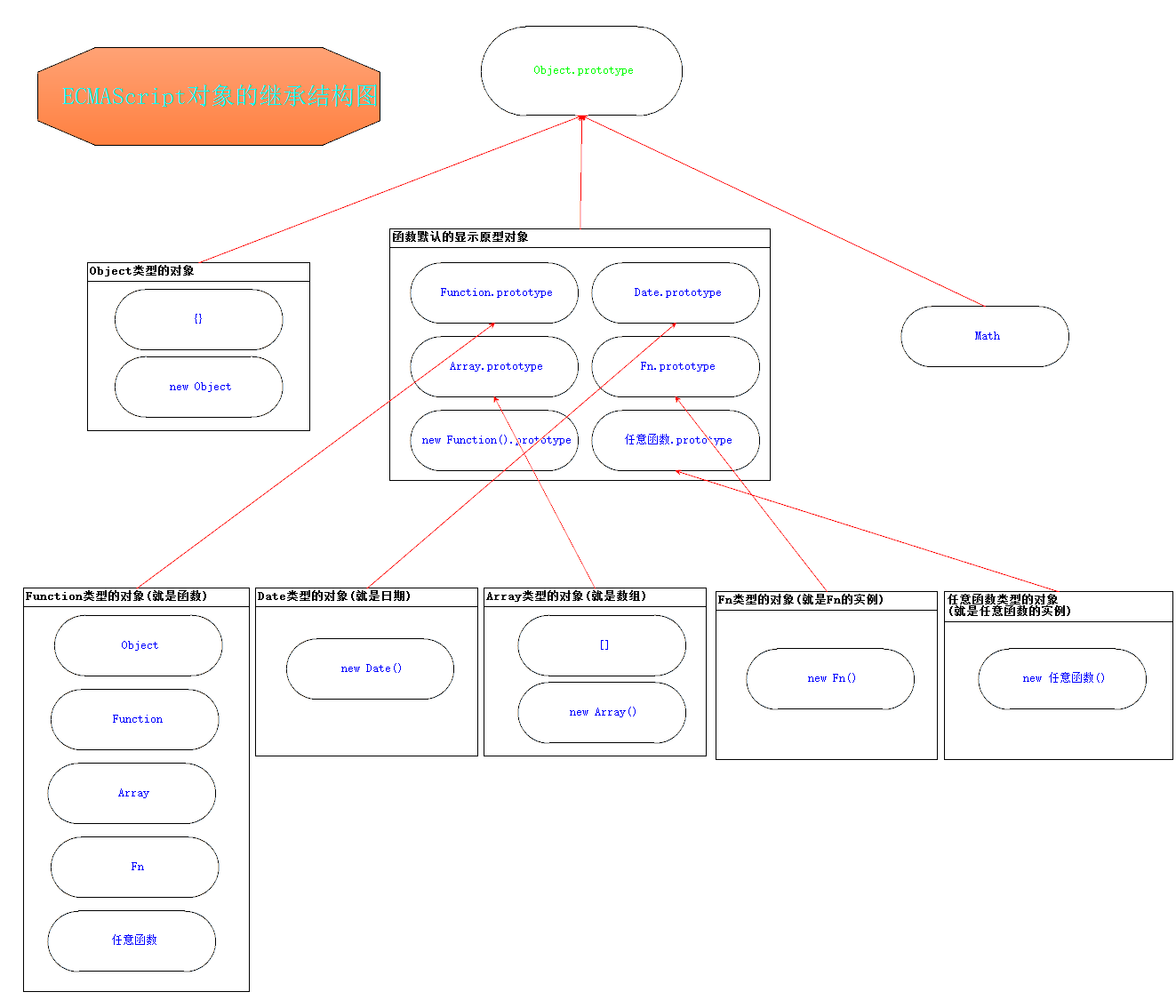
3.3 所有的正则，都被看作是RegExp的实例，所以都继承 RegExp.prototype

Eg:function fn() {}

1 fn 是Function的实例，所以继承 Function.prototype

2 fn.prototype 继承 Object.prototype

3 new fn() 是fn的实例, 所以继承 fn.prototype



#### 25、什么是事件冒泡/捕获？

答：

事件冒泡：子元素事件的触发会影响父元素事件；

开关事件冒泡：

A，开启事件冒泡：element.addEventListener(eventName，handler，false)；

B，关闭事件冒泡：假设传统方式事件的返回值为e，就可以通过e.stopPropagation()来关闭事件冒泡；

事件捕获：父元素的事件会影响子元素的事件；

开启事件捕获：element.addEventListener(eventName，handler，true)

#### 26、如何阻止事件冒泡和默认事件?

举例说明什么是事件冒泡，何时阻止事件冒泡：

如果<p>是在<div>里面，那么呢，<P>有一个onclick事件，<div>也有onclick事件，为了触发<P>的点击事件时，不触发父元素的点击事件，那么就需要调用如下函数，阻止事件冒泡。

阻止事件冒泡函数：

function stopBubble(e){

  if(e&&e.stopPropagation){//非IE

   e.stopPropagation();

  }

  else{//IE

   window.event.cancelBubble=true;

  }

 }

阻止默认事件：

function stopDefault( e ) {

         //阻止默认浏览器动作(W3C)

         if ( e && e.preventDefault )

             e.preventDefault();

         //IE中阻止函数器默认动作的方式

         else

             window.event.returnValue = false;

         return false;

     }

#### 27、面向对象和类的区别？

答：简单的说类是对象的模版。

在 js 中没有类， 所以在js 中所谓的 类 就是构造函数， 对象就是由构造函数创建出来的实例对象。面向对象就是使用面向对象的方式处理问题， 面向对象是对面向过程进行封装。

面向对象有三大特性

抽象性， 需要通过核心数据和特定环境才能描述对象的具体意义

封装性， 封装就是将数据和功能组合到一起， 在js中对象就是键值对的集合， 对象将属性和方法封装起来， 方法将过程封装起来

继承性， 将别人的属性和方法成为自己的， 传统继承基于模板(类)， js中继承基于 构造函数

#### 28、简述for in 循环的特点及使用场景？

答：for...in 语句用于对数组或者对象的属性进行循环操作。

for ... in 循环中的代码每执行一次，就会对数组的元素或者对象的属性进行一次操作。

for (变量 in 对象){

在此执行代码

}

“变量”用来指定变量，指定的变量可以是数组元素，也可以是对象的属性。

注意：for in循环不会按照属性的下标来排列输出。

对象的概念，面向对象编程的程序实际就是多个对象的集合，我们可以把所有的事物都抽象成对象，在程序设计中可以看作：对象=属性+方法。属性就是对象的数据，而方法就是对象的行为。

类的概念，类是对象的模版，而对象是类的实例化。举个例子，汽车设计图可以看作是类，而具体的汽车就是对象。再比如有一个类是表示人，然后可以通过人这个模版来实例化出张三、李四。

#### 29、例举强制类型转换和隐式类型转换？

答：强制

转化成字符串 toString() String()

转换成数字 Number()、 parseInt()、 parseFloat()

转换成布尔类型 Boolean()

隐式

拼接字符串 例子 var str = "" + 18

- / % === ==

#### 30、split() join() 的区别

Split()是把一串字符（根据某个分隔符）分成若干个元素存放在一个数组里

即切割成数组的形式；

join() 是把数组中的字符串连成一个长串，可以大体上认为是Split()的逆操作

#### 31、数组方法pop() push() unshift() shift()

Push()尾部添加 pop() 删除并返回数组的最后一个元素

Unshift() 头部添加 shift() 头部删除

#### 32、怎样判断一个JavaScript变量是array还是obiect？

1、如果你只是用typeof来检查该变量，不论是array还是object，都将返回‘object'。

此问题的一个可行的答案是检查该变量是不是object，并且检查该变量是否有数字长度（当为空array时长度也可能为0）。

然而，参数对象【arguments object】（传给制定函数的所有参数），也可能会适用于上述方法，技术上来说，参数对象并不是一个array。

此外，当一个对象有a.length属性的时候，这个方法也不成立。

代码如下:

// Real array 正在的数组

var my\_array = [];

// Imposter! 冒名顶替的！

var my\_object = {};

my\_object.length = 0;

// Potentially faulty 潜在的错误

function is\_this\_an\_array(param) {

if (typeof param === 'object' && !isNaN(param.length)) {

console.log('Congrats, you have an array!');

}

else {

console.log('Bummer, not an array');

}

}

// Works 成功

is\_this\_an\_array(my\_array);

// Works, but is incorrect 成功了，但是不正确

is\_this\_an\_array(my\_object);

2、回答这个问题的另一个答案是用一个更加隐蔽的方法，调用toString( )方法试着将该变量转化为代表其类型的string。

该方法对于真正的array可行；参数对象转化为string时返回[object Arguments]会转化失败；此外，

对于含有数字长度属性的object类也会转化失败。

代码如下:

// Real array 真正的数组

var my\_array = [];

// Imposter! 冒名顶替的！

var my\_object = {};

my\_object.length = 0;

// Rock solid 坚如磐石（检验函数）

function is\_this\_an\_array(param) {

if (Object.prototype.toString.call(param) === '[object Array]') {

console.log('Congrats, you have an array!');

}

else {

console.log('Bummer, not an array');

}

}

// Works 成功了

is\_this\_an\_array(my\_array);

// Not an array, yay! 不是数组（array）！

is\_this\_an\_array(my\_object);

3、此外，可能在不可靠的多框架DOM环境中，instanceof是个完美合适的操作。

扩展阅读："Instanceof Considered Harmful…"

http://perfectionkills.com/instanceof-considered-harmful-or-how-to-write-a-robust-isarray

代码如下:

var my\_array = [];

if (my\_array instanceof Array) {

console.log('Congrats, you have an array!');

}

4、对于Javascript 1.8.5（ECMAScript 5），变量名字.isArray( )可以实现这个目的

https://developer.mozilla.org/en-US/docs/JavaScript/Reference/Global\_Objects/Array/isArray

代码如下:

var my\_array = [];

if (Array.isArray(my\_array)) {

console.log('Congrats, you have an array!');

}

#### 33、解释jsonp的原理，以及为什么不是真正的ajax

jsonp的原理：动态创建script标签，回调函数

JSONP(JSON with Padding)是[JSON](http://baike.baidu.com/view/136475.htm" \t "http://baike.baidu.com/_blank)的一种“使用模式”，可用于解决主流浏览器的跨域数据访问的问题。利用 <script> 元素的这个开放策略，网页可以得到从其他来源动态产生的 JSON 资料，而这种使用模式就是所谓的 JSONP。用 JSONP 抓到的资料并不是 JSON，而是任意的JavaScript，用 JavaScript 直译器执行而不是用 JSON 解析器解析。

jsonp为什么不是ajax：

ajax和jsonp这两种技术在调用方式上“看起来”很像，目的也一样，都是请求一个url，然后把服务器返回的数据进行处理，因此jquery和ext等框架都把jsonp作为ajax的一种形式进行了封装；

2、但ajax和jsonp其实本质上是不同的东西。ajax的核心是通过XmlHttpRequest获取非本页内容，而jsonp的核心则是动态添加<script>标签来调用服务器提供的js脚本。

3、所以说，其实ajax与jsonp的区别不在于是否跨域，ajax通过服务端代理一样可以实现跨域，jsonp本身也不排斥同域的数据的获取。

4、还有就是，jsonp是一种方式或者说非强制性协议，如同ajax一样，它也不一定非要用json格式来传递数据，如果你愿意，字符串都行，只不过这样不利于用jsonp提供公开服务。

总而言之，jsonp不是ajax的一个特例，哪怕jquery等巨头把jsonp封装进了ajax，也不能改变着一点！

#### 34、JavaScript的事件流模型都有什么，以及怎么阻止他们？

答：1、原始事件模型

普通的事件绑定，比如事件赋值，按钮上绑定事件等

2、DOM事件模型

addEventListener("eventType"，"handler"，"true!false")；

removeEventListner("eventType"，"handler"，"true!false")；

气泡模型（与ie有点区别），冒泡和捕获

3、IE模型

气泡模型

attachEvent( "eventType"，"handler")

detachEvent("eventType"，"handler" )

与dom2不同的是eventType有on前缀

#### 35、Javascript 中的垃圾回收机制

答：在Javascript中，如果一个对象不再被引用，那么这个对象就会被GC回收。如果两个对象互相引用，而不再被第3者所引用，那么这两个互相引用的对象也会被回收。因为函数a被b引用，b又被a外的c引用，这就是为什么  函数a执行后不会被回收的原因。

#### 36、请解释一下浏览器的同源策略

为什么会有同源策略？

我们都知道JavaScript可以操作web文档的内容，试想，如果不对这一点加以限制，那么JS可以做的事情就太多了，危险性也太高，所以就针对它可以操作哪些文档的内容有了一个限制，这个限制就是同源策略。

同源策略在什么情况下会起作用呢？

当web页面使用多个<iframe>元素或者打开其他浏览器窗口的时候，这一策略就会起作用。

同源策略的含义：

脚本只能读取和所属文档来源相同的窗口和文档的属性。

这里就涉及到了一个浏览器如何判断两者是否同源以及如何判断脚本来源的问题。

注意一点：脚本本身的来源并不作为判断是否同源的依据，而是将脚本所属文档的来源作为判断依据。

1. 判断脚本来源

例如：文档A中通过script的src引用了一个外部脚本，这个脚本是google提供的，也是从google的主机上加载到文档A中的，那么这个脚本的所属文档是谁呢，答案是文档A。

2. 判断是否同源

理解了脚本来源，接着理解怎么判断是否同源：如果两个文档在协议、主机以及载入文档的URL端口这三点中有一点不同，就认为他们不同源。

#### 37、事件绑定和普通事件有什么区别

事件绑定是指把事件注册到具体的DOM元素之上，普通事件指的是可以用来注册的事件

事件绑定例子：

DOM 元素.addEventListener( 事件类型，事件处理函数)

普通事件例子：

DOM 元素.事件类型 = function(){事件处理函数}

对于事件绑定来说，同一个 DOM 元素上面绑定同样的事件类型时，可以绑定多个事件处理函数，实际解决了多人开发情况下造成的时间处理覆盖的情况（污染），而如果使用普通事件来解决这个问题的时候会造成后写的代码将前写的代码进行了重写操作，造成环境污染。

实际开发的时候可以视情况而定，来选择绑定事件的方式。

#### 38、怎么判断一个变量是否null/undefined

Undefined:

if (typeof(reValue) == "undefined") {

    alert("undefined");

}

typeof 返回的是字符串，有六种可能："number"、"string"、"boolean"、"object"、"function"、 "undefined"

Null:

var exp = null;

if (!exp && typeof(exp)!=”undefined” && exp!=0)

{

alert(“is null”);

}

我们在 DOM 应用中，一般只需要用 (!exp) 来判断就可以了，因为 DOM 应用中，可 能返回 null，可能返回 undefined，如果具体判断 null 还是 undefined 会使程序过于复杂。

#### 39、”==”和“===”的不同

”==”判断值是否相等，会自动转换类型；

“===”是判断值及类型是否完全相等，不会自动转换类型。

#### 40、Javascript中callee和caller的作用？

caller是返回一个对函数的引用，该函数调用了当前函数；

function a(){

b();

};

function b(){

alert(b.**caller**);

};

a(); //结果就是弹出函数a和内容

callee是返回正在被执行的function函数，也就是所指定的function对象的正文。

先来看一个阶乘函数：

可以看到用到了递归函数

但是问题来了，要是改变了函数名，里面的函数名也要随着改变，这样很不方便所以我们用callee来试试

function Aaa(n){

if(n<=1){

**return 1;**

}else{

**return n\*Aaa(n-1);**

}

}

function Aaa(n){

if(n<=1){

**return 1;**

}else{

**return n\*arguments.callee(n-1)**

}

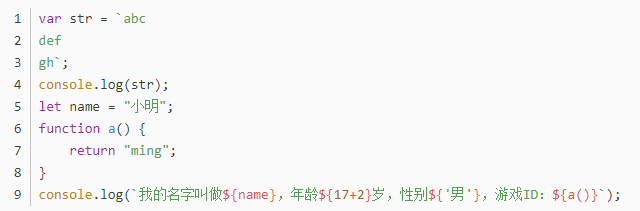
}

### 1.1.4 es6部分（必会）

1. **新增声明命令**let**和**const

在es6中通常用 let 和 const 来声明，let 表示变量、const 表示常量。

特点：

* let 和 const 都是块级作用域。以{}代码块作为作用域范围 只能在代码块里面使用。
* 不存在变量提升，只能先声明再使用，否则会报错。在代码块内，在声明变量之前，该变量都是不可用的。这在语法上，称为“暂时性死区”（temporal dead zone，简称 TDZ）。
* 在同一个代码块内，不允许重复声明。
* const声明的是一个只读常量，在声明时就需要赋值。（如果 const 的是一个对象，对象所包含的值是可以被修改的。抽象一点儿说，就是对象所指向的地址不能改变，而变量成员是可以修改的。）

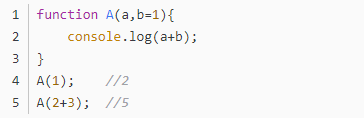
1. **模板字符串（Template String）**

用一对反引号(`)标识，它可以当作普通字符串使用，也可以用来定义多行字符串，也可以在字符串中嵌入变量，js表达式或函数，变量、js表达式或函数需要写在${ }中。

1. **函数的扩展**

* 函数的默认参数

ES6为参数提供了默认值。在定义函数时便初始化了这个参数，以便在参数没有被传递进去时使用。



* **箭头函数**

在es6中，提供了一种简洁的函数写法，我们称作“箭头函数”。

写法：函数名=(形参)=>{……}     当函数体中只有一个表达式时，{}和return可以省略，当函数体中形参只有一个时，()可以省略。

特点：箭头函数中的this始终指向箭头函数定义时的离this最近的一个函数，如果没有最近的函数就指向window。

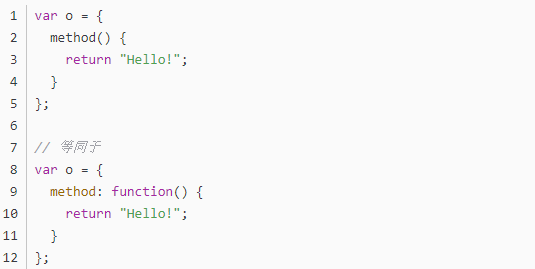


1. **对象的扩展**

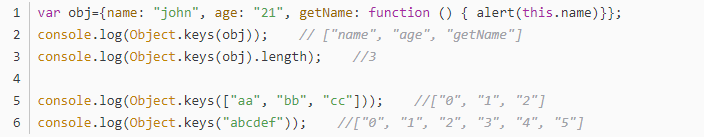
* 属性的简写。ES6 允许在对象之中，直接写变量。这时，属性名为变量名, 属性值为变量的值。



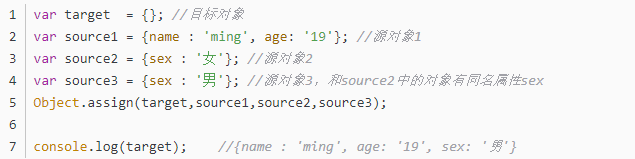
* 方法的简写。省略冒号与function关键字。



* Object.keys()方法，获取对象的所有属性名或方法名（不包括原形的内容），返回一个数组。

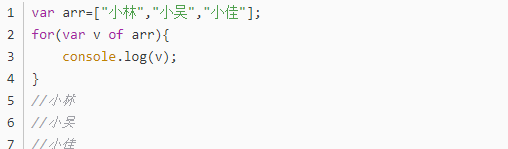


* Object.assign ()，assign方法将多个原对象的属性和方法都合并到了目标对象上面。可以接收多个参数，第一个参数是目标对象，后面的都是源对象。



1. **for...of  循环**

是遍历所有数据结构的统一的方法。for...of循环可以使用的范围包括数组、Set 和 Map 结构、某些类似数组的对象（比如arguments对象、DOM NodeList 对象）、Generator 对象，以及字符串。



1. **import和export**

ES6标准中，JavaScript原生支持模块(module)了。这种将JS代码分割成不同功能的小块进行模块化，将不同功能的代码分别写在不同文件中，各模块只需导出公共接口部分，然后通过模块的导入的方式可以在其他地方使用。

export用于对外输出本模块（一个文件可以理解为一个模块）变量的接口。

import用于在一个模块中加载另一个含有export接口的模块。

import和export命令只能在模块的顶部，不能在代码块之中。



1. **Promise对象**

* Promise是异步编程的一种解决方案，将异步操作以同步操作的流程表达出来，避免了层层嵌套的回调函数。
* 它有三种状态，分别是pending-进行中、resolved-已完成、rejected-已失败。
* Promise 构造函数包含一个参数和一个带有 resolve（解析）和 reject（拒绝）两个参数的回调。在回调中执行一些操作（例如异步），如果一切都正常，则调用 resolve，否则调用 reject。对于已经实例化过的 promise 对象可以调用 promise.then() 方法，传递 resolve 和 reject 方法作为回调。then()方法接收两个参数：onResolve和onReject，分别代表当前 promise 对象在成功或失败时。



promise的执行过程：

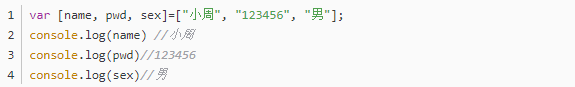


1. **解构赋值**

ES6 允许按照一定模式，从数组和对象中提取值，对变量进行赋值，这被称为解构（Destructuring）。

* 数组的解构赋值

数组中的值会自动被解析到对应接收该值的变量中，数组的解构赋值要一一对应 如果有对应不上的就是undefined



* 对象的解构赋值

对象的解构与数组有一个重要的不同。数组的元素是按次序排列的，变量的取值由它的位置决定；而对象的属性没有次序，变量必须与属性同名，才能取到正确的值。



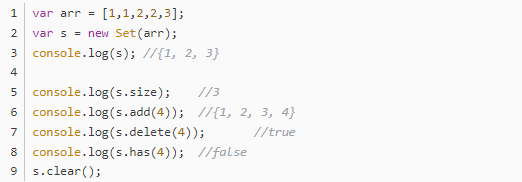
1. **set数据结构**

Set数据结构，类似数组。所有的数据都是唯一的，没有重复的值。它本身是一个构造函数。

属性和方法：

* size 数据的长度
* add() 添加某个值，返回 Set 结构本身。
* delete() 删除某个值，返回一个布尔值，表示删除是否成功。
* has() 查找某条数据，返回一个布尔值。
* clear() 清除所有成员，没有返回值。

应用：数组去重



1. **Spread Operator 展开运算符(...)**

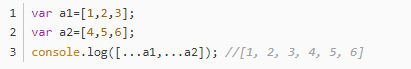
* 将字符串转成数组



* 将集合转成数组



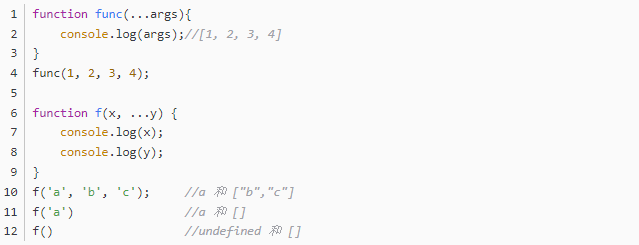
* 两个数组的合并



* 在函数中，用来代替arguments参数

rest参数  …变量名称

rest 参数是一个数组 ，它的后面不能再有参数，不然会报错



### 1.1.5程序题（JS程序）

#### 1、写出代码对下列数组去重并从大到小排列{5,2,3,6,8,6,5,4,7,1,9}

function fn(arr){

for (var i = 0; i < arr.length-1; i++) {

for (var j = 0; j < arr.length-1-i; j++) {

if(arr[j]<arr[j+1]){

var temp = arr[j];

arr[j]=arr[j+1];

arr[j+1]=temp;

}

}

}

for (i = 0; i < arr.length; i++) {

var c=arr[i];

for (var s = i+1; s < arr.length; s++) {

if(arr[s]==c)

arr.splice(s,1);

s--;

}

}

}

return arr;

}

console.log(fn([5,2,3,6,8,6,5,4,7,1,9]).toString());

#### 2、用JavaScript实现冒泡排序。数据为23、45、18、37、92、13、24

//升序算法

function sort(arr){

for (var i = 0; i <arr.length; i++) {

for (var j = 0; j <arr.length-i; j++) {

if(arr[j]>arr[j+1]){

var c=arr[j];//交换两个变量的位置

arr[j]=arr[j+1];

arr[j+1]=c;

}

};

};

return arr.toString();

}

console.log(sort([23,45,18,37,92,13,24]));

#### 3、用js实现随机选取10–100之间的10个数字，存入一个数组，并排序。

function randomNub(aArray, len, min, max) {

if (len >= (max - min)) {

return '超过' + min + '-' + max + '之间的个数范围' + (max - min - 1) + '个的总数';

}

if (aArray.length >= len) {

aArray.sort(function(a, b) {

return a - b

});

return aArray;

}

var nowNub = parseInt(Math.random() \* (max - min - 1)) + (min + 1);

for (var j = 0; j < aArray.length; j++) {

if (nowNub == aArray[j]) {

randomNub(aArray, len, min, max);

return;

}

}

aArray.push(nowNub);

randomNub(aArray, len, min, max);

return aArray;

}

var arr=[];

randomNub(arr,10,10,100);

#### 4、已知数组var stringArray = [“This”, “is”, “Baidu”, “Campus”]，Alert出”This is Baidu Campus”。

答案：var stringArray = ["This", "is", "Baidu", "Campus"]

alert(stringArray.join(""))

#### 5、已知有字符串foo=”get-element-by-id”,写一个function将其转化成驼峰表示法”getElementById”。

function combo(msg){

    var arr=msg.split("-");

    for(var i=1;i<arr.length;i++){

        arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);

    }

    msg=arr.join("");

    return msg;

}

#### 6、下面这段JS输出什么，并简述为什么？

function Foo() {

var i = 0；

return function () {

console.log(i++)；

}

}

var f1 = Foo()，

f2 = Foo()；

f1()；

f1()；

f2()；

console.log(i)；

0 //f1=Foo() 相当于f1赋值为函数Foo()的返回值f1=function（）{

console.log(i++)

}

1 //因为f1=了一个function所以有了作用域，f2和f1不同，不在一个内存中

0

报错 //i为Foo内部的变量全局不可访问，全局中没有i变量所以会报错

#### 7、有这样一个URL：http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e，请写一段JS程序提取URL中的各个GET参数(参数名和参数个数不确定)，将其按key-value形式返回到一个json结构中，如{a:’1′, b:’2′, c:”, d:’xxx’, e:undefined}。

|  |  |
| --- | --- |
|  | function serilizeUrl(url) {  var urlObject = {};  if (/\?/.test(url)) {  var urlString = url.substring(url.indexOf("?") + 1);  var urlArray = urlString.split("&");  for (var i = 0, len = urlArray.length; i < len; i++) {  var urlItem = urlArray[i];  var item = urlItem.split("=");  urlObject[item[0]] = item[1];  }  return urlObject;  }  return null;  } |

#### 8、var numberArray = [3,6,2,4,1,5]; （考察基础API）

1) 实现对该数组的倒排，输出[5,1,4,2,6,3]

var numberArray = [3,6,2,4,1,5];

alert( numberArray .reverse())

1. 实现对该数组的降序排列，输出[6,5,4,3,2,1]

var numberArray = [3,6,2,4,1,5];

// a-b输出从小到大排序，b-a输出从大到小排序。

numberArray .sort(function(a,b){

return b-a});

console.log((numberArray .join()));

#### 9、输出今天的日期，以YYYY-MM-DD的方式，比如今天是2014年9月26日，则输出2014-09-26

|  |  |
| --- | --- |
|  | var d = new Date();  // 获取年，getFullYear()返回4位的数字  var year = d.getFullYear();  // 获取月，月份比较特殊，0是1月，11是12月  var month = d.getMonth() + 1;  // 变成两位  month = month < 10 ? '0' + month : month;  // 获取日  var day = d.getDate();  day = day < 10 ? '0' + day : day;  alert(year + '-' + month + '-' + day); |

#### 10、将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的{$id}替换成10，{$nny （使用正则表达式）

答案：”<tr><td>{$id}</td><td>{$id}\_{$name}</td></tr>”.replace(/{\$id}/g, ’10′).replace(/{\$name}/g, ‘Tony’);

#### 11、请写出下面输出的值

Console.log(undefined || 1)；//值\_\_\_**1**\_\_

Console.log(null || NaN)；//值\_\_**NaN**\_\_\_

Console.log(0 && 1)；//值\_**\_0**\_\_\_

Console.log(0 && 1 || 0)；//值\_\_**0**\_\_\_

#### 12、如何垂直居中一个浮动元素？

|  |  |
| --- | --- |
|  | // 方法一：已知元素的高宽  #div1{      background-color:#6699FF;      width:200px;      height:200px;      position: absolute;        //父元素需要相对定位      top: 50%;      left: 50%;      margin-top:-100px ;   //二分之一的height，width      margin-left: -100px;      }    //方法二:未知元素的高宽      #div1{      width: 200px;      height: 200px;      background-color: #6699FF;        margin:auto;      position: absolute;        //父元素需要相对定位      left: 0;      top: 0;      right: 0;      bottom: 0;      } |

#### 13、如何垂直居中一个img?

|  |  |
| --- | --- |
|  | #container     //<img>的容器设置如下  {      display:table-cell;      text-align:center;      vertical-align:middle;  } |

#### 14、以下js的运行结果是什么，为什么？

var txt='hx';

function hello(){

var txt;

fn();//world 函数名与变量名重复的时候，以函数名为主

var fn=function(){alert('hello')}

function fn(){alert('world');}

alert(txt);//undefined 局部变量，只是声明，没有赋值

fn();//hello 先进行声明，后赋值，执行fn=function(){alert('hello')}

}

hello();

#### 15、看下列代码，将会输出什么?(变量声明提升)

|  |  |
| --- | --- |
|  | var foo = 1;  function(){      console.log(foo);      var foo = 2;      console.log(foo);  } |

答案：输出undefined 和 2。上面代码相当于：

|  |  |
| --- | --- |
|  | var foo = 1;  function(){      var foo;      console.log(foo); //undefined      foo = 2;      console.log(foo); // 2;  } |

函数声明与变量声明会被JavaScript引擎隐式地提升到当前作用域的顶部，但是只提升名称不会提升赋值部分。

#### 16、把两个数组合并，并删除第二个元素。

|  |  |
| --- | --- |
|  | var array1 = ['a','b','c'];  var bArray = ['d','e','f'];  var cArray = array1.concat(bArray);  cArray.splice(1,1); |

#### 17、写一个function，清除字符串前后的空格。（兼容所有浏览器）

//使用自带接口trim()，考虑兼容性：

|  |  |
| --- | --- |
|  | if (!String.prototype.trim) {   String.prototype.trim = function() {   return this.replace(/^\s+/, "").replace(/\s+$/,"");   }  }   // test the function  var str = " \t\n test string ".trim();  alert(str == "test string"); // alerts "true" |

#### 18、Javascript中, 以下哪条语句一定会产生运行错误？答案( B )

1. var \_变量=NaN;B、var 0bj = [];C、var obj = //; D、var obj = {};

#### 19、以下两个变量a和b，a+b的哪个结果是NaN？ 答案( C )

A、var a=undefind; b=NaN

B、var a=‘123’; b=NaN

C、var a =undefined , b =NaN

D、var a=NaN , b='undefined'

#### 20、var a=10; b=20; c=4; ++b+c+a++ 以下哪个结果是正确的答案( B )

1. 34 B、35 C、36 D、37

#### 21、写出程序运行的结果？

for(i=0, j=0; i<10, j<6; i++, j++){

k = i + j;}

结果：10

#### 22、阅读以下代码，请分析出结果：

var arr = new Array(1 ,3 ,5);

arr[4]='z';

arr2 = arr.reverse();

arr3 = arr.concat(arr2);

alert(arr3);

弹出提示对话框：z,,5,3,1,z,,5,3,1

#### 23、截取字符串abcdefg的efg

alert('abcdefg'.substring(4));

#### 24、判断一个字符串中出现次数最多的字符，统计这个次数

答：var str = 'asdfssaaasasasasaa';

var json = {};

for (var i = 0; i < str.length; i++) {

        if(!json[str.charAt(i)]){

                json[str.charAt(i)] = 1;

        }else{

                json[str.charAt(i)]++;

        }

};

var iMax = 0;

var iIndex = '';

for(var i in json){

        if(json[i]>iMax){

                iMax = json[i];

                iIndex = i;

        }

}

alert('出现次数最多的是:'+iIndex+'出现'+iMax+'次');

#### 25、将数字 12345678 转化成 RMB形式 如： 12,345,678

|  |  |
| --- | --- |
|  | //个人方法；  //思路：先将数字转为字符， str= str + '' ;  //利用反转函数，每三位字符加一个 ','最后一位不加； re()是自定义的反转函数，最后再反转回去！  for(var i = 1; i <= re(str).length; i++){      tmp += re(str)[i - 1];      if(i % 3 == 0 && i != re(str).length){          tmp += ',';      }  } |

#### 26、加减运算

|  |  |
| --- | --- |
|  | alert('5'+3); //53 string  alert('5'+'3'); //53 string  alert('5'-3); //2 number  alert('5'-'3'); //2 number |

#### 27、计算字符串字节数：

|  |  |
| --- | --- |
|  | new function(s){       if(!arguments.length||!s) return null;       if(""==s) return 0;       var l=0;       for(var i=0;i<s.length;i++){           if(s.charCodeAt(i)>255) l+=2; else l+=1;  //charCodeAt()得到的是unCode码       }     //汉字的unCode码大于 255bit 就是两个字节       alert(l);  }("hello world!"); |

#### 28、看下列代码,输出什么？

var a = new Object();

a.value = 1;

b = a;

b.value = 2;

alert(a.value);

执行完后输出结果为2

## 1.2 前端性能优化

#### 1、你如何对网站的文件和资源进行优化？

文件合并：（目的是减少http请求）：Web性能优化最佳实践中最重要的一条是减少HTTP 请求，它也是YSlow中比重最大的一条规则。减少HTTP请求的方案主要有合并JavaScript和CSS文件、CSS Sprites、图像映射 （Image Map）和使用Data URI来编码图片。CSS Sprites和图像映射现在已经随处可见了，但由于IE6和IE7不支持Data URI以及性能问题，这项技术尚未大量使用。目前大部分网页中的JavaScript和CSS文件数量和开发时一致，少量的网页会根据实际情况采取本地合并，这些合并中相当多的是有选择的手动完成，每次新的合并都需要重新在本地完成并上传到服务器，比较的随意和繁琐，同样文件的压缩也有类似的情况。而利用服务端的合并和压缩，我们就可以按照开发的逻辑尽可能让文件的颗粒度变小，利用网页中URL的规则来自动实现文件的合并和压缩，这会相当的灵活和高效。参照<http://www.iamued.com/qianduan/1462.html>

**文件最小化/文件压缩**：目的是直接减少文件下载的体积；常用的工具是YUI Compressor。参考<http://www.cnblogs.com/Darren_code/archive/2011/12/31/property.html>

**使用CDN托管：**"其基本思路是尽可能避开互联网上有可能影响数据传输速度和稳定性的瓶颈和环节，使内容传输的更快、更稳定。通过在网络各处放置节点服务器所构成的在现有的互联网基础之上的一层智能虚拟网络，CDN系统能够实时的根据网络流量和各节点的连接、负载状况以及到用户的距离和响应时间等综合信息将用户的请求重新导向离用户最近的服务节点上。" 形象点说：古代打仗大家一定都知道，由于古代的交通很不发达，所以当外族进攻的时候往往不能及时的反击，等朝廷征完兵再把兵派往边境的时候那些侵略者却是早已不见了踪影，这个让古代的帝王很是郁闷。后来帝王们学聪明了，都将大量的兵员提前派往边境驻扎，让他们平时屯田，战时当兵，这样的策略起到了很显著的作用。

缓存的使用：（并且多个域名来提供缓存）

#### 2、一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。

**图片懒加载，**在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。

如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。

如果图片为css图片，可以使用CSSsprite，SVGsprite，Iconfont、Base64等技术。

如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。

如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

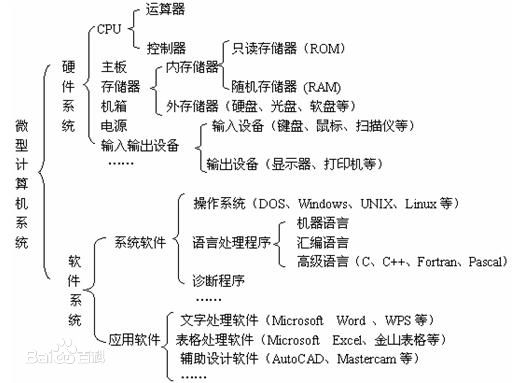
#### 3、图项目中图片处理相关的优化，项目中用到的优化方案，图片大小达到多少的时候选择处理？

答：1、首先了解在web开发中常见的图片有那些格式。JPG 通常使用的背景图片，照片图片，商品图片等等。这一类型的图片都属于大尺寸图片或较大尺寸图片一般使用的是这种格式。PNG 这种格式的又分为两种 一种PNG-8，一种 PNG-24。PNG-8格式不支持半透明，也是IE6兼容的图片存储方式。PNG-24图片质量要求较高的半透明或全透明背景，保存成PNG-24更合适（为了兼容IE可以试用js插件pngfix）一般是背景图标中使用的多。GIF 这种格式显而易见的是在需要gif动画的时候使用了。2．优化方案 样式代替图片例如：半透明、圆角、阴影、高光、渐变等。这些效果主流的浏览器都能够完美支持，而对于那些低端浏览器，我们并不会完全抛弃他们，“渐进增强”则是一个很好的解决方案。 精灵图CSS Sprites，将同类型的图标或按钮等背景图合到一张大图中，减少页面请求。 字体图标Icon Font，将图标做成字体文件。优点是图标支持多个尺寸，兼容所有浏览器，减少页面请求等。美中不足的是只支持纯色的icon。SVG，对于绝大多数图案、图标等，矢量图更小，且可缩放而无需生成多套图。现在主流浏览器都支持SVG了，所以可放心使用！ Base64将图片转化为base64编码格式，资源内嵌于CSS或HTML中，不必单独请求。Base64格式data：[][；charset=][；base64]，Base64 在CSS中的使用.demoImg{ background-image： url("data：image/jpg；base64，/9j/4QMZRXhpZgAASUkqAAgAAAA L....")； }Base64 在HTML中的使用<img width="40" height="30" src="data：image/jpg；base64，/9j/4QMZRXhpZgAASUkqAAgAAAAL...." /> 图片响应式通常图片加都是可以通过lazy加载的形式来的，那么可以在加载的时候来判断屏幕的尺寸来达到加载大图还是小图的目的来达到优化。像资源的优化方法

## 1.3计算机相关术语

#### 1、关于计算机相关术语的介绍

了解计算机相关术语的目的：作为一个计算机相关专业的学生来说，大学开设的课程有： 计算机基础、网页设计、计算机组成原理、数据结构、C语言、C++、java、.net、计算机网络、高等数学、线性代数、离散数学、概率论、操作系统、软件测试、linux、汇编语言等。了解一定的计算机相关术语有助于更好的体现我们的专业性、技术性。



计算机组成图

#### 2、http超文本传输协议

[超文本传输协议](http://baike.baidu.com/item/%E8%B6%85%E6%96%87%E6%9C%AC%E4%BC%A0%E8%BE%93%E5%8D%8F%E8%AE%AE" \t "http://baike.baidu.com/_blank)（HTTP，HyperText Transfer Protocol)是[互联网](http://baike.baidu.com/item/%E4%BA%92%E8%81%94%E7%BD%91" \t "http://baike.baidu.com/_blank)上应用最为广泛的一种[网络协议](http://baike.baidu.com/item/%E7%BD%91%E7%BB%9C%E5%8D%8F%E8%AE%AE" \t "http://baike.baidu.com/_blank)。所有的[WWW](http://baike.baidu.com/item/WWW" \t "http://baike.baidu.com/_blank)文件（即超文本文件（Hypertext），是指具有超链接功能的文件，它可以将文件中已经定义好的关键字（Keyword），经过鼠标的点取（Click），便可以得到该关键字的相关解释，这种方法使用户使用起来更感舒适。类似于早期使用的WIN32下的HELP文件。）都必须遵守这个标准。设计HTTP最初的目的是为了提供一种发布和接收[HTML](http://baike.baidu.com/item/HTML" \t "http://baike.baidu.com/_blank)页面的方法。

工作原理：

一次HTTP操作称为一个事务，其工作过程可分为四步：

1. 首先客户机与服务器需要建立连接。只要单击某个超级链接，HTTP的工作就开始了。 2、建立连接后，客户机发送一个请求给服务器，请求方式的格式为：统一资源标识符（URL）、协议[版本号](http://baike.baidu.com/item/%E7%89%88%E6%9C%AC%E5%8F%B7" \t "http://baike.baidu.com/_blank)，后边是MIME信息包括请求修饰符、客户机信息和可能的内容。

3、服务器接到请求后，给予相应的响应信息，其格式为一个状态行，包括信息的协议版本号、一个成功或错误的代码，后边是MIME信息包括服务器信息、实体信息和可能的内容。

4、客户端接收服务器所返回的信息通过浏览器显示在用户的显示屏上，然后客户机与服务器断开连接。

注意：如果在以上过程中的某一步出现错误，那么产生错误的信息将返回到客户端，由显示屏输出。对于用户来说，这些过程是由HTTP自己完成的，用户只要用鼠标点击，等待信息显示就可以了。



http工作流程图

报文格式：

请求报文格式如下：

请求行 － 通用信息头 － 请求头 － 实体头 － 报文主体

应答报文格式如下：

状态行 － 通用信息头 － 响应头 － 实体头 － 报文主体

协议功能：

HTTP协议（HyperText Transfer Protocol，超文本传输协议）是用于从WWW服务器传输超文本到本地浏览器的传输协议。它可以使浏览器更加高效，使网络传输减少。它不仅保证计算机正确快速地传输超文本文档，还确定传输文档中的哪一部分，以及哪部分内容首先显示(如文本先于图形)等。

HTTP是客户端浏览器或其他程序与[Web服务](http://baike.baidu.com/item/Web%E6%9C%8D%E5%8A%A1" \t "http://baike.baidu.com/_blank)器之间的应用层通信协议。在Internet上的Web服务器上存放的都是超文本信息，客户机需要通过HTTP协议传输所要访问的超文本信息。HTTP包含命令和传输信息，不仅可用于Web访问，也可以用于其他因特网/内联网应用系统之间的通信，从而实现各类应用资源超媒体访问的集成。

我们在浏览器的地址栏里输入的网站地址叫做URL (Uniform Resource Locator，[统一资源定位符](http://baike.baidu.com/item/%E7%BB%9F%E4%B8%80%E8%B5%84%E6%BA%90%E5%AE%9A%E4%BD%8D%E7%AC%A6" \t "http://baike.baidu.com/_blank))。就像每家每户都有一个门牌地址一样，每个网页也都有一个Internet地址。当你在

浏览器的地址框中输入一个URL或是单击一个[超级链接](http://baike.baidu.com/item/%E8%B6%85%E7%BA%A7%E9%93%BE%E6%8E%A5" \t "http://baike.baidu.com/_blank)时，URL就确定了要浏览的地址。浏览器通过超文本传输协议(HTTP)，将Web服务器上站点的网页代码提取出来，并翻译成漂亮的网页。

#### 3、TCP协议

TCP协议是网络协议中最基本的协议之一，TCP（Transmission Control Protocol [传输控制协议](http://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E6%8E%A7%E5%88%B6%E5%8D%8F%E8%AE%AE" \t "http://baike.baidu.com/_blank)）是一种面向连接的、可靠的、基于字节流的[传输层](http://baike.baidu.com/item/%E4%BC%A0%E8%BE%93%E5%B1%82" \t "http://baike.baidu.com/_blank)通信协议。

连接建立:

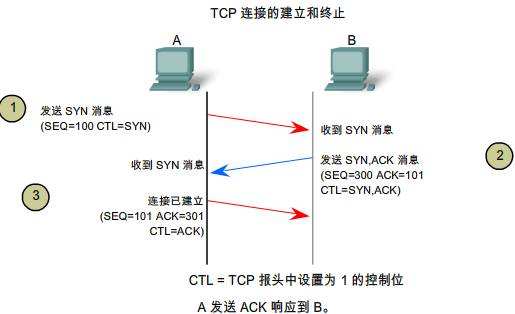
TCP是因特网中的传输层协议，使用[三次握手协议](http://baike.baidu.com/item/%E4%B8%89%E6%AC%A1%E6%8F%A1%E6%89%8B%E5%8D%8F%E8%AE%AE" \t "http://baike.baidu.com/_blank)建立连接，完成三次握手，[客户端](http://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF" \t "http://baike.baidu.com/_blank)与服务器开始传送[数据](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE" \t "http://baike.baidu.com/_blank)。

TCP三次握手：

第一次握手：建立连接时，[客户端](http://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF" \t "http://baike.baidu.com/_blank)发送[syn](http://baike.baidu.com/item/syn" \t "http://baike.baidu.com/_blank)包（syn=j）到[服务器](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)，并进入[SYN\_SENT](http://baike.baidu.com/item/SYN_SENT" \t "http://baike.baidu.com/_blank)状态，等待服务器确认；SYN：同步序列编号（***Synchronize Sequence Numbers***）。

[第二次握手](http://baike.baidu.com/item/%E7%AC%AC%E4%BA%8C%E6%AC%A1%E6%8F%A1%E6%89%8B" \t "http://baike.baidu.com/_blank)：[服务器](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)收到[syn](http://baike.baidu.com/item/syn" \t "http://baike.baidu.com/_blank)包，必须确认客户的SYN（[ack](http://baike.baidu.com/item/ack" \t "http://baike.baidu.com/_blank)=j+1），同时自己也发送一个SYN包（syn=k），即SYN+ACK包，此时服务器进入[SYN\_RECV](http://baike.baidu.com/item/SYN_RECV" \t "http://baike.baidu.com/_blank)状态；

第三次握手：[客户端](http://baike.baidu.com/item/%E5%AE%A2%E6%88%B7%E7%AB%AF" \t "http://baike.baidu.com/_blank)收到[服务](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1" \t "http://baike.baidu.com/_blank)器的SYN+ACK包，向[服务器](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)发送确认包ACK([ack](http://baike.baidu.com/item/ack" \t "http://baike.baidu.com/_blank)=k+1），此包发送完毕，客户端和服务器进入[ESTABLISHED](http://baike.baidu.com/item/ESTABLISHED" \t "http://baike.baidu.com/_blank)（TCP连接成功）状态，完成三次握手。



TCP连接建立图

TCP协议的优缺点：

优点：TCP发送的包有序号，对方收到包后要给一个反馈，如果超过一定时间还没收到反馈就自动执行超时重发，因此TCP最大的优点是可靠。

缺点：很简单，就是麻烦，如果数据量比较小的话建立连接的过程反而占了大头，不断地重发也会造成网络延迟，因此比如[视频聊天](https://www.baidu.com/s?wd=%E8%A7%86%E9%A2%91%E8%81%8A%E5%A4%A9&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPj99mhnYnHP9nhF9myFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EP1m3PjcknH6" \t "https://zhidao.baidu.com/question/_blank)通常就使用UDP，因为丢失一些包也没关系，速度流畅才是重要的。

#### 4、计算机网络的分层体系结构

物理层：物理接口规范,传输比特流,网卡是工作在物理层的.

数据链路层：成帧,保证帧的无误传输,MAC地址,形成EHTHERNET帧 数据链路层在不可靠的物理介质上提供可靠的传输。该层的作用包括：物理地址寻址、数据的成帧、[流量控制](https://www.baidu.com/s?wd=%E6%B5%81%E9%87%8F%E6%8E%A7%E5%88%B6&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1dhujc1uARsnWDsmWcdPhc0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtYnWD4nWnLnf" \t "https://zhidao.baidu.com/question/_blank)、数据的检错、重发等。

网络层：路由选择,流量控制,IP地址,形成IP包

传输层：端口地址,如HTTP对应80端口.TCP和UDP工作于该层,还有

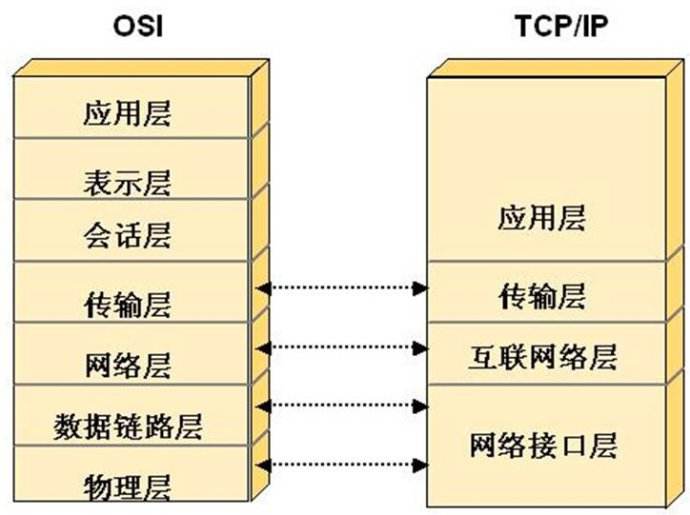
差错校验和流量控制.

会话层:组织两个会话进程之间的通信,并管理数据的交换使用ETBIOS

和WINSOCK协议.QQ等软件进行通讯因该是工作在会话层的.

表示层：使得不同操作系统之间通信成为可能.

应用层：对应于各个应用软件



计算机网络的分层体系结构图

#### 5、计算机存储器相关知识

存储器：是计算机的重要组成部分. 它可分为：

计算机内部的存储器（简称内存）

计算机外部的存储器（简称外存）

[内存储器](https://www.baidu.com/s?wd=%E5%86%85%E5%AD%98%E5%82%A8%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykuyckm1bzm1w-rH03uH-B0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnWm4njmsnjTs" \t "https://zhidao.baidu.com/question/_blank)从功能上可以分为：读写存储器RAM、只读存储器ROM两大类

RAM和ROM的区别：

RAM：（Ramdom Access Memory）易挥发性[随机存取存储器](https://www.baidu.com/s?wd=%E9%9A%8F%E6%9C%BA%E5%AD%98%E5%8F%96%E5%AD%98%E5%82%A8%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPyfYuHcLrHP9nhcYuynY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjD1P1ndnHf" \t "https://zhidao.baidu.com/question/_blank)，高速存取，读写时间相等，且与地址无关，如计算机内存等。RAM表示的是读写存储器，可以与任一存储单元进行读或写操作，计算机关闭电源后其内的信息将不在保存，再次开机需要重新装入，通常用来存放操作系统，各种正在运行的软件、输入和输出数据、中间结果及与外存交换信息等，我们常说的内存主要是指RAM。

ROM：（Read Only Memory）[只读存储器](https://www.baidu.com/s?wd=%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPyfYuHcLrHP9nhcYuynY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjD1P1ndnHf" \t "https://zhidao.baidu.com/question/_blank)。断电后信息不丢失，如计算机启动用的[BIOS芯片](https://www.baidu.com/s?wd=BIOS%E8%8A%AF%E7%89%87&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPyfYuHcLrHP9nhcYuynY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjD1P1ndnHf" \t "https://zhidao.baidu.com/question/_blank)。存取速度很低，（较RAM而言）且不能改写。由于不能改写信息，不能升级，现已很少使用。ROM表示的是[只读存储器](https://www.baidu.com/s?wd=%E5%8F%AA%E8%AF%BB%E5%AD%98%E5%82%A8%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YkPyfYuHcLrHP9nhcYuynY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3ErjD1P1ndnHf" \t "https://zhidao.baidu.com/question/_blank)，即：它只能读出信息，不能写入信息，计算机关闭电源后其内的信息仍旧保存，一般用它存储固定的系统软件和字库等。

#### 6、浏览器

一、浏览器相关知识介绍：

浏览器是指可以显示网页[服务器](http://baike.baidu.com/item/%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)或者[文件系统](http://baike.baidu.com/item/%E6%96%87%E4%BB%B6%E7%B3%BB%E7%BB%9F" \t "http://baike.baidu.com/_blank)的[HTML文件](http://baike.baidu.com/item/HTML%E6%96%87%E4%BB%B6" \t "http://baike.baidu.com/_blank)（[标准通用标记语言](http://baike.baidu.com/item/%E6%A0%87%E5%87%86%E9%80%9A%E7%94%A8%E6%A0%87%E8%AE%B0%E8%AF%AD%E8%A8%80" \t "http://baike.baidu.com/_blank)的一个应用）内容，并让用户与这些文件交互的一种[软件](http://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6" \t "http://baike.baidu.com/_blank)。

它用来显示在[万维网](http://baike.baidu.com/item/%E4%B8%87%E7%BB%B4%E7%BD%91" \t "http://baike.baidu.com/_blank)或局域网等内的文字、图像及其他信息。这些文字或图像，可以是连接其他网址的超链接，用户可迅速及轻易地浏览各种信息。大部分网页为[HTML](http://baike.baidu.com/item/HTML" \t "http://baike.baidu.com/_blank)格式。

国内网民计算机上常见的网页浏览器有，[QQ浏览器](http://baike.baidu.com/item/QQ%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[Internet Explorer](http://baike.baidu.com/item/Internet%20Explorer" \t "http://baike.baidu.com/_blank)、[Firefox](http://baike.baidu.com/item/Firefox" \t "http://baike.baidu.com/_blank)、[Safari](http://baike.baidu.com/item/Safari" \t "http://baike.baidu.com/_blank)，[Opera](http://baike.baidu.com/item/Opera/25140" \t "http://baike.baidu.com/_blank)、[Google Chrome](http://baike.baidu.com/item/Google%20Chrome" \t "http://baike.baidu.com/_blank)、[百度浏览器](http://baike.baidu.com/item/%E7%99%BE%E5%BA%A6%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[搜狗浏览器](http://baike.baidu.com/item/%E6%90%9C%E7%8B%97%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[猎豹浏览器](http://baike.baidu.com/item/%E7%8C%8E%E8%B1%B9%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[360浏览器](http://baike.baidu.com/item/360%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[UC浏览器](http://baike.baidu.com/item/UC%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[傲游浏览器](http://baike.baidu.com/item/%E5%82%B2%E6%B8%B8%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、[世界之窗浏览器](http://baike.baidu.com/item/%E4%B8%96%E7%95%8C%E4%B9%8B%E7%AA%97%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)等，浏览器是最经常使用到的客户端程序。

移动端产品有（移动端的浏览器）：百度、搜狗、UC、腾讯

内核：

IE内核。包括[360安全浏览器](http://baike.baidu.com/item/360%E5%AE%89%E5%85%A8%E6%B5%8F%E8%A7%88%E5%99%A8" \t "http://baike.baidu.com/_blank)、IE、 Greenbrowser、 Maxthon2、世界之窗、刚开始的搜狗浏览器。

Chrome内核，如 Chrome浏览器。

双核(IE和chrome/webkit内核).。双核的意思是一般网页用chrome内核(即webkit或高速模式)打开，网银等指定的网页用IE内核打开。 如360高速浏览器,搜狗高速浏览器，并不是1个网页同时用2个内核处理。

Firefox。

二、浏览器的主要构成（High Level Structure）

　　浏览器的主要组件包括：

　　1. 用户界面 － 包括地址栏、后退/前进按钮、书签目录等，也就是你所看到的除了用来显示你所请求页面的主窗口之外的其他部分。

　　2. 浏览器引擎 － 用来查询及操作渲染引擎的接口。

　　3. 渲染引擎 － 用来显示请求的内容，例如，如果请求内容为html，它负责解析html及css，并将解析后的结果显示出来。

　　4. 网络 － 用来完成网络调用，例如http请求，它具有平台无关的接口，可以在不同平台上工作。

　　5. UI后端 － 用来绘制类似组合选择框及对话框等基本组件，具有不特定于某个平台的通用接口，底层使用操作系统的用户接口。

　　6. JS解释器 － 用来解释执行JS代码。

1. 数据存储 － 属于持久层，浏览器需要在硬盘中保存类似cookie的各种数据，HTML5定义了web database技术，这是一种轻量级完整的客户端存储技术

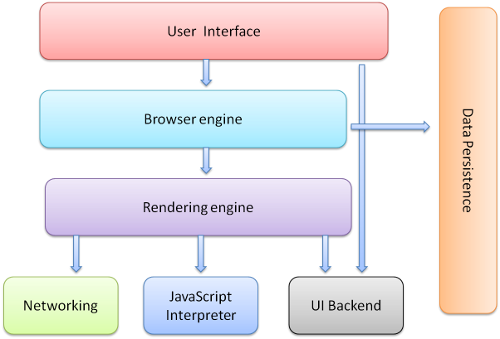


图1：浏览器主要组件

　　需要注意的是，不同于大部分浏览器，Chrome为每个Tab分配了各自的渲染引擎实例，每个Tab就是一个独立的进程。

　　对于构成浏览器的这些组件，后面会逐一详细讨论。

三、渲染引擎（The rendering engine）

　　渲染引擎的职责就是渲染，即在浏览器窗口中显示所请求的内容。

　　默认情况下，渲染引擎可以显示html、xml文档及图片，它也可以借助插件（一种浏览器扩展）显示其他类型数据，例如使用PDF阅读器插件，可以显示PDF格式，将由专门一章讲解插件及扩展，这里只讨论渲染引擎最主要的用途——显示应用了CSS之后的html及图片

详情参照：http://blog.csdn.net/finish\_dream/article/details/52304276

#### 7、服务器

一、介绍

服务器，也称伺服器，是提供计算服务的设备。由于服务器需要响应服务请求，并进行处理，因此一般来说服务器应具备承担服务并且保障服务的能力。

服务器的构成包括[处理器](http://baike.baidu.com/item/%E5%A4%84%E7%90%86%E5%99%A8" \t "http://baike.baidu.com/_blank)、[硬盘](http://baike.baidu.com/item/%E7%A1%AC%E7%9B%98" \t "http://baike.baidu.com/_blank)、[内存](http://baike.baidu.com/item/%E5%86%85%E5%AD%98" \t "http://baike.baidu.com/_blank)、[系统](http://baike.baidu.com/item/%E7%B3%BB%E7%BB%9F" \t "http://baike.baidu.com/_blank)[总线](http://baike.baidu.com/item/%E6%80%BB%E7%BA%BF" \t "http://baike.baidu.com/_blank)等，和通用的计算机架构类似，但是由于需要[提供](http://baike.baidu.com/item/%E6%8F%90%E4%BE%9B" \t "http://baike.baidu.com/_blank)高可靠的服务，因此在处理能力、稳定性、可靠性、安全性、可扩展性、可管理性等方面要求较高。

在网络环境下，根据服务器提供的服务类型不同，分为文件服务器，[数据库服务器](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)，应用程序服务器，WEB服务器等。

二、分类：

按照体系架构来区分，服务器主要分为两类：非x86服务器和x86服务器

按应用层次划分：**1、入门级服务器2、工作组服务器3、部门级服务器4、企业级服务器5、典型服务器应用（**办公[OA](http://baike.baidu.com/item/OA/25368" \t "http://baike.baidu.com/_blank)服务器、ERP服务器、[WEB服务器](http://baike.baidu.com/item/WEB%E6%9C%8D%E5%8A%A1%E5%99%A8" \t "http://baike.baidu.com/_blank)、[数据库](http://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93" \t "http://baike.baidu.com/_blank)服务器、财务服务器、打印服务器、集群服务器、视频监控服务器、[游戏](http://baike.baidu.com/item/%E6%B8%B8%E6%88%8F" \t "http://baike.baidu.com/_blank)服务器、论坛服务器）

1. 特性：1、可扩展性2、易使用性3、可用性4、易管理性
2. 外形：1、机架式2、刀片3、塔式4、机柜式
3. 操作系统：服务器平台的操作系统。Unix[操作系统](http://baike.baidu.com/item/%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F" \t "http://baike.baidu.com/_blank)，由于是Unix的后代，大多都有较好的作服务器平台的功能。

#### 8、线程与进程的区别

进程：

狭义定义：进程是正在运行的程序的实例。广义定义：进程是一个具有一定独立功能的程序关于某个数据集合的一次运行活动。它是[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3rHR1n1ubnywbmWf4P1b0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrjbLPjc3nHb1nHf1PWbdnjms" \t "https://zhidao.baidu.com/question/_blank)动态执行的基本单元，在传统的[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3rHR1n1ubnywbmWf4P1b0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrjbLPjc3nHb1nHf1PWbdnjms" \t "https://zhidao.baidu.com/question/_blank)中，进程既是基本的[分配单元](https://www.baidu.com/s?wd=%E5%88%86%E9%85%8D%E5%8D%95%E5%85%83&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3rHR1n1ubnywbmWf4P1b0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBtkrjbLPjc3nHb1nHf1PWbdnjms" \t "https://zhidao.baidu.com/question/_blank)，也是基本的执行单元。进程是一个实体，进程是一个“执行中的程序”

进程的特点：并发性：任何进程都可以同其他进程一起并发执行

独立性：进程是一个能独立运行的基本单位，同时也是系统分配资源和调度的独立单位；

异步性：由于进程间的相互制约，使进程具有执行的间断性，即进程按各自独立的、不可预知的速度向前推进

结构特征：进程由程序、数据和进程控制块三部分组成。

多个不同的进程可以包含相同的程序：一个程序在不同的数据集里就构成不同的进程，能得到不同的结果；但是执行过程中，程序不能发生改变。

线程：

线程是进程中的实体，一个进程可以拥有多个线程，一个线程必须有一个父进程。线程不拥有系统资源，只有运行必须的一些数据结构；它与父进程的其它线程共享该进程所拥有的全部资源。线程可以创建和撤消线程，从而实现程序的并发执行。一般，线程具有就绪、阻塞和运行三种基本状态。

有时候，线程也称作轻量级进程。就象进程一样，线程在程序中是独立的、并发的执行路径，每个线程有它自己的堆栈、自己的程序计数器和自己的局部变量。但是，与分隔的进程相比，进程中的线程之间的隔离程度要小。它们共享内存、文件句柄和其它每个进程应有的状态。

线程的基本操作：

派生：线程在进程内派生出来，它即可由进程派生，也可由线程派生。

阻塞（Block）：如果一个线程在执行过程中需要等待某个事件发生，则被阻塞。

激活（unblock）：如果阻塞线程的事件发生，则该线程被激活并进入就绪队列。

调度（schedule）：选择一个就绪线程进入执行状态。

结束（Finish）：如果一个线程执行结束，它的寄存器上下文以及堆栈内容等将被释放。

二者的区别：

1）地址空间和其它资源（如打开文件）：进程间相互独立，同一进程的各线程间共享。某进程内的线程在其它进程不可见。

2）通信：进程间通信IPC，线程间可以直接读写进程数据段（如全局变量）来进行通信——需要进程同步和互斥手段的辅助，以保证数据的一致性。

3）调度和切换：线程上下文切换比进程上下文切换要快得多。

4）在多线程OS中，进程不是一个可执行的实体。

补充了解：一个程序至少有一个进程,一个进程至少有一个线程. 线程的划分尺度小于进程，使得多线程程序的并发性高。另外，进程在执行过程中拥有独立的内存单元，而多个线程共享内存，从而极大地提高了程序的运行效率。

线程在执行过程中与进程还是有区别的。每个独立的线程有一个程序运行的入口、顺序执行序列和程序的出口。但是线程不能够独立执行，必须依存在应用程序中，由应用程序提供多个线程执行控制。

从逻辑角度来看，多线程的意义在于一个应用程序中，有多个执行部分可以同时执行。但操作系统并没有将多个线程看做多个独立的应用，来实现进程的调度和管理以及资源分配。这就是进程和线程的重要区别。

进程是具有一定独立功能的程序关于某个数据集合上的一次运行活动,进程是系统进行资源分配和调度的一个独立单位.线程是进程的一个实体,是CPU调度和分派的基本单位,它是比进程更小的能独立运行的基本单位.线程自己基本上不拥有系统资源,只拥有一点在运行中必不可少的资源(如程序计数器,一组寄存器和栈),但是它可与同属一个进程的其他的线程共享进程所拥有的全部资源.

一个线程可以创建和撤销另一个线程;同一个进程中的多个线程之间可以并发执行.进程和线程的主要差别在于它们是不同的操作系统资源管理方式。进程有独立的地址空间，一个进程崩溃后，在保护模式下不会对其它进程产生影响，而线程只是一个进程中的不同执行路径。线程有自己的堆栈和局部变量，但线程之间没有单独的地址空间，一个线程死掉就等于整个进程死掉，所以多进程的程序要比多线程的程序健壮，但在进程切换时，耗费资源较大，效率要差一些。但对于一些要求同时进行并且又要共享某些变量的并发操作，只能用线程，不能用进程。

#### 9、经典编程算法

1. 快速排序算法

2、堆排序算法

3、归并排序

4、二分查找算法

5、BFPRT（线性查找算法）

6、DFS（深度优先搜索）

7、BFS（广度优先搜索）

8、Floyd-Warshall all-pairs最短路径算法

#### 10、经典排序算法

1、插入排序—直接插入排序(Straight Insertion Sort)

2、插入排序—希尔排序（Shell`s Sort）

3、选择排序—简单选择排序（Simple Selection Sort）

4、选择排序—堆排序（Heap Sort）

5、交换排序—冒泡排序（Bubble Sort）

6、快速排序（Quick Sort）

7、归并排序（Merge Sort）

8、桶排序/基数排序(Radix Sort)

9、堆排序

10、计数排序

#### 11、黑盒、白盒、灰盒测试

**白盒测试：**

白盒测试也称为结构测试或逻辑驱动测试，是针对被测单元内部是如何进行工作的测试。它根据程序的控制结构设计测试用例，主要用于软件或程序验证。白盒测试法检查程序内部逻辑结构，对所有的逻辑路径进行测试，是一种穷举路径的测试方法，但即使每条路径都测试过了，但仍然有可能存在错误。因为：穷举路径测试无法检查出程序本身是否违反了设计规范，即程序是否是一个错误的程序；穷举路径测试不可能检查出程序因为遗漏路径而出错；穷举路径测试发现不了一些与数据相关的错误。

白盒测试需要遵循的原则有： 1. 保证一个模块中的所有独立路径至少被测试一次；2. 所有逻辑值均需要测试真（true）和假（false）；两种情况；3. 检查程序的内部数据结构，保证其结构的有效性；4. 在上下边界及可操作范围内运行所有循环。

白盒测试方法有：

静态测试、动态测试、单元测试、代码检查、同行评审、技术评审

**黑盒测试：**

黑盒测试又称为功能测试、数据驱动测试或基于规格说明书的测试，是一种从用户观点出发的测试。测试人员一般把被测程序当作一个黑盒子。

　　黑盒测试主要测到的错误类型有：不正确或遗漏的功能；接口、界面错误；性能错误；数据结构或外部数据访问错误；初始化或终止条件错误等等。

　　常用的黑盒测试方法有：等价类划分法；边界值分析法；因果图法；场景法；正交实验设计法；判定表驱动分析法；错误推测法；功能图分析法。

两者之间的区别：

黑盒测试着重测试软件功能。

黑盒测试并不能取代白盒测试，它是与白盒测试互补的测试方法，它很可能发现白盒测试不易发现的其他类型错误。

**灰盒测试 (Gray-Box Testing)**

灰盒测试更像是白盒测试和黑盒测试的混合测试，现阶段对灰盒测试没有更明确的定义，但更多的时候，我们的测试做的就是灰盒测试，即既会做黑盒测试又会做白盒测试。

#### 12、二叉排序树

定义：二叉排序树或者是一棵空树，或者是具有下列性质的[二叉树](http://baike.baidu.com/item/%E4%BA%8C%E5%8F%89%E6%A0%91" \t "http://baike.baidu.com/_blank)：

（1）若左子树不空，则左子树上所有结点的值均小于或等于它的[根结](http://baike.baidu.com/item/%E6%A0%B9%E7%BB%93" \t "http://baike.baidu.com/_blank)点的值；

（2）若右子树不空，则右子树上所有结点的值均大于或等于它的根结点的值；

（3）左、右子树也分别为二叉排序树；

详见：<http://m.blog.csdn.net/yxb_yingu/article/details/51336197>

## 1.4 git使用

#### 1、关于git（简介）

Git是Linux之父Linus的第二个伟大的作品，它最早是在Linux上开发的，被用来管理Linux核心的源代码。后来慢慢地有人将其移植到了Unix、Windows、Max OS等操作系统中。

Git是一个分布式的版本控制系统，与集中式的版本控制系统不同的是，每个人都工作在通过克隆建立的本地版本库中。也就是说每个人都拥有一个完整的版本库，查看提交日志、提交、创建里程碑和分支、合并分支、回退等所有操作都直接在本地完成而不需要网络连接。

对于Git仓库来说，每个人都有一个独立完整的仓库，所谓的远程仓库或是服务器仓库其实也是一个仓库，只不过这台主机24小时运行，它是一个稳定的仓库，供他人克隆、推送，也从服务器仓库中拉取别人的提交。

Git是目前世界上最先进的分布式版本控制系统，没有之一，对，没有之一!

#### git常用命令总结

git init # 在本地新建一个仓库

git add # 将工作区的修改提交到暂存区

git commit # 将暂存区的修改提交到当前分支

git reset # 回退到某一个版本

git stash # 保存某次修改

git pull # 从远程更新代码

git push # 将本地代码更新到远程分支上

git reflog # 查看历史命令

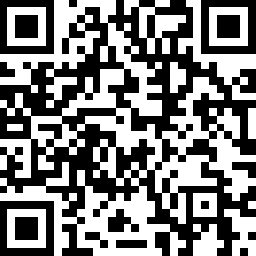
git status # 查看当前仓库的状态

git diff # 查看修改

git log # 查看提交历史

git revert # 回退某个修改

#### git命令详解请参考以下链接

<https://www.cnblogs.com/my--sunshine/p/7093412.html>

微信扫描二维码快速查看！

# 第二部分（框架部分）

**2.1、流行框架前端热潮知识**

### 前端框架与类库的区别。

**1、总括前端框架与类库的区别**

框架：(1)、框架是一种特殊的、已经实现了的WEB应用，你只需要对它填充具体的业务逻辑。这里框架是起主导作用的，由它来根据具体的应用逻辑来调用你的代码。

(2)封装好的代码，不过会功能更加多样，一个框架会包含多个类库，并且框架面向的顶层的开发，而类库更多的是面向底层的开发。

(3)、注重整体，框架帮助你解决“代码如何组织”的问题，面对中/大型项目，特别是需要多人共同参与的项目，选择一个合适的框架有助于写出规范的。

(4)、目前常见的前端框架：Bootstrap、angular.js、vue.js、react.js、node.js、amaze UI

类库：(1)、类库是一些函数的集合，通过调用开放出来的API获取相应的功能

它能帮助你写WEB应用。起主导作用的是你的代码，由你来决定何时使用类库。

封装好的代码方法

(2)、注重细节，类库帮助你解决“如何把代码写得更少/巧/强壮”的问题

对于小型应用来说，因为[业务逻辑](https://www.baidu.com/s?wd=%E4%B8%9A%E5%8A%A1%E9%80%BB%E8%BE%91&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLPv7WrjmdnyP9nyfsuym0IAYqnWm3PW64rj0d0AP8IA3qPjfsn1bkrjKxmLKz0ZNzUjdCIZwsrBtEXh9GuA7EQhF9pywdQhPEUiqkIyN1IA-EUBt1njn4rjf1n104PHbYrj64nWf" \t "https://zhidao.baidu.com/question/_blank)简单，代码总量不会太大，组织不会是太大得问题，所以用类库就够了。

(3)、常见的类库：jquery、zepto

**2、jquery和javascript的区别？**

jQuery是JavaScript的一个封装集合。封装了很多JavaScript的方法，也就是说，jQuery里面的内容都是JavaScript语句。只是封装起来让我们学习和使用的简单一些.

JavaScript的优点和缺点：

优点： 1、性能：由于JavaScript运行在客户端，节省了web服务器的请求时间和带宽2、轻量级的脚本语言，比较容易学习3、运行在用户机器上，运行结果和处理相对比较快。4、可以使用第三方附加组件来检查代码片段。

缺点： 1、安全问题：由于JavaScript在客户端运行，可能被用于黑客目的。2渲染问题：在不同浏览器中的处理结果可能不同。

jQuery的优点和缺点：

优点:1、使用jQuery最大的好处是少量的代码做更多的事情。与JavaScript相比，jQuery的语法更加简单。通过jQuery，可以很容易地浏览文档、选择元素、处理事件以及添加效果等，同时还允许开发者定制插件。 2、jQuery消除了JavaScript跨平台兼容问题。相比其他JavaScript和JavaScript库，jQuery更容易使用。jQuery有一个庞大的库/函数。jQuery有良好的文档和帮助手册。jQuery支持AJAX。

缺点： 由于不是原生JavaScript语言，理解起来可能会受到限制。项目中需要包含jQuery库文件。如果包含多个版本的jQuery库，可能会发生冲突。

**3、谈谈小程序的双向绑定和vue的异同？**

但小程序直接this.data的属性是不可以同步到视图的，必须调用this.setData()方法！

### 2.1.2、VUE

1. **VUE的适用场景**

对浏览器兼容要求不高，vuejs是到IE9；

对MVVM有一定的经验；

加载速度要求高；

对性能要求比较高；

需要组件化开发；

喜欢对原生js对象操作；

SPA

使用vue的项目有：苏宁易购触屏版的购物车结算页面、有桌面端（比如 [饿了么安全应急响应中心](https://link.zhihu.com/?target=https://security.ele.me" \t "https://www.zhihu.com/question/_blank)）也有移动端（比如 [饿了么招聘](https://link.zhihu.com/?target=https://jobs-mobile.ele.me" \t "https://www.zhihu.com/question/_blank)）

1. **VUE是什么**

vue.js（读音 /vjuː/，类似于 view） 是一套构建用户界面的渐进式框架

与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与[单文件组件](http://cn.vuejs.org/v2/guide/single-file-components.html)和 [Vue 生态系统支持的库](http://github.com/vuejs/awesome-vue" \l "libraries--plugins)结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

1. **Vue父子组件间传值**

具体操作流程：http://www.cnblogs.com/daiwenru/p/6694530.html

总结：子组件中需要以某种方式例如点击事件的方法来触发一个自定义事件

将需要传的值作为其自定义事件的第二个参数，该值将作为实参传给响应自定义事件的方法

在父组件中注册子组件并在子组件标签上绑定对自定义事件的监听

1. **React 和 Vue 的区别**

**React 和 Vue 的相似之处：**

（1）、使用 Virtual DOM

（2）、提供了响应式（Reactive）和组件化（Composable）的视图组件。

（3）、将注意力集中保持在核心库，而将其他功能如路由和全局状态管理交给相关的库。

**Vue与Angular、React的对比：**

1、vue.js更轻量，gzip后只有20K+，angular:56K ,react:44K

2、vue.js更易上手，学习曲线平稳

3、吸收两家之长，有angular的指令和react组件化思想

**优缺点：**

优点：体积小。接口灵活。侵入性好，可用于页面的一部分，而不是整个页面。扩展性好。源码规范简洁。代码较为活跃，作者是中国人，可在官方论坛中文提问。github9000+。基于组件化的开发。它是一个轻量级mvvm框架、数据驱动+组件化的前端开发、社区完善

缺点：社区不大，如果有问题可以读源码。功能仅限于view层，Ajax等功能需要额外的库。对开发人员要求较高。开发的话，需要webpack，不然很难用，最好配合es6。

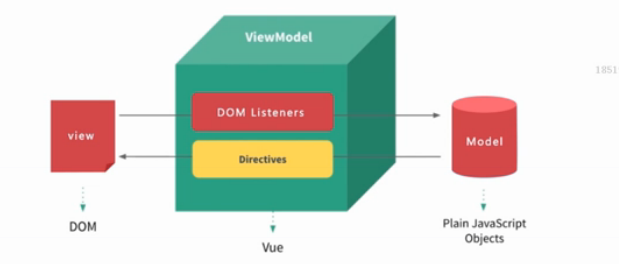
1. **Vue.js核心思想：**

1、数据驱动（基于操作dom的方式）

（1）、Dom是数据的一种自然映射（双向数据绑定）手动改变DOM非常麻烦使用vue.js之后，只需要改变数据，通过改变Directives指令,当数据发生变化，会通过数据指令去修改对应的DOM

（2）、Vue.js还会对事件进行一定的监听，当我们改变视图（view）的时候通过DOM Listeners来改变数据

通过以上两点就实现了数据的双向绑定



2、组件化

Vue-cli:(它是vue的脚手架工具)

作用：帮助我们完成基础的代码（包括：目录结构、本地调试、代码部署、热加载、单元测试）

组件的设计原则：

页面上每一个独立的可视/可交互区域视为一个组件

每个组件对应一个工程目录，组件所需要的各种资源在这个目录下就近维护

展示面不过是组件的容器，组件可以嵌套自由组合形成完整的页面

1. **什么是MVVM：**

MVVM是Model-View-ViewModel的缩写。MVVM是一种设计思想。Model 层代表数据模型，也可以在Model中定义数据修改和操作的业务逻辑；View 代表UI 组件，它负责将数据模型转化成UI 展现出来，ViewModel 是一个同步View 和 Model的对象。

在MVVM架构下，View 和 Model 之间并没有直接的联系，而是通过ViewModel进行交互，Model 和 ViewModel 之间的交互是双向的， 因此View 数据的变化会同步到Model中，而Model 数据的变化也会立即反应到View 上。

ViewModel 通过双向数据绑定把 View 层和 Model 层连接了起来，而View 和 Model 之间的同步工作完全是自动的，无需人为干涉，因此开发者只需关注业务逻辑，不需要手动操作DOM, 不需要关注数据状态的同步问题，复杂的数据状态维护完全由 MVVM 来统一管理。

1. **MVVM和MVC的区别：**

mvc和mvvm其实区别并不大。都是一种设计思想。主要就是mvc中Controller演变成mvvm中的viewModel。mvvm主要解决了mvc中大量的DOM 操作使页面渲染性能降低，加载速度变慢，影响用户体验。

1. **MVVM和jquery的区别：**

* vue数据驱动，通过数据来显示视图层而不是节点操作。
* jQuery是一个快速、简洁的JavaScript框架 , jQuery是基于事件驱动

1. **Vue的优点是什么：**

* 低耦合。视图（View）可以独立于Model变化和修改，一个ViewModel可以绑定到不同的"View"上，当View变化的时候Model可以不变，当Model变化的时候View也可以不变。
* 可重用性。你可以把一些视图逻辑放在一个ViewModel里面，让很多view重用这段视图逻辑。
* 独立开发。开发人员可以专注于业务逻辑和数据的开发（ViewModel），设计人员可以专注于页面设计。
* 可测试。界面素来是比较难于测试的，而现在测试可以针对ViewModel来写。

1. **Vue组件之间的传值：**

* 父组件往子组件传值props
* 子组件往父组件传值，通过emit事件
* 不同组件之间传值，通过eventBus（小项目少页面用eventBus，大项目多页面使用 vuex）

1. **Vue路由之间跳转有哪些：**

* 声明式（标签跳转）
* 编程式（ js跳转）

1. **vue-cli怎样使用自定义组件? 遇到过哪些问题：**

第一步：在components目录新建你的组件文件（indexPage.vue），script一定要export default {}

第二步：在需要用的页面（组件）中导入：import indexPage from '@/components/indexPage.vue'

第三步：注入到vue的子组件的components属性上面,components:{indexPage}

第四步：在template视图view中使用，

例如有indexPage命名，使用的时候则index-page

1. **vue如何实现按需加载配合webpack设置：**

webpack中提供了require.ensure()来实现按需加载。以前引入路由是通过import 这样的方式引入，改为const定义的方式进行引入。

不进行页面按需加载引入方式import home from ../../common/home.vue'

进行页面按需加载的引入方式：const home = r => require.ensure( [], () => r (require('../../common/home.vue')))

1. **vuex是什么？怎么使用？哪种功能场景使用它？**

vuex是vue框架中状态管理。

使用方式: 在main.js引入store，注入。新建一个目录store，….. export 。

场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

1. **vuex有哪几种属性？**

有五种，分别是 State、 Getter、Mutation 、Action、 Module

* vuex的State特性
  + A、Vuex就是一个仓库，仓库里面放了很多对象。其中state就是数据源存放地，对应于一般Vue对象里面的data
  + B、state里面存放的数据是响应式的，Vue组件从store中读取数据，若是store中的数据发生改变，依赖这个数据的组件也会发生更新
  + C、它通过mapState把全局的 state 和 getters 映射到当前组件的 computed 计算属性中
* vuex的Getter特性
  + A、getters 可以对State进行计算操作，它就是Store的计算属性
  + B、 虽然在组件内也可以做计算属性，但是getters 可以在多组件之间复用
  + C、 如果一个状态只在一个组件内使用，是可以不用getters
* vuex的Mutation特性
  + Action 类似于 mutation，不同在于：Action 提交的是 mutation，而不是直接变更状态；Action 可以包含任意异步操作。
* vuex的Action特性
  + 包含任意异步操作，通过提交 mutation 间接更变状态
* vuex的Module特性
  + 将 store 分割成模块，每个模块都具有state、mutation、action、getter、甚至是嵌套子模块

1. **不用Vuex会带来什么问题？**

* 可维护性会下降，想修改数据要维护三个地方；
* 可读性会下降，因为一个组件里的数据，根本就看不出来是从哪来的；
* 增加耦合，大量的上传派发，会让耦合性大大增加，本来Vue用Component就是为了减少耦合，现在这么用，和组件化的初衷相背。

1. **v-show和v-if指令的共同点和不同点？**

* v-show指令是通过修改元素的display的CSS属性让其显示或者隐藏
* v-if指令是直接销毁和重建DOM达到让元素显示和隐藏的效果

1. **如何让CSS只在当前组件中起作用？**

将当前组件的<style>修改为<style scoped>

1. **<keep-alive></keep-alive>的作用是什么?**

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例，主要用于保留组件状态或避免重新渲染。

1. **Vue中引入组件的步骤?**
2. 采用ES6的import ... from ...语法或CommonJS的require()方法引入组件

2) 对组件进行注册

vue.component('My-component',{

template:"<div>my name is template!</div>"

});

3) 使用组件<my-component></my-component>

1. **指令v-el的作用是什么?**

提供一个在页面上已存在的 DOM 元素作为 Vue 实例的挂载目标.可以是 CSS 选择器，也可以是一个 HTMLElement 实例 , vue2.0当中改为ref属性

1. **在Vue中使用插件的步骤?**

1) 采用ES6的import ... from ...语法或CommonJSd的require()方法引入插件

2} 使用全局方法Vue.use( plugin )使用插件,可以传入一个选项对象Vue.use(MyPlugin, { someOption: true })

1. **请列举出3个Vue中常用的生命周期钩子函数?（举两例）**

* created: 实例已经创建完成之后调用,在这一步,实例已经完成数据观测, 属性和方法的运算, watch/event事件回调. 然而, 挂载阶段还没有开始, $el属性目前还不可见
* mounted: el被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。如果 root 实例挂载了一个文档内元素，当 mounted 被调用时 vm.$el 也在文档内。

1. **active-class是哪个组件的属性？**

vue-router模块的router-link组件。

1. **怎么定义vue-router的动态路由以及如何获取传过来的动态参数？**

* 在router目录下的index.js文件中，对path属性加上/:id。
* 使用router对象的params.id。

1. **vue-router有哪几种导航钩子？**
   * 1. 全局导航钩子：router.beforeEach(to,from,next)

作用：跳转前进行判断拦。

2) 组件内的钩子；

3) 单独路由独享组件

1. **vue生命周期？**

Vue生命周期总共分为8个阶段：创建前/后，载入前/后，更新前/后，销毁前/后。

官网链接：

[https://cn.vuejs.org/v2/guide/instance.html#生命周期图示](https://cn.vuejs.org/v2/guide/instance.html" \l "生命周期图示)

* 创建前/后： 在beforeCreate阶段，vue实例的挂载元素el和数据对象data都为undefined，还未初始化。在created阶段，vue实例的数据对象data有了，el还没有。
* 载入前/后：在beforeMount阶段，vue实例的$el和data都初始化了，但还是挂载之前为虚拟的dom节点，data.message还未替换。在mounted阶段，vue实例挂载完成，data.message成功渲染。
* 更新前/后：当data变化时，会触发beforeUpdate和updated方法。
* 销毁前/后：在执行destroy方法后，对data的改变不会再触发周期函数，说明此时vue实例已经解除了事件监听以及和dom的绑定，但是dom结构依然存在

1. **什么是vue生命周期？**

Vue 实例从创建到销毁的过程，就是生命周期。也就是从开始创建、初始化数据、编译模板、挂载Dom→渲染、更新→渲染、卸载等一系列过程，我们称这是 Vue 的生命周期。

1. **vue生命周期的作用是什么？**

它的生命周期中有多个事件钩子，让我们在控制整个Vue实例的过程时更容易形成好的逻辑。

1. **第一次页面加载会触发哪几个钩子？**

第一次页面加载时会触发 beforeCreate, created, beforeMount, mounted 这几个钩子。

1. **DOM 渲染在 哪个周期中就已经完成？**

DOM 渲染在 mounted 中就已经完成了。

1. **简单描述每个周期具体适合哪些场景？**

* beforecreate : 可以在这加个loading事件，在加载实例时触发
* created : 初始化完成时的事件写在这里，如在这结束loading事件，异步请求也适宜在这里调用
* mounted : 挂载元素，获取到DOM节点
* updated : 如果对数据统一处理，在这里写上相应函数
* beforeDestroy : 可以做一个确认停止事件的确认框

1. **说出至少4种vue当中的指令和它的用法？**

* v-if：判断是否隐藏；
* v-for：数据循环；
* v-bind:class：绑定一个属性；
* v-model：实现双向绑定

1. **vue-loader是什么？使用它的用途有哪些？**

* vue-loader是解析.vue文件的一个加载器。
* 用途：js可以写es6、style样式可以scss或less、template可以加jade等。

1. **scss是什么？**

css的预编译

1. **scss在vue-cli中的安装使用步骤是？**

* 第一步：先装css-loader、node-loader、sass-loader等加载器模块
* 第二步：在build目录找到webpack.base.config.js，在那个extends属性中加一个拓展.scss
* 第三步：在同一个文件，配置一个module属性
* 第四步：然后在组件的style标签加上lang属性 ，例如：lang=”scss”

1. **scss有哪几大特性？**

* 可以用变量，例如（$变量名称=值）；
* 可以用混合器，例如（）
* 可以嵌套

1. **v-for为什么使用key？**

因为vue在更新渲染dom的时候是根据新旧dom数进行对比的，使用key来给每个节点做一个唯一标识，Diff算法就可以正确的识别此节点，找到正确的位置区插入新的节点。

1. **为什么避免 v-if 和 v-for 用在一起？**

当 Vue 处理指令时，v-for 比 v-if 具有更高的优先级，通过v-if 移动到容器元素，不会再重复遍历列表中的每个值。取而代之的是，我们只检查它一次，且不会在 v-if 为否的时候运算 v-for。

1. **VNode是什么？**

Vue在 页面上渲染的节点，及其子节点称为“虚拟节点 (Virtual Node)”，简写为“VNode”。

1. **虚拟 DOM是什么？**

“虚拟 DOM”是由 Vue 组件树建立起来的整个 VNode 树的称呼。

1. **vue中利用索引修改数组的时候，页面会跟着同步吗？**

利用索引修改数组的时候，页面不会进行同步，此时应该利用vue.set的方法进行设置数据

1. **vue首屏加载过慢如何解决？**

* 路由懒加载，会将原来打包一个app.js的文件打包成多个文件
* 异步组件，按需加载
* 组件异步加载，将子组件
* webpack开启gzip压缩
* 如果图片过多，开启图片懒加载
* 使用cdn资源
* 如果首页是登录页，做多入口

1. **路由中如何去除url上的‘#’？**

路由有两种模式，一种为hash模式，另一种为history模式，开启history模式后自动去除#，开启history模式需要后台配合

官网地址：<https://router.vuejs.org/zh/guide/essentials/history-mode.html>

1. **vue中的单项数据流和双向数据绑定是什意思？**

* 单项数据流是指数据是单向的，父组件的数据传递给子组件，只能单项绑定，不可以在子组件修改父组件的数据
* 双向数据绑定：是指数据和页面进行双向绑定，相互影响

1. **vue中双向数据绑定的原理是什么？**

vue双向数据绑定的原理主要通过数据劫持Object.defineProperty和发布订阅模式实现的，通过Object.defineProperty监听数据发生变化然后通知订阅者（watcher），订阅者触发响应的回调

1. **为什么vue组件中的data必须是函数？**

因为如果默认为data是对象的话，对象为引用类型，这样的话，所有复用的组件都是引用的同一个数据，但是如果是函数的话，每次函数都会先创建一个新的数据，从而使每个组件的数据独立

1. **你知道webpack中babel、plugin、loader都有什么作用吗？**

* babel用来出来es6转es5
* plugin配置webpack的一些插件
* loader用来配置解析处理第三方文件的

1. **keep-alive的作用？**

<keep-alive></keep-alive> 包裹动态组件时，会缓存不活动的组件实例，主要用于保留组件状态或避免重新渲染。

1. **$route和router的区别？**

* $route : 包括path，params，hash，query，fullPath，matched，name等路由信息参数
* $router: 是路由的跳转方法，钩子函数等

### **2.1.3、React**

**1、谈谈你对 React 的了解？**

React 是用于构建前端页面的 JavaScript 库。React 主要用于构建 UI，由 Facebook 于 2013 年开源。

React 特点：

1. 声明式设计 −React 采用声明范式，可以轻松描述应用。
2. 高效 −React 通过对 DOM 的模拟，最大限度地减少与 DOM 的交互。
3. 灵活 −React 可以与已知的库或框架很好地配合。
4. JSX − JSX 是 JavaScript 扩展语法。React 开发不一定使用 JSX, 但我们建议使用它。
5. 组件 − 通过 React 构建组件，代码更容易复用，能够很好的应用在大项目的开发中。
6. 单向数据流 − React 实现了单向响应的数据流，从而减少了重复代码，这也是它为什么 比传统数据绑定更简单。

**2、什么是 JSX？为什么浏览器无法读取 JSX？（必会）**

JSX 是 JavaScript XML 的简写，它利用 JavaScript 的表现力和类似 HTML 的模板语法，得 HTML 文件非常容易理解。此文件能使应用非常可靠，并能够提高其性能。浏览器只能处理 JavaScript 对象，而不能读取常规 JavaScript 对象中的 JSX，所以为了使浏览器能够读取 JSX，首先，需要用 Babel 转换器将 JSX 文件转换为 JavaScript 对象，然后再将其传给浏览器

**3、shouldComponentUpdate 是做什么的？（必会）**

shouldComponentUpdate 这个方法用来判断是否需要调用 render 方法重新绘制 dom，因为 DOM 的描绘非常消耗性能，如果我们能在shouldComponentUpdate 方法中能够写出更优化的 dom diff 算法，可以极大的提高性能。

**4、React 性能优化是哪个周期函数？（必会）**

shouldComponentUpdate

**5、React 中 keys 的作用是什么？（必会）**

Keys 是 React 用于追踪哪些列表中元素被修改、被添加或者被移除的辅助标识。在开发过程中，我们需要保证某个元素的 key 在其同级元素中具有唯一性。

在 React Diff 算法中 React 会借助元素的 Key 值来判断该元素是新近创建的还是被移动而来的元素，从而减少不必要的元素重渲染；

此外，React 还需要借助 Key 值来判断元素与本地状态的关联关系，因此我们绝不可忽视转换函数中 Key 的重要性。

**6、React 中 refs 的作用是什么？（必会）**

Refs 是 React 提供给我们的安全访问 DOM 元素或者某个组件实例的句柄，我们可以为元素添加 ref 属性然后在回调函数中接受该元素在 DOM 树中的句柄，该值会作为回调函数的第一个参数返回。

**7、请列举 React 中定义组件的方法？（必会）**

1. 函数式定义的无状态组件

2. es6 中 extends React.Component 定义的组件

**8、调用 setState 之后发生了什么？（必会）**

1. 代码中调用 setState 函数之后，React 会将传入的参数对象与组件当前的状态合并，然后触发所谓的调和过程（Reconciliation）。

2. 经过调和过程，React 会以相对高效的方式根据新的状态构建 React 元素树并且着手重新渲染整个 UI 界面；

3. 在 React 得到元素树之后，React 会自动计算出新的树与老树的节点差异，然后根据差异对界面进行最小化重渲染；

4. 在差异计算算法中，React 能够相对精确地知道哪些位置发生了改变以及应该如何改变，这就保证了按需更新，而不是全部重新渲染。

**9、你怎么理解 redux 的 state 的？（必会）**

Store 对象包含所有数据。如果想得到某个时点的数据，就要对 Store 生成快照。这种时点的数据集合，就叫做 State。 当前时刻的 State，可以通过 store.getState() 拿到。唯一改变 state 的方法就是触发 action

**10、除了在构造函数中绑定 this，还有其它方式吗？（必会）**

你可以使用属性初始值设定项(property initializers)来正确绑定回调，create-React-app 也是默认支持的。在回调中你可以使用箭头函数，但问题是每次组件渲染时都会创建一个新的回调

**11、(在构造函数中)调用 super(props)的目的是什么？（必会）**

在 super() 被调用之前，子类是不能使用 this 的，在 ES2015 中，子类必须在 constructor 中调用 super()。传递 props 给 super() 的原因则是便于(在子类中)能在 constructor 访问 this.props

**12、简述 flux 思想？（必会）**

1. 用户访问 View

2. View 发出用户的 Action

3. Dispatcher 收到 Action,要求 Store 进行相应的更新

4. Store 更新后，发出一个"change"事件

5. View 收到"change"事件后，更新页面

**13、事件在 React 中的处理方式？（必会）**

React 元素的事件处理和 DOM 元素类似。但是有一点语法上的不同

1. React 事件绑定属性的命名采用驼峰式写法，而不是小写。

2. 如果采用 JSX 的语法你需要传入一个函数作为事件处理函数，而不是一个字符串(DOM 元素的写法)

**14、列出 Redux 的核心方法？（必会）**

1. Action – 这是一个用来描述发生了什么事情的对象

2. Reducer – 这是一个确定状态将如何变化的地方

3. Store – 整个程序的状态/对象树保存在 Store 中

4. View – 只显示 Store 提供的数据

**15、(组件的)状态(state)和属性(props)之间有何不同？（必会）**

State 是一种数据结构，用于组件挂载时所需数据的默认值。State 可能会随着时间的推移而发生突变，但多数时候是作为用户事件行为的结果。Props(properties 的简写)则是组件的配置。

props 由父组件传递给子组件，并且就子组件而言，props 是不可变的,组件不能改变自身的 props，但是可以把其子组件的 props 放在一起(统一管理)

**16、何为受控组件(controlledcomponent) ？（必会）**

在 HTML 中，类似 <input>,<textarea>和<select> 这样的表单元素会维护自身的状态，并基于用户的输入来更新，当用户提交表单时，前面提到的元素的值将随表单一起被发送，但在 React中会有些不同，包含表单元素的组件将会在 state 中追踪输入的值，并且每次调用回调函数时，如 onChange 会更新 state，重新渲染组件，一个输入表单元素，它的值通过 React 的这种方式来控制，这样的元素就被称为”受控元素 ”

**17、何为高阶组件(higherordercomponent) ？（必会）**

高阶组件是一个以组件为参数并返回一个新组件的函数。HOC 运行你重用代码、逻辑和引导抽象，最常见的可能是 Redux 的 connect 函数，除了简单分享工具库和简单的组合，HOC 最好的方式是共享 React 组件之间的行为，如果你发现你在不同的地方写了大量代码来做同一件事时，就应该考虑将代码重构为可重用的 HOC

**18、React 中组件如何进行数据传值？（必会）**

1. 父级传递子级：把数据挂载子组件的属性上，子组件通过 this.props 来接收父组件的数据。

2. 子级传递父级：父级需要定义一个修改数据的方法，把修改数据的方法传给子组件，当子组件需要修改父级数据时，调用父级传过来的修改方法

3. 兄弟组件传递：属于同一个父级，父组件分别和这两个组件传递。比如子组件 A 操作执行父组件方法，父组件进行修改，然后把信息传给子组件 B

4. Context 跨组件传递数据：顶级组件向最里面组件进行传值

**19、解释 Reducer 的作用（必会）**

Reducers 是纯函数，它规定应用程序的状态怎样因响应 ACTION 而改变。Reducers 通过接受先前的状态和 action 来工作，然后它返回一个新的状态。它根据操作的类型确定需要执行哪种更新，然后返回新的值。如果不需要完成任务，它会返回原来的状态

**20、redux 有什么缺点（必会）**

一个组件所需要的数据，必须由父组件传过来，而不能像 flux 中直接从 store 取；当一个组件相关数据更新时，即使父组件不需要用到这个组件，父组件还是会重新 render，可能会有效率影响，或者需要写复杂的 shouldComponentUpdate 进行判断

**21、了解 redux 么，说一下 redux（必会）**

redux 是一个应用数据流框架，主要是解决了组件间状态共享的问题，原理是集中式管理，

主要有三个核心方法，action，store，reducer

三大原则：

1. 唯一数据源(整个应用的 state 被储存在一棵 object tree 中，并且这个 object tree 只存在于唯一一个 store 中)

2. reducer 必须是纯函数（输入必须对应着唯一的输出）

3. State 是只读的, 想要更改必须经过派发 action

**22、Redux 的工作流程**

使用通过 reducer 创建出来的 Store 发起一个 Action，reducer 会执行相应的更新 state 的方法，当

state 更新之后，view 会根据 state 做出相应的变化

1. 提供 getState()获取到 state

2. 通过 dispatch(action)发起 action 更新 state

3. 通过 subscribe()注册监听器

**23、vue 和 React 的区别（必会）**

监听数据变化的实现原理不同；数据流的不同；React 组合不同功能的方式是通过 HoC(高阶组件），Vue 组合不同功能的方式是通过 mixin；组件通信的不同；模板渲染方式的不同；渲染过程不同等

**24、React 生命周期函数有哪些？（必会）**

1. Mounting 挂载阶段

constructor()、componentWillMount()组件挂载到页面之前render()创建虚拟 DOM，进行 diff 运算，更新 DOM 树.不可进行 setState()、componentDidMount()组件挂载到页面之后，可以在此请求数据

2. Updateing 更新阶段

componentWillReceiveProps()父级数据发生变化

shouldComponentUpdate()、性能优化

3. Unmounting 卸载阶段

componentWillUnmount 组件卸载和销毁之前立刻停用、可以在此销毁定时器，取消网络请求，消除创建的相关 DOM 节点等

**25、运行阶段生命周期调用顺序？（必会）**



**26、React 中 component 和 pureComponent 区别是什么？（必会）**

1. Component 是 ReactApp 的基本构建的单位，也是 React 中的基本代码复用单位

2. PureComponent 与 Component 在除了其 shouldComponentUpdate 方法的实现之外几乎完全相同。

3. PureComponent 已经替我们实现了 shouldComponentUpdate 方法。对于 PureComponent 而言， 当其 props 或者 state 改变之时，新旧 props 与 state 将进行浅对比（shallow comparison）。另一方面，Component 默认的情况下其shouldComponentUpdate 方法并不进行新旧 props 与 state 的对比

**27、什么是无状态组件，与有状态组件的区别？（必会）**

无状态组件主要用来定义模板，接收来自父组件 props 传递过来的数据，使用{props.xxx}的表达式把 props 塞到模板里面

有状态组件主要用来定义交互逻辑和业务数据，使用{this.state.xxx}的表达式把业务数据挂载到容器组件的实例上（有状态组件也可以叫做容器组件，无状态组件也可以叫做展示组件），然后传递 props 到展示组件，展示组件接收到 props，把 props 塞到模板里面

**28、调用 render 时，DOM 一定会更新吗，为什么？（必会）**

不一定更新,React 组件中存在两类 DOM，render 函数被调用后，React 会根据 props 或者 state重新创建一棵 virtual DOM 树，虽然每一次调用都重新创建，但因为创建是发生在内存中，所以很快不影响性能。而 virtual dom 的更新并不意味着真实 DOM 的更新，React 采用 diff 算法将 virtual DOM 和真实 DOM 进行比较，找出需要更新的最小的部分，这时 Real DOM 才可能发生修改,所以每次 state 的更改都会使得 render 函数被调用，但是页面 DOM 不一定发生修改

**29、在哪些生命周期中可以修改组件的 state？（必会）**

componentDidMount 和 componentDidUpdate

constructor、

componentWillMount 中 setState 会发生错误：

setState 只能在 mounted 或 mounting组件中执行componentWillUpdate 中 setState 会导致死循环

**30、connect()前两个参数是什么？（必会）**

1. mapStateToProps(state, ownProps)

允 许 我 们 将 store 中 的 数 据 作 为 props 绑 定 到 组 件 中 ， 只 要 store 更 新 了 就 会 调 用 mapStateToProps 方法mapStateToProps 返回的结果必须是 object 对象，该对象中的值将会更新到组件中

2. mapDispatchToProps(dispatch, [ownProps])

允许我们将 action 作为 props 绑定到组件中，如果不传这个参数 redux 会把 dispatch 作为属性注入给组件，可以手动当做store.dispatch 使用 mapDispatchToProps 希望你返回包含对应 action 的 object 对象

**31、React-router 的原理（高薪常问）**

1. BrowserRouter 或 hashRouter 用来渲染 Router 所代表的组件

2. Route 用来匹配组件路径并且筛选需要渲染的组件

3. Switch 用来筛选需要渲染的唯一组件

4. Link 直接渲染某个页面组件

5. Redirect 类似于 Link，在没有 Route 匹配成功时触发

**32、React 的 diff 原理（高薪常问）**

diff（翻译差异）：计算一棵树形结构转换成另一棵树形结构的最少操作

1. 把树形结构按照层级分解，只比较同级元素

2. 给列表结构的每个单元添加唯一的 key 属性，方便比较

3. React 只会匹配相同 class 的 component（这里面的 class 指的是组件的名字）

4. 合并操作，调用 component 的 setState 方法的时候, React 将其标记为 dirty.到每一个事件循环结束, React 检查所有标记 dirty 的 component 重新绘制

5. 选择性子树渲染。开发人员可以重写 shouldComponentUpdate 提高 diff 的性能

**33、为什么建议传递给 setState 的参数是一个 callback 而不是一个对**

**象（高薪常问）**

因为 this.props 和 this.state 的更新可能是异步的，不能依赖它们的值去计算下一个 state。

**34、redux 中间件原理（高薪常问）**

redux 的中间件就是对 store 的 dispatch 做了个升级，升级之后 dispatch 就可以对象和函数都可以接收，这个时候当调用 dispatch 方法给 dispatch 方法传递的参数是一个对象的话，那么 dispatch 就会把这个对象直接传递给 store，跟之前我们写 dispatch 传递给它一个对象没什么 区别，但是如果传递给 dispatch 方法是一个函数的话，这个时候 dispatch 已经升级了，它就不会把这个函数直接传递给 store，它会先让这个函数执行，执行完了之后需要调用 store 的时候再去调用 store。所以 dispatch 在这里会根据参数的不同执行不同的事情

**35、React 性能优化的方案（高薪常问）**

减少 render 方法的调用; 避免使用状态提升来共享 state,此时应该使用 redux 解决方案; 保持稳定的 dom 结构，尽量避免 dom 节点跨层级移动操作; 使用 css 来隐藏节点，而不是真的移除或添加 DOM 节点等

**36、为什么虚拟 DOM 会提高性能?说下他的原理（高薪常问）**

虚拟 dom 相当于在 js 和真实 dom 中间加了一个缓存，利用 dom 的 diff 算法避免了没有必要的 dom 操作，从而提高性能

VirtualDOM 工作过程有三个简单的步骤：

1、每当底层数据发生改变时，整个 UI 都将在 VirtualDOM 描述中重新渲染

2、然后计算之前 DOM 表示与新表示的之间的差异

3、完成计算后，将只用实际更改的内容更新 realDOM

**37、setState 何时同步何时异步？（高薪常问）**

1. setState 只在合成事件（React 为了解决跨平台，兼容性问题，自己封装了一套事件机制，代理了原生的事件，像在 jsx 中常见的 onClick、onChange 这些都是合成事件）和钩子函数（生命周期）中是“异步”的，在原生事件和 setTimeout 中都是同步的

2. setState 的“异步”并不是说内部由异步代码实现，其实本身执行的过程和代码都是同步的，只是合成事件和钩子函数的调用顺序在更新之前，导致在合成事件和钩子函数中没法立马拿到更新后的值，形式了所谓的“异步”，当然可以通过第二个参数 setState(partialState,callback) 中的callback 拿到更新后的结果

3. setState 的批量更新优化也是建立在“异步”（合成事件、钩子函数）之上的，在原生事件和setTimeout 中不会批量更新，在“异步”中如果对同一个值进行多次 setState ，setState 的批量更新策略会对其进行覆盖，取最后一次的执行，如果是同时 setState 多个不同的值，在更新时会对其进行合并批量更新

### 2.1.4、微信小程序

**1、微信小程序主要目录和文件的作用？**

* project.config.json 项目配置文件，用得最多的就是配置是否开启https校验；
* App.js 设置一些全局的基础数据等；
* App.json 底部tab, 标题栏和路由等设置；
* App.wxss 公共样式，引入iconfont等；
* pages 里面包含一个个具体的页面；
* index.json (配置当前页面标题和引入组件等)；
* index.wxml (页面结构)；
* index.wxss (页面样式表)；
* index.js (页面的逻辑，请求和数据处理等)；

**2、wxml与标准的html的异同？**

* 都是用来描述页面的结构；
* 都由标签、属性等构成；
* 标签名字不一样，且小程序标签更少，单一标签更多；
* 多了一些 wx:if 这样的属性以及 {{ }} 这样的表达式
* WXML仅能在微信小程序开发者工具中预览，而HTML可以在浏览器内预览
* 组件封装不同， WXML对组件进行了重新封装，
* 小程序运行在JS Core内，没有DOM树和window对象，小程序中无法使用window对象和document对象。

**3、WXSS和CSS的异同？**

* 都是用来描述页面的样子；
* WXSS 具有 CSS 大部分的特性，也做了一些扩充和修改；
* WXSS新增了尺寸单位，WXSS 在底层支持新的尺寸单位 rpx；
* WXSS 仅支持部分 CSS 选择器；
* WXSS 提供全局样式与局部样式

**4、怎么封装微信小程序的数据请求的？**

* 在根目录下创建utils目录及api.js文件和apiConfig.js文件；
* 在apiConfig.js 封装基础的get, post 和 put， upload等请求方法，设置请求体，带上token和异常处理等；
* 在api中引入apiConfig.js封装好的请求方法，根据页面数据请求的urls, 设置对应的方法并导出；
* 在具体的页面中导入；

**5、小程序页面间有哪些传递数据的方法？**

* 使用全局变量实现数据传递
* 页面跳转或重定向时，使用url带参数传递数据
* 使用组件模板 template传递参数
* 使用缓存传递参数
* 使用数据库传递数据

**6、谈谈小程序的生命周期函数？**

* onLoad() 页面加载时触发，只会调用一次，可获取当前页面路径中的参数。
* onShow() 页面显示/切入前台时触发，一般用来发送数据请求；
* onReady() 页面初次渲染完成时触发, 只会调用一次，代表页面已可和视图层进行交互。
* onHide() 页面隐藏/切入后台时触发, 如底部 tab 切换到其他页面或小程序切入后台等。
* onUnload() 页面卸载时触发，如redirectTo或navigateBack到其他页面时。

1. **简述微信小程序原理？**

* 小程序本质就是一个单页面应用，所有的页面渲染和事件处理，都在一个页面内进行，但又可以通过微信客户端调用原生的各种接口；
* 它的架构，是数据驱动的架构模式，它的UI和数据是分离的，所有的页面更新，都需要通过对数据的更改来实现；
* 它从技术讲和现有的前端开发差不多，采用JavaScript、WXML、WXSS三种技术进行开发；
* 功能可分为webview和appService两个部分；
* webview用来展现UI，appService有来处理业务逻辑、数据及接口调用；
* 两个部分在两个进程中运行，通过系统层JSBridge实现通信，实现UI的渲染、事件的处理等。

1. **路由谈谈原生小程序开发、wepy、mpvue的对比**

* 如果是新项目，且没有旧的 h5 项目迁移，则考虑用小程序原生开发，好处是相比于第三方框架，坑少。
* 而如果有 老的 h5 项目是 vue 开发 或者 也有 h5 项目也需要小程序开发，则比较适合 wepy 或者 mpvue 来做迁移或者开发，近期看wepy几乎不更新了，所以推荐美团的mpvue。
* 而如果如果团队前端强大，自己做一套框架也没问题。

**9、小程序的双向绑定和vue有何不同？**

小程序直接 this.data 的属性是不可以同步到视图的，必须调用：

**this.setData({**

**// 这里设置**

**})**

**10、哪些方法可以提高微信小程序的应用速度？**

1、提高页面加载速度

2、用户行为预测

3、减少默认 data 的大小

4、组件化方案

**11、微信小程序的优劣势？**

**优势：**

* 即用即走，不用安装，省流量，省安装时间，不占用桌面
* 依托微信流量，天生推广传播优势
* 开发成本比 App 低

**劣势：**

* 用户留存，即用即走是优势，也存在一些问题
* 入口相对传统 App 要深很多
* 限制较多,页面大小不能超过2M。不能打开超过10个层级的页面

**12、如何实现下拉刷新**

* 首先在全局 config 中的 window 配置 enablePullDownRefresh
* 在 Page 中定义 onPullDownRefresh 钩子函数,到达下拉刷新条件后，该钩子函数执行，发起请求方法
* 请求返回后，调用 wx.stopPullDownRefresh 停止下拉刷新

参考地址：https://juejin.im/post/5a781c756fb9a063606eb742

### 2.1.5、NodeJS

**1、NodeJS的适用场景**

1. RESTful API

这是NodeJS最理想的应用场景，可以处理数万条连接，本身没有太多的逻辑，只需要请求API，组织数据进行返回即可。它本质上只是从某个数据库中查找一些值并将它们组成一个响应。由于响应是少量文本，入站请求也是少量的文本，因此流量不高，一台机器甚至也可以处理最繁忙的公司的API需求。

2. 统一Web应用的UI层

目前MVC的架构，在某种意义上来说，Web开发有两个UI层，一个是在浏览器里面我们最终看到的，另一个在server端，负责生成和拼接页面。

不讨论这种架构是好是坏，但是有另外一种实践，面向服务的架构，更好的做前后端的依赖分离。如果所有的关键业务逻辑都封装成REST调用，就意味着在上层只需要考虑如何用这些REST接口构建具体的应用。那些后端程序员们根本不操心具体数据是如何从一个页面传递到另一个页面的，他们也不用管用户数据更新是通过Ajax异步获取的还是通过刷新页面。

3. 大量Ajax请求的应用

例如个性化应用，每个用户看到的页面都不一样，缓存失效，需要在页面加载的时候发起Ajax请求，NodeJS能响应大量的并发请求。　　总而言之，NodeJS适合运用在高并发、I/O密集、少量业务逻辑的场景。

**2、NodeJS是什么**

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。

**NodeJS的特点**

其特点为：

1. 它是一个Javascript运行环境

2. 依赖于Chrome V8引擎进行代码解释

3. 事件驱动

4. 非阻塞I/O

5. 轻量、可伸缩，适于实时数据交互应用

6. 单进程，单线程

**NodeJS的优缺点：**

优点：1. 高并发（最重要的优点）2. 适合I/O密集型应用

缺点：1. 不适合CPU密集型应用；CPU密集型应用给Node带来的挑战主要是：由于JavaScript单线程的原因，如果有长时间运行的计算（比如大循环），将会导致CPU时间片不能释放，使得后续I/O无法发起；

2. 只支持单核CPU，不能充分利用CPU

3. 可靠性低，一旦代码某个环节崩溃，整个系统都崩溃

4. 开源组件库质量参差不齐，更新快，向下不兼容

5. Debug不方便，错误没有stack trace

原因：单进程，单线程

解决方案：（1）分解大型运算任务为多个小任务，使得运算能够适时释放，不阻塞I/O调用的发起；（2）Nnigx反向代理，负载均衡，开多个进程，绑定多个端口；

（3）开多个进程监听同一个端口，使用cluster模块；

**3、NodeJS非阻塞I/O模型执行流程**

主线程：

1.执行node的代码，把代码放入队列

2.事件循环程序（主线程）把队列里面的同步代码都先执行了，

3.同步代码执行完成，执行异步代码

4.异步代码分2种状况，

(1)、异步非io setTimeout() setInterval()

判断是否可执行，如果可以执行就执行，不可以跳过。

(2)、异步io 文件操作

会从线程池当中去取一条线程，帮助主线程去执行。

5主线程会一直轮训，队列中没有代码了，主线程就会退出。

子线程：被放在线程池里面的线程，用来执行异步io操作

1.在线程池里休息

2.异步io的操作来了，执行异步io操作。

3.子线程会把异步io操作的callback函数，扔回给队列

4.子线程会回到线程池了去休息。

callback，在异步io代码执行完成的时候被扔回主线程。

**4、node核心内置类库(事件，流，文件，网络等)**

1.node的构架是什么样子的?

答案: 主要分为三层，应用app >> V8及node内置架构 >> 操作系统. V8是node运行的环境，可以理解为node虚拟机．node内置架构又可分为三层: 核心模块(javascript实现) >> c++绑定 >> libuv + CAes + http.

2. node有哪些核心模块?

答案: EventEmitter, Stream, FS, Net和全局对象

3. node有哪些全局对象?

答案: process, console, Buffer和exports

4. process有哪些常用方法?

答案: process.stdin, process.stdout, process.stderr, process.on, process.env, process.argv, process.arch, process.platform, process.exit

5. console有哪些常用方法?

答案: console.log/console.info, console.error/console.warning, console.time/console.timeEnd, console.trace, console.table

6. node有哪些定时功能?

答案: setTimeout/clearTimeout, setInterval/clearInterval, setImmediate/clearImmediate, process.nextTick

7. 什么是EventEmitter?

参考答案: EventEmitter是node中一个实现观察者模式的类，主要功能是监听和发射消息，用于处理多模块交互问题.

8. EventEmitter有哪些典型应用?

参考答案: 1) 模块间传递消息 2) 回调函数内外传递消息 3) 处理流数据，因为流是在EventEmitter基础上实现的. 4) 观察者模式发射触发机制相关应用

9. 怎么捕获EventEmitter的错误事件?

参考答案: 监听error事件即可．如果有多个EventEmitter,也可以用domain来统一处理错误事件.

**5、[nodejs中流(stream)的理解](https://segmentfault.com/a/1190000000519006)**

1. 什么是Stream?

答案: stream是基于事件EventEmitter的数据管理模式．由各种不同的抽象接口组成，主要包括可写，可读，可读写，可转换等几种类型．

2. Stream有什么好处?

答案: 非阻塞式数据处理提升效率，片断处理节省内存，管道处理方便可扩展等.

3. Stream有哪些典型应用?

答案: 文件，网络，数据转换，音频视频等.

4. 怎么捕获Stream的错误事件?

答案: 监听error事件，方法同EventEmitter.

5. 有哪些常用Stream,分别什么时候使用?

答案: Readable为可被读流，在作为输入数据源时使用；Writable为可被写流,在作为输出源时使用；Duplex为可读写流,它作为输出源接受被写入，同时又作为输入源被后面的流读出．Transform机制和Duplex一样，都是双向流，区别时Transfrom只需要实现一个函数\_transfrom(chunk, encoding, callback);而Duplex需要分别实现\_read(size)函数和\_write(chunk, encoding, callback)函数.

6. 实现一个Writable Stream?

答案: 三步走:1)构造函数call Writable 2)　继承Writable 3) 实现\_write(chunk, encoding, callback)函数

**NodeJs面试题大全详见：**http://www.cnblogs.com/meteorcn/p/node\_mianshiti\_interview\_question.html

### 2.1.6、less/sass 预处理

1. **为什么要使用less/sass 预处理器（优点、缺点）**
2. 、 CSS无法递归式定义
3. 、CSS的mixin式复用性支持不够

 (3)、预编译可缓解多浏览器兼容造成的冗余(4)css并不能算是一们真正意义上的“编程”语言，它本身无法未完成像其它编程语言一样的嵌套、继承、设置变量等工作。为了解决css的不足，开发者们想到了编写一种对css进行预处理的“中间语言”，可以实现一些“编程”语言才有的功能，然后自动编译成css供浏览识别，这样既一定程度上弥补了css的不足，也无需一种新的语言来代替css以供浏览器识别。于是css预处理语言就应运而生了。

1. 、CSS 预处理语言，它扩展了 CSS 语言，增加了变量、Mixin、函数等特性，使 CSS 更易维护和扩展。

预处理器的缺点：(1)、CSS的好处在于简便、随时随地被使用和调试。预编译CSS步骤的加入，让我们开发工作流中多了一个环节，调试也变得更麻烦了。(2)、预编译很容易造成后代选择器的滥用因此，实际项目中衡量预编译方案时，还是得想想，比起带来的额外维护开销，预编译有没有解决更大的麻烦。降低了自己对最终代码的控制力。(3)、更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。

1. **less/sass区别**
2. sass是基于Ruby的，然后是在[服务器端](https://www.baidu.com/s?wd=%E6%9C%8D%E5%8A%A1%E5%99%A8%E7%AB%AF&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykmyf3nWTkPHfYnH9WmH790ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHRkPjDdPH61rHckPjmsnWfvn0" \t "https://zhidao.baidu.com/question/_blank)处理的。很多开发者不会选择LESS因为[JavaScript引擎](https://www.baidu.com/s?wd=JavaScript%E5%BC%95%E6%93%8E&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykmyf3nWTkPHfYnH9WmH790ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHRkPjDdPH61rHckPjmsnWfvn0" \t "https://zhidao.baidu.com/question/_blank)需要额外的时间来处理代码然后输出修改过的CSS到[浏览器](https://www.baidu.com/s?wd=%E6%B5%8F%E8%A7%88%E5%99%A8&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Ykmyf3nWTkPHfYnH9WmH790ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHRkPjDdPH61rHckPjmsnWfvn0" \t "https://zhidao.baidu.com/question/_blank)。

(2)关于变量在LESS和Sass中的唯一区别就是，LESS用@，Sass用$

**1、[原生app,WEBAPP,混合app的差异](http://www.cnblogs.com/huanghundaxia/p/5734209.html)**

**什么叫做原生App?**

原生App是专门针对某一类移动设备而生的，它们都是被直接安装到设备里，而用户一般也是通过网络商店或者卖场来获取

原生应用是特别为某种操作系统开发的，比如iOS、Android、黑莓等等，它们是在各自的移动设备上运行的。

优点：

可访问手机所有功能（GPS、摄像头）；

速度更快、性能高、整体用户体验不错；

可线下使用（因为是在跟Web相对地平台上使用的）；

支持大量图形和动画;

容易发现（在App Store里面）和重新发现（应用图标会一直在主页上）；

应用下载能创造盈利（当然App Store抽取20-30% 的营收）。

缺点：

开发成本高；

支持设备非常有限（一般是哪个系统就在哪个平台专属设备上用）；

上线时间不确定（App Store审核过程不一）；

内容限制（App Store限制）；

获得新版本时需重新下载应用更新。

什么叫做移动Web App?

一般说来，移动Web App都是需要用到网络的，它们利用设备上的浏览器(比如iPhone的Safari)来运行，而且它们不需要在设备上下载后安装。

Web应用本质上是为移动浏览器设计的基于Web的应用，它们是用普通Web开发语言开发的，可以在各种智能手机浏览器上运行。

优点：

支持设备广泛；

较低的开发成本；

可即时上线；

无内容限制

用户可以直接使用最新版本（自动更新，不需用户手动更新）。

缺点：

表现略差（对联网的要求比较大）；

用户体验没那么炫；

图片和动画支持性不高；

没法在App Store中下载、无法通过应用下载获得盈利机会；

要求联网；

对手机特点有限制（摄像头、GPS等）。

对于这些缺点，如果能把HTML 5的优点用到Web上的话就会得到很大改善，尽管技术在提高，目前它还不能做原生应用可以做的每件事。

有些公司，比如金融时报继原生应用后也开发了Web应用，用户通过浏览器来访问他们的应用，因为他们的应用要采取应用内购买，而App Store是不允许应用内购买的，不然它们就无法拿到那20%-30% 的抽成了。

什么是混合app?

Hybrid App是指介于web-app、native-app这两者之间的app,它虽然看上去是一个Native App，但只有一个UI WebView，里面访问的是一个Web App，比如街旁网最开始的应用就是包了个客户端的壳，其实里面是HTML5的网页，后来才推出真正的原生应用。再彻底一点的，如掌上百度和淘宝客户端 Android版，走的也是Hybrid App的路线，不过掌上百度里面封装的不是WebView，而是自己的浏览内核，所以体验上更像客户端，更高效。

Hybrid App（混合模式移动应用）兼具“Native App良好用户交互体验的优势”和“Web App跨平台开发的优势”。很多人不知道市场上一些主流移动应用都是基于Hybrid App的方式开发，比如国外有Facebook、国内有百度搜索等。

综合一下就是：“Hybrid App同时使用网页语言与程序语言开发，通过应用商店区分移动操作系统分发，用户需要安装使用的移动应用”。总体特性更接近Native App但是和Web App区别较大。只是因为同时使用了网页语言编码，所以开发成本和难度比Native App要小很多。因此说，Hybrid App兼具了Native App的所有优势，也兼具了Web App使用HTML5跨平台开发低成本的优势

混合应用大家都知道是原生应用和Web应用的结合体，采用了原生应用的一部分、Web应用的一部分，所以必须，部分在设备上运行、部分在Web上运行。

不过混合应用中比例很自由，比如Web 占90%，原生占10%；或者各占50%。

优点：

兼容多平台；

顺利访问手机的多种功能；

App Store中可下载（Web应用套用原生应用的外壳）；

可线下使用。

缺点：

不确定上线时间；

用户体验不如本地应用；

性能稍慢（需要连接网络）；

技术还不是很成熟。

比如Facebook现在的应用属于混合应用它可以在许多App Store畅通无阻，但是掺杂了大量Web特性，所以它运行速度比较慢，而现在为了提高性能FB又决定采用原生应用。



以下是列举的一些重点差异：

● 开发难度。移动web和混合App开发难度对于web开发者来说相对较低，而且可以充分利用现有的web开发工具和工作流程

● 发布渠道和更新方式。混合App可以在应用商店App Store发布，但可以自主更新，而原生App的更新必须通过应用商店App Store。

● 移动设备本地API访问。混合App可以通过JavaScript API访问到移动设备的摄像头、GPS；而原生App可以通过原生编程语言访问设备所有功能。

● 跨平台和可移植性。基于浏览器的移动web最好的可移植性和跨平台表现；混合App也能节省跨平台的时间和成本，只需编写一次核心代码就可部署到多个平台，而原生App的跨平台性能最差。

● 搜索引擎友好。只有移动web对搜索引擎友好，可与在线营销无缝整合。

● 货币化。混合App除广告外，还支持付费下载及程序内购买；原生App的程序内购买金额2012年首次超过下载收费。

● 消息推送。只有混合App和原生App支持消息推送，这能增加用户忠诚度。

**结论总结**：选择合适的应用需要考虑些什么，比如：1、是否需要使用某些设备的特殊功能如：摄像头、摄像头闪光灯或者重力加速器2、你的开发预算3、你的应用是否一定需要网络3你的应用的目标硬件设备是所有的移动设备还是仅仅只是一部分而已4、你自己已经熟悉的开发语言5、这个应用对于性能要求是否苛刻6、如何靠这个应用赢利

（详见：http://www.cnblogs.com/huanghundaxia/p/5734209.html）

1. **ionic流行开发框架（混合APP）**

Ionic介绍：

1. ionic是一个用来开发混合手机应用的，开源的，免费的代码库。可以优化html、css和js的性能，构建高效的应用程序，而且还可以用于构建Sass和AngularJS的优化。ionic会是一个可以信赖的框架。
2. ionic 是一个专注于用WEB开发技术，基于HTML5创建类似于手机平台原生应用的一个开发框架。绑定了AngularJS和Sass。这个框架的目的是从web的角度开发手机应用，基于PhoneGap的编译平台，可以实现编译成各个平台的应用程序。

ionic的开发添加[android](http://baike.baidu.com/item/android/60243" \t "http://baike.baidu.com/item/_blank)和[ios](http://baike.baidu.com/item/ios/45705" \t "http://baike.baidu.com/item/_blank)环境。

ionic提供很多[css](http://baike.baidu.com/item/css/5457" \t "http://baike.baidu.com/item/_blank)组件 和javascript UI库。

ionic可以支持定制android和ios的插件，也支持服务端REST的敏捷开发。

**ionic 特点：**

1.ionic 基于Angular语法，简单易学。

2.ionic 是一个轻量级框架。

3.ionic 完美的融合下一代移动框架，支持 Angularjs 的特性， MVVM ，代码易维护。

4.ionic 提供了漂亮的设计，通过 SASS 构建应用程序，它提供了很多 UI 组件来帮助开发者开发强大的应用。

5.ionic 专注原生，让你看不出混合应用和原生的区别

6.ionic 提供了强大的命令行工具。

7.ionic 性能优越，运行速度快。

Ionic中文教程连接：http://www.runoob.com/ionic/ionic-tutorial.html

### 2.1.7、前端工具介绍或使用方法

1. **前端工程化工具**

发展历程：grunt -->gulp -->webpack

gulp[较之grunt的优](http://blog.csdn.net/u011413061/article/details/48834967)势：

1、易用， Gulp相比Grunt更简洁，而且遵循代码优于配置策略，维护Gulp更像是写代码。

2、高效， Gulp相比Grunt更有设计感，核心设计基于Unix流的概念，通过管道连接，不需要写中间文件。

3、高质量 ，Gulp的每个插件只完成一个功能，这也是Unix的设计原则之一，各个功能通过流进行整合并完成复杂的任务。例如：Grunt的imagemin插件不仅压缩图片，同时还包括缓存功能。他表示，在Gulp中，缓存是另一个插件，可以被别的插件使用，这样就促进了插件的可重用性。目前官方列出的有673个插件。

4、易学 ，Gulp的核心API只有5个，掌握了5个API就学会了Gulp，之后便可以通过管道流组合自己想要的任务。

5、流 ，使用Grunt的I/O过程中会产生一些中间态的临时文件，一些任务生成临时文件，其它任务可能会基于临时文件再做处理并生成最终的构建后文件。而使用Gulp的优势就是利用流的方式进行文件的处理，通过管道将多个任务和操作连接起来，因此只有一次I/O的过程，流程更清晰，更纯粹。

6、代码优于配置 ，维护Gulp更像是写代码，而且Gulp遵循CommonJS规范，因此跟写Node程序没有差别。

1. **Webpack**

1、什么是Webpack

WebPack可以看做是模块打包机：它做的事情是，分析你的项目结构，找到JavaScript模块以及其它的一些浏览器不能直接运行的拓展语言（Scss，TypeScript等），并将其打包为合适的格式以供浏览器使用。

2、为什要使用WebPack

今的很多网页其实可以看做是功能丰富的应用，它们拥有着复杂的JavaScript代码和一大堆依赖包。为了简化开发的复杂度，前端社区涌现出了很多好的实践方法

a:模块化，让我们可以把复杂的程序细化为小的文件;

b:类似于TypeScript这种在JavaScript基础上拓展的开发语言：使我们能够实现目前版本的JavaScript不能直接使用的特性，并且之后还能能装换为JavaScript文件使浏览器可以识别；

c:scss，less等CSS预处理器

.........

这些改进确实大大的提高了我们的开发效率，但是利用它们开发的文件往往需要进行额外的处理才能让浏览器识别,而手动处理又是非常反锁的，这就为WebPack类的工具的出现提供了需求。

3、WebPack和Grunt以及Gulp相比有什么特性

其实Webpack和另外两个并没有太多的可比性，Gulp/Grunt是一种能够优化前端的开发流程的工具，而WebPack是一种模块化的解决方案，不过Webpack的优点使得Webpack可以替代Gulp/Grunt类的工具。

Grunt和Gulp的工作方式是：在一个配置文件中，指明对某些文件进行类似编译，组合，压缩等任务的具体步骤，这个工具之后可以自动替你完成这些任务。

Webpack的工作方式是：把你的项目当做一个整体，通过一个给定的主文件（如：index.js），Webpack将从这个文件开始找到你的项目的所有依赖文件，使用loaders处理它们，最后打包为一个浏览器可识别的JavaScript文件。

优点：模块化

在webpack看来一切都是模块！这就是它不可不说的优点，包括你的JavaScript代码，也包括CSS和fonts以及图片等等，只要通过合适的loaders，它们都可以被当做模块被处理。

Webpack：[https://segmentfault.com/a/1190000006178770#articleHeader8](https://segmentfault.com/a/1190000006178770" \l "articleHeader8)

1. **svn和git的区别**

1、GIT是分布式的，SVN是集中式的

2、git是每个历史版本都存储完整的文件,便于恢复,svn是存储差异文件,历史版本不可恢复。(核心)

3、git可离线完成大部分操作,svn则不能。

4、git有着更优雅的分支和合并实现。

5、git有着更强的撤销修改和修改历史版本的能力

6、Git下载下来后，在本地不必联网就可以看到所有的log，很方便学习，SVN却需要联网；git速度更快,效率更高。

7、Git 没有一个全局的版本号，而SVN有。

Svn本地管理详见：<http://www.cnblogs.com/pingwen/p/5152684.html>

Git版本管理详见：http://www.cnblogs.com/grimm/p/5368759.html

1. **Git/GitHub**

**Git**是一个开源的分布式版本控制系统，用以有效、高速的处理各种规模的项目版本管理， 它是 Linux Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件，后来得到广泛的使用。

**GitHub** 是一个面向开源及私有[软件](http://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6/12053" \t "http://baike.baidu.com/item/github/_blank)项目的托管平台，因为只支持 Git 作为唯一的版本库格式进行托管，故名 GitHub。除了 Git 代码仓库托管及基本的 Web 管理界面以外，还提供了订阅、讨论组、文本渲染、在线文件编辑器、协作图谱（报表）、代码片段分享（Gist）等功能。目前，其注册用户已经超过350万，托管版本数量也是非常之多，其中不乏知名开源项目 Ruby on Rails、jQuery、python 等。

**2.2、如何表述项目**

### 2.2.1、项目描述方式

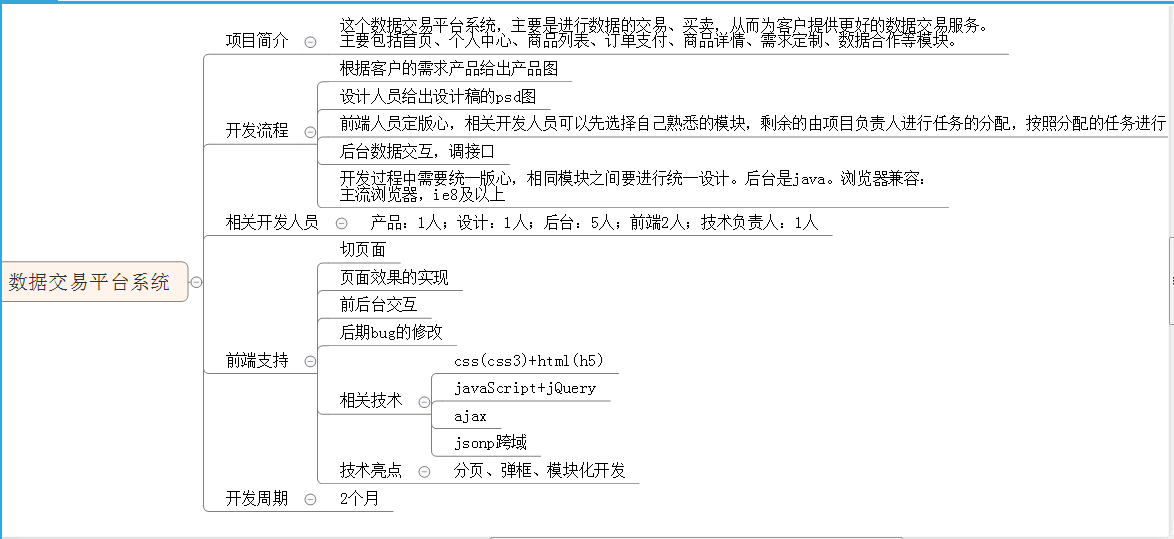
**1、项目功能模块分析**

项目功能模块分析结合自己的知识技术水平以及面试公司会用到的相关技术点阐述我们的项目，吸引面试官，让他感觉你跟他们公司的气质很适合。

1. **项目如何开展分配**

从项目的需求分析、原型定制、人员分配、技术定性、技术讨论、注意事项等展开我们开发这个项目的一个流程分析，可以加上开发过程中遇到的问题，怎么解决的，或者加上自己的见解等。

### 2.2.2、项目口述、面述话术模板



**1、基于项目功能模块分析**

1、介绍项目主要是干嘛的，例如：数据交易平台系统主要是主要是进行数据的交易、买卖，从而为客户提供更好的数据交易服务。主要包括首页、个人中心、商品列表、订单支付、商品详情、需求定制、数据合作等模块。

2、介绍自己负责的部分和使用的技术，例如：在这个项目中我负责的主要是个人中心这个模块的开发，包括静态页面和数据交互。开发过程中主要用到的技术有H5+css3、ajax、插件（分页、弹框、模块化开发）。

3、然后结合自己的项目介绍，说什么技术实现了什么功能（如果产品上线可以给面试官看）4、此时你基本上介绍完毕，面试官如果对你介绍的感兴趣，他会问你的，如果他没有问，你可以问下该公司主要业务是什么，用什么技术开发，寻找共同点，展开话题

**2、从项目如何开展分配分析**

1、首先介绍项目主要是干嘛的，例如：数据交易平台系统主要是进行数据的交易、买卖，从而为客户提供更好的数据交易服务。主要包括首页、个人中心、商品列表、订单支付、商品详情、需求定制、数据合作等模块。

2、该项目结合用户提出的需求进行制定，技术人员讨论技术点的实现，结合考虑多方面的内容，定制产品模型。

3、产品模型出来之后，ui设计师，设计出来psd图，同时前端人员可以做些准备工作（比如定版心，定框架、浏览器兼容版本等）

4、前端人员任务分配：可以先自行选择，然后项目经理进行2次分配。

5、确定时间节点，开发周期等比如：静态页面完成时间、与后台交互时间、测试时间等。

6、可以说一下在开发过程中遇到的一些问题，比如双边距问题，输入框默认值问题，讲一下怎么解决的，总结自己的感想。

7、介绍基本完成后可以适当与面试官进行交流，问下该公司主要业务是什么，用什么技术开发，寻找共同点，展开下一步的话题，可以占据主动性。

# 第三部分（非技术部分）

**3.1、人事面试**

**1、请你自我介绍一下你自己？**

回答提示：

一般人回答这个问题过于平常，只说姓名、年龄、爱好、工作经验，这些在简历上都有。其实，企业最希望知道的是求职者能否胜任工作，包括：最强的技能、最深入研究的知识领域、个性中最积极的部分、做过的最成功的事，主要的成就等，这些都可以和学习无关，也可以和学习有关，但要突出积极的个性和做事的能力，说得合情合理企业才会相信。企业很重视一个人的礼貌，求职者要尊重考官，在回答每个问题之后都说一句“谢谢”，企业喜欢有礼貌的求职者。

**2、你觉得你个性上最大的优点是什么？**

**回答提示：**

沉着冷静、条理清楚、立场坚定、顽强向上、乐于助人和关心他人、适应能力和幽默感、乐观和友爱。

1. **说说你最大的缺点？**

**回答提示：**

这个问题企业问的概率很大，通常不希望听到直接回答的缺点是什么等，如果求职者说自己小心眼、爱忌妒人、非常懒、脾气大、工作效率低，企业肯定不会录用你。绝对不要自作聪明地回答“我最大的缺点是过于追求完美”，有的人以为这样回答会显得自己比较出色，但事实上，他已经岌岌可危了。业喜欢求职者从自己的优点说起，中间加一些小缺点，最后再把问题转回到优点上，突出优点的部分，企业喜欢聪明的求职者。

**4、你对加班的看法？**

**回答提示：**

实际上好多公司问这个问题，并不证明一定要加班，只是想测试你是否愿意为公司奉献。

**回答样本：**

如果是工作需要我会义不容辞加班，我现在单身，没有任何家庭负担，可以全身心的投入工作。但同时，我也会提高工作效率，减少不必要的加班。

**5、你对薪资的要求？**

**回答提示：**

如果你对薪酬的要求太低，那显然贬低自己的能力；如果你对薪酬的要求太高，那又会显得你分量过重，公司受用不起。一些雇主通常都事先对求聘的职位定下开支预算，因而他们第一次提出的价钱往往是他们所能给予的最高价钱，他们问你只不过想证实一下这笔钱是否足以引起你对该工作的兴趣。如果你自己必须说出具体数目，请不要说一个宽泛的范围，那样你将只能得到最低限度的数字。最好给出一个具体的数字，这样表明你已经对当今的人才市场作了调查，知道像自己这样学历的雇员有什么样的价值。

**回答样本一：**

我对工资没有硬性要求，我相信贵公司在处理我的问题上会友善合理。我注重的是找对工作机会，所以只要条件公平，我则不会计较太多。

**回答样本二：**

我受过系统的软件编程的训练，不需要进行大量的培训，而且我本人也对编程特别感兴趣。因此，我希望公司能根据我的情况和市场标准的水平，给我合理的薪水。

**6、你的职业规划？**

**回答提示：**

这是每一个应聘者都不希望被问到的问题，但是几乎每个人都会被问到，比较多的答案是“管理者”。但是近几年来，许多公司都已经建立了专门的技术途径。这些工作地位往往被称作“顾问”、“参议技师”或“高级软件工程师”等等。当然，说出其他一些你感兴趣的职位也是可以的，比如产品销售部经理，生产部经理等一些与你的专业有相关背景的工作。要知道，考官总是喜欢有进取心的应聘者，此时如果说“不知道”，或许就会使你丧失一个好机会。最普通的回答应该是“我准备在技术领域有所作为”或“我希望能按照公司的管理思路发展”。

**7、你还有什么问题要问吗？**

**回答提示：**

企业的这个问题看上去可有可无，其实很关键，企业不喜欢说“没问题”的人，因为其很注重员工的个性和创新能力。企业不喜欢求职者问个人福利之类的问题，如果有人这样问：贵公司对新入公司的员工有没有什么培训项目，我可以参加吗？或者说贵公司的晋升机制是什么样的？企业将很欢迎，因为体现出你对学习的热情和对公司的忠诚度以及你的上进心。

**8、如果通过这次面试我们单位录用了你，但工作一段时间却发现你根本不适合这个职位，你怎么办？**

**回答提示：**

一段时间发现工作不适合我，有两种情况：①如果你确实热爱这个职业，那你就要不断学习，虚心向领导和同事学习业务知识和处事经验，了解这个职业的精神内涵和职业要求，力争减少差距；②你觉得这个职业可有可无，那还是趁早换个职业，去发现适合你的，你热爱的职业，那样你的发展前途也会大点，对单位和个人都有好处。

**9、在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？**

**回答提示：**

①原则上我会尊重和服从领导的工作安排，同时私底下找机会以请教的口吻，婉转地表达自己的想法，看看领导是否能改变想法。②如果领导没有采纳我的建议，我也同样会按领导的要求认真地去完成这项工作。③还有一种情况，假如领导要求的方式违背原则，我会坚决提出反对意见，如领导仍固执己见，我会毫不犹豫地再向上级领导反映。

**10、在完如果你的工作出现失误，给本公司造成经济损失，你认为该怎么办？**

**回答提示：**

①我本意是为公司努力工作，如果造成经济损失，我认为首要的问题是想方设法去弥补或挽回经济损失。如果我无能力负责，希望单位帮助解决。②分清责任，各负其责，如果是我的责任，我甘愿受罚；如果是一个我负责的团队中别人的失误，也不能幸灾乐祸，作为一个团队，需要互相提携共同完成工作，安慰同事并且帮助同事查找原因总结经验。③总结经验教训，一个人的一生不可能不犯错误，重要的是能从自己的或者是别人的错误中吸取经验教训，并在今后的工作中避免发生同类的错误。检讨自己的工作方法、分析问题的深度和力度是否不够，以致出现了本可以避免的错误。

**11、谈谈你对跳槽的看法？**

**回答提示：**

①正常的“跳槽”能促进人才合理流动，应该支持。②频繁的跳槽对单位和个人双方都不利，应该反对。

**12、工作中你难以和同事、上司相处，你该怎么办？**

回答提示：

①我会服从领导的指挥，配合同事的工作。②我会从自身找原因，仔细分析是不是自己工作做得不好让领导不满意，同事看不惯。还要看看是不是为人处世方面做得不好，如果是这样的话 我会努力改正。③如果我找不到原因，我会找机会跟他们沟通，请他们指出我的不足，有问题就及时改正。④作为优秀的员工，应该时刻以大局为重，即使在一段时间内，领导和同事对我不理解，我也会做好本职工作，虚心向他们学习，我相信，他们会看见我在努力，总有一天会对我微笑的。

**13、你对于我们公司了解多少？**

回答提示：

在去公司面试前上网查一下该公司主营业务。如回答：贵公司有意改变策略，加强与国外大厂的OEM合作，自有品牌的部分则透过海外经销商。

**14、请说出你选择这份工作的动机？**

回答提示：

这是想知道面试者对这份工作的热忱及理解度，并筛选因一时兴起而来应试的人，如果是无经验者，可以强调“就算职种不同，也希望有机会发挥之前的经验”。

**15、你最擅长的技术方向是什么？**

回答提示：

说和你要应聘的职位相关的课程，表现一下自己的热诚没有什么坏处。

**16、你能为我们公司带来什么呢？**

回答提示：

企业很想知道未来的员工能为企业做什么，求职者应再次重复自己的优势，然后说：“就我的能力，我可以做一个优秀的员工在组织中发挥能力，给组织带来高效率和更多的收益”。企业喜欢求职者就申请的职位表明自己的能力，比如申请营销之类的职位，可以说：“我可以开发大量的新客户，同时，对老客户做更全面周到的服务，开发老客户的新需求和消费。”等等。

**17、最能概括你自己的三个词是什么？**

回答提示：

我经常用的三个词是：适应能力强，有责任心和做事有始终，结合具体例子向主考官解释，

**18、为什么要离职?**

回答提示：

回答这个问题时一定要小心，就算在前一个工作受到再大的委屈，对公司有多少的怨言，都千万不要表现出来，尤其要避免对公司本身主管的批评，避免面试官的负面情绪及印象。建议此时最好的回答方式是将问题归咎在自己身上，例如觉得工作没有学习发展的空间，自己想在面试工作的相关产业中多加学习，或是前一份工作与自己的生涯规划不合等等，回答的答案最好是积极正面的。

回答样本：

我希望能获得一份更好的工作，如果机会来临，我会抓住。我觉得目前的工作，已经达到顶峰，即沒有升迁机会。

**19、对工作的期望与目标何在？**

回答提示：

这是面试者用来评断求职者是否对自己有一定程度的期望、对这份工作是否了解的问题。对于工作有确实学习目标的人通常学习较快，对于新工作自然较容易进入状况，这时建议你，最好针对工作的性质找出一个确实的答案，如业务员的工作可以这样回答：“我的目标是能成为一个超级业务员，将公司的产品广泛的推销出去，达到最好的业绩成效；为了达到这个目标，我一定会努力学习，而我相信以我认真负责的态度，一定可以达到这个目标。”其他类的工作也可以比照这个方式来回答，只要在目标方面稍微修改一下就可以了。

**20、就你申请的这个职位，你认为你还欠缺什么？**

回答提示：

企业喜欢问求职者弱点，但精明的求职者一般不直接回答。他们希望看到这样的求职者：继续重复自己的优势，然后说：“对于这个职位和我的能力来说，我相信自己是可以胜任的，只是缺乏经验，这个问题我想我可以进入公司以后以最短的时间来解决，我的学习能力很强，我相信可以很快融入公司的企业文化，进入工作状态。”企业喜欢能够巧妙地躲过难题的求职者。

**21、你通常如何处理別人的批评？**

回答提示：

①沈默是金，不必说什么，否则情况更糟，不过我会接受建设性的批评。②我会等大家冷靜下来再讨论。

**22、怎样对待自己的失败？**

回答提示：

我们大家生来都不是十全十美的，我相信我有第二个机会改正我的错误。

**23、什么会让你有成就感？**

回答提示：

为贵公司竭力效劳，尽我所能，完成一个项目。

**24、你为什么愿意到我们公司来工作？**

回答提示：

对于这个问题，你要格外小心，如果你已经对该单位作了研究，你可以回答一些详细的原因，像“公司本身的高技术开发环境很吸引我。”、“我同公司出生在同样的时代，我希望能够进入一家与我共同成长的公司。”、“你们公司一直都稳定发展，在近几年来在市场上很有竞争力。”、“我认为贵公司能够给我提供一个与众不同的发展道路。”这都显示出你已经做了一些调查，也说明你对自己的未来有了较为具体的远景规划。

**25、你和别人发生过争执吗？你是怎样解决的？**

回答提示：

这是面试中最险恶的问题，其实是考官布下的一个陷阱，千万不要说任何人的过错，应知成功解决矛盾是一个协作团体中成员所必备的能力。假如你工作在一个服务行业，这个问题简直成了最重要的一个环节。你是否能获得这份工作，将取决于这个问题的回答。考官希望看到你是成熟且乐于奉献的。他们通过这个问题了解你的成熟度和处世能力。在没有外界干涉的情况下，通过妥协的方式来解决才是正确答案。

**26、对这项工作，你有哪些可预见的困难？**

回答提示：

①不宜直接说出具体的困难，否则可能令对方怀疑应聘者不行。②可以尝试迂回战术，说出应聘者对困难所持有的态度——工作中出现一些困难是正常的，也是难免的，但是只要有坚忍不拔的毅力、良好的合作精神以及事前周密而充分的准备，任何困难都是可以克服。

分析：

一般问这个问题，面试者的希望就比较大了，因为已经在谈工作细节，但常规思路中的回答，又被面试官“骗”了。当面试官询问这个问题的时候，有两个目的。第一，看看应聘者是不是在行，说出的困难是不是在这个职位中一般都不可避免的问题。第二，是想看一下应聘者解决困难的手法对不对，及公司能否提供这样的资源。而不是想了解应聘者对困难的态度。

**27、如果我录用你，你将怎样开展工作？**

回答提示：

①如果应聘者对于应聘的职位缺乏足够的了解，最好不要直接说出自己开展工作的具体办法。②可以尝试采用迂回战术来回答，如“首先听取领导的指示和要求，然后就有关情况进行了解和熟悉，接下来制定一份近期的工作计划并报领导批准，最后根据计划开展工作。”。

分析：

这个问题的主要目的也是了解应聘者的工作能力和计划性、条理性，而且重点想要知道细节。如果向思路中所讲的迂回战术，面试官会认为回避问题，如果引导了几次仍然是回避的话，此人绝对不会录用了。

**28、在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？与上级意见不一是，你将怎么办？**

回答提示：

①原则上我会尊重和服从领导的工作安排；同时私底下找机会以请教的口吻，婉转地表达自己的想法，看看领导是否能改变想法。②如果领导没有采纳我的建议，我也同样会按领导的要求认真地去完成这项工作。③还有一种情况，假如领导要求的方式违背原则，我会坚决提出反对意见，如领导仍固执己见，我会毫不犹豫地再向上级领导反映。

**29、你工作经验欠缺，如何能胜任这项工作？**

常规思路：

①如果招聘单位对应届毕业生的应聘者提出这个问题，说明招聘公司并不真正在乎“经验”，关键看应聘者怎样回答。②对这个问题的回答最好要体现出应聘者的诚恳、机智、果敢及敬业。③如“作为应届毕业生，在工作经验方面的确会有所欠缺，因此在读书期间我一直利用各种机会在这个行业里做兼职。我也发现，实际工作远比书本知识丰富、复杂。但我有较强的责任心、适应能力和学习能力，而且比较勤奋，所以在兼职中均能圆满完成各项工作，从中获取的经验也令我受益非浅。请贵公司放心，学校所学及兼职的工作经验使我一定能胜任这个职位。”

点评：

这个问题思路中的答案尚可，突出自己的吃苦能力和适应性以及学习能力（不是学习成绩）为好。

**30、您在前一家公司的离职原因是什么？**

回答提示：

①最重要的是：应聘者要使找招聘单位相信，应聘者在过往的单位的“离职原因”在此家招聘单位里不存在。②避免把“离职原因”说得太详细、太具体。③不能掺杂主观的负面感受，如“太辛苦”、“人际关系复杂”、“管理太混乱”、“公司不重视人才”、“公司排斥我们某某的员工”等。④但也不能躲闪、回避，如“想换换环境”、“个人原因”等。⑤不能涉及自己负面的人格特征，如不诚实、懒惰、缺乏责任感、不随和等。⑥尽量使解释的理由为应聘者个人形象添彩。⑦相关例子：如“我离职是因为这家公司倒闭；我在公司工作了三年多，有较深的感情；从去年始，由于市场形势突变，公司的局面急转直下；到眼下这一步我觉得很遗憾，但还要面对显示，重新寻找能发挥我能力的舞台。”同一个面试问题并非只有一个答案，而同一个答案并不是在任何面试场合都有效，关键在应聘者掌握了规律后，对面试的具体情况进行把握，有意识地揣摩面试官提出问题的心理背景，然后投其所好。

分析：

除非是薪资太低，或者是最初的工作，否则不要用薪资作为理由。“求发展”也被考官听得太多，离职理由要根据每个人的真实离职理由来设计，但是在回答时一定要表现得真诚。实在想不出来的时候，家在外地可以说是因为家中有事，须请假几个月，公司又不可能准假，所以辞职，这个答案一般面试官还能接受。

**31、为了做好你工作份外之事，你该怎样获得他人的支持和帮助？**

回答提示：

每个公司都在不断变化发展的过程中，你当然希望你的员工也是这样。你希望得到那些希望并欢迎变化的人，因为这些人明白，为了公司的发展，变化是公司日常生活中重要组成部分。这样的员工往往很容易适应公司的变化，并会对变化做出积极的响应。

**32、果你在这次面试中没有被录用，你怎么打算？**

回答提示：

现在的社会是一个竞争的社会，从这次面试中也可看出这一点，有竞争就必然有优劣，有成功必定就会有失败。往往成功的背后有许多的困难和挫折，如果这次失败了也仅仅是一次而已，只有经过经验经历的积累才能塑造出一个完全的成功者。我会从以下几个方面来正确看待这次失败：①要敢于面对，面对这次失败不气馁，接受已经失去了这次机会就不会回头这个现实，从心理意志和精神上体现出对这次失败的抵抗力。要有自信，相信自己经历了这次之后经过努力一定能行，能够超越自我。②善于反思，对于这次面试经验要认真总结，思考剖析，能够从自身的角度找差距。正确对待自己，实事求是地评价自己，辩证的看待自己的长短得失，做一个明白人。③走出阴影，要克服这一次失败带给自己的心理压力，时刻牢记自己弱点，防患于未然，加强学习，提高自身素质。④认真工作，回到原单位岗位上后，要实实在在、踏踏实实地工作，三十六行、行行出状元，争取在本岗位上做出一定的成绩。⑤再接再厉，成为国家公务员一直是我的梦想，以后如果有机会我仍然后再次参加竞争。

**33、谈谈你过去做过的成功案例？(工作中遇到什么问题)**

回答提示：

举一个你最有把握的例子，把来龙去脉说清楚，而不要说了很多却没有重点。切忌夸大其词，把别人的功劳到说成自己的，很多主管为了确保要用的人是最适合的，会打电话向你的前一个主管征询对你的看法及意见，所以如果说谎，是很容易穿梆的。

**34、如何安排自己的时间？会不会排斥加班？**

回答提示：

基本上，如果上班工作有效率，工作量合理的话，应该不太需要加班。可是我也知道有时候很难避免加班，加上现在工作都采用责任制，所以我会调配自己的时间，全力配合。

分析：

虽然不会有人心甘情愿的加班，但依旧要表现出高配合度的诚意。

**35、这个职务的期许？**

回答提示：

希望能借此发挥我的所学及专长，同时也吸收贵公司在这方面的经验，就公司、我个人而言，缔造“双赢”的局面。

分析：

回答前不妨先询问该公司对这项职务的责任认定及归属，因为每一家公司的状况不尽相同，以免说了一堆理想抱负却发现牛头不对马嘴。

**36、什么选择我们这家公司？**

回答提示：

曾经在报章杂志看过关于贵公司的报道，与自己所追求的理念有志一同。而贵公司在业界的成绩也是有目共睹的，而且对员工的教育训练、升迁等也都很有制度。

分析：

去面试前先做功课，了解一下该公司的背景，让对方觉得你真的很有心想得到这份工作，而不只是探探路。

**37、谈谈如何适应办公室工作的新环境？**

回答提示：

①办公室里每个人有各自的岗位与职责，不得擅离岗位。②根据领导指示和工作安排，制定工作计划，提前预备，并按计划完成。③多请示并及时汇报，遇到不明白的要虚心请教。④抓间隙时间，多学习，努力提高自己的政治素质和业务水平。

**38、工作中学习到了些什么？**

回答提示：

这是针对转职者提出的问题，建议此时可以配合面试工作的特点作为主要依据来回答，如业务工作需要与人沟通，便可举出之前工作与人沟通的例子，经历了哪些困难，学习到哪些经验，把握这些要点做陈述，就可以轻易过关了。

**39、除了本公司外，还应聘了哪些公司？**

回答提示：

很奇怪，这是相当多公司会问的问题，其用意是要概略知道应徵者的求职志向，所以这并非绝对是负面答案，就算不便说出公司名称，也应回答“销售同种产品的公司”，如果应聘的其他公司是不同业界，容易让人产生无法信任的感觉。

**40、何时可以到职？**

回答提示：

大多数企业会关心就职时间，最好是回答“如果被录用的话，到职日可按公司规定上班”，但如果还未辞去上一个工作、上班时间又太近，似乎有些强人所难，因为交接至少要一个月的时间，应进一步说明原因，录取公司应该会通融的。