**1.JS中的是是非非**

**JS是一门计算机编程语言，是一门动态语言也称为脚本语言，是解析型编程语言。为什么是脚本？因为他本身不能执行，就是没有main函数或主程序的入口，必须被他的宿主环境，也就是解析环境解析执行他。为是什么是解析型？因为JS不会进行编程，链接，汇编等一系统的过程生成某个文件，再执行，它就是以字符串形式加载执行..**

**2.JS中的真真假假**

**空，null,undefined,false,0,””,'',NaN都为假,其他都为真**

**3.函数，类，对象，构造器有什么区别?**

**在js中函数，类，对象，构造器可以说是没有任何区别的，我们可以说函数既为对象，对象既为类，类既为构造器。 但一般情况下，我们称对象为那个类的实例，这是为了减少混乱。**

**function Person(){**

**}**

**var person = new Person();**

**//Person我们可以称为函数，类，对象，构造器(不带参数)**

**//person称为对象的引用**

**4.懂this关键字，你懂JS五分之一.**

**this永远指向调用该方法(函数)的那个对象.(必背)**

**首先要知道this是什么？**

**this就是js中的一个对象。是一个对象，不是一个函数，此是的对象你可以看作是一个对象的引用，引用谁？引用调用方法的那个对象.**

**6.prototype和constructor的在JS框架中的影响**

**prototype为原型,是一个对象。,。**

**constructor为构造器,是一个函数**

**到现在为止，你要分清楚，什么时候的对象为对象，什么时候的对象是一个函数.也难为JS解析器，能解析运行如此灵活的代码。**

**prototype可以说他是一个类的静态的属性，他指向这个对象的实例。换句话说prototype是一个对象。prototype指向的对象和我们new的对象有何区别？建立了一种等价桥梁关系，但不是同一个。当我们在函数prototype上绑定属性的时候，那么属性和值就绑定到了prototype对象上，并没有正真的绑定到那个对象上去。当需要访问那个对象上的这个属性的时候，JS解析器，首先会去从对象本身上去找这个属性，然后再到原型对象上找。**

**constructor指向一个对象的构造器。(什么是构造器？自己去复习。),由此可以看出他是一个对象级别的属性。也就是要使用constructor这个属性，必须需要一个对象。**

**那么prototype是一个对象有没有constructor属性？当然有，既然constructor是指构造器，那有没有一个prototype属性？当然有，以此类推,下面这两段代码是正确的.**

**constructor.prototype.constructor.prototype.constructor……**

**prototype.constructor.prototype.constructor.prototype……**

**其实我也不知道到底可以连多长？感兴趣可以试一下。顺便告诉你用递归算法。**

**在众多的JS代码中，当我们已知一个对象，要求他的构造器，或者已知一个构造器，求他的对象。应入门了。**

**7.==和===将判断进行到底.**

**==和===神像形不象.不要去考虑，一眼就看出是做断判是否相等。**

**有木有区别?**

**==判断变量是否相等。**

**===判断变量的值相等。**

**以此类推:====用来干嘛的？？用来报错的.你懂的.这里只想告诉你，只有==和===**

**JS中变量是弱类型的**

**都是var 类型的变量，但他们的值不一样，一个是整形，一个是字符串型.**

**var类型和var类型相比，当然是true,整形和字符串型比，当然是false.**

**==用来比较他们值是否一样。不会是计较值的类型，只要他们的变量是var类型就可以了。当然是这废话，难不成，你不还能声明第二种变量类型的变量？.**

**===用来比较，会去判断他们的值是否为同一类型。如果不是，没有的比较。JS中变量的值是强类型的，有整形，字符串，数字，布尔等.**

**8.typeof和instanceof完成了判断未成的使命**

**typeof用来判断基本数据类型**

**instanceof 用来判断对象类型是否为某一个类型号**

**9.5种继承让你如虎添翼.**

**i.对象冒充**

**ii.apply**

**iii.call**

**iv.prototype**

**v.for循环方式**

**用法:永远只须记住，JS中的继承只是属性和方法的拷贝.**

**10.DOM模型结构改变你的思想**

**DOM是一种思想，一种将数据以树状结构数据的思想。学习DOM只需要掌握任何一个节点都有一个父节点和0到多个子节点.任何一个节点都有一个页面上的标签表现形式和一个内存对应着一个标签对象.页面只是展示数据的地方，内存中才是DOM对象数据保存地方。任何一个DOM对象只能有一个父节点对象。父子关系可以随时改变。**

**必须:**

**获取对象:查**

**对象操作:增，修，删**

**内容操作:innerHTML,innerText等**

**事件操作:mouse,key**

**样式操作:id,tag,class**

**属性操作:attribute**

**11.回调函数减少编写代码**

**什么是回调函数?**

**在JS中函数名是用来标识一个函数的。既我们可以传递给某个函数一个函数名(函数名柄),然后由那个函数来自动调用我们的函数完成相关的处理。**

**调用者与被调用者分开，我们不需要关心调用者，和被调用者。回调可用于通知机制，事件。**

**12.函数和arguments**

**函数名就是函数的句柄，指针，函数名是唯一的，这也成就了JS中没有函数重载。只有函数覆盖。函数名才是唯一标识函数的。**

**永远需牢访，JS中的函数调用就是在函数名后面加对括号()**

**函数调用参数可有可无，无论什么情况下，参数都会保存在函数体中的arguments对象中，你可以直接使用他，他是一个对象，参数是以数组形式存放。**

**13.闭包(匿名函数)为开发开辟了捷径**

**闭包是函数内调用函数外的参数，一般也可称为匿名函数，但两者有所不同。**

**闭包的价值在于可以作为函数对象或者匿名函数，对于类型系统而言这就意味着不仅要表示数据还要表示代码。支持闭包的多数语言都将函数作为第一级对象，就是说这些函数可以存储到变量中、作为参数传递给其他函数，最重要的是能够被函数动态地创建和返回.**

**匿名函数减少了变量名称的冲突，为没有权限作用域的JS提供了权限作用域。当我们需要提供一个接口，但又不想让这个接口对象为全局变量就需要用匿名函数:**

**(function(){**

**局部代码，外部永远无法访问，除非你提供一个入口**

**})();**

**14.函数声明和函数表达式**

**对于函数声明，js解析器会优先读取，确保在所有代码执行之前声明已经被解析，而函数表达式，如同定义其它基本类型的变量一样，只在执行到某一句时也会对其进行解析，所以在实际中，它们还是会有差异的，具体表现在，当使用函数声明的形式来定义函数时，可将调用语句写在函数声明之前，而后者，这样做的话会报错。**

**15.XMLHTTPRequest相关问题**

（1）HTTP状态码

（2）onreadystate的取值以及意义

（3）responetext等 open send  
 360百科：<http://baike.so.com/doc/5509290-5745036.html>

**HTTP状态码：**  
 用以表示网页服务器HTTP相应状态的2位数字代码。所有状态码的第一个数字代表了响应的5种状态之一。

1xx消息

这一类型的状态码，代表请求已被接受，需要继续处理。这类响应是临时响应，只包含状态行和某些可选的响应头信息，并以空行结束。由于 HTTP/1.0 协议中没有定义任何 1xx 状态码，所以除非在某些试验条件下，服务器禁止向此类客户端发送 1xx 响应。

2xx成功

这一类型的状态码，代表请求已成功被服务器接收、理解、并接受。

3xx重定向

这类状态码代表需要客户端采取进一步的操作才能完成请求。通常，这些状态码用来重定向，后续的请求地址（重定向目标）在本次响应的 Location 域中指明。

当且仅当后续的请求所使用的方法是 GET 或者 HEAD 时，用户浏览器才可以在没有用户介入的情况下自动提交所需要的后续请求。客户端应当自动监测无限循环重定向（例如：A->A，或者A->B->C->A），因为这会导致服务器和客户端大量不必要的资源消耗。按照 HTTP/1.0 版规范的建议，浏览器不应自动访问超过5次的重定向。

4xx请求错误

这类的状态码代表了客户端看起来可能发生了错误，妨碍了服务器的处理。除非响应的是一个 HEAD 请求，否则服务器就应该返回一个解释当前错误状况的实体，以及这是临时的还是永久性的状况。这些状态码适用于任何请求方法。浏览器应当向用户显示任何包含在此类错误响应中的实体内容。

如果错误发生时客户端正在传送数据，那么使用TCP的服务器实现应当仔细确保在关闭客户端与服务器之间的连接之前，客户端已经收到了包含错误信息的数据包。如果客户端在收到错误信息后继续向服务器发送数据，服务器的TCP栈将向客户端发送一个重置数据包，以清除该客户端所有还未识别的输入缓冲，以免这些数据被服务器上的应用程序读取并干扰后者。

5xx服务器错误

这类状态码代表了服务器在处理请求的过程中有错误或者异常状态发生，也有可能是服务器意识到以当前的软硬件资源无法完成对请求的处理。除非这是一个HEAD 请求，否则服务器应当包含一个解释当前错误状态以及这个状况是临时的还是永久的解释信息实体。浏览器应当向用户展示任何在当前响应中被包含的实体。

这些状态码适用于任何响应方法。

**常见数字：**  
 200：英文提示：OK

请求已成功，请求所希望的响应头或数据体将随此响应返回。

401：英文提示：Unauthorized，当前请求需要用户验证。该响应必须包含一个适用于被请求资源的 WWW-Authenticate 信息头用以询问用户信息。客户端可以重复提交一个包含恰当的 Authorization 头信息的请求。如果当前请求已经包含了 Authorization 证书，那么401响应代表着服务器验证已经拒绝了那些证书。如果401响应包含了与前一个响应相同的身份验证询问，且浏览器已经至少尝试了一次验证，那么浏览器应当向用户展示响应中包含的实体信息，因为这个实体信息中可能包含了相关诊断信息。参见RFC 2617。

404：请求失败，请求所希望得到的资源未被在服务器上发现。没有信息能够告诉用户这个状况到底是暂时的还是永久的。假如服务器知道情况的话，应当使用410状态码来告知旧资源因为某些内部的配置机制问题，已经永久的不可用，而且没有任何可以跳转的地址。404这个状态码被广泛应用于当服务器不想揭示到底为何请求被拒绝或者没有其他适合的响应可用的情况下。  
 500：服务器遇到了一个未曾预料的状况，导致了它无法完成对请求的处理。一般来说，这个问题都会在服务器的程序码出错时出现。

502：作为网关或者代理工作的服务器尝试执行请求时，从上游服务器接收到无效的响应。  
 503：由于临时的服务器维护或者过载，服务器当前无法处理请求。这个状况是临时的，并且将在一段时间以后恢复。如果能够预计延迟时间，那么响应中可以包含一个 Retry-After 头用以标明这个延迟时间。如果没有给出这个 Retry-After 信息，那么客户端应当以处理500响应的方式处理它。

注意：503状态码的存在并不意味着服务器在过载的时候必须使用它。某些服务器只不过是希望拒绝客户端的连接。  
 504：作为网关或者代理工作的服务器尝试执行请求时，未能及时从上游服务器（URI标识出的服务器，例如HTTP、FTP、LDAP）或者辅助服务器（例如DNS）收到响应。

注意：某些代理服务器在DNS查询超时时会返回400或者500错误