**1.几种基本数据类型?复杂数据类型?值类型和引用数据类型?堆栈数据结构?**

基本数据类型：Undefined、Null、Boolean、Number、String  
值类型：数值、布尔值、null、undefined。  
引用类型：对象、数组、函数。  
堆栈数据结构：是一种支持后进先出(LIFO)的集合,即后被插入的数据,先被取出!  
js数组中提供了以下几个方法可以让我们很方便实现堆栈：  
shift:从数组中把第一个元素删除，并返回这个元素的值。  
unshift: 在数组的开头添加一个或更多元素，并返回新的长度  
push:在数组的中末尾添加元素，并返回新的长度  
pop:从数组中把最后一个元素删除，并返回这个元素的值。

**2.声明函数作用提升?声明变量和声明函数的提升有什么区别?**

(1) 变量声明提升：变量申明在进入执行上下文就完成了。  
只要变量在代码中进行了声明，无论它在哪个位置上进行声明， js引擎都会将它的声明放在范围作用域的顶部；  
  
(2) 函数声明提升：执行代码之前会先读取函数声明，意味着可以把函数申明放在调用它的语句后面。  
只要函数在代码中进行了声明，无论它在哪个位置上进行声明， js引擎都会将它的声明放在范围作用域的顶部；  
  
(3) 变量or函数声明：函数声明会覆盖变量声明，但不会覆盖变量赋值。  
同一个名称标识a，即有变量声明var a，又有函数声明function a() {}，不管二者声明的顺序，函数声明会覆盖变量声明，也就是说，此时a的值是声明的函数function a() {}。注意：如果在变量声明的同时初始化a，或是之后对a进行赋值，此时a的值变量的值。eg: var a; var c = 1; a = 1; function a() { return true; } console.log(a);

**3.判断数据类型?**

typeof返回的类型都是字符串形式，可以判断function的类型；在判断除Object类型的对象时比较方便。  
判断已知对象类型的方法： instanceof，后面一定要是对象类型，并且大小写不能错，该方法适合一些条件选择或分支。

**4.异步编程？**

方法1：回调函数，优点是简单、容易理解和部署，缺点是不利于代码的阅读和维护，各个部分之间高度耦合（Coupling），流程会很混乱，而且每个任务只能指定一个回调函数。  
  
方法2：时间监听，可以绑定多个事件，每个事件可以指定多个回调函数，而且可以“去耦合”（Decoupling），有利于实现模块化。缺点是整个程序都要变成事件驱动型，运行流程会变得很不清晰。  
  
方法3：发布/订阅，性质与“事件监听”类似，但是明显优于后者。  
  
方法4：Promises对象，是CommonJS工作组提出的一种规范，目的是为异步编程提供统一接口。  
简单说，它的思想是，每一个异步任务返回一个Promise对象，该对象有一个then方法，允许指定回调函数。

**5.事件流?事件捕获？事件冒泡？**

事件流：从页面中接收事件的顺序。也就是说当一个事件产生时，这个事件的传播过程，就是事件流。  
  
IE中的事件流叫事件冒泡；事件冒泡：事件开始时由最具体的元素接收，然后逐级向上传播到较为不具体的节点（文档）。对于html来说，就是当一个元素产生了一个事件，它会把这个事件传递给它的父元素，父元素接收到了之后，还要继续传递给它的上一级元素，就这样一直传播到document对象（亲测现在的浏览器到window对象，只有IE8及下不这样  
  
事件捕获是不太具体的元素应该更早接受到事件，而最具体的节点应该最后接收到事件。他们的用意是在事件到达目标之前就捕获它；也就是跟冒泡的过程正好相反，以html的click事件为例，document对象（DOM级规范要求从document开始传播，但是现在的浏览器是从window对象开始的）最先接收到click事件的然后事件沿着DOM树依次向下传播，一直传播到事件的实际目标；

**6.如何清除一个定时器?**

window.clearInterval();  
window.clearTimeout();

**7.如何添加一个dom对象到body中?innerHTML和innerText区别?**

body.appendChild(dom元素)；  
innerHTML:从对象的起始位置到终止位置的全部内容,包括Html标签。  
innerText:从起始位置到终止位置的内容, 但它去除Html标签   
分别简述五个window对象、属性  
  
成员对象   
window.event window.document window.history   
window.screen window.navigator window.external  
Window对象的属性如下：  
window //窗户自身  
window.self //引用本窗户window=window.self   
window.name //为窗户命名   
window.defaultStatus //设定窗户状态栏信息   
window.location //URL地址,配备布置这个属性可以打开新的页面

**8.数据持久化技术(ajax)?简述ajax流程**

1)客户端产生js的事件  
2)创建XMLHttpRequest对象  
3)对XMLHttpRequest进行配置  
4)通过AJAX引擎发送异步请求  
5)服务器端接收请求并且处理请求，返回html或者xml内容  
6)XML调用一个callback()处理响应回来的内容  
7)页面局部刷新

**9.回调函数?**

回调函数就是一个通过函数指针调用的函数。如果你把函数的指针（地址）作为参数传递给另一个函数，当这个指针被用来调用其所指向的函数时，我们就说这是回调函数。回调函数不是由该函数的实现方直接调用，而是在特定的事件或条件发生时由另外的一方调用的，用于对该事件或条件进行响应。

**10.什么是闭包?\* 堆栈溢出有什么区别？ 内存泄漏? 那些操作会造成内存泄漏？怎么样防止内存泄漏？**

闭包：就是能够读取其他函数内部变量的函数。  
堆栈溢出：就是不顾堆栈中分配的局部数据块大小，向该数据块写入了过多的数据，导致数据越界，结果覆盖了别的数据。经常会在递归中发生。  
内存泄露是指：用动态存储分配函数内存空间，在使用完毕后未释放，导致一直占据该内存单元。直到程序结束。指任何对象在您不再拥有或需要它之后仍然存在。  
  
造成内存泄漏：  
setTimeout 的第一个参数使用字符串而非函数的话，会引发内存泄漏。  
闭包、控制台日志、循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）  
防止内存泄露：  
1、不要动态绑定事件；  
2、不要在动态添加，或者会被动态移除的dom上绑事件，用事件冒泡在父容器监听事件；  
3、如果要违反上面的原则，必须提供destroy方法，保证移除dom后事件也被移除，这点可以参考Backbone的源代码，做的比较好；  
4、单例化，少创建dom，少绑事件。

**11.平时工作中怎么样进行数据交互?如果后台没有提供数据怎么样进行开发?mock数据与后台返回的格式不同意怎么办?**

由后台编写接口文档、提供数据接口实、前台通过ajax访问实现数据交互；  
在没有数据的情况下寻找后台提供静态数据或者自己定义mock数据；  
返回数据不统一时编写映射文件 对数据进行映射。

**12 简述ajax执行流程**

基本步骤：**var** xhr **=null**;*//创建对象* **if**(window.XMLHttpRequest){

xhr **=** **new** XMLHttpRequest();}**else**{

xhr **=** **new** ActiveXObject("Microsoft.XMLHTTP");}xhr.open(“方式”,”地址”,”标志位”);*//初始化请求*  xhr.setRequestHeader(“”,””);*//设置http头信息* xhr.onreadystatechange **=function**(){}*//指定回调函数* xhr.send();*//发送请求*

**13.自执行函数?用于什么场景？好处?**

自执行函数:1、声明一个匿名函数2、马上调用这个匿名函数。  
作用：创建一个独立的作用域。  
  
好处：防止变量弥散到全局，以免各种js库冲突。隔离作用域避免污染，或者截断作用域链，避免闭包造成引用变量无法释放。利用立即执行特性，返回需要的业务函数或对象，避免每次通过条件判断来处理  
  
场景：一般用于框架、插件等场景

**14.html和xhtml有什么区别?**

HTML是一种基本的WEB网页设计语言，XHTML是一个基于XML的标记语言。  
1.XHTML 元素必须被正确地嵌套。  
2.XHTML 元素必须被关闭。  
3.标签名必须用小写字母。  
4.空标签也必须被关闭。  
5.XHTML 文档必须拥有根元素。

**15. 什么是构造函数？与普通函数有什么区别?**

构造函数：是一种特殊的方法、主要用来创建对象时初始化对象，总与new运算符一起使用，创建对象的语句中构造函数的函数名必须与类名完全相同。  
与普通函数相比只能由new关键字调用，构造函数是类的标示

**16. 通过new创建一个对象的时候，函数内部有哪些改变**

**function** Person(){}Person.prototype.friend **=** [];Person.prototype.name **=** '';*// var a = new Person();// a.friend[0] = '王琦';// a.name = '程娇';// var b = new Person();// b.friend？// b.name?*

1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。   
2、属性和方法被加入到 this 引用的对象中。  
3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。

**17.事件委托？有什么好处?**

（1）利用冒泡的原理，把事件加到父级上，触发执行效果  
（2）好处：新添加的元素还会有之前的事件；提高性能。

**18.window.onload ==? DOMContentLoaded ?**

一般情况下，DOMContentLoaded事件要在window.onload之前执行，当DOM树构建完成的时候就会执行DOMContentLoaded事件，而window.onload是在页面载入完成的时候，才执行，这其中包括图片等元素。大多数时候我们只是想在DOM树构建完成后，绑定事件到元素，我们并不需要图片元素，加上有时候加载外域图片的速度非常缓慢。

**19.节点类型?判断当前节点类型?**

1. 元素节点   
2. 属性节点   
3. 文本节点   
8. 注释节点   
9. 文档节点   
通过nodeObject.nodeType判断节点类型：其中，nodeObject 为DOM节点（节点对象）。该属性返回以数字表示的节点类型，例如，元素节点返回 1，属性节点返回 2 。

**20.如何合并两个数组？数组删除一个元素?**

*//三种方法。* （1）**var** arr1**=**[1,2,3];

**var** arr2**=**[4,5,6];

arr1 **=** arr1.concat(arr2);

console.log(arr1);

（2）**var** arr1**=**[1,2,3];

**var** arr2**=**[4,5,6];

Array.prototype.push.apply(arr1,arr2);

console.log(arr1);

（3）**var** arr1**=**[1,2,3];

**var** arr2**=**[4,5,6];

**for** (**var** i**=**0; i **<** arr2.length; i**++**) {

arr1.push( arr2[i] );

}

console.log(arr1);

**21.强制转换 显式转换 隐式转换?**

*//强制类型转换：*Boolean(0) *// => false - 零* Boolean(**new** object()) *// => true - 对象* Number(**undefined**) *// => NaN* Number(**null**) *// => 0* String(**null**) *// => "null"*parseInt( )parseFloat( )JSON.parse( )JSON.stringify ( )

隐式类型转换：  
在使用算术运算符时，运算符两边的数据类型可以是任意的，比如，一个字符串可以和数字相加。之所以不同的数据类型之间可以做运算，是因为JavaScript引擎在运算之前会悄悄的把他们进行了隐式类型转换的  
（例如：x+"" //等价于String(x)  
+x //等价于Number(x)  
x-0 //同上  
!!x //等价于Boolean(x),是双叹号）  
  
显式转换：   
如果程序要求一定要将某一类型的数据转换为另一种类型，则可以利用强制类型转换运算符进行转换，这种强制转换过程称为显示转换。  
显示转换是你定义让这个值类型转换成你要用的值类型，是底到高的转换。例 int 到float就可以直接转，int i=5,想把他转换成char类型，就用显式转换（char）i

**22. Jq中如何实现多库并存?**

Noconfict 多库共存就是“$ ”符号的冲突。   
  
方法一： 利用jQuery的实用函数$.noConflict();这个函数归还$的名称控制权给另一个库，因此可以在页面上使用其他库。这时，我们可以用"jQuery "这个名称调用jQuery的功能。 $.noConflict();   
jQuery('#id').hide();   
.....  
//或者给jQuery一个别名   
var $j=jQuery   
$j('#id').hide();   
.....  
  
方法二： (function($){})(jQuery)   
  
方法三： jQuery(function($){})   
通过传递一个函数作为jQuery的参数，因此把这个函数声明为就绪函数。 我们声明$为就绪函数的参数，因为jQuery总是吧jQuery对象的引用作为第一个参数传递，所以就保证了函数的执行。

**23.Jq中get和eq有什么区别？**

get() :取得其中一个匹配的元素。num表示取得第几个匹配的元素，get多针对集合元素，返回的是DOM对象组成的数组 eq():获取第N个元素，下标都是从0开始，返回的是一个JQuery对象

**24.如何通过原生js 判断一个元素当前是显示还是隐藏状态?**

**if**( document.getElementById("div").css("display")**===**'none')**if**( document.getElementById("div").css("display")**===**'block')$("#div").is(":hidden"); *// 判断是否隐藏*$("#div").is(":visible")

**25.Jq如何判断元素显示隐藏？**

*//第一种：使用CSS属性* **var** display **=**$('#id').css('display'); **if**(display **==** 'none'){ alert("我是隐藏的！"); }*//第二种：使用jquery内置选择器* **<**div id**=**"test"**>** **<**p**>**仅仅是测试所用**<**/p> </div>**if**($("#test").is(":hidden")){ $("#test").show(); *//如果元素为隐藏,则将它显现 }else{ $("#test").hide(); //如果元素为显现,则将其隐藏 }//第三种：jQuery判断元素是否显示 是否隐藏***var** node**=**$('#id');**if**(node.is(':hidden')){　　*//如果node是隐藏的则显示node元素，否则隐藏*　　node.show();　}**else**{

　　node.hide();}

**26.移动端上什么是点击穿透?**

点击穿透现象有3种：  
点击穿透问题：点击蒙层（mask）上的关闭按钮，蒙层消失后发现触发了按钮下面元素的click事件跨页面点击穿透问题：如果按钮下面恰好是一个有href属性的a标签，那么页面就会发生跳转另一种跨页面点击穿透问题：这次没有mask了，直接点击页内按钮跳转至新页，然后发现新页面中对应位置元素的click事件被触发了  
  
解决方案：  
1、只用touch  
最简单的解决方案，完美解决点击穿透问题  
把页面内所有click全部换成touch事件（ touchstart 、’touchend’、’tap’）  
  
2、只用click  
下下策，因为会带来300ms延迟，页面内任何一个自定义交互都将增加300毫秒延迟  
  
3、tap后延迟350ms再隐藏mask  
改动最小，缺点是隐藏mask变慢了，350ms还是能感觉到慢的  
  
4、pointer-events  
比较麻烦且有缺陷， 不建议使用mask隐藏后，给按钮下面元素添上 pointer-events: none; 样式，让click穿过去，350ms后去掉这个样式，恢复响应缺陷是mask消失后的的350ms内，用户可以看到按钮下面的元素点着没反应，如果用户手速很快的话一定会发现

**27.Jq绑定事件的几种方式？on bind ?**

jQuery中提供了四种事件监听方式，分别是bind、live、delegate、on，对应的解除监听的函数分别是unbind、die、undelegate、off  
  
Bind( )是使用频率较高的一种，作用就是在选择到的元素上绑定特定事件类型的监听函数;  
  
Live( )可以对后生成的元素也可以绑定相应的事件,处理机制就是把事件绑定在DOM树的根节点上，而不是直接绑定在某个元素上;  
  
Delegate( )采用了事件委托的概念，不是直接为子元素绑定事件，而是为其父元素（或祖先元素也可）绑定事件，当在div内任意元素上点击时，事件会一层层从event target向上冒泡，直至到达你为其绑定事件的元素；  
  
on( )方法可以绑定动态添加到页面元素的事件，on()方法绑定事件可以提升效率；

**28.Jq中如何将一个jq对象转化为dom对象？**

方法一：  
jQuery对象是一个数据对象，可以通过[index]的方法，来得到相应的DOM对象。   
如：var $v =$("#v") ; //jQuery对象   
var v=$v[0]; //DOM对象   
alert(v.checked) //检测这个checkbox是否被选中   
  
方法二：  
jQuery本身提供，通过.get(index)方法，得到相应的DOM对象   
如：var $v=$("#v"); //jQuery对象   
var v=$v.get(0); //DOM对象   
alert(v.checked) //检测这个checkbox是否被选中

**29.Jq中有几种选择器?分别是什么?**

层叠选择器、基本过滤选择器、内容过滤选择器、可视化过滤选择器、属性过滤选择器、子元素过滤选择器、表单元素选择器、表单元素过滤选择器

**30.Jq中怎么样编写插件?**

*//第一种是类级别的插件开发：//1.1 添加一个新的全局函数 添加一个全局函数，我们只需如下定义：* jQuery.foo **=** **function**() {

alert('This is a test. This is only a test.'); };

*//1.2 增加多个全局函数 添加多个全局函数，可采用如下定义：* jQuery.foo **=** **function**() {

alert('This is a test. This is only a test.'); }; jQuery.bar **=** **function**(param) {

alert('This function takes a parameter, which is "' **+** param **+** '".'); }; 调用时和一个函数的一样的**:**jQuery.foo();jQuery.bar();或者$.foo();$.bar('bar');*//1.3 使用jQuery.extend(object);* jQuery.extend({

foo**:** **function**() {

alert('This is a test. This is only a test.');

},

bar**:** **function**(param) {

alert('This function takes a parameter, which is "' **+** param **+**'".');

}

}); *//1.4 使用命名空间// 虽然在jQuery命名空间中，我们禁止使用了大量的javaScript函数名和变量名。// 但是仍然不可避免某些函数或变量名将于其他jQuery插件冲突，因此我们习惯将一些方法// 封装到另一个自定义的命名空间。*jQuery.myPlugin **=** { foo**:function**() {

alert('This is a test. This is only a test.');

},

bar**:function**(param) {

alert('This function takes a parameter, which is "' **+** param **+** '".');

} }; *//采用命名空间的函数仍然是全局函数，调用时采用的方法：* $.myPlugin.foo(); $.myPlugin.bar('baz');*//通过这个技巧（使用独立的插件名），我们可以避免命名空间内函数的冲突。//第二种是对象级别的插件开发//形式1：* (**function**($){

$.fn.extend({

pluginName**:function**(opt,callback){

*// Our plugin implementation code goes here.*  }

}) })(jQuery);

*//形式2：*(**function**($) {

$.fn.pluginName **=** **function**() {

*// Our plugin implementation code goes here.*  }; })(jQuery);*//形参是$，函数定义完成之后,把jQuery这个实参传递进去.立即调用执行。//这样的好处是,我们在写jQuery插件时,也可以使用$这个别名,而不会与prototype引起冲突*

**31.$('div+.ab')和$('.ab+div') 哪个效率高？**

$('div+.ab')效率高

**32.$.map和$.each有什么区别**

map()方法主要用来遍历操作数组和对象，会返回一个新的数组。$.map()方法适用于将数组或对象每个项目新阵列映射到一个新数组的函数；  
each()主要用于遍历jquery对象，返回的是原来的数组，并不会新创建一个数组。

**33.编写一个 getElementsByClassName 封装函数?**

**<**body**>** **<**input type**=**"submit" id **=** "sub" **class=**"ss confirm btn" value**=**"提交"**/>** **<**script**>** window.onload **=** **function**(){ *//方法一*  **var** Opt **=** document.getElementById('sub');

**var** getClass **=** **function**(className,tagName){

**if**(document.getElementsByTagName){

**var** Inp **=** document.getElementsByTagName(tagName);

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

**if**((**new** RegExp('(\\s|^)' **+**className **+**'(\\s|$)')).test(Inp[i].className)){

**return** Inp[i];

}

}

}**else** **if**(document.getElementsByClassName){

**return** document.getElementsByClassName(className);

}

} *//方法二* **var** aa **=** getClass("confirm", "input");

**function** getClass(className, targetName){

**var** ele **=** [];

**var** all **=** document.getElementsByTagName(targetName **||** "\*");

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

**if**(all[i].className.match(**new** RegExp('(\\s|^)'**+**confirm**+**'(\\s|$)'))){

ele[ele.length] **=** all[i];

}

}

**return** ele;

}*//方法三* **function** getObjsByClass(tagName, className){

**if**(document.getElementsByClassName){

alert("document.getElementsByClassName");

**return** document.getElementsByClassName(className);

}**else**{

**var** el **=** [];

**var** \_el **=** document.getElementsByTagName(tagName);

**for**(**var** i**=**0; i**<**\_el.length; i**++**){

**if**(\_el[i].className.indexOf(className) **>** **-**1){

alert(\_el[i]);

el[\_el.length] **=** \_el[i];

}

}

alert(el);

**return** el;

}

}

}

**<**/script>**<**/body>

**34.简述下工作流程**

我在之前的公司工作流程大概是这样的：公司定稿会结束以后，会进行简单的技术研讨，然后我们前端会进行先期的技术准备。前端切图人员会进行psd设计稿切图，并且将css文件进行整合。我们主要编写JS部分，其中包括搭建前端框架（大项目），编写js业务和数据持久化操作，我们也会编写js插件并且进行封装方便使用，还有就是编写JS前端组建和JS测试单元，最后将完成的JS部分与切图人员提供的HTML页面进行整合。最后对完成的页面进行功能测试、页面兼容、产品还原。然后对产品进行封存，提交测试。如果出现BUG会返回给我们开发人员进行修改，再提交测试，最后测试成功，进行版本封存。等到程序全部上线的时候进行线上测试。

**35.一般使用什么版本控制工具?svn如何对文件加锁**

svn加锁目的：为了避免多个人同一时间对同一个文件改动的相互覆盖，版本控制系统就必须有一套冲突处理机制。  
  
svn加锁两种策略：乐观加锁：所有签出的文件都是可读写的，对文件的修改不必获得文件的锁，当你修改完文件签入时，会首先要求你更新本地文件，版本控制系统不会覆盖你的本地修改，而是会让你自己合并冲突后签入。  
  
严格加锁：所有签出的文件都是只读的，任何对文件的修改必须要获得文件的锁，如果其他人没有拥有该文件的锁，那么版本控制系统就会授权给你文件的锁，并将文件设置为可编辑的。  
  
svn两种加锁步骤：乐观加锁：选择你想要获取锁定的文件，然后右键菜单点击TortoiseSVN 选取获取锁定。  
  
严格加锁：在想要采取严格加锁的文件或目录上点击右键，使用TortoiseSVN 属性菜单，点击新建属性，选择需要锁定。

**36. git 和 svn的区别?**

SVN是集中式版本控制系统，版本库是集中放在中央服务器的，而干活的时候，用的都是自己的电脑，所以首先要从中央服务器哪里得到最新的版本，然后干活，干完后，需要把自己做完的活推送到中央服务器。集中式版本控制系统是必须联网才能工作，如果在局域网还可以，带宽够大，速度够快，如果在互联网下，如果网速慢的话，就纳闷了。  
  
Git是分布式版本控制系统，那么它就没有中央服务器的，每个人的电脑就是一个完整的版本库，这样，工作的时候就不需要联网了，因为版本都是在自己的电脑上。既然每个人的电脑都有一个完整的版本库，那多个人如何协作呢？比如说自己在电脑上改了文件A，其他人也在电脑上改了文件A，这时，你们两之间只需把各自的修改推送给对方，就可以互相看到对方的修改了。

**37. jquery和zepto有什么区别?**

1.针对移动端程序，Zepto有一些基本的触摸事件可以用来做触摸屏交互（tap事件、swipe事件），Zepto是不支持IE浏览器的，这不是Zepto的开发者Thomas Fucks在跨浏览器问题上犯了迷糊，而是经过了认真考虑后为了降低文件尺寸而做出的决定，就像jQuery的团队在2.0版中不再支持旧版的IE（6 7 8）一样。因为Zepto使用jQuery句法，所以它在文档中建议把jQuery作为IE上的后备库。那样程序仍能在IE中，而其他浏览器则能享受到Zepto在文件大小上的优势，然而它们两个的API不是完全兼容的，所以使用这种方法时一定要小心，并要做充分的测试。  
  
2.Dom操作的区别：添加id时jQuery不会生效而Zepto会生效。  
  
3.zepto主要用在移动设备上，只支持较新的浏览器，好处是代码量比较小，性能也较好。  
jquery主要是兼容性好，可以跑在各种pc，移动上，好处是兼容各种浏览器，缺点是代码量大，同时考虑兼容，性能也不够好。

**38. $(function(){})和window.onload 和 $(document).ready(function(){})**

window.onload:用于当页面的所有元素，包括外部引用文件，图片等都加载完毕时运行函数内的函数。load方法只能执行一次，如果在js文件里写了多个，只能执行最后一个。  
  
$(document).ready(function(){})和$(function(){})都是用于当页面的标准DOM元素被解析成DOM树后就执行内部函数。这个函数是可以在js文件里多次编写的，对于多人共同编写的js就有很大的优势，因为所有行为函数都会执行到。而且$(document).ready()函数在HMTL结构加载完后就可以执行，不需要等大型文件加载或者不存在的连接等耗时工作完成才执行，效率高。

**39. Jq中 attr 和 prop 有什么区别**

对于HTML元素本身就带有的固有属性，在处理时，使用prop方法。  
对于HTML元素我们自己自定义的DOM属性，在处理时，使用attr方法。

**40. 简述下 this 和定义属性和方法的时候有什么区别?Prototype？**

this表示当前对象，如果在全局作用范围内使用this，则指代当前页面对象window； 如果在函数中使用this，则this指代什么是根据运行时此函数在什么对象上被调用。 我们还可以使用apply和call两个全局方法来改变函数中this的具体指向。  
  
prototype本质上还是一个JavaScript对象。 并且每个函数都有一个默认的prototype属性。  
  
在prototype上定义的属性方法为所有实例共享，所有实例皆引用到同一个对象，单一实例对原型上的属性进行修改，也会影响到所有其他实例。

**41. 什么是预编译语音|预编译处理器?**

Sass是一种CSS预处理器语言，通过编程方式生成CSS代码。因为可编程，所以操控灵活性自由度高，方便实现一些直接编写CSS代码较困难的代码。  
  
同时，因为Sass是生成CSS的语言，所以写出来的Sass文件是不能直接用的，必须经过编译器编译成CSS文件才能使用。  
  
CSS 预处理器是一种语言用来为 CSS 增加一些编程的的特性，无需考虑浏览器的兼容性问题，例如你可以在 CSS 中使用变量、简单的程序逻辑、函数等等在编程语言中的一些基本技巧，可以让你的 CSS 更见简洁，适应性更强，代码更直观等诸多好处。最常用的css预处理器有sass、less css、 stylus。

**42.ajax 和 jsonp ？**

ajax和jsonp的区别：  
相同点：都是请求一个url  
不同点：ajax的核心是通过xmlHttpRequest获取内容  
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。

**43.ajax执行流程？**

1. 创建XMLHttpRequest对象,也就是创建一个异步调用对象  
2. 创建一个新的HTTP请求,并指定该HTTP请求的方法、URL及验证信息  
3. 设置响应HTTP请求状态变化的函数  
4. 发送HTTP请求  
5. 获取异步调用返回的数据  
6. 使用JavaScript和DOM实现局部刷新

**44.xhr对象 status ? readystate?**

status是XMLHttpRequest对象的一个属性，表示响应的HTTP状态码。  
readyState是XMLHttpRequest对象的一个属性，用来标识当前XMLHttpRequest对象处于什么状态。

**45.readystate 0~4**

0：未初始化状态：此时，已经创建了一个XMLHttpRequest对象  
1： 准备发送状态：此时，已经调用了XMLHttpRequest对象的open方法，并且XMLHttpRequest对象已经准备好将一个请求发送到服务器端  
2：已经发送状态：此时，已经通过send方法把一个请求发送到服务器端，但是还没有收到一个响应  
3：正在接收状态：此时，已经接收到HTTP响应头部信息，但是消息体部分还没有完全接收到  
4：完成响应状态：此时，已经完成了HTTP响应的接收

**46.说出几个http协议状态码?**

200, 201, 302, 304, 400, 404, 500  
  
200：请求成功  
201：请求成功并且服务器创建了新的资源  
302：服务器目前从不同位置的网页响应请求，但请求者应继续使用原有位置来响应以后的请求。  
304：自从上次请求后，请求的网页未修改过。服务器返回此响应时，不会返回网页内容。  
400：服务器不理解请求的语法。  
404：请求的资源（网页等）不存在  
500： 内部服务器错误

**47.上一个项目是什么？主要负责哪些？购物车流程?支付功能?**

主要负责哪些就讲主要做哪些功能模块：  
  
1）商品模块:  
1、商品列表：商品排序 商品筛选 商品过滤 商品查询 商品推荐  
2、商品详情:类型推荐 商品简介 商品详情 商品评价 售后维护   
  
2)购物车模块：商品编号、数量、价格、总额、运费、运输选项、运费总计、从购物车删除选项、更新数量、结账、继续购物、商品描述、库存信息

**48.sessionStorage和localstroage与cookie之间有什么关联, cookie最大存放多少字节**

三者共同点：都是保存在浏览器端，且同源的。  
  
区别:  
1、cookie在浏览器和服务器间来回传递。而sessionStorage和localStorage不会自动把数据发给服务器，仅在本地保存  
  
2、存储大小限制也不同，cookie数据不能超过4k，sessionStorage和localStorage 但比cookie大得多，可以达到5M  
  
3、数据有效期不同，sessionStorage：仅在当前浏览器窗口关闭前有效，自然也就不可能持久保持；localStorage：始终有效，窗口或浏览器关闭也一直保存，因此用作持久数据；cookie只在设置的cookie过期时间之前一直有效，即使窗口或浏览器关闭  
  
4、作用域不同，sessionStorage不在不同的浏览器窗口中共享，即使是同一个页面(即数据不共享)；localStorage 在所有同源窗口中都是共享的；cookie也是在所有同源窗口中都是共享的( 即数据共享 )。

**49.ajax中 get 和 post 有什么区别?**

get和post都是数据提交的方式。  
get的数据是通过网址问号后边拼接的字符串进行传递的。post是通过一个HTTP包体进行传递数据的。  
get的传输量是有限制的，post是没有限制的。  
get的安全性可能没有post高，所以我们一般用get来获取数据，post一般用来修改数据。

**50.Gc机制是什么？为什么闭包不会被回收变量和函数？**

1、Gc垃圾回收机制;  
2、外部变量没释放，所以指向的大函数内的小函数也释放不了

**51.简述下你理解的面向对象？**

万物皆对象，把一个对象抽象成类,具体上就是把一个对象的静态特征和动态特征抽象成属性和方法,也就是把一类事物的算法和数据结构封装在一个类之中,程序就是多个对象和互相之间的通信组成的.  
  
面向对象具有封装性,继承性,多态性。  
封装:隐蔽了对象内部不需要暴露的细节,使得内部细节的变动跟外界脱离,只依靠接口进行通信.封装性降低了编程的复杂性. 通过继承,使得新建一个类变得容易,一个类从派生类那里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而继承和实现接口和运行时的类型绑定机制 所产生的多态,使得不同的类所产生的对象能够对相同的消息作出不同的反应,极大地提高了代码的通用性.   
  
总之,面向对象的特性提高了大型程序的重用性和可维护性.

**52.this是什么 在不同场景中分别代表什么**

（1）function a(){ this ?} //This:指向windows  
  
（2）function b(){ return function(){ this ?}}b()(); //This:指向windows  
  
（3）function c(){ return {s:function(){this}}}c().s(); //This:指向object  
  
由于其运行期绑定的特性，JavaScript 中的 this 含义要丰富得多，它可以是全局对象、当前对象或者任意对象，这完全取决于函数的调用方式。

**53.你对数据校验是怎么样处理的？jquery.validate？**

通俗的说，就是为保证数据的完整性，用一种指定的算法对原始数据计算出的一个校验值。接收方用同样的算法计算一次校验值，如果和随数据提供的校验值一样，就说明数据是完整的。   
用正则表达式来处理;  
jquery.validate：为表单验证插件

**54.如何对登录的账号密码进行加密?**

Md5

**55.在jq中 mouseover mouseenter mouseout mouseleave 和 hover有什么关联?**

mouseenter与mouseover：  
不论鼠标指针穿过被选中元素或其子元素，都会触发mouseover事件。  
只有在鼠标指针穿过被选元素时，才会触发mouseentr事件。  
  
mouseout与mouseleave：  
不论鼠标离开被选元素还是任何子元素，都会触发mouseout事件。  
只有在鼠标指针离开被选元素时，才会触发mouseleave事件。  
  
hover:  
hover是一个符合方法，相当于mouseenter+mouseleave。

**56.jsonp原理？ 缺点?**

工作原理：使用script标签实现跨域访问，可在url中指定回调函数，获取JSON数据并在指定的回调函数中执行jquery实现jsop。  
  
缺点：只支持GET方式的jsonp实现，是一种脚本注入行为存在一定的安全隐患。如果返回的数据格式有问题或者返回失败了，并不会报错。

**57.除了jsonp 还有什么跨域方式**

javascript跨域有两种情况：   
1、基于同一父域的子域之间，如：[http://a.c.com](https://link.zhihu.com/?target=http://a.c.com" \t "/Users/shina/Documents\\x/_blank)和[http://b.c.com](https://link.zhihu.com/?target=http://b.c.com" \t "/Users/shina/Documents\\x/_blank)   
2、基于不同的父域之间，如：[http://www.a.com](https://link.zhihu.com/?target=http://www.a.com" \t "/Users/shina/Documents\\x/_blank)和[http://www.b.com](https://link.zhihu.com/?target=http://www.b.com" \t "/Users/shina/Documents\\x/_blank)   
3、端口的不同，如：[http://www.a.com:8080](https://link.zhihu.com/?target=http://www.a.com:8080" \t "/Users/shina/Documents\\x/_blank)和[http://www.a.com:8088](https://link.zhihu.com/?target=http://www.a.com:8088" \t "/Users/shina/Documents\\x/_blank)   
4、协议不同，如：[http://www.a.com](https://link.zhihu.com/?target=http://www.a.com" \t "/Users/shina/Documents\\x/_blank)和[https://www.a.com](https://link.zhihu.com/?target=https://www.a.com" \t "/Users/shina/Documents\\x/_blank)   
  
对于情况3和4，需要通过后台proxy来解决，具体方式如下：   
a、在发起方的域下创建proxy程序   
b、发起方的js调用本域下的proxy程序   
c、proxy将请求发送给接收方并获取相应数据   
d、proxy将获得的数据返回给发起方的js   
  
代码和ajax调用一致，其实这种方式就是通过ajax进行调用的  
而情况1和2除了通过后台proxy这种方式外，还可以有多种办法来解决：   
1、document.domain+iframe（只能解决情况1）：   
a、在发起方页面和接收方页面设置document.domain，并将值设为父域的主域名(window.location.hostname)   
b、在发起方页面创建一个隐藏的iframe，iframe的源是接收方页面   
c、根据浏览器的不同，通过iframe.contentDocument || iframe.contentWindow.document来获得接收方页面的内容   
d、通过获得的接收方页面的内容来与接收方进行交互   
这种方法有个缺点，就是当一个域被攻击时，另一个域会有安全漏洞出现。

**58.如何使用storage 对js文件进行缓存**

由于sessionStorage - 针对一个 session 的数据存储，所以我们一般利用localStorage储存js文件，只有在第一次访问该页面的时候加载js文件，以后在访问的时候加载本地localStorage执行

**59.如何确保ajax或连接不走缓存路径**

在Ajax中使用Get请求数据不会有页面缓存的问题，而使用POST请求可是有时候页面会缓存我们提交的信息，导致我们发送的异步请求不能正确的返回我们想要的数据  
  
$.post(url,data ,ranNum:Math.random()} ,function(data){}）  
  
ranNum : 这个是防止缓存的核心，每次发起请求都会用Math.random()方法生成一个随机的数字，这样子就会刷新url缓存

**60.split() join()?**

前者是切割成数组的形式，  
后者是将数组转换成字符串

**61.slice() splice()?**

slice() 方法可从已有的数组中返回选定的元素。  
splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。

**62.typeof？typeof [ ]返回数据类型是？**

*//判断基本数据类型；var a=[];typeof a输出object；//本来判断一个对象类型用typeof是最好的，不过对于Array类型是不适用的，//可以使用 instanceof操作符：* **var** arrayStr**=new** Array("1","2","3","4","5");

alert(arrayStr **instanceof** Array); *//当然以上在一个简单的页面布局里面是没有问题的，如果是复杂页面情况，//入获取的是frame内部的Array对象，可以用这个函数判断：* **function** isArray(obj) {

**return** Object.prototype.toString.call(obj) **===** '[object Array]';

}

**63.disabled readyonly?**

readonly只针对input(text / password)和textarea有效，而disabled对于所有的表单元素都有效，当表单元素在使用了disabled后，当我们将表单以POST或GET的方式提交的话，这个元素的值不会被传递出去，而readonly会将该值传递出去。

**64.同步异步?**

1、进程同步：就是在发出一个功能调用时，在没有得到结果之前，该调用就不返回。也就是必须一件一件事做,等前一件做完了才能做下一件事  
  
2、异步的概念和同步相对。当一个异步过程调用发出后，调用者不能立刻得到结果。实际处理这个调用的部件在完成后，通过状态、通知和回调来通知调用者。

**65.promise**

Promise的构造函数接收一个参数，是函数，并且传入两个参数：resolve，reject，分别表示异步操作执行成功后的回调函数和异步操作执行失败后的回调函数。

**66.函数fn1 函数fn2 函数fn3，如果想在三个函数都执行完成后执行某一个事件应该如何实现?**

*//1、设置事件监听。//2、回调函数：***function** fn1(){

console.log("执行fn1");

fn2();}**function** fn2(){

console.log("执行fn2");

fn3();}**function** fn3(){

console.log("执行fn3");

mou();}**function** mou(){

console.log("执行某个函数");}fn1();

**67.JavaScript提供了哪几种“异步模式”？**

1、回调函数（callbacks）  
2、事件监听  
3、Promise对象

**68.什么是移动端的300ms延迟？什么是点击穿透？解决方案?**

移动端300ms延迟：假定这么一个场景。用户在 浏览器里边点击了一个链接。由于用户可以进行双击缩放或者双击滚动的操作，当用户一次点击屏幕之后，浏览器并不能立刻判断用户是确实要打开这个链接，还是想要进行双击操作。因此，浏览器 就等待 300 毫秒，以判断用户是否再次点击了屏幕。也就是说，当我们点击页面的时候移动端浏览器并不是立即作出反应，而是会等上一小会儿才会出现点击的效果。  
  
点击穿透：假如页面上有两个元素A和B。B元素在A元素之上。我们在B元素的touchstart事件上注册了一个回调函数，该回调函数的作用是隐藏B元素。我们发现，当我们点击B元素，B元素被隐藏了，随后，A元素触发了click事件。这是因为在移动端浏览器，事件执行的顺序是touchstart > touchend > click。而click事件有300ms的延迟，当touchstart事件把B元素隐藏之后，隔了300ms，浏览器触发了click事件，但是此时B元素不见了，所以该事件被派发到了A元素身上。如果A元素是一个链接，那此时页面就会意外地跳转。  
  
300ms延迟解决方案：  
(1) 禁用缩放，在html文档头部加meta标签如下：  
<meta name=”viewport” content=”user-scalable=no”/>  
  
(2) 更改默认的视口宽度 （响应式布局，消除了站点上可能存在的双击绽放的请求）  
<meta name=”viewport” content=”width=device-width”/>  
  
(3) Css touch-action  
touch-action:none;在该元素上的操作不会触发用户代理的任何行为，无需进行3000ms延迟判断。  
  
(4) FastClick为解决移动端浏览器300毫秒延迟开发的一个轻量级的库  
点击穿透解决方案：  
（1）只用touch  
（2）只用click  
（3）tap后延迟350ms再隐藏mask  
（4）pointer-events

**69.变量作用域?**

*//变量作用域：一个变量的作用域是程序源代码中定义这个变量的区域。全局变量拥有全局作用域，//在js代码中任何地方都是有定义的。在函数内声明的变量只在函数体内有定义，它们是局部变量，//作用域是局部性的。函数参数也是局部变量，它们只在函数体内有定义。***var** a **=** "";window.b**=**''”;**function**(e) {

**var** c**=** "";

d**=**"";

e**=**"";}**function** go() {

console.info(**this**);*//window* **return** **function**() {

console.info(**this**); *// window* **return** {

b**:function**(){

console.info(**this**); *//b的父对象* }

}

}}go()().b();

**70.call & apply 两者之间的区别**

call和apply都是改变this指向的方法，区别在于call可以写多个参数，而apply只能写两个参数，第二个参数是一个数组，用于存放要传的参数。

**71.call和apply 有什么好处？**

用call和apply:实现更好的继承和扩展，更安全。

**72.谁是c的构造函数?**

**function** ab() {

**this**.say **=** ""; } ab.constructor **=** {} ab.name **=** ''; **var** c **=** **new** ab(); *//构造函数默认指向函数本身,ab是一个类,它的构造函数是它本身，//然后ab.constructor={};ab的构造函数就指向{}了，c是ab的实例化对象，c的构造函数就是{}//通过使用new的时候,创建对象发生了那些改变? 当使用new操作时，会马上开辟一个块内存，//创建一个空对象，并将this指向这个对象。接着，执行构造函数ab()，对这个空对象进行构造//（构造函数里有什么属性和方法都一一给这个空白对象装配上去，这就是为何它叫构造函数了）。*

**73.sass和less有什么区别?**

1.编译环境不一样 Sass的安装需要Ruby环境，是在服务端处理的，而Less是需要引入less.js来处理Less代码输出css到浏览器，也可以在开发环节使用Less，然后编译成css文件，直接放到项目中。   
  
2.变量符不一相，less是@，而scss是$,而且它们的作用域也不一样，less是块级作用域   
  
3.输出设置，Less没有输出设置，sass提供4种输出选项，nested,compact,compressed和expanded nested：嵌套缩进的css代码(默认) expanded：展开的多行css代码 compact：简洁格式的css代码 compressed：压缩后的css代码   
  
4.sass支持条件语句，可以使用if{}else{},for{}循环等等，而less不行   
  
5.引用外部css文件，sass引用外部文件必须以\_开头，文件名如果以下划线\_形状，sass会认为该文件是一个引用文件，不会将其编译为css文件。less引用外部文件和css中的@import没什么差异。   
  
6.sass和less的工具库不同。sass有工具库Compass, 简单说，sass和Compass的关系有点像Javascript和jQuery的关系,Compass是sass的工具库。在它的基础上，封装了一系列有用的模块和模板，补充强化了sass的功能。less有UI组件库Bootstrap,Bootstrap是web前端开发中一个比较有名的前端UI组件库，Bootstrap的样式文件部分源码就是采用less语法编写。   
  
总结：不管是sass，还是less，都可以视为一种基于CSS之上的高级语言，其目的是使得CSS开发更灵活和更强大，sass的功能比less强大,基本可以说是一种真正的编程语言了，less则相对清晰明了,易于上手,对编译环境要求比较宽松。考虑到编译sass要安装Ruby,而Ruby官网在国内访问不了,个人在实际开发中更倾向于选择less。

**74.bootstrap好处？**

自适应和响应式布局，12栅格系统，统一的界面风格和css样式有利于团队开发。编写灵活、稳定、高质量的 HTML 和 CSS 代码的规范。

**75.开发时如何对项目进行管理?gulp?**

本人开发时，利用gulp等前端工作流管理工具管理项目。 gulp是新一代的前端项目构建工具，你可以使用gulp及其插件对你的项目代码（less,sass）进行编译，还可以压缩你的js和css代码，甚至压缩你的图片，能够合并文件，压缩文件，语法检查，监听文件变化，测试，转换二进制，转换图片等一系列功能。gulp仅有少量的API，所以非常容易学习。实现良好的项目管理。

**76.压缩合并目的？http请求的优化方式？**

1）Web性能优化最佳实践中最重要的一条是减少HTTP请求。而减少HTTP请求的最主要的方式就是，合并并压缩JavaScript和CSS文件。   
  
CSS Sprites（CSS精灵）：把全站的图标都放在一个图像文件中，然后用CSS的background-image和background-position属性定位来显示其中的一小部分。   
  
合并脚本和样式表; 图片地图：利用image map标签定义一个客户端图像映射，（图像映射指带有可点击区域的一幅图像）具体看：[http://club.topsage.com/thread-2527479-1-1.html](https://link.zhihu.com/?target=http://club.topsage.com/thread-2527479-1-1.html" \t "/Users/shina/Documents\\x/_blank)   
  
图片js/css等静态资源放在静态服务器或CDN服时，尽量采用不用的域名，这样能防止cookie不会互相污染，减少每次请求的往返数据。   
  
css替代图片, 缓存一些数据   
少用location.reload()：使用location.reload() 会刷新页面，刷新页面时页面所有资源 (css，js，img等) 会重新请求服务器。建议使用location.href="当前页url" 代替location.reload() ，使用location.href 浏览器会读取本地缓存资源。

**77.ajax请求方式有几种（8种）？**

1）$.get(url,[data],[callback])   
2）$.getJSON(url,[data],[callback])   
3）$.post(url,[data],[callback],[type])   
4）$.ajax(opiton)   
5）$.getScript( url, [callback] )   
6）jquery对象.load( url, [data], [callback] )   
7）serialize() 与 serializeArray()

**78.如何copy一个dom元素？**

原生Js方法：var div = document.getElementsByTagName('div')[0];  
var clone = div.cloneNode();  
Jquery方法：$('div').clone();  
在默认情况下，.clone()方法不会复制匹配的元素或其后代元素中绑定的事件。不过，可以为这个方法传递一个布尔值参数，将这个参数设置为true, 就可以连同事件一起复制，即.clone(true)。

**79.数组的排序方法（sort）？排序？汉字排序？**

数组的排序方法：reverse()和sort()。reverse()方法会对反转数组项的顺序。  
Eg:var values = [0, 1, 5, 10, 15]; values.sort();//0,1,10,15,5  
var values = [1, 2, 3, 4, 5]; values.reverse();//5,4,3,2,1  
  
js中的排序（详情参考：[http://www.tuicool.com/articles/IjInMbU](https://link.zhihu.com/?target=http://www.tuicool.com/articles/IjInMbU" \t "/Users/shina/Documents\\x/_blank)）  
利用sort排序, 冒泡排序, 快速排序, 插入排序, 希尔排序, 选择排序  
归并排序  
  
localeCompare() 方法用于字符串编码的排序  
localeCompare 方法：返回一个值，指出在当前的区域设置中两个字符串是否相同。

**80.简述一下你理解的面向对象？**

面向对象是基于万物皆对象这个哲学观点. 把一个对象抽象成类,具体上就是把一个对象的静态特征和动态特征抽象成属性和方法,也就是把一类事物的算法和数据结构封装在一个类之中,程序就是多个对象和互相之间的通信组成的.  
  
面向对象具有封装性,继承性,多态性。  
  
封装:隐蔽了对象内部不需要暴露的细节,使得内部细节的变动跟外界脱离,只依靠接口进行通信.封装性降低了编程的复杂性. 通过继承,使得新建一个类变得容易,一个类从派生类那里获得其非私有的方法和公用属性的繁琐工作交给了编译器. 而 继承和实现接口和运行时的类型绑定机制 所产生的多态,使得不同的类所产生的对象能够对相同的消息作出不同的反应,极大地提高了代码的通用性.   
  
总之,面向对象的特性提高了大型程序的重用性和可维护性.

**81.如何创建一个对象？**

1. 工厂模式  
2. 构造函数模式  
3. 原型模式  
4. 混合构造函数和原型模式  
5. 动态原型模式  
6. 寄生构造函数模式  
7. 稳妥构造函数模式  
  
程序的设计模式?工厂模式?发布订阅?  
1）设计模式并不是某种语言的某块代码，设计模式是一种思想，提供给在编码时候遇到的各种问题是可以采取的解决方案，更倾向于一种逻辑思维，而不是万能代码块。  
设计模式主要分三个类型:创建型、结构型和行为型。  
创建型模式：单例模式，抽象工厂模式，建造者模式，工厂模式与原型模式。  
结构型模式：适配器模式，桥接模式，装饰者模式，组合模式，外观模式，享元模式以及代理模式。  
行为型模式：模板方法模式，命令模式，迭代器模式，观察者模式，中介者模式，备忘录模式，解释器模式，状态模式，策略模式，职责链模式和访问者模式。  
  
2）与创建型模式类似，工厂模式创建对象（视为工厂里的产品）是无需指定创建对象的具体类。  
工厂模式定义一个用于创建对象的接口，这个接口由子类决定实例化哪一个类。该模式使一个类的实例化延迟到了子类。而子类可以重写接口方法以便创建的时候指定自己的对象类型。  
  
3）观察者模式又叫做发布订阅模式，它定义了一种一对多的关系，让多个观察者对象同时监听某一个主题对象，这个主题对象的状态发生改变时就会通知所有观察着对象。它是由两类对象组成，主题和观察者，主题负责发布事件，同时观察者通过订阅这些事件来观察该主体，发布者和订阅者是完全解耦的，彼此不知道对方的存在，两者仅仅共享一个自定义事件的名称。  
( 设计模式实在是太高深了，小伙伴门结合网上实例自行学习，我实在是无能为力啊 )

**82.commonjs?requirejs?AMD|CMD|UMD?**

1.CommonJS就是为JS的表现来制定规范，NodeJS是这种规范的实现，webpack 也是以CommonJS的形式来书写。因为js没有模块的功能，所以CommonJS应运而生。但它不能在浏览器中运行。 CommonJS定义的模块分为:{模块引用(require)} {模块定义(exports)} {模块标识(module)}   
  
2.RequireJS 是一个JavaScript模块加载器。 RequireJS有两个主要方法(method): define()和require()。这两个方法基本上拥有相同的定义(declaration) 并且它们都知道如何加载的依赖关系，然后执行一个回调函数(callback function)。与require()不同的是， define()用来存储代码作为一个已命名的模块。 因此define()的回调函数需要有一个返回值作为这个模块定义。这些类似被定义的模块叫作AMD (Asynchronous Module Definition，异步模块定义)。   
  
3.AMD 是 RequireJS 在推广过程中对模块定义的规范化产出 AMD异步加载模块。它的模块支持对象 函数 构造器 字符串 JSON等各种类型的模块。 适用AMD规范适用define方法定义模块。  
  
4.CMD是SeaJS 在推广过程中对模块定义的规范化产出  
AMD与CDM的区别：  
（1）对于于依赖的模块，AMD 是提前执行(好像现在也可以延迟执行了)，CMD 是延迟执行。  
（2）AMD 推崇依赖前置，CMD 推崇依赖就近。  
（3）AMD 推崇复用接口，CMD 推崇单用接口。  
（4）书写规范的差异。  
  
5.umd是AMD和CommonJS的糅合。  
AMD 浏览器第一的原则发展 异步加载模块。  
CommonJS模块以服务器第一原则发展，选择同步加载，它的模块无需包装(unwrapped modules)。这迫使人们又想出另一个更通用的模式UMD ( Universal Module Definition ), 希望解决跨平台的解决方案。UMD先判断是否支持Node.js的模块( exports )是否存在，存在则使用Node.js模块模式。

**83. js的几种继承方式？**

1.使用对象冒充实现继承   
2.采用call、Apply方法改变函数上下文实现继承   
3.原型链方式继承

**84. JavaScript原型，原型链 ? 有什么特点？**

在JavaScript中,一共有两种类型的值,原始值和对象值.每个对象都有一个内部属性[[prototype]],我们通常称之为原型.原型的值可以是一个对象,也可以是null.如果它的值是一个对象,则这个对象也一定有自己的原型.这样就形成了一条线性的链,我们称之为原型链.   
  
访问一个对象的原型可以使用ES5中的Object.getPrototypeOf方法,或者ES6中的\_\_proto\_\_属性. 原型链的作用是用来实现继承,比如我们新建一个数组,数组的方法就是从数组的原型上继承而来的。

**85. eval是做什么的？**

它的功能是把对应的字符串解析成JS代码并运行； 应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

**86. null，undefined 的区别？**

undefined表示变量声明但未初始化的值，null表示准备用来保存对象，还没有真正保存对象的值。从逻辑角度看，null表示一个空对象指针。

**87. JSON 的了解？**

JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小。

**88. js延迟加载的方式有哪些？**

defer和async、动态创建DOM方式（用得最多）、按需异步载入js

**89. ajax 是什么?**

异步javascript和XML，是指一种创建交互式网页应用的网页开发技术。通过后台与服务器进行少量数据交换，AJAX可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。

**90. 同步和异步的区别?**

javascript同步表示sync，指：代码依次执行 javascript异步表示async，指：代码执行不按顺序，‘跳过’执行，待其他某些代码执行完后再来执行，成为异步。

**91. 如何解决跨域问题?**

Jsonp、iframe、window.name、window.postMessage、服务器上设置代理页面

**92. 异步加载的方式有哪些？**

(1) defer，只支持IE  
(2) async：true  
(3) 创建script，插入到DOM中，加载完毕后callBack

**93. jQuery与jQuery UI 有啥区别？**

jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。  
jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等。

**94. 你有哪些性能优化的方法？**

(1) 减少http请求次数：CSS Sprites, JS、CSS源码压缩、图片大小控制合适；网页Gzip， CDN托管，data缓存 ，图片服务器。  
  
(2) 前端模板 JS+数据，减少由于HTML标签导致的带宽浪费，前端用变量保存AJAX请求结果，每次操作本地变量，不用请求，减少请求次数   
  
(3) 用innerHTML代替DOM操作，减少DOM操作次数，优化javascript性能。  
  
(4) 当需要设置的样式很多时设置className而不是直接操作style。  
  
(5) 少用全局变量、缓存DOM节点查找的结果。减少IO读取操作。  
  
(6) 避免使用CSS Expression（css表达式)又称Dynamic properties(动态属性)。  
  
(7) 图片预加载，将样式表放在顶部，将脚本放在底部 加上时间戳。  
  
(8) 避免在页面的主体布局中使用table，table要等其中的内容完全下载之后才会显示出来，显示比div+css布局慢。

**95. 一个页面从输入 URL 到页面加载显示完成，这个过程中都发生了什么？（流程说的越详细越好）**

查找浏览器缓存   
DNS解析、查找该域名对应的IP地址、重定向（301）、发出第二个GET请求  
进行HTTP协议会话  
客户端发送报头(请求报头)  
服务器回馈报头(响应报头)  
html文档开始下载  
文档树建立，根据标记请求所需指定MIME类型的文件  
文件显示  
  
浏览器这边做的工作大致分为以下几步：  
加载：根据请求的URL进行域名解析，向服务器发起请求，接收文件（HTML、JS、CSS、图象等）。  
解析：对加载到的资源（HTML、JS、CSS等）进行语法解析，建议相应的内部数据结构（比如HTML的DOM树，JS的（对象）属性表，CSS的样式规则等等）

**96. ajax的缺点**

1、ajax不支持浏览器back按钮。  
2、安全问题 AJAX暴露了与服务器交互的细节。   
3、对搜索引擎的支持比较弱。  
4、破坏了程序的异常机制。  
5、不容易调试

来源：<https://www.qdfuns.com/notes/40816/795e40bec4bcf47ca37f983ce62f1ea4.html>