## 1、javascript的typeof返回哪些数据类型

string number ~~array~~ object function Boolean undefiined

数组（Array）、日期对象(Date)、正则(RegExp)、 Math =>object

**考点：使用typeof检测数据类型’**

**扩展：如何检测数组类型？**

Array.isArray(); 浏览器兼容性：IE9+

toString.call([]);//”[object Array]”

[] instanceof Array

var arr=[];

arr.constructor;//Array

## 2、例举3种强制类型转换和2种隐式类型转换?

强制类型转换：自己通过函数来进行数据类型转换

举例：（parseInt,parseFloat,Number()）

隐式类型转换：JS引擎自动帮我们转换的

举例：==、 console.log()、 alert() 、if() 、+-\*/

**扩展：通过==比较两边的值是否相等的结果？**

1==’1’

null==undefined

## 3、split() join() 的区别

split()将字符串按照指定的字符分割成一个数组，并返回

join()将数组用指定的字符连接成一个字符串，并返回

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

栈方法：

push()尾部添加，返回 数组长度

pop()尾部删除，返回 被删除的元素

队列方法：

unshift()头部添加 ，返回 数组长度

shift()头部删除，返回被删除的元素

## 5、事件绑定和普通事件有什么区别

普通事件：给html元素添加一个特定的属性（比如：onclick）

事件绑定：js代码中通过标记(id tag class)获取元素，给元素添加特定的方法(比如onclick)

传统事件绑定和符合W3C标准的事件绑定有什么区别？

div1.onclick=function(){};

<button onmouseover=””></button>

1、如果说给同一个元素绑定了两次或者多次相同类型的事件，那么后面的绑定会覆盖前面的绑定

2、不支持DOM事件流 事件捕获阶段🡺目标元素阶段=>事件冒泡阶段

addEventListener

1. 如果说给同一个元素绑定了两次或者多次相同类型的事件，所以的绑定将会依次触发
2. 支持DOM事件流的
3. 进行事件绑定传参不需要on前端

addEventListener(“click”,function(){},false);//此时的事件就是在事件冒泡阶段执行

ie9开始，ie11 edge：addEventListener

ie9以前：attachEvent/detachEvent

1. 进行事件类型传参需要带上on前缀
2. 这种方式只支持事件冒泡，不支持事件捕获

事件绑定是指把事件注册到具体的元素之上，普通事件指的是可以用来注册的事件

## 6、IE和DOM事件流的区别

**IE9以前**：[attachEvent(“onclick”)、detachEvent](http://www.baidu.com/link?url=fVtfIQaNBwj3KUO_767M8Y2AoHQa2q1pSY0iwTzkigv1ue0E8r5tBI4bezzlCu01KrbygrzNAJ-jw5H8FTc-1a" \t "_blank)(“onclick”)

**IE9开始**跟DOM事件流是一样的，都是addEventListener

**比较attachEvent和addEventListener：**

1、attachEvent只支持事件冒泡 addEventListener既支持事件冒泡，也支持事件捕获  
2、参数：attachEvent事件类型需要on前缀 addEventListener事件类型不需要on前缀

3、如果使用attachEvent对一个元素的目标阶段绑定了多次事件，那么会按照绑定顺序的相反顺序进行触发

如果使用addEventListener对一个元素的目标阶段绑定了多次事件，那么会按照绑定顺序进行触发

## 7、IE和标准下有哪些兼容性的写法

a、获取事件对象：var ev = ev || window.event

var ev=ev?ev:window.evnet;

srcElement：IE9之前的浏览器用来获取事件目标元素

target：IE9+、ff、chrome用来获取事件的目标元素

b、获取事件目标元素：var target = ev.srcElement||ev.target

c、innerText

## 8、call和apply的区别

**考点：call和apply的用法**

call和apply相同点：改变函数中this的指向

不同点：函数参数的传递形式

call将函数参数依次传入

apply将函数参数用一个数组的形式传入

**无参数调用：**

**function** fn(){  
 alert(**this**.name);  
}  
**var** p1={name:1};  
fn.call(p1);  
fn.apply(p1);

**有参数调用：**

**function** fn2(*name*,*age*){  
 **this**.name=*name*;  
 **this**.age=*age*;  
}  
**var** p1={};  
fn2.call(p1,"张三",20);  
fn2.apply(p1,["张三",20]);

## 9、如何实现js中的继承

**考点：继承的多种方式（参考 高设6.3）**

**1、原型继承的第一种方式：**

**function** Cat(*name*){  
 **this**.name=*name*;  
}  
//原型继承  
Cat.prototype.say=**function**(){  
 alert("你好，我是一只猫，我叫："+**this**.name);  
}

**2、原型继承第二种方式：**

**function** Cat(*name*) {  
 **this**.name = *name*;  
}  
**function** Animal() {}  
Animal.prototype.run = **function** () {  
 alert("动物跑");  
};  
Cat.prototype = **new** Animal();

Cat.prototype.constructor=Cat;

**3、借用构造函数**

**function** Cat(*name*,*age*) {  
 Animal.call(**this**,*name*,*age*);  
}  
**function** Animal(*name*,*age*) {  
 **this**.name = *name*;  
 **this**.age=*age*;  
}

**4、经典继承**

**function create**(*obj*) {

if(Object.create) {

return Object.create(obj);

} else {

**function** F(){};

F.prototype = obj;

return new F();

}

}

## 10、JavaScript this、闭包、作用域

this：指向调用上下文

作用域：定义一个函数就开辟了一个局部作用域，整个js执行环境有一个全局作用域

闭包：一个函数可以访问其他函数中的变量（闭包是一个受保护的变量空间）

**var** f=(**function** fn(){  
 **var** name=1;  
 **return function**(){  
 name++;  
 console.log(name);  
 }  
})();

## 11、事件委托是什么

利用事件冒泡的原理，将事件绑定在父容器中，让父容器代为触发

## 12、闭包是什么，有什么特性，对页面有什么影响

闭包就是能够读取其他函数内部变量的函数。

闭包的缺点：

1 更多的内存消耗

2 性能问题（跨作用域访问）

3滥用闭包函数会造成内存泄露，因为闭包中引用到的包裹函数中定义的变量都永远不会被释放，所以我们应该在必要的时候，及时释放这个闭包函数

闭包是一种特殊的对象。它由两部分构成：函数，以及创建该函数的环境。

可以把闭包简单理解成 “定义在一个函数内部的函数”，闭包就是将函数内部和函数外部连接起来的一座桥梁。闭包有如下特性：

a. JavaScript允许你使用在当前函数以外定义的变量

b. 即使外部函数已经返回，当前函数仍然可以引用在外部函数所定义的变量

c. 闭包可以更新外部变量的值

d. 用闭包模拟私有方法

由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包，否则会造成网页的性能问题

## 13、如何阻止事件冒泡和默认事件

**阻止事件冒泡：**

IE9+ FF Chrome：e. stopPropagation();

window.event.canceBubble=true;//ie9之前

**默认行为**：html标签所具有的默认行为，比如：

a、点击a标签，就会默认跳转到指定的页面

b、点击submit按钮，就会自动提交表单

适用场景：

1、异步操作

2、提交表单之前对表单进行一些基本的验证，比如邮箱是否合法，用户名是不是满足指定的格式

为了不让a点击之后跳转，我们就要给他的点击事件进行阻止

3、文本框获得焦点

**阻止默认行为：**

IE9之前：window.event.returnValue=false;

IE9+ FF Chrome： e.preventDefault();

## 14、添加 删除 替换 插入到某个节点的方法

obj.appendChild()

[insertBefore](https://developer.mozilla.org/en-US/docs/Web/API/Node/insertBefore)

obj.insertBefore(newElement, referenceElement )

参数1：被插入的元素

参数2：目标元素

obj.replaceChild(newChild, oldChild)//替换

obj.removeChild(child)//删除

## 15、javascript的本地对象，内置对象和宿主对象

本地对象为Array RegExp等可以new实例化

内置对象为global Math 等不可以实例化的

宿主为浏览器自带的document,window 等

## 16、document load 和document ready的区别

document.onload 是在结构和样式加载完才执行js

document.ready原生中没有这个方法，jquery中有 $().ready(function)

DOMCententLoaded事件：页面的文档结构（DOM数）加载完之后就会触发

window.onload：不仅仅要在结构和样式加载完，还要执行完所有的外部样式、图片这些资源文件，全部加载完才会触发window.onload事件

## 17、”==”和“===”的不同

==：会自动转换类型

===：先判断左右两边的数据类型，如果数据类型不一致，直接返回false

之后才会进行两边值的判断

1==”1”

null==undefined;//==true

## 18、javascript的同源策略

一段脚本只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

http,ftp:协议

**关键词解释：**

主机名：localhost、www.baidu.com

协议：http https ftp

端口：一个网站对应着一个端口， http协议的默认端口：80

https协议的默认端口是8083

同源策略带来的麻烦：ajax在不同域名下的请求无法实现，

如果说想要请求其他来源的js文件，或者json数据，那么可以通过jsonp来解决

[跨域解决方式](http://www.cnblogs.com/2050/p/3191744.html)一

[跨域解决方式二](http://www.cnblogs.com/rainman/archive/2011/02/20/1959325.html)

## 19、编写一个数组去重的方法

var arr=[1,1,3,4,2,4,7];

=>[1,3,4,2,7]

**思路1：**

1. 先创建一个空数组，用来保存最终的结果
2. 循环数组元素，判断元素在新数组中是否有相同元素，如果没有就插入到新数组中
3. 返回新数组

**代码1：**

**function** unique(*arr*){  
 **var** result=[];  
 **for** (**var** i = 0, len = *arr*.length; i < len; i++) {  
 **var** arri = *arr*[i];  
 **if**(result.indexOf(arri)<0){  
 result.push(arri);  
 }  
   
 }  
 **return** result;  
}

## 20、JavaScript是一门什么样的语言，它有哪些特点？

弱类型、脚本语言、面向对象、

没有标准答案。

运行环境：JS引擎（v8(Chrome)/SpiderMonkey(FireFox)/JavaScriptCore(Safari)

/Chakra(IE)）

**语言特性：**

1、面向对象：原型继承、构造函数、原型链

2、动态语言：弱类型语言

//动态语言的特性  
var *num*=10;//num是一个数字类型  
*num*="jim";//此时num又变成一个字符串类型

//我们把一个变量用来保存不同数据类型的语言称之为一个动态语言  
//静态语言：c# java c c++ OC

//int a;  
//静态语言在声明一个变量就已经确定了这个变量的数据类型，  
// 而且在任何时候都不可以改变他的数据类型

## 21、JavaScript的数据类型都有什么？

基本数据类型：number、string、boolean、undefined、null

复杂数据类型：Object(Array,Date,RegExp,Function)

## 22、已知ID的Input输入框，希望获取这个输入框的输入值，怎么做？(不使用第三方框架)

|  |  |
| --- | --- |
|  | document.getElementById(“ID”).value |

## 23、希望获取到页面中所有的checkbox怎么做？(不使用第三方框架)

**var** domList = document.getElementsByTagName("input");  
**var** ckList = [];//返回的所有的checkbox  
**var** len = domList.length;  
**for** (**var** i = 0; i < len; i++) {  
 **if** (domList[i].type == "checkbox") {  
 ckList.push(domList[len]);  
 }  
}

## 24、设置一个已知ID的DIV的html内容为xxxx，字体颜色设置为黑色(不使用第三方框架)

|  |  |
| --- | --- |
|  | var dom = document.getElementById(“ID”);  dom.innerHTML = “xxxx”  dom.style.color = “#000”;//”black” |

## 25、当一个DOM节点被点击时候，我们希望能够执行一个函数，应该怎么做？

HTML事件绑定：<div onclick=”test()”></div>

DOM0事件绑定：xxx.onclick = test

DOM2事件绑定：addEventListener(div1, ‘click’, test)

**扩展：Javascript的事件流模型都有什么？**

**事件流：**

从最不确定的元素**(最外层容器)**到目标元素，再由目标元素到最不确定的元素**(最外层容器)；**

也就是说先经历事件捕获，到目标元素，再经历事件冒泡

“事件冒泡”：事件开始由最具体的元素接受，然后逐级向上传播

“事件捕捉”：事件由最不具体的节点先接收，然后逐级向下，一直到最具体的

## 26、看下列代码输出为何？解释原因。

|  |  |
| --- | --- |
|  | var a;  alert(typeof a); // “undefined”  //alert(b); // 报错  b=10;  alert(typeof b);//”number” |

**答案：undefined、number**

**undefined**产生情况：

1. 一个变量定义了却没有被赋值
2. 想要获取一个对象上不存在的属性或者方法:
3. 一个数组中没有被赋值的元素
4. 调用函数，参数未传

扩展：not defined语法错误

## 27、看下列代码,输出什么？解释原因。

|  |  |
| --- | --- |
|  | var a = null;  alert(typeof a); //object |

## 28、看下列代码,输出什么？解释原因。

|  |  |
| --- | --- |
|  | var undefined;//此时undefined这个变量的值是undefined  undefined **==** null; // true  1 == true;   // true  此时会把布尔类型的值转换为数字类型 true=1 false=0  2 == true;   // false  0 == false;  // true  **0 == ''**;     // true  **NaN == NaN**;  // false isNaN  [] == false; // true 解释：会把[]和false都通过Number()转换为数字类型  [] == ![];   // true 解释：![]：false |
|  |  |

一个是number一个是string时，会尝试将string转换为number

一个是number一个是boolean，将boolean转换为number，结果：true：1 false：0

一个是object 另一个是string或number，将Object转换成number或string

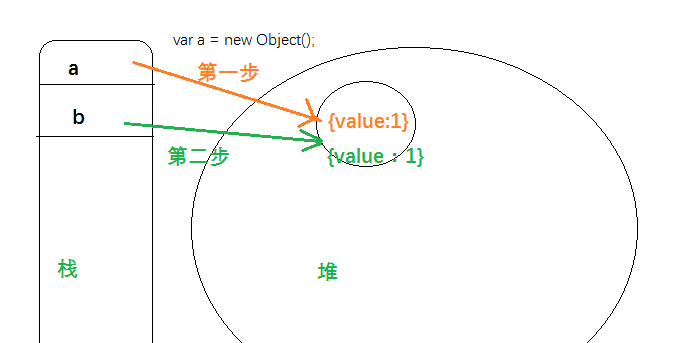
所以，对于0、空字符串的判断，建议使用 “===” 。“===”会先判断两边的值类型，类型不匹配时为false。

### 28.2、看下列代码会有什么样的输出？

|  |  |
| --- | --- |
|  | var foo = "11"+2-"1";  console.log(foo);//112-1=111  console.log(typeof foo);//”number”  **考点：**  1、数字和字符串都可以用加法运算符，数字和字符串相加，结果就是一个字符串  2、但是减法运算符只能用于两个数字之间，想要执行减法运算，必须把两边数字都变成数字类型的  答案：”111”、”number” |

## 29、看代码给答案。

|  |  |
| --- | --- |
|  | var a = new Object();  a.value = 1;  b = a; //b.value=1  b.value = 2;//b.value=2;a.value=2，因为a和b指向同一块引用类型的值  alert(a.value); |

答案：2（考察引用数据类型细节）

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

**考点：数组的join方法的使用**

答案：alert(stringArray.join(“ ”))

### 30.2、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("-");//[get,element,by,id]      for(var i=1;i<arr.length;i++){          arr[i]=arr[i].[0].toUpperCase()+arr[i].substring(1);//Element      }      msg=arr.join("");//msg=” getElementById”      return msg;  } |

## 31、var numberArray = [3,6,2,4,1,5];

### 1) 实现对该数组的倒排，输出[5,1,4,2,6,3]

**function** reverseArray(*arr*){  
 **var** result=[];  
 //方法1：  
 /\*for (var i = arr.length - 1; i >= 0; i--) {  
 result.push(arr[i]);  
 }\*/  
 //方法2：  
 **for** (**var** i = 0, len = *arr*.length; i < len; i++) {  
 result.unshift(*arr*[i]);  
 }  
 **return** result;  
}

### 2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

[冒泡排序过程演示](http://blog.csdn.net/u012545279/article/details/17412219)

**function** sortDesc(*arr*) {  
 **for** (**var** i = 0, len = *arr*.length; i < len; i++) {  
 **for** (**var** j = i + 1, len2 = *arr*.length; j < len2; j++) {  
 //>就是降序 <就是升序  
 **if** (*arr*[j] > *arr*[i]) {  
 **var** temp = *arr*[j];  
 *arr*[j] = *arr*[i];  
 *arr*[i] = temp;  
 }  
 }  
 }  
 **return** *arr*;  
}

## 32、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

**考点：日期对象Date相关API的使用**

|  |  |
| --- | --- |
|  | 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); |

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

**考点：正则表达式、字符串的replace方法的使用**

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

## 34、为了保证页面输出安全，我们经常需要对一些特殊的字符进行转义，请写一个函数escapeHtml，将<, >, &, “进行转义

|  |  |
| --- | --- |
|  | function escapeHtml(str) {  //[<>”&]:中括号中字符只要其中的一个出现就代表满足条件  //给replace第二个参数传递一个回调函数，回调函数中参数就是匹配结果，如果匹配不到就是null  return str.replace(/[<>”&]/g, function(match) {      switch (match) {        case “<”:           return “&lt;”;        case “>”:          return “&gt;”;        case “&”:          return “&amp;”;        case “\””://双引号包裹一个单引号：“’” 单引号包裹一个双引号 ‘””’           return “&quot;”;       }    });  } |

## 35、foo = foo||bar ，这行代码是什么意思？为什么要这样写？

这种写法称之为短路表达式

相当于：

**var** foo;  
**if**(foo){  
 foo=foo;  
}**else**{  
 foo=bar;  
}

答案：常用于函数参数的空判断

短路表达式：作为”&&”和”||”操作符的操作数表达式，这些表达式在进行求值时，只要最终的结果已经可以确定是真或假，求值过程便告终止，这称之为短路求值。

**考点：if条件的真假判定**

记住以下是false的情况：空字符串、false、undefined、null、0

## 36、看下列代码，将会输出什么?

**考点：1、变量作用域 2、变量声明提升**

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

答案：输出undefined 和 2。

## 37、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。

|  |  |
| --- | --- |
|  | var iArray = [];  function getRandom(istart, iend){          var iChoice = iend - istart +1;          return Math.floor(**Math.random()** \* iChoice+ istart);  }  Math.random()就是获取0-1之间的随机数（永远获取不到1）  for(var i=0; i<10; i++){  var result= getRandom(10,100);          iArray.push(result);  }  **iArray.sort();** |
|  |  |

## 38、把两个数组合并，并删除第二个元素。

**考点：1、数组的concat、splice用法**

splice() 方法删除数组的元素，或者向数组中添加元素，然后返回被删除的项目。

参数1：从何处开始删除数组的元素（使用负数则从数组的末尾开始）

参数2：要删除的元素个数（如果是0，就代表不删除）

参数3，4，5。。。：要添加的元素

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

## 39、怎样添加、移除、移动、复制、创建和查找节点（使用原生JS实现）

1）创建新节点

createDocumentFragment()    //创建一个DOM文档片段

createElement()   //创建一个具体的元素

createTextNode()   //创建一个文本节点

2）添加、移除、替换、插入

appendChild()      //追加

removeChild()      //移除

replaceChild()      //替换

insertBefore()      //插入

3）查找

getElementsByTagName()    //通过标签名称

getElementsByName()     //通过元素的Name属性的值

getElementsByTagName() // 通过类名查找

getElementById()        //通过元素Id，唯一性

## 40、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}。

答案：

**function** serlize(*url*){  
 **var** result={};  
 //1、寻找？后面的字符串  
 *url*=*url*.substr(*url*.indexOf("?")+1);  
 //2、将字符串用&分隔  
 **var** args=*url*.split("&");//[“a=1”,”b=2”]  
 **for** (**var** i = 0, len = args.length; i < len; i++) {  
 **var** arg = args[i];

var item = arg.split('=');  
 //3、对象的键=值  
 result[item[0]]= item[1];  
 }  
 **return** result;  
}  
serlize('http://item.taobao.com/item.htm?a=1&b=2&c=&d=xxx&e');

## 41、正则表达式构造函数var reg=new RegExp(“xxx”)与正则表达字面量var reg=//有什么不同？匹配邮箱的正则表达式？

[RegExp](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/RegExp)

答案：当使用RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。

邮箱的正则匹配：

|  |  |
| --- | --- |
|  | var regMail = /^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+((.[a-zA-Z0-9\_-]{2,3}){1,2})$/; |

### 41.2、.看下面代码，给出输出结果。

for(var i=1;i<=3;i++){

  setTimeout(function(){

      console.log(i);

  },0);

}

答案：4 4 4。

**考点：setTimeout的执行原理——Javascript事件处理器在线程空闲之前不会运行。**

**[JavaScript运行机制：事件循环](http://www.ruanyifeng.com/blog/2014/10/event-loop.html)**

**追问，**如何让上述代码输出1 2 3？

**代码1：用立即执行函数**

**for**(**var** i=1;i<=3;i++){  
 setTimeout((**function**(num){  
 **return function**(){  
 console.log(*num*);  
 }  
  
 })(i),0);  
}

**代码2：使用闭包**

**for**(**var** i=1;i<=3;i++){  
 setTimeout((**function**(){  
 **var** j=i;  
 **return function**(){  
 console.log(j);  
 };  
 })(),0);  
}

## 42、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(IE9以下浏览器不支持)：

**考点：1、原型扩展 2、正则表达式 3、字符串的replace方法**

**if**(**typeof** String.prototype.trim !="function"){  
 String.prototype.trim=**function**(){  
 **return this**.replace(/^\s+|\s+$/g,"");  
 }  
}  
**var** str=" hello ";

## 43、Javascript中callee和caller的作用？

arguments.callee：获得当前函数的引用

fn.caller是返回一个对函数的引用，该函数调用了当前函数；（被废弃）

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。

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

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

//正确答案：BC

## 45、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 //拼写

B、var a=‘123’; b=NaN

C、var a =undefined , b =NaN

D、var a=NaN , b='undefined'

解析：A、拼写错误 B、结果是字符串 D、结果是字符串

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

**考点：++运算符的使用 21+4+10=35**

A、34 B、35 C、36 D、37

//var a=10; b=20; c=4;

//++b+c+a++  
//21+4+10=35;

var a = 1;

a++ + a++ = ？

a = 1;

a++ + ++a = ?

a = 1;

++a + a++ = ?

a = 1;

++a + ++a = ?

## 47、下面的JavaScript语句中，（ D ）实现检索当前页面中的表单元素中的所有文本框，并将它们全部清空

A. for(vari=0;i< form1.elements.length;i++) {

if(form1.elements.type==”text”)

form1.elements.value=”";

}

B. for(vari=0;i<document.forms.length;i++) {

if(forms[0].elements.type==”text”)

forms[0].elements.value=”";

}

C. if(document.form.elements.type==”text”)

form.elements.value=”";

D. for(vari=0;i<document.forms.length; i++){

var form= document.forms[i];

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

if(form [j].type==”text”)

form.elements [j].value=”";

}

}

## 48、要将页面的状态栏中显示“已经选中该文本框”，下列JavaScript语句正确的是（ A ）

A. window.status=”已经选中该文本框”

B. document.status=”已经选中该文本框”

C. window.screen=”已经选中该文本框”

D. document.screen=”已经选中该文本框”

## 49、以下哪条语句会产生运行错误：（A、D）

A.var obj = ();B.var obj = [];C.var obj = {};D.var obj = //;

## 50、以下哪个单词不属于javascript保留字：（B）

A.withB.parentC.classD.void

## 51、请选择结果为真的表达式：（C）

**考点：1、instanceof：用于检测某个对象是不是某个构造函数的实例**

**2、==、===用于数据类型的判断**

A.null instanceof ObjectB.null === undefinedC.null == undefinedD.NaN == NaN

## 52、Javascript中, 如果已知HTML页面中的某标签对象的id=”username”，用\_\_\_\_document.getElementById(‘username’)\_\_\_ \_方法获得该标签对象。

## 53、typeof运算符返回值中有一个跟javascript数据类型不一致，它是\_\_\_\_\_\_\_\_”function”\_\_\_\_\_\_\_\_。

**考点：type运算符**

## 54、定义了一个变量，但没有为该变量赋值，如果alert该变量，javascript弹出的对话框中显示\_\_\_undefined\_\_\_\_\_\_ 。

## 55、分析代码，得出正确的结果。

var a=10, b=20 , c=30;

++a;

a++;

e=++a+(++b)+(c++)+a++;//13+21+30+13

alert(e);

弹出提示对话框：77

var *a*=10, *b*=20 , *c*=30;  
++*a*;//a=11  
*a*++;//a=11  
*e*=++*a*+(++*b*)+(*c*++)+*a*++;  
//a=12 13+21+30+13=77  
alert(e);

## 56、写出函数DateDemo的返回结果，系统时间假定为今天

function DateDemo(){

var d, s="今天日期是：";

d = new Date();

s += d.getMonth() + "/";

s += d.getDate() + "/";

s += d.**getFullYear**();

return s;}

**考点：Date对象的api使用，**

**注意点：getMonth()打印的是比当前月份小1的数字**

结果：今天日期是：

## 57、写出程序运行的结果？

for(var i=0, j=0; i<10, j<6; i++, j++){  
 //循环结束条件：j=5 i=5  
 k = i + j;//k=10  
}  
//结果：10

**[for循环](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Statements/for)**

**解析：最终的结果其实就是程序运行的最后一个表达式的值**

## 58、阅读以下代码，请分析出结果：

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

arr[4]='z'; //arr=[1,3,5,undefined,’z’]

arr2 = arr.**reverse**(); //arr2=[’z’,undefined,5,3,1];

//arr=[’z’,undefined,5,3,1]

arr3 = arr.concat(arr2);

alert(arr3);

**考点：reverse 方法颠倒数组中元素的位置，并返回该数组的引用。**

答案：弹出：z,,5,3,1,z,,5,3,1

## 59、补充按钮事件的函数，确认用户是否退出当前页面，确认之后关闭窗口； <html>

**考点：confirm的用法**

<head>

<script type=”text/javascript” >

function closeWin(){

//在此处添加代码

**if**(confirm("确定要退出吗？")){  
 window.close();  
}

}

</script>

</head>

<body>

<input type=”button”value=”关闭窗口”onclick=”closeWin()”/>

</body>

</html>

## 60、写出简单描述html标签（不带属性的开始标签和结束标签）的正则表达式，并将以下字符串中的html标签去除掉

**考点：正则表达式以及replace()的使用**

var str = “<div>这里是div<p>里面的段落</p></div>”;

//

<scripttype=”text/javascript”>

var reg = /<\/?\w+\/?>/gi;//img br hr

|  |  |
| --- | --- |
| *x*? | 匹配问号前面的内容出现0 或 1 次。 |

var str = “<div>这里是div<p>里面的段落</p></div>”;

alert(str.replace(reg,”"));

</script>

## 61、完成foo()函数的内容，要求能够弹出对话框提示当前选中的是第几个单选框。

<html>

<head>

<metahttp-equiv=”Content-Type” content=”text/html;charset=utf-8″ />

</head>

<body>

<script type=”text/javascript” >

function foo() {

//在此处添加代码

var rdo =**document.form1.radioGroup;//表单下面的所有的单选框**

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

if(rdo[i].checked){

alert(“您选择的是第”+(i+1)+”个单选框”);

}

}

}

document.getElementsByTagName(“input”)[4].onclick = function(e) {

e.preventDefault();

foo();

};

</script>

<body>

<form name=”form1″ >

<input type=”radio” name=”radioGroup” />

<input type=”radio” name=”radioGroup”/>

<input type=”radio” name=”radioGroup”/>

<input type=”radio” name=”radioGroup”/>

<input type=”submit”/>

</form>

</body>

</html>

## 62、完成函数showImg()，要求能够动态根据下拉列表的选项变化，更新图片的显示

**考点：1、下拉框切换：onchange事件 2、通过value获取下拉框的值**

<script type=”text/javascript” >

function showImg (oSel) {

//在此处添加代码

var str = oSel.value;

document.getElementById(“pic”).src= str+”.jpg”;

}

</script>

<img id=”pic”src=”img1.jpg”width=”200″ height=”200″ />

<br />

<select id=”sel”onchange=”showImg(this)”>

<option value=”img1“>城市生活</option>

<option value=”img2“>都市早报</option>

<option value=”img3“>青山绿水</option>

</select>

## 63、截取字符串abcdefg的efg

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

## 64、列举浏览器对象模型BOM里常用的至少4个对象，并列举window对象的常用方法至少5个

对象：window document location screen history navigator

方法：alert() confirm() prompt() open() close()

## 65、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.getElementsByName 根据元素name查找元素

document.getElementsByTagName 根据指定的元素名查找元素

## 66、希望获取到页面中所有的checkbox怎么做？(不使用第三方框架)

var domList = document.getElementsByTagName(‘input’)

var checkBoxList = [];

var len = domList.length;　　//缓存到局部变量

while (len--) {

　　if (domList[len].type == ‘checkbox’) {

　　checkBoxList.push(domList[len]);

　　}

}

## 68、javascript中有哪几种数据类型，分别写出中文和英文。

string boolean number null undefined object

字符串 布尔 数值 空值 未定义 对象

## 69、javascript中==和===的区别是什么？举例说明。

==会自动进行类型转换，比如：1==”1”，数字和字符串比较，会将非数字转变为数字

===不会，比如：1===”1”由于两边数据类型不一致，所以结果必然是false

## 70、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;

}

匿名函数：没有函数名称，无法通过函数名称调用

function(){}:只能自己执行自己

**第三种（Function构造函数方式）：**

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

## 71、Javascript如何实现继承？

原型链继承，借用构造函数继承，组合继承，寄生式继承，寄生组合继承

## 72、Javascript创建对象的几种方式？

构造函数方式，原型模式，混合构造函数原型模式，工厂方式，动态原型方式

**混合构造函数+原型模式：**

**function** Robot(*name*){  
 **this**.name=*name*;  
}  
Robot.prototype.say=**function**(){  
 alert("大家好，我叫"+**this**.name);  
};  
**var** alphaGo=**new** Robot("阿尔法狗");  
alphaGo.say();

**工厂方式：**

**function** create(*name*,*age*){  
 **var** o={};  
 o.name=*name*;  
 o.age=*age*;

**return** o;  
}  
**var** p1=create("张三",20);

**动态原型方式：**

**function** Person(*name*,*work*){  
 **this**.name=*name*;  
 **if**(*work*){  
 Person.prototype.doWork=**function**(){  
 alert("我正在从事"+*work*+"的工作");  
 }  
 }  
}  
**var** p1=**new** Person("姚明");  
**var** p2=**new** Person("喵喵","程序猿鼓励师");

## 73、把 Script 标签 放在页面的最底部的body封闭之前 和封闭之后有什么区别？浏览器会如何解析它们？

如果说放在body的封闭之前，将会阻塞其他资源的加载

如果放在body封闭之后，不会影响body内元素的加载

## 74、iframe的优缺点？

优点：

1. 解决加载缓慢的第三方内容如图标和广告等的加载问题

2、并行加载脚本

缺点：

1、iframe会阻塞主页面的onload事件

2、即时内容为空，加载也需要时间，因为需要http请求

3、不便于SEO

4、内外网页维护麻烦

## 75、请你谈谈Cookie的弊端？

什么是Cookie？Cookie是指某些网站为了辨别用户身份或进行session跟踪而储存在用户本地浏览器终端上的数据。一般来说，Cookie通过HTTP Headers从服务器端返回到浏览器上。首先，服务器端在响应中利用Set-Cookie header来创建一个Cookie ，然后，浏览器在它的请求中通过Cookie header包含这个已经创建的Cookie，并且把它返回至服务器，从而完成浏览器的验证。  
  
**Cookie的保存形式**  
IE 浏览器将站点的 Cookie 保存在文件名格式为 @.txt 的文件中，其中 是您的帐户名。例如，如果您的名称为 user，您访问的站点为 www.codetc.com，那么该站点的 Cookie 将保存在名为 user@codetc.com.txt 的文件中。（该文件名可能包含一个顺序的编号，如 user@codetc.com [1].txt。） Cookie 文本文件是与用户相关的，所以会按照帐户分别保存。  
  
**Cookie的限制**  
一个 Cookie 大约占用 50 个字符的基本空间开销（用于保存有效期信息等），再加上其中保存的值的长度，其总和接近 4K 的限制。大多数浏览器只允许每个站点保存 20 个 Cookie。  
  
**为什么选择把信息保存到cookie中**  
由于session在使用过程中会造成极大的网络负担，随之带来的就是性能问题，所认我们可以把session以Cookie的形式保存在客户端。当然有时候也是为了完成某些特定的功能而使用cookie，比如[实现记住密码和自动登录](http://www.codetc.com/article-87-1.html)。  
  
**优点**：是极高的扩展性和可用性  
通过良好的编程，控制保存在cookie中的session对象的大小。  
通过加密和安全传输技术（SSL），减少cookie被破解的可能性。  
只在cookie中存放不敏感数据，即使被盗也不会有重大损失。  
控制cookie的生命期，使之不会永远有效。偷盗者很可能拿到一个过期的cookie。  
  
**缺点**：数量限制和安全问题  
Cookie是有数量和长度限制的。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉。  
安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。  
有些状态不可能保存在客户端。例如，为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

缺点：

1.`Cookie`数量和长度的限制。每个domain最多只能有20条cookie，每个cookie长度不能超过4KB，否则会被截掉。

2.安全性问题。如果cookie被人拦截了，那人就可以取得所有的session信息。即使加密也与事无补，因为拦截者并不需要知道cookie的意义，他只要原样转发cookie就可以达到目的了。

3.有些状态不可能保存在客户端。例如，

一、为了防止重复提交表单，我们需要在服务器端保存一个计数器。如果我们把这个计数器保存在客户端，那么它起不到任何作用。

二、比如用户密码输入错误超过3次，我们应该在后端数据库中保存错误数据

## 76、DOM操作——怎样添加、移除、移动、复制、创建和查找节点。

1. 创建新节点

createDocumentFragment() // 创建一个DOM片段

createElement() // 创建一个具体的元素

createTextNode() // 创建一个文本节点

2. 添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore() // 在已有的子节点前插入一个新的子节点

3. 查找

getElementsByTagName() // 通过标签名称

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)

getElementById() // 通过元素Id，唯一性

## 77、js延迟加载的方式有哪些？

**1. defer和async**

defer：当文档加载完成之后执行

async：异步执行脚本

**2. 动态创建DOM方式**

创建script，插入到DOM中，加载完毕后callBack）

## 78、documen.write和 innerHTML 的区别？

document.write 指定位置输出

dom.innerHTML 可以重绘指定元素的内容

[document.write和innerHTML的区别](http://stackoverflow.com/questions/10085109/when-should-one-use-innerhtml-and-when-document-write-in-javascript)

## 79、哪些操作会造成内存泄漏？

内存泄漏指任何对象在您不再拥有或需要它之后仍然存在。

垃圾回收器定期扫描对象，并计算引用了每个对象的其他对象的数量。如果一个对象的引用数量为 0（没有其他对象引用过该对象），或对该对象的惟一引用是循环的，那么该对象的内存即可回收。

1. 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。

2. 闭包

3. 控制台日志

[内存泄漏的几种情况](http://www.cnblogs.com/sprying/archive/2013/05/31/3109517.html)

## 81、split() join() 的区别

答：前者是切割成数组的形式，后者是将数组转换成字符串

## 82、数组方法pop() push() unshift() shift()各表示什么意思？

答：push()尾部添加、pop()尾部删除、Unshift()头部添加、shift()头部删除

## 83、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;// json[str[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');

## 84、javascript的typeof返回哪些数据类型

Object number function boolean underfind

## 85、例举3种强制类型转换和2种隐式类型转换?

强制（parseInt,parseFloat,number）

隐式（== – ===）

## 86、split() join() 的区别

前者是切割成数组的形式，后者是将数组转换成字符串

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

push()尾部添加 shift() 尾部删除

unshift() 头部添加 shift() 头部删除

## 93、写一个获取非行间样式的函数

dom.style.color;//行内的color属性的值

function getStyle(obj,attr)

{

**if(obj.currentStyle) //ie9之前**

**{**

**return obj.currentStyle[attr];**

**}**

**else{ //ie9+ 标准浏览器**

**window.getComputedStyle(obj,null)[attr];**

**}**

}

## 95、闭包是什么，有什么特性，对页面有什么影响

闭包就是能够读取其他函数内部变量的函数。

[闭包](http://blog.csdn.net/gaoshanwudi/article/details/7355794) 此链接可查看（问这个问题的不是一个公司）

## 96、解释jsonp的原理，以及为什么不是真正的ajax

动态创建script标签，回调函数

Ajax是页面无刷新请求数据操作

## 99、字符串反转，如将 '12345678' 变成 '87654321'

|  |  |
| --- | --- |
|  | //大牛做法；  //思路：先将字符串转换为数组 split()，利用数组的反序函数 reverse()颠倒数组，再利用 join() 转换为字符串  var str = '12345678';  str = str.split('').reverse().join('');  [Ajax](http://1187163537.iteye.com/blog/2176588)  [Ajax交互](http://blog.csdn.net/chenmoquan/article/details/38560649)  [阿里-面试题](https://yq.aliyun.com/articles/5993%233) |

## 100、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 += ',';      }  } |

## 101、生成5个1-10之间不同的随机数；

**考点：Math.random()：结果0-1之间**

**function** getFiveRandomNumber(){  
 **var** arr=[];  
 **while**(arr.length<5){  
 **var** number=Math.floor(Math.random()\*10) + 1;  
 **if**(arr.indexOf(number)<0){  
 arr.push(number);  
 }  
 }  
 **return** arr;  
}

## 102、去掉数组中重复的数字

**方法一；**

|  |  |
| --- | --- |
|  | //思路：每遍历一次就和之前的所有做比较，不相等则放入新的数组中！  //这里用的原型 个人做法；  **Array.prototype.unique** = function(){      var len = this.length,          newArr = [],          flag = 1;      for(var i = 0; i < len; i++, flag = 1){          for(var j = 0; j < i; j++){              if(this[i] == this[j]){                  flag = 0;        //找到相同的数字后，不执行添加数据              }          }          flag ? newArr.push(this[i]) : '';      }      return newArr;  } |

**方法二：**

|  |  |
| --- | --- |
|  | (function(arr){      var len = arr.length,          newArr = [],          flag;      for(var i = 0; i < len; i+=1, flag = 1){          for(var j = 0; j < i; j++){              if(arr[i] == arr[j]){                  flag = 0;              }          }          flag?newArr.push(arr[i]):'';      }      alert(newArr);  })([1, 1, 22, 3, 4, 55, 66]); |

## 103、阶乘函数；9\*8\*7\*6\*5…\*1

|  |  |
| --- | --- |
|  | //原型方法  Number.prototype.N = function(){      var re = 1;      for(var i = 1; i <= this; i++){          re \*= i;      }      return re;  }  var num = 5;  alert(num.N()); |

## 104、window.location.search返回的是什么？

    答：查询(参数)部分。除了给动态语言赋值以外，我们同样可以给静态页面,并使用javascript来获得相信应的参数值

        返回值：?ver=1.0&id=timlq 也就是问号后面的！

//url:http://www.sina.com/getage?number=1&year=2016

## 105、window.location.hash 返回的是什么？

答：反正当前页面的**锚点**

如果网页的地址为：http://www.sina.com/getage?#age

window.location.hash就会返回”#age”

## 106、window.location.reload() 作用？

    答：刷新当前页面。

## 107、阻止冒泡函数

|  |  |
| --- | --- |
|  | function stopPropagation(e) {      e = e || window.event;      if(e.stopPropagation) { //W3C阻止冒泡方法          e.stopPropagation();      } else {          e.cancelBubble = true; //IE阻止冒泡方法      }  }  document.getElementById('need\_hide').onclick = function(e) {      stopPropagation(e);  } |

## 108、什么是闭包？ 写一个简单的闭包？；

    答：闭包就是能够读取其他函数内部变量的函数。在本质上，闭包就是将函数内部和函数外部连接起来的一座桥梁。

|  |  |
| --- | --- |
|  | function outer(){      var num = 1;      function inner(){          var n = 2;          alert(n + num);      }      return inner;  }  outer()(); |

## 109、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。

## 110、看题做答：

|  |  |
| --- | --- |
|  | function f1(){      var tmp = 1;      this.x = 3;      console.log(tmp);    //A      console.log(this.x)；     //B  }  var obj = new f1(); //1 ==>1 3  console.log(obj.x)     //2 ==>3  console.log(f1());        //3 ==>1 3 undefined |

    分析：

        这道题让我重新认识了对象和函数，首先看代码（1），这里实例话化了 f1这个类。相当于执行了 f1函数。所以这个时候 A 会输出 1， 而 B 这个时候的 this 代表的是 实例化的当前对象 obj B 输出 3.。 代码（2）毋庸置疑会输出 3， 重点 代码（3）首先这里将不再是一个类，它只是一个函数。那么 A输出 1， B呢？这里的this 代表的其实就是window对象，那么this.x 就是一个全局变量 相当于在外部 的一个全局变量。所以 B 输出 3。最后代码由于f没有返回值那么一个函数如果没返回值的话，将会返回 underfined ，所以答案就是 ： 1， 3， 3， 1， 3， underfined 。

## 111、下面输出多少？

|  |  |
| --- | --- |
|  | var o1 = new Object();  var o2 = o1;  o2.name = "CSSer";  console.log(o1.name); |

  如果不看答案，你回答正确了的话，那么说明你对javascript的数据类型了解的还是比较清楚了。js中有两种数据类型，分别是：基本数据类型和引用数据类型（object Array）。对于保存基本类型值的变量，变量是按值访问的，因为我们操作的是变量实际保存的值。对于保存引用类型值的变量，变量是按引用访问的，我们操作的是变量值所引用（指向）的对象。答案就清楚了：  //CSSer;

## 112、下面代码的输出结果？

|  |  |
| --- | --- |
|  | function changeObjectProperty (o) {      o.siteUrl = "http://www.csser.com/";      o = new Object();      o.siteUrl = "http://www.popcg.com/";  }  var CSSer = new Object();  changeObjectProperty(CSSer);  console.log(CSSer.siteUrl); // |

    如果CSSer参数是按引用传递的，那么结果应该是"http://www.popcg.com/"，但实际结果却仍是"http://www.csser.com/"。事实是这样的：在函数内部修改了引用类型值的参数，该参数值的原始引用保持不变。我们可以把参数想象成局部变量，当参数被重写时，这个变量引用的就是一个局部变量，局部变量的生存期仅限于函数执行的过程中，函数执行完毕，局部变量即被销毁以释放内存。

    （补充：内部环境可以通过作用域链访问所有的外部环境中的变量对象，但外部环境无法访问内部环境。每个环境都可以向上搜索作用域链，以查询变量和函数名，反之向下则不能。）

## 113、输出多少？

|  |  |
| --- | --- |
|  | var a = 6;  setTimeout(function () {      var a = 666;//**由于变量a是一个局部变量**      alert(a);      // 输出666，  }, 1000);  console.log(a);  a = 66; |

因为var a = 666;定义了局部变量a，并且赋值为666，根据变量作用域链，  
全局变量处在作用域末端，优先访问了局部变量，从而覆盖了全局变量 。

|  |  |
| --- | --- |
|  | var a = 6;  setTimeout(function () {  //变量声明提前      alert(a);      // 输出undefined      var a = 666;  }, 1000);  a = 66; |
|  |  |

因为var a = 666;定义了局部变量a，同样覆盖了全局变量，但是在alert(a);之前  
a并未赋值，所以输出undefined。

|  |  |
| --- | --- |
|  | var *a* = 6; setTimeout(function(){  alert(a);  var a = 66; }, 1000); *a* = 666; alert(*a*); //结果：666 undefined |

记住： 异步处理，一切OK 声明提前

## 114、输出多少？

|  |  |
| --- | --- |
|  | function setN(obj){      obj.name='屌丝';      obj = new Object();      obj.name = '腐女';  };  var per = new Object();  setN(per);  alert(per.name);  //屌丝 内部 |

## 115、JS的继承性

|  |  |
| --- | --- |
|  | window.color = 'red';  var o = {color: 'blue'};  function sayColor(){      alert(this.color);  }  考点：1、this的指向  2、call的用法  sayColor(); //red  sayColor.call(this); //red this指向的是window对象  sayColor.call(window); //red  sayColor.call(o); //blue |

## 117、加减运算

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

## 118、什么是同源策略？

    指： 同协议、端口、域名的安全策略，由网景(Netscape)公司提出来的安全协议！

## 119、call和apply的区别是什么？

参数形式不同，call(obj, pra, pra)后面是单个参数。apply(obj, [args])后面是数组。

## 121、结果是什么？

**考点：一个函数作为一个普通函数和作为一个构造函数**

|  |  |
| --- | --- |
|  | function foo(){      foo.a = function(){alert(1)};      this.a = function(){alert(2)};      a = function(){alert(3)};      var a = function(){alert(4)};  };  foo.prototype.a = function(){alert(5)};  foo.a = function(){alert(6)};  foo.a(); //6  var obj = new foo();  obj.a(); //2  foo.a(); //1 |

## 122、输出结果

**考点：变量作用域、this的使用**

|  |  |
| --- | --- |
|  | var a = 5;  function test(){      a = 0;      alert(a);      alert(this.a); //没有定义 a这个属性      var a;      alert(a)  }  test(); // 0, 5, 0  new test(); // 0, undefined, 0 //由于类它自身没有属性a， 所以是undefined |

## 123、计算字符串字节数：

**考点：字符串的charCodeAt方法**

**[charCodeAt](https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/String/charCodeAt)**

|  |  |
| --- | --- |
|  | function getByte(s){       if(!arguments.length||!s) return null;       if(""==s) return 0;     //无效代码，因为上一句!s已经判断过       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uniCode码       }     //汉字的uniCode码大于 255bit 就是两个字节       alert(l);  }  getByte("hello world!"); |

## 124、以下代码将会产生什么样的结果？

**考点：!（逻辑非）运算符的使用**

|  |  |
| --- | --- |
|  | var bool = !!2; alert(bool)；//true; |

技巧：编程中经常使用双向非操作可以把字符串和数字转换为布尔值。

## 125、声明对象，添加属性，输出属性

**考点：this的使用**

|  |  |
| --- | --- |
|  | var obj = {          name: 'leipeng',          showName: function(){              alert(this.name);          }      }  obj.showName(); |

## 126、匹配输入的字符：第一个必须是字母或下划线开头，后面就是字母和数字或者下划线构成，长度5-20

**考点：正则表达式**

|  |  |
| --- | --- |
|  | var reg = /^[a-zA-Z\_][a-zA-Z0-9\_]{4,19}$/,              name1 = 'leipeng',              name2 = '0leipeng',              name3 = '你好leipeng',              name4 = 'hi';            alert(reg.test(name1));          alert(reg.test(name2));          alert(reg.test(name3));          alert(reg.test(name4)); |

## 127、检测变量类型

**考点：typeof运算符**

|  |  |
| --- | --- |
|  | function checkStr(str){          return typeof str == 'string';      }      alert(checkStr('leipeng')); |

## 128、如何在HTML中添加事件，几种方法？

1、标签之中直接添加 onclick="alert(‘欢迎来到传智播客学习’)";

2、标签之中添加 onclick="fun()";

## 129、BOM对象有哪些，列举window对象？

    1、window对象 ，是JS的最顶层对象，其他的BOM对象都是window对象的属性；

    2、document对象，文档对象；

    3、location对象，浏览器当前URL信息；

    4、navigator对象，浏览器本身信息；

    5、screen对象，客户端屏幕信息；

    6、history对象，浏览器访问历史信息；

## 130、请问代码实现 outerHTML

//说明：outerHTML其实就是innerHTML再加上本身；

|  |
| --- |
| <!doctype html>   <html>    <head>      <meta charset="UTF-8">      <title>Document</title>    </head>    <body>      <div id="outer">         hello      </div>    <script>     Function getOuerHTML(element) {  Var d = document.createElement(“div”);  d.appendChild(element);  d = null;  return d.innerHTML;  }      function $(id){     return document.getElementById(id);     }  Console.log(getOuterHTML($(“d”)));    </script>   </body>   </html> |
|  |

## 131、JS中的简单继承 （使用call方法实现！）

**考点：call()**

|  |  |
| --- | --- |
|  | //顶一个父母类，注意：类名都是首字母大写的哦！    function Parent(name, money){              this.name = name;              this.money = money;              this.info = function(){                  alert('姓名： '+this.name+' 钱： '+ this.money);              }          }          //定义孩子类          function Children(name){  **Parent.call(this, name); //继承 姓名属性，不要钱。**              this.info = function(){                  alert('姓名： '+this.name);              }          }          //实例化类          var per = new Parent('parent', 800000000000);          var chi = new Children('child');          per.info();          chi.info(); |

## 132、bind(), live(), delegate()的区别

    bind： 绑定事件，对新添加的事件不起作用，方法用于将一个处理程序附加到每个匹配元素的事件上并返回jQuery对象。

    live： 方法将一个事件处理程序附加到与当前选择器匹配的所有元素（包含现有的或将来添加的）的指定事件上并返回jQuery对象。

delegate： 方法基于一组特定的根元素将处理程序附加到匹配选择器的所有元素（现有的或将来的）的一个或多个事件上。

jQuery1.7+推荐使用：on() off()

## 134、简述link和import的区别？

[link和import的区别](http://www.cnblogs.com/cnwebdeveloper/archive/2011/04/19/2020792.html)

区别1：link是XHTML标签，除了加载CSS外，还可以定义RSS等其他事务；@import属于CSS范畴，只能加载CSS。

区别2：link引用CSS时，在页面载入时同时加载；@import需要页面网页完全载入以后加载。

区别3：link是XHTML标签，无兼容问题；@import是在CSS2.1提出的，低版本的浏览器不支持。

区别4：link支持使用Javascript控制DOM去改变样式；而@import不支持。

## 136、 解析URL成一个对象？

String.prototype.urlQueryString = function(){

                var url = this.split('?')[1].split('&'),

                    len = url.length;

                this.url = {};

                for(var i = 0; i < len; i += 1){

                    var cell = url[i].split('='),

                        key = cell[0],

                        val = cell[1];

                    this.url[''+key+''] = val;

                }

                return this.url;

}

            var url = '?name=12&age=23';

            console.log(url.urlQueryString().age);

## 137、看下列代码输出什么？

var foo = "11"+2-"1";  
console.log(foo);  
console.log(typeof foo);

执行完后foo的值为111，foo的类型为Number。

## 138、看下列代码,输出什么？

**考点：内存、引用类型**

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

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

　　window.onload()方法是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，相当于**DOMContentLoaded事件**，不必等到加载完毕。

## 143、你如何优化自己的代码？

A、代码重用（函数封装）

B、避免使用过多的全局变量（命名空间，封闭空间，模块化mvc..）

C、拆分函数避免函数过于臃肿：**单一职责**原则

D、将面向过程的编程方式改为使用面向对象编程

E、适当的注释，尤其是一些复杂的业务逻辑或者是计算逻辑，都应该写出这个业务逻辑的具体过程

F、内存管理，尤其是闭包中的变量释放

## 144、请描述出下列代码运行的结果

**考点：this的使用**

function d(){

console.log(this);

}

d();//window

## 145、需要将变量e的值修改为“a+b+c+d”,请写出对应的代码

var e=”abcd”;

设计一段代码能够遍历下列整个DOM节点

<div>

<p>

<span><a/></span>

<span><a/></span>

</p>

<ul>

<li></li>

<li></li>

</ul>

</div>

## 147、使用js实现这样的效果：在文本域里输入文字时，当按下enter键时不换行，而是替换成“{{enter}}”,(只需要考虑在行尾按下enter键的情况).

**考点：onkeydown事件、keyCode属性**

textarea.onkeydown=function(e){

e.preventDefault();//为了阻止enter键的默认换行效果

if(e.keycode==”enter键码”){ //键码：108

testarea.value+=”{{enter}}”;

}

}

## 148、以下代码中end字符串什么时候输出

**考点：while循环、setTimeout的使用**

var t=true;

setTimeout(function(){

console.log(123);

t=false;

},1000);

while(t){}// **此时是一个死循环，永远不可能执行setTimeout中的回调函数**

console.log(‘end’);

## 149、specify(‘hello,world’)//=>’h,e,l,l,o,w,o,r,l,d’实现specify函数

**考点：split方法和join方法的使用**

function specify(str){

return str= "hello,world".split(",").join("").split("").join(",");

}

## 150、请将一个URL的search部分参数与值转换成一个json对象

//search部分的参数格式：a=1&b=2&c=3

function getJsonFromUrlSearch(*search*){  
 var item;  
 var result={};  
 if(*search*.indexOf('&')<0){  
 item=*search*.split('=');  
 result[item[0]]=item[1];  
 return result;  
 }  
 var splitArray=*search*.split('&');  
 for (var i = 0; i < splitArray.length; i++) {  
 var obj = splitArray[i];  
 item=obj.split('=');  
 result[item[0]]=item[1];  
 }  
 return result;  
}  
var *c*=getJsonFromUrlSearch("a=1&b=2&c=3");

## 151、请用原生js实现jquery的get\post功能

**1、创建XMLHttpRequest对象**

**function** createXMLHTTPRequest() {

                //1.创建XMLHttpRequest对象

                //这是XMLHttpReuquest对象无部使用中最复杂的一步

                //需要针对IE和其他类型的浏览器建立这个对象的不同方式写不同的代

**var** xmlHttpRequest;

**if** (window.XMLHttpRequest) {

                    //针对FireFox，Mozillar，Opera，Safari，IE7，IE8

                   xmlHttpRequest = **new** XMLHttpRequest();

                    //针对某些特定版本的mozillar浏览器的BUG进行修正

**if** (xmlHttpRequest.overrideMimeType) {

                        xmlHttpRequest.overrideMimeType("text/xml");

                    }

                } **else** **if** (window.ActiveXObject) {

                    //针对IE6，IE5.5，IE5

                    //两个可以用于创建XMLHTTPRequest对象的控件名称，保存在一个的数组中

                    //排在前面的版本较新

**var** activexName = [ "MSXML2.XMLHTTP", "Microsoft.XMLHTT" ];

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

**try** {

                            //取出一个控件名进行创建，如果创建成功就终止循

                            //如果创建失败，回抛出异常，然后可以继续循环，继续尝创建

                           xmlHttpRequest = **new** ActiveXObject(activexNamei]);

**if**(xmlHttpRequest){

**break**;

                           }

                        } **catch** (e) {

                        }

                    }

                }

**return** xmlHttpRequest;

            }

**2、get请求**

1. **function** get(){
2. **var** req = createXMLHTTPRequest();
3. **if**(req){
4. req.open("GET", "http://test.com/?keywords=手机", **true**);
5. req.onreadystatechange = **function**(){
6. **if**(req.readyState == 4){
7. **if**(req.status == 200){
8. alert("success");
9. }**else**{
10. alert("error");
11. }
12. }
13. }
14. req.send(**null**);
15. }
16. }

**3、post请求**

**function** post(){

**var** req = createXMLHTTPRequest();

**if**(req){

        req.open("POST", "http://test.com/", **true**);

        req.setRequestHeader("Content-Type","application/x-www-form-urlencoded; charset=gbk;");

        req.send("keywords=手机");

        req.onreadystatechange = **function**(){

**if**(req.readyState == 4){

**if**(req.status == 200){

                    alert("success");

                }**else**{

                    alert("error");

                }

            }

        }

    }

}

## 152、请简要描述web前端性能需要考虑哪方面，你的优化思路是什么？

//参见雅虎14web优化规则

//减少http请求：

//1、小图弄成大图，

//2、合理的设置缓存

//3、资源合并、压缩

//将外部的js文件置底

//内存管理：合理释放闭包函数

## 153、简述readyonly与disabled的区别

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

## 156、写出3个使用this的典型应用

构造函数中使用this，原型对象中使用this，对象字面量使用this

## 157、请尽可能详尽的解释ajax的工作原理

**思路：**先解释异步，再解释ajax如何使用

Ajax的原理简单来说通过XmlHttpRequest对象来向服务器发异步请求，从服务器获得数据，然后用javascript来操作DOM而更新页面。这其中最关键的一步就是从服务器获得请求数据。要清楚这个过程和原理，我们必须对 XMLHttpRequest有所了解。

　XMLHttpRequest是ajax的核心机制，它是在IE5中首先引入的，是一种支持异步请求的技术。简单的说，也就是javascript可以及时向服务器提出请求和处理响应，而不阻塞用户。达到无刷新的效果。

## 158、为什么扩展javascript内置对象不是好的做法？

因为扩展内置对象会影响整个程序中所使用到的该内置对象的原型属性

## 159、请解释一下javascript的同源策略

域名、协议、端口相同

## 161、浏览器标准模式和怪异模式之间的区别是什么？

标准模式是指，浏览器按W3C标准解析执行代码；

怪异模式则是使用浏览器自己的方式解析执行代码，因为不同浏览器解析执行的方式不一样，所以我们称之为怪异模式。

浏览器解析时到底使用标准模式还是怪异模式，与你网页中的DTD声明直接相关，DTD声明定义了标准文档的类型（标准模式解析）文档类型，会使浏览器使用相应的方式加载网页并显示，忽略DTD声明,将使网页进入怪异模式

## 162、如果设计中使用了非标准的字体，你该如何去实现？

先通过font-face定义字体，再引用

@font-face

{

font-family: myFirstFont;

src: url('Sansation\_Light.ttf'),

url('Sansation\_Light.eot'); /\* IE9+ \*/

}

## 164、module(12,5)//2 实现满足这个结果的modulo函数

function modulo(*a*,*b*){  
 return a%b;//return a/b;  
}

## 165、HTTP协议中，GET和POST有什么区别？分别适用什么场景 ？

get传送的数据长度有限制，post没有

get通过url传递，在浏览器地址栏可见，post是在报文中传递

适用场景：

post一般用于表单提交

get一般用于数据查询，或者严格要求不是那么高的场景

## 166、HTTP状态消息200 302 304 403 404 500分别表示什么

200：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。

302：请求的资源临时从不同的 URI响应请求。由于这样的重定向是临时的，客户端应当继续向原有地址发送以后的请求。只有在Cache-Control或Expires中进行了指定的情况下，这个响应才是可缓存的

304：如果客户端发送了一个带条件的 GET 请求且该请求已被允许，而文档的内容（自上次访问以来或者根据请求的条件）并没有改变，则服务器应当返回这个状态码。304响应禁止包含消息体，因此始终以消息头后的第一个空行结尾。

403：服务器已经理解请求，但是拒绝执行它。

404：请求失败，请求所希望得到的资源未被在服务器上发现。

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端的源代码出现错误时出现。

## 167、HTTP协议中，header信息里面，怎么控制页面失效时间（last-modified,cache-control,Expires分别代表什么）

|  |  |
| --- | --- |
| Last-Modified | 文 档的最后改动时间。客户可以通过If-Modified-Since请求头提供一个日期，该请求将被视为一个条件GET，只有改动时间迟于指定时间的文档 才会返回，否则返回一个304（Not Modified）状态。Last-Modified也可用setDateHeader方法来设置。 |
| Expires | 应该在什么时候认为文档已经过期，从而不再缓存它？ |

## 168、HTTP协议目前常用的有哪几个？KEEPALIVE从哪个版本开始出现的？

http1.0、http1.1

http1.1 keeplive

## 170、列举常用的web页面开发，调试以及优化工具

sublime vscode webstorm hbuilder dw

httpwatch=>ie

ff:firebug

chrome:

## 171、解释什么是sql注入

## 173、请列举js数组类型中的常用方法

|  |  |
| --- | --- |
| **方法** | **描述** |
| [concat()](http://www.w3school.com.cn/jsref/jsref_concat_array.asp) | 连接两个或更多的数组，并返回结果。 |
| [join()](http://www.w3school.com.cn/jsref/jsref_join.asp) | 把数组的所有元素放入一个字符串。元素通过指定的分隔符进行分隔。 |
| [pop()](http://www.w3school.com.cn/jsref/jsref_pop.asp) | 删除并返回数组的最后一个元素 |
| [push()](http://www.w3school.com.cn/jsref/jsref_push.asp) | 向数组的末尾添加一个或更多元素，并返回新的长度。 |
| [reverse()](http://www.w3school.com.cn/jsref/jsref_reverse.asp) | 颠倒数组中元素的顺序。 |
| [shift()](http://www.w3school.com.cn/jsref/jsref_shift.asp) | 删除并返回数组的第一个元素 |
| [slice()](http://www.w3school.com.cn/jsref/jsref_slice_array.asp) | 从某个已有的数组返回选定的元素 |
| [sort()](http://www.w3school.com.cn/jsref/jsref_sort.asp) | 对数组的元素进行排序 |
| [splice()](http://www.w3school.com.cn/jsref/jsref_splice.asp) | 删除元素，并向数组添加新元素。 |
| [toSource()](http://www.w3school.com.cn/jsref/jsref_tosource_array.asp) | 返回该对象的源代码。 |
| [toString()](http://www.w3school.com.cn/jsref/jsref_toString_array.asp) | 把数组转换为字符串，并返回结果。 |
| [toLocaleString()](http://www.w3school.com.cn/jsref/jsref_toLocaleString_array.asp) | 把数组转换为本地数组，并返回结果。 |
| [unshift()](http://www.w3school.com.cn/jsref/jsref_unshift.asp) | 向数组的开头添加一个或更多元素，并返回新的长度。 |
| [valueOf()](http://www.w3school.com.cn/jsref/jsref_valueof_array.asp) | 返回数组对象的原始值 |

## 174、FF与IE中如何阻止事件冒泡，如何获取事件对象，以及如何获取触发事件的元素

## 175、列举常用的js框架以及分别适用的领域

jquery：简化了js的一些操作，并且提供了一些非常好用的API

jquery ui、jquery-easyui：在jqeury的基础上提供了一些常用的组件 日期，下拉框，表格这些组件

require.js、sea.js（阿里的玉帛）==>模块化开发使用的

zepto：精简版的jquery，常用于手机web前端开发 提供了一些手机页面实用功能,touch

ext.js：跟jquery差不多，但是不开源，也没有jquery轻量

angular、knockoutjs、avalon(去哪儿前端架构师：司徒正美)：MV\*框架，适合用于单页应用开发(SPA)

## 177、js可否实现面向对象编程，如果可以如何实现js对象的继承

答案：可以

javascript中面向对象的相关知识点：构造函数、原型、原型链（对象的原型链、函数的原型链）、继承、call、apply

实现继承的几种方式

**原型链**

## 180、如何获取对象a拥有的所有属性（可枚举的、不可枚举的，不包括继承来的属性）

第一种方式：Object.keys

浏览器兼容性：IE9+

第二种方式：for…in

问题：即可以获取自身属性、也可以获取继承的属性，所有需要进行过滤

for(o in obj){

if(obj. hasOwnProperty (o)){

//把o这个属性放入到一个数组中

}

}

## 181、有下面这样一段HTML结构，使用css实现这样的效果：

左边容器无论宽度如何变动，右边容器都能自适应填满父容器剩余的宽度。

<div class=”warp”>

<div class=”left”></div>

<div class=”right”></div>

</div>

## 182、下面这段代码想要循环输出结果01234，请问输出结果是否正确，如果不正确，请说明为什么，并修改循环内的代码使其输出正确结果

**考点：setTimeout的执行原理**

for(var i=0;i<5;++i){

setTimeout(function(){

console.log(i+’’);

},100);

}

**正确答案：**

for (var i = 0; i < 5; i++) {

setTimeout((function(e){

console.log(e);

})(i), 100)

};

## 184、JavaScript以下哪条语句会产生运行错误

A. var obj = (); B. var obj = []; C. var obj = {}; D. var obj = //;

答案：AD

## 185、以下哪些是javascript的全局函数：（ABCDE）

A. escape 函数可对字符串进行编码，这样就可以在所有的计算机上读取该字符串。ES3中反对使用该方法，应用使用 decodeURI() 和 decodeURIComponent() 替代它。

B. parseFloat parseFloat() 函数可解析一个字符串，并返回一个浮点数。

该函数指定字符串中的首个字符是否是数字。如果是，则对字符串进行解析，直到到达数字的末端为止，然后以数字返回该数字，而不是作为字符串。

C. eval 函数可计算某个字符串，并执行其中的的 JavaScript 代码。

D. setTimeout

E. alert

## 186、关于IE的window对象表述正确的有：（CD）

A. window.opener属性本身就是指向window对象

window.opener返回打开当前窗口的那个窗口的引用.

如果当前窗口是由另一个窗口打开的, **window.opener**保留了那个窗口的引用. 如果当前窗口不是由其他窗口打开的, 则该属性返回 null.

B. window.reload()方法可以用来刷新当前页面 //正确答案：应该是location.reload或者window.location.reload

C. window.location=”a.html”和window.location.href=”a.html”的作用都是把当前页面替换成a.html页面

D. 定义了全局变量g；可以用window.g的方式来存取该变量

## 187、描述错误的是：D

A、Http状态码302表示暂时性转移 对

B、DomContentLoaded事件早于onload事件 //正确

当 onload 事件触发时，页面上所有的DOM，样式表，脚本，图片，flash都已经加载完成了。

当 DOMContentLoaded 事件触发时，仅当DOM加载完成，不包括样式表，图片，flash。

C、IE678不支持事件捕获

**解析：IE9开始就出现了addEventListener方法**

D、localStorage 存储的数据在电脑重启后丢失 //**错误，因为没有时间限制**

try...catch 语句。(在 IE5+、Mozilla 1.0、和 Netscape 6 中可用)

## 188、关于link和@import的区别正确的是 AB

A: link属于XHTML标签，而@import是CSS提供的；

B：页面被加载时，link会同时被加载，而后者引用的CSS会等到页面被加载完再加载

C：import只在IE5以上才能识别 而link是XHTML标签，无兼容问题

**解析：CSS2.1开始支持import**

D: link方式的样式的权重高于@import的权重

**解析：权重一样，只取决于书写顺序的先后**

## 189、下面正确的是 A

A: 跨域问题能通过JsonP方案解决

B：不同子域名间仅能通过修改window.name解决跨域 //还可以通过script标签src jsonp

C：只有在IE中可通过iframe嵌套跨域 //任何浏览器都可以使用iframe

D：MediaQuery属性是进行视频格式检测的属性是做响应式的

## 189、不用任何插件，如何实现一个tab栏切换？

通过改变不同层的css设置层的显示和隐藏

## 190、基本数据类型的专业术语以及单词拼写

## 191、变量的命名规范以及命名推荐

## 192、三种弹窗的单词以及三种弹窗的功能

alert

confirm

prompt

## 193、console.log( 8 | 1 ); 输出值是多少？

**考点：| 位运算符**

答案：9

## 194、只允许使用 + - \* / 和 Math.\* ，求一个函数 y = f(x, a, b);当x > 100 时返回 a 的值，否则返回 b 的值，不能使用 if else 等条件语句，也不能使用|,?:,数组。

答案：

function f(x, a, b) {

var temp = Math.ceil(Math.min(Math.max(x - 100, 0), 1));

return a \* temp + b \* (1 - temp);

}

console.log(f(-10, 1, 2));

## 196、一个div，有几种方式得到这个div的jQuery对象？<div class='aabbcc' id='nodesView'></div>想直接获取这个div的dom对象，如何获取？dom对象如何转化为jQuery对象？

var domView=document.getElementById(“nodesView”)

document.getElementsByClassName(“aabbcc”);

document.querySelector(“.aabbcc#nodesView”);

转换为jquery对象：$( domView)

## 197、主流浏览器内核

IE trident

火狐gecko

谷歌苹果webkit

Opera：Presto

## 198、如何显示/隐藏一个dom元素？请用原生的JavaScript方法实现

dom.style.display=”none”;

dom.style.display=””;

## 199、JavaScript有哪几种数据类型

Number String Boolean Null Undefined Object

## 200、jQuery框架中$.ajax()的常用参数有哪些？

**type**

类型：String

默认值: "GET")。请求方式 ("POST" 或 "GET")， 默认为 "GET"。注意：其它 HTTP 请求方法，如 PUT 和 DELETE 也可以使用，但仅部分浏览器支持。

**url**

类型：String

默认值: 当前页地址。发送请求的地址。

**success**

类型：Function

请求成功后的回调函数。

参数：由服务器返回，并根据 dataType 参数进行处理后的数据；描述状态的字符串。

这是一个 Ajax 事件。

**options**

类型：Object

可选。AJAX 请求设置。所有选项都是可选的。

**async**

类型：Boolean

默认值: true。默认设置下，所有请求均为异步请求。如果需要发送同步请求，请将此选项设置为 false。

注意，同步请求将锁住浏览器，用户其它操作必须等待请求完成才可以执行。

**beforeSend(XHR)**

类型：Function

发送请求前可修改 XMLHttpRequest 对象的函数，如添加自定义 HTTP 头。

XMLHttpRequest 对象是唯一的参数。

这是一个 Ajax 事件。如果返回 false 可以取消本次 ajax 请求。

**cache**

类型：Boolean

默认值: true，dataType 为 script 和 jsonp 时默认为 false。设置为 false 将不缓存此页面。

jQuery 1.2 新功能。

**contentType**

类型：String

默认值: "application/x-www-form-urlencoded"。发送信息至服务器时内容编码类型。

默认值适合大多数情况。如果你明确地传递了一个 content-type 给 $.ajax() 那么它必定会发送给服务器（即使没有数据要发送）。

**data**

类型：String

发送到服务器的数据。将自动转换为请求字符串格式。GET 请求中将附加在 URL 后。查看 processData 选项说明以禁止此自动转换。必须为 Key/Value 格式。如果为数组，jQuery 将自动为不同值对应同一个名称。如 {foo:["bar1", "bar2"]} 转换为 '&foo=bar1&foo=bar2'。

**dataFilter**

类型：Function

给 Ajax 返回的原始数据的进行预处理的函数。提供 data 和 type 两个参数：data 是 Ajax 返回的原始数据，type 是调用 jQuery.ajax 时提供的 dataType 参数。函数返回的值将由 jQuery 进一步处理。

**dataType**

类型：String

预期服务器返回的数据类型。如果不指定，jQuery 将自动根据 HTTP 包 MIME 信息来智能判断，比如 XML MIME 类型就被识别为 XML。在 1.4 中，JSON 就会生成一个 JavaScript 对象，而 script 则会执行这个脚本。随后服务器端返回的数据会根据这个值解析后，传递给回调函数。可用值:

* "xml": 返回 XML 文档，可用 jQuery 处理。
* "html": 返回纯文本 HTML 信息；包含的 script 标签会在插入 dom 时执行。
* "script": 返回纯文本 JavaScript 代码。不会自动缓存结果。除非设置了 "cache" 参数。注意：在远程请求时(不在同一个域下)，所有 POST 请求都将转为 GET 请求。（因为将使用 DOM 的 script标签来加载）
* "json": 返回 JSON 数据 。
* "jsonp": JSONP 格式。使用 JSONP 形式调用函数时，如 "myurl?callback=?" jQuery 将自动替换 ? 为正确的函数名，以执行回调函数。
* "text": 返回纯文本字符串

**error**

类型：Function

默认值: 自动判断 (xml 或 html)。请求失败时调用此函数。

有以下三个参数：XMLHttpRequest 对象、错误信息、（可选）捕获的异常对象。

如果发生了错误，错误信息（第二个参数）除了得到 null 之外，还可能是 "timeout", "error", "notmodified" 和 "parsererror"。

这是一个 Ajax 事件。

## 写一个post请求并带有发送数据和返回数据的样例

*$*.ajax({  
 url:"1.html",  
 data:{name:"张三",age:18},//post数据  
 dataType:"json",  
 type:"POST",  
 success:function(*data*){  
 //data：返回的数据  
 },  
 error:function(){  
 //异常处理  
 }  
});

## 201、JavaScript数组元素添加、删除、排序等方法有哪些？

Array.concat( ) 连接数组

Array.join( ) 将数组元素连接起来以构建一个字符串

Array.length 数组的大小

Array.pop( ) 删除并返回数组的最后一个元素

Array.push( ) 给数组添加元素

Array.reverse( ) 颠倒数组中元素的顺序

Array.shift( ) 将元素移出数组

Array.slice( ) 返回数组的一部分

Array.sort( ) 对数组元素进行排序

Array.splice( ) 插入、删除或替换数组的元素

Array.toLocaleString( ) 把数组转换成局部字符串

Array.toString( ) 将数组转换成一个字符串

Array.unshift( ) 在数组头部插入一个元素

## 202、如何添加html元素的事件，有几种方法？请列举

a、直接在标签里添加：<div onclick=”alert(你好)”>这是一个层</div>

b、在元素上通过js添加:

c、使用事件注册函数添加

## 203、JavaScript的循环语句有哪些？

while for do while for…in

## 204、作用域-编译期执行期以及全局局部作用域问题

理解js执行主要的两个阶段：预解析和执行期

## 205、闭包：下面这个ul，如何点击每一列的时候alert其index？

<ul id="test">

<li>这是第一条</li>

<li>这是第二条</li>

<li>这是第三条</li>

</ul>

//非闭包实现

var *lis*=document.querySelectorAll('li');  
document.querySelector('#test').onclick=function(*e*){  
 for (var i = 0; i < *lis*.length; i++) {  
 var li = *lis*[i];  
 if(li==*e*.target){  
 alert(i);  
 }  
 }  
};

//闭包实现

var *lis*=document.querySelectorAll('li');  
for (var *i* = 0; *i* < *lis*.length; *i*++) {  
 var *li* = *lis*[*i*];  
 *li*.onclick=(function(*index*){  
 return function(*e*){  
 alert(*index*);  
 };  
 })(*i*);  
}

## 206、列出3条以上ff和IE的脚本兼容问题

1、在IE下可通过document.frames["id"];得到该IFRAME对象，

而在火狐下则是通过document.getElementById("content\_panel\_if").contentWindow;

2、IE的写法： \_tbody=\_table.childNodes[0]

在FF中，firefox会在子节点中包含空白则第一个子节点为空白""， 而ie不会返回空白

可以通过if("" != node.nodeName)过滤掉空白子对象

3、模拟点击事件

if(document.all){ //ie下

document.getElementById("a3").click();

}

else{ //非IE

var evt = document.createEvent("MouseEvents");

evt.initEvent("click", true, true);

document.getElementById("a3").dispatchEvent(evt);

}

4、事件注册

if (isIE){window.attachEvent("onload", init);}else{window.addEventListener("load", init, false);}

## 209、如现在有一个效果，有显示用户头像、用户昵称、用户其他信息；当用户鼠标移到头像上时，会弹出用户的所有信息；如果是你，你会如何实现这个功能，请用代码实现？

//答案见：J:\代码,PPT,笔记,电子书\面试题\面试题02.html

## 210、call与apply有什么作用？又有什么什么区别？用callee属性实现函数递归？

apply的参数是数组,call的参数是单个的值，除此之外，两者没有差别，重点理解this的改变，callee已经不推荐使用

## 211、用正则表达式，写出由字母开头，其余由数字、字母、下划线组成的6~30的字符串？

var reg=/^[a-ZA-Z][\da-zA-Z\_]{5,29}/;

## 212、列举浏览器对象模型BOM里常用的至少4个对象，并列举window对象的常用方法至少5个 （10分）

对象：window document location screen history navigator

方法：alert() confirm() prompt() open() close() setInterval() setTimeout() clearInterval() clearTimeout()

(详细参见：J:\代码,PPT,笔记,电子书\面试题\window对象方法.png)

## 213、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；

用法：fn.caller

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。

用法：arguments.callee

## 214、对于apply和call两者在作用上是相同的，即是调用一个对象的一个方法，以另一个对象替换当前对象。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。

但两者在参数上有区别的。对于第一个参数意义都一样，但对第二个参数：?apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。?如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3]) 。

## 215、在Javascript中什么是伪数组？如何将伪数组转化为标准数组？

伪数组（类数组）：无法直接调用数组方法或期望length属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的argument参数，还有像调用getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。

可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

## 216、写一个函数可以计算 sum(5,0,-5);输出0; sum(1,2,3,4);输出10;

function calc(){  
 var result=0;  
 for (var i = 0; i < arguments.length; i++) {  
 var obj = arguments[i];  
 result+=obj;  
 }  
 return result;  
}  
alert(calc(1,2,3,4));

Js基本功

## 217、事件代理怎么实现？

在元素的父节点注册事件，通过事件冒泡，在父节点捕获事件

## 218、《正则》写出正确的正则表达式匹配固话号，区号3-4位，第一位为0，中横线，7-8位数字，中横线，3-4位分机号格式的固话号

常用正则表达式语法要熟悉

/0[0-9]{2,3}-\d{7,8}/

## 219、《算法》 一下A,B可任选一题作答，两题全答加分

A:农场买了一只羊，第一年是小羊，第二年底生一只，第三年不生，第四年底再生一只，第五年死掉。

## B: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}

先去重再排序

去重方法参考：J:\代码,PPT,笔记,电子书\面试题

## 220、请写出一张图片的HTML代码，已知道图片地址为“images/abc.jpg”,宽100px，高50px

## 221、请写一个正则表达式：要求最短6位数，最长20位，阿拉伯数和英文字母（不区分大小写）组成

^(?=.\*\d)(?=.\*[a-z])(?=.\*[A-Z])[a-zA-Z\d]{6,20}$

## 222、统计1到400亿之间的自然数中含有多少个1？比如1-21中，有1、10、11、12、13、14、15、16、17、18、19、20、21这么多自然数有13个1

答案参考：J:\代码,PPT,笔记,电子书\面试题\面试题\_222.html

## 223、删除与某个字符相邻且相同的字符，比如fdaffdaaklfjklja字符串处理之后成为“fdafdaklfjklja”

答案参考：J:\代码,PPT,笔记,电子书\面试题\面试题\_223.html

## 224、请写出三种以上的Firefox有但InternetExplorer没有的属性和函数

1、在IE下可通过document.frames["id"];得到该IFRAME对象，

而在火狐下则是通过document.getElementById("content\_panel\_if").contentWindow;

2、IE的写法： \_tbody=\_table.childNodes[0]

在FF中，firefox会在子节点中包含空白则第一个子节点为空白""， 而ie不会返回空白

可以通过if("" != node.nodeName)过滤掉空白子对象

3、模拟点击事件

if(document.all){ //ie下

document.getElementById("a3").click();

}

else{ //非IE

var evt = document.createEvent("MouseEvents");

evt.initEvent("click", true, true);

document.getElementById("a3").dispatchEvent(evt);

}

4、事件注册

if (isIE){window.attachEvent("onload", init);}else{window.addEventListener("load", init, false);}

## 225、请写出一个程序，在页面加载完成后动态创建一个form表单，并在里面添加一个input对象并给它任意赋值后义post方式提交到：http://127.0.0.1/save.php

答案参考：J:\代码,PPT,笔记,电子书\面试题\面试题\_225.html

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

面试经常遇到的排序，查找算法要熟悉

## 227、解释一下什么叫闭包，并实现一段闭包代码

简单理解就是函数的嵌套形成闭包，闭包包括函数本身及其外部作用域

## 228、简述一下什么叫事件委托以及其原理

在元素的父节点注册事件，通过事件冒泡，在父节点捕获事件

## 229、前端代码优化的方法

var User = { 对象

count = 1，属性

getCount：function（）{ 方法

return this.count;

}

}

console.log(User.getCount());

var func = User.getCount;

console.log(func());

1 undefined（window）;

## 230、下列JavaScript代码执行后，依次alert的结果是

(function test(){

var a=b=5;

alert(typeof a);

alert(typeof b);

})();

alert(typeof a);

alert(typeof b);

//number number undefined number

## 231、下列JavaScript代码执行后，iNum的值是

var iNum = 0;

for(var i = 1; i< 10; i++){

if(i % 5 == 0){

continue;

}

iNum++;

}

分析：

i=1 1

i=2 2

i=3 3

i=4 4

i=5

i=6 6

i=7 7

i=8 8

i=9 9

## 232、输出结果是多少？

1） var a;

var b = a \* 0;

if (b == b) {

console.log(b \* 2 + "2" - 0 + 4);

} else {

console.log(!b \* 2 + "2" - 0 + 4);

}

答案：26

扩展：关于乘法操作符：J:\代码,PPT,笔记,电子书\面试题\乘性操作符.png

2） <script>

var a = 1;

</script>

<script>

var a;

var b = a \* 0;

if (b == b) { **//b=0**

console.log(b \* 2 + "2" - 0 + 4);

} else {

console.log(!b \* 2 + "2" - 0 + 4);

}

</script>

答案：6

3） var t = 10;

function test(t){

var t = t++;//此时的t是一个局部变量，全局变量没有任何变化

console.log(t);//此时的结果又是多少？

}test(t);

console.log(t);

答案：10

4） var t = 10;

function test(test){

var t = test++;

}test(t);

console.log(t);

答案：10

6） var t = 10;

function test(test){

t = test++;

}test(t);

console.log(t);

答案：10

7） var t = 10;

function test(test){

t = t + test;//undefined+10=NaN

console.log(t);

var t = 3;

}test(t);

console.log(t);

答案：NaN 10

8）var a;

var b = a / 0;

if (b == b) {//b=NaN

console.log(!b \* 2 + "2" - 0 + 4);

} else {

console.log(!b \* 2 + "2" - 0 + 4);

}

答案：26

9）<script>

var a = 1;

</script>

<script>

var a;

var b = a / 0;

if (b == b) { **//b=Infinity**

console.log(b \* 2 + "2" + 4);

} else {

console.log(!b \* 2 + "2" + 4);

}

</script>

答案：Infinity24

## 233、用程序实现找到html中id名相同的元素？

<body>

<form id='form1'>

<div id='div1'></div>

<div id='div2'></div>

<div id='div3'></div>

<div id='div4'></div>

<div id='div5'></div>

<div id='div3'>id名重复的元素</div>

</form>

</body>

## 234、下列JavaScript代码执行后，运行的结果是

<button id='btn'>点击我</button>

var btn = document.getElementById('btn');

var handler = {

id: '\_eventHandler',

exec: function(){

alert(this.id);

}

}

btn.addEventListener('click', handler.exec);

答案：btn，因为handler.exec是由btn这个按钮执行的

## 235、☆☆☆下列JavaScript代码执行后，依次alert的结果是

var obj = {proto: {a:1,b:2}};

function F(){};

F.prototype = obj.proto;

var f = new F();

obj.proto.c = 3;

obj.proto = {a:-1, b:-2};

alert(f.a);//1

alert(f.c);//3

delete F.prototype['a'];

alert(f.a);//undefined

alert(obj.proto.a);//-1

## 236、下列JavaScript代码执行后的效果是

<ul id='list'>

<li>item</li>

<li>item</li>

<li>item</li>

<li>item</li>

<li>item</li>

</ul>

var items = document.querySelectorAll('#list>li');

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

setTimeout(function(){

items[i].style.backgroundColor = '#fee';

}, 5);

}

答案：异常

## 237、下列JavaScript代码执行后的li元素的数量是

<ul>

<li>Item</li>

<li></li>

<li></li>

<li>Item</li>

<li>Item</li>

</ul>

var items = document.getElementsByTagName('li');

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

if(items[i].innerHTML == ''){

items[i].parentNode.removeChild(items[i]);

}

}

答案：4个

## 238、程序中捕获异常的方法？

window.error

try{}catch(){}finally{}

## 239、将字符串”<tr><td>{$id}</td><td>{$name}</td></tr>”中的{$id}替换成10，{$name}替换成Tony （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’);

## 240、给String对象添加一个方法，传入一个string类型的参数，然后将string的每个字符间价格空格返回，例如：

addSpace(“hello world”) // -> ‘h e l l o ?w o r l d’

String.prototype.spacify = function(){

return this.split('').join(' ');

};

## 241、写出函数DateDemo的返回结果，系统时间假定为今天

function DateDemo(){

var d, s="今天日期是：";

d = new Date();

s += d.getMonth() + "/";

s += d.getDate() + "/";

s += d.getFullYear();

return s;

}

结果：今天日期是：7/17/2010

## 242、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);

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

答案：alert(stringArray.join(“”))

## 244、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;

}

## 245、.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]

2)实现对该数组的降序排列，输出[6,5,4,3,2,1]

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;

}

## 246、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

答案：

array1=array1.concat(bArray)

array1.splice(1,1)

## 247、如何消除一个数组里面重复的元素？

var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];

function deRepeat(){

var newArr=[];

var obj={};

var index=0;

var l=arr.length;

for(var i=0;i<l;i++){

if(obj[arr[i]]==undefined)

{

obj[arr[i]]=1;

newArr[index++]=arr[i];

}

else if(obj[arr[i]]==1)

}

return newArr;

}

var newArr2=deRepeat(arr);

alert(newArr2); //输出1,2,3,4,5,6,9,25

## 248、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。

var iArray = [];

funtion getRandom(istart, iend){

var iChoice = istart - iend +1;

return Math.floor(Math.random() \* iChoice + istart;

}

for(var i=0; i<10; i++){

iArray.push(getRandom(10,100));

}

iArray.sort();

## 249、正则表达式构造函数var reg=new RegExp(“xxx”)与正则表达字面量var reg=//有什么不同？匹配邮箱的正则表达式？

答案：当使用RegExp()构造函数的时候，不仅需要转义引号（即\”表示”），并且还需要双反斜杠（即\\表示一个\）。使用正则表达字面量的效率更高。?

## 250、1 var regMail = /^([a-zA-Z0-9\_-])+@([a-zA-Z0-9\_-])+((.[a-zA-Z0-9\_-]{2,3}){1,2})$/;

正则表达式对象3 – 清除空格

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"

## 251、数组和字符串

<script lang="JavaScript" type="text/javascript">

function outPut(s) {

document.writeln(s);

}

var a = "lashou";

var b = a;

outPut(b);

a = "拉手";

outPut(a);

outPut(b);

var a\_array = [1, 2, 3];

var b\_array = a\_array;

outPut(b\_array);

a\_array[3] = 4;

outPut(a\_array);

outPut(b\_array);

</script>

输出结果：

答案：lashou 拉手 lashou 1,2,3 1,2,3,4 1,2,3,4

## 252、下列控制台都输出什么

第1题：

function setName(){

name="张三";

}

setName();

console.log(name);

答案："张三"

## 253、第2题：

//考点：1、变量声明提升 2、变量搜索机制

var a=1;

function test(){

console.log(a);

var a=1;

}

test();

答案：undefined

## 254、第3题：

var b=2;

function test2(){

window.b=3;

console.log(b);

}

test2();

答案：3

## 255、第4题：

c=5;//声明一个全局变量c

function test3(){

window.c=3;

console.log(c); //答案：undefined，原因：由于此时的c是一个局部变量c，并且没有被赋值

var c;

console.log(window.c);//答案：3，原因：这里的c就是一个全局变量c

}

test3();

## 256、第5题：

var arr = [];

arr[0] = 'a';

arr[1] = 'b';

arr[10] = 'c';

alert(arr.length); //答案：11

console.log(arr[5]); //答案：undefined

## 257、第6题：

var a=1;

console.log(a++); //答案：1

console.log(++a); //答案：3

## 258、第7题：

console.log(null==undefined); //答案：true

console.log("1"==1); //答案：true，因为会将数字1先转换为字符串1

console.log("1"===1); //答案：false，因为数据类型不一致

## 259、第8题：

typeof 1; "number"

typeof "hello"; "string"

typeof /[0-9]/; "object"

typeof {}; "object"

typeof null; "object"

typeof undefined; "undefined"

typeof [1,2,3]; "object"

typeof function(){}; //"function"

## 260、第9题：

parseInt(3.14); //3

parseFloat("3asdf"); //3

parseInt("1.23abc456");

parseInt(true);//"true" NaN

## 261、第10题：

//考点：函数声明提前

function bar() {

return foo;

foo = 10;

function foo() {}

//var foo = 11;

}

alert(typeof bar());//"function"

## 262、第11题：考点：函数声明提前

var foo = 1;

function bar() {

foo = 10;

return;

function foo() {}

}

bar();

alert(foo);//答案：1

## 263、第12题：

console.log(a);//是一个函数

var a = 3;

function a(){}

console.log(a);////3

## 264、第13题：

//考点：对arguments的操作

function foo(a) {

arguments[0] = 2;

alert(a);//答案：2，因为：a、arguments是对实参的访问，b、通过arguments[i]可以修改指定实参的值

}

foo(1);

## 265、第14题：

function foo(a) {

alert(arguments.length);//答案：3，因为arguments是对实参的访问

}

foo(1, 2, 3);

## 266、第15题

bar();//报错

var foo = function bar(name) {

console.log("hello"+name);

console.log(bar);

};

//alert(typeof bar);

foo("world");//"hello"

console.log(bar);//undefined

console.log(foo.toString());

bar();//报错

## 267、第16题

function test(){

console.log("test函数");

}

setTimeout(function(){

console.log("定时器回调函数");

}, 0)

test();

function foo(){

var name="hello";

}

# 七、JS高级

## JQuery一个对象可以同时绑定多个事件，这是如何实现的？

jQuery可以给一个对象同时绑定多个事件，低层实现方式是使用addEventListner或attachEvent兼容不同的浏览器实现事件的绑定，这样可以给同一个对象注册多个事件。

## 知道什么是webkit么? 知道怎么用浏览器的各种工具来调试和debug代码么?

Webkit是浏览器引擎，包括html渲染和js解析功能，手机浏览器的主流内核，与之相对应的引擎有Gecko（Mozilla Firefox 等使用）和Trident（也称MSHTML，IE 使用）。

对于浏览器的调试工具要熟练使用，主要是页面结构分析，后台请求信息查看，js调试工具使用，熟练使用这些工具可以快速提高解决问题的效率

## 如何测试前端代码? 知道BDD, TDD, Unit Test么? 知道怎么测试你的前端工程么(mocha, sinon, jasmin, qUnit..)?

了解BDD行为驱动开发与TDD测试驱动开发已经单元测试相关概念，

## 4、前端templating(Mustache, underscore, handlebars)是干嘛的, 怎么用?

Web 模板引擎是为了使用户界面与业务数据（内容）分离而产生的，

Mustache 是一个 logic-less （轻逻辑）模板解析引擎，它的优势在于可以应用在 Javascript、PHP、Python、Perl 等多种编程语言中。

Underscore封装了常用的JavaScript对象操作方法，用于提高开发效率。

Handlebars 是 JavaScript 一个语义模板库，通过对view和data的分离来快速构建Web模板。

## 5、简述一下 Handlebars 的基本用法？

没有用过的话说出它是干什么的即可

官网：<http://handlebarsjs.com/>

参考：J:\代码,PPT,笔记,电子书\面试题\handlebarDemo

## 6、简述一下 Handlerbars 的对模板的基本处理流程， 如何编译的？如何缓存的？

学习技术不仅要会用，还有熟悉它的实现机制，这样在开发中遇到问题时才能更好的解决

## 7、用js实现千位分隔符?

原生js的熟练度，实践经验，实现思路

## 8、检测浏览器版本版本有哪些方式？

IE与标准浏览器判断，IE不同版本的判断，userAgent var ie = /\*@cc\_on !@\*/false;

## 9、我们给一个dom同时绑定两个点击事件，一个用捕获，一个用冒泡，你来说下会执行几次事件，然后会先执行冒泡还是捕获

对两种事件模型的理解

## 10、实现一个函数clone，可以对JavaScript中的5种主要的数据类型（包括Number、String、Object、Array、Boolean）进行值复制

* 考察点1：对于基本数据类型和引用数据类型在内存中存放的是值还是指针这一区别是否清楚
* 考察点2：是否知道如何判断一个变量是什么类型的
* 考察点3：递归算法的设计

|  |  |
| --- | --- |
|  | // 方法一：  Object.prototype.clone = function(){     var o = this.constructor === Array ? [] : {};     for(var e in this){       o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];     }     return o;  }  //方法二：     /\*\*       \* 克隆一个对象       \* @param Obj       \* @returns       \*/      function clone(Obj) {          var buf;          if (Obj instanceof Array) {              buf = [];//创建一个空的数组              var i = Obj.length;              while (i--) {                  buf[i] = clone(Obj[i]);              }              return buf;          }else if (Obj instanceof Object){              buf = {};//创建一个空对象              for (var k in Obj) { //为这个对象添加新的属性                  buf[k] = clone(Obj[k]);              }              return buf;          }else{ //普通变量直接赋值              return Obj;          }      } |

## 11、如何消除一个数组里面重复的元素？

|  |  |
| --- | --- |
|  | var arr=[1,2,3,3,4,4,5,5,6,1,9,3,25,4];          function deRepeat(){              var newArr=[];              var obj={};              var index=0;              var l=arr.length;              for(var i=0;i<l;i++){                  if(obj[arr[i]]==undefined)                    {                      obj[arr[i]]=1;                      newArr[index++]=arr[i];                    }                  else if(obj[arr[i]]==1)                    continue;              }              return newArr;          }          var newArr2=deRepeat(arr);          alert(newArr2); //输出1,2,3,4,5,6,9,25 |

## 12、小贤是一条可爱的小狗(Dog)，它的叫声很好听(wow)，每次看到主人的时候就会乖乖叫一声(yelp)。从这段描述可以得到以下对象：

|  |  |
| --- | --- |
|  | function Dog() {        this.wow = function() {                 alert(’Wow’);        }        this.yelp = function() {                this.wow();        }  } |

小芒和小贤一样，原来也是一条可爱的小狗，可是突然有一天疯了(MadDog)，一看到人就会每隔半秒叫一声(wow)地不停叫唤(yelp)。请根据描述，按示例的形式用代码来实。（继承，原型，setInterval）

|  |  |
| --- | --- |
|  | function MadDog() {      this.yelp = function() {            var self = this;            setInterval(function() {                  self.wow();            }, 500);        }  }  MadDog.prototype = new Dog();  //for test  var dog = new Dog();  dog.yelp();  var madDog = new MadDog();  madDog.yelp(); |

## 13、下面这个ul，如何点击每一列的时候alert其index?（闭包）

|  |  |
| --- | --- |
|  | <ul id=”test”>  <li>这是第一条</li>  <li>这是第二条</li>  <li>这是第三条</li>  </ul> |
|  | // 方法一：  var lis=document.getElementById('2223').getElementsByTagName('li');  for(var i=0;i<3;i++)  {      lis[i].index=i;      lis[i].onclick=function(){          alert(this.index);      };  }  //方法二：  var lis=document.getElementById('2223').getElementsByTagName('li');  for(var i=0;i<3;i++){      lis[i].index=i;      lis[i].onclick=(function(a){          return function() {              alert(a);          }      })(i);  } |

## 14、编写一个JavaScript函数，输入指定类型的选择器(仅需支持id，class，tagName三种简单CSS选择器，无需兼容组合选择器)可以返回匹配的DOM节点，需考虑浏览器兼容性和性能。

/\*\*\* @param selector {String} 传入的CSS选择器。\* @return {Array}\*/

|  |  |
| --- | --- |
|  | var query = function(selector) {  var reg = /^(#)?(\.)?(\w+)$/img;  var regResult = reg.exec(selector);  var result = [];  //如果是id选择器  if(regResult[1]) {  if(regResult[3]) {  if(typeof document.querySelector === "function") {  result.push(document.querySelector(regResult[3]));      }else {        result.push(document.getElementById(regResult[3]));      }     }     }     //如果是class选择器     else if(regResult[2]) {      if(regResult[3]) {         if(typeof document.getElementsByClassName === 'function') {           var doms = document.getElementsByClassName(regResult[3]);           if(doms) {             result = converToArray(doms);           }         }       //如果不支持getElementsByClassName函数       else {         var allDoms = document.getElementsByTagName("\*") ;         for(var i = 0, len = allDoms.length; i < len; i++) {           if(allDoms[i].className.search(new RegExp(regResult[2])) > -1) {             result.push(allDoms[i]);           }         }       }    }  }    //如果是标签选择器    else if(regResult[3]) {      var doms = document.getElementsByTagName(regResult[3].toLowerCase());      if(doms) {        result = converToArray(doms);      }    }    return result;    }    function converToArray(nodes){      var array = null;      try{        array = Array.prototype.slice.call(nodes,0);//针对非IE浏览器      }catch(ex){       array = new Array();      for( var i = 0 ,len = nodes.length; i < len ; i++ ) {        array.push(nodes[i])      }    }    return array;  } |

## 15、请评价以下代码并给出改进意见。

|  |  |
| --- | --- |
|  | if(window.addEventListener){      var addListener = function(el,type,listener,useCapture){          el.addEventListener(type,listener,useCapture);    };  }  else if(document.all){      addListener = function(el,type,listener){          el.attachEvent("on"+type,function(){            listener.apply(el);        });     }  } |

* 不应该在if和else语句中声明addListener函数，应该先声明；
* 不需要使用window.addEventListener或document.all来进行检测浏览器，应该使用能力检测；
* 由于attachEvent在IE中有this指向问题，所以调用它时需要处理一下

改进如下：

|  |
| --- |
| function addEvent(elem, type, handler){  　　if(elem.addEventListener){  　　　　elem.addEventListener(type, handler, false);  　　}else if(elem.attachEvent){  　　　　elem['temp' + type + handler] = handler;  　　　　elem[type + handler] = function(){  　　　　elem['temp' + type + handler].apply(elem);  　　};  　　elem.attachEvent('on' + type, elem[type + handler]);    }else{  　　elem['on' + type] = handler;  　　}  } |

## 16、给String对象添加一个方法，传入一个string类型的参数，然后将string的每个字符间价格空格返回，例如：

addSpace(“hello world”) // -> ‘h e l l o  w o r l d’

|  |  |
| --- | --- |
|  | String.prototype.spacify = function(){        return this.split('').join(' ');      }; |

接着上述问题答案提问，1）直接在对象的原型上添加方法是否安全？尤其是在Object对象上。(这个我没能答出？希望知道的说一下。)　2）函数声明与函数表达式的区别？

答案：在js中，解析器在向执行环境中加载数据时，对函数声明和函数表达式并非是一视同仁的，解析器会率先读取函数声明，并使其在执行任何代码之前可用（可以访问），至于函数表达式，则必须等到解析器执行到它所在的代码行，才会真正被解析执行。

## 17、定义一个log方法，让它可以代理console.log的方法。

可行的方法一：

|  |
| --- |
| function log(msg)　{      console.log(msg);  }  log("hello world!") // hello world! |

如果要传入多个参数呢？显然上面的方法不能满足要求，所以更好的方法是：

|  |
| --- |
| function log(){      console.log.apply(console, arguments);  }; |

到此，追问apply和call方法的异同。

对于apply和call两者在作用上是相同的，即是调用一个对象的一个方法，以另一个对象替换当前对象。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。

但两者在参数上有区别的。对于第一个参数意义都一样，但对第二个参数： apply传入的是一个参数数组，也就是将多个参数组合成为一个数组传入，而call则作为call的参数传入（从第二个参数开始）。 如 func.call(func1,var1,var2,var3)对应的apply写法为：func.apply(func1,[var1,var2,var3]) 。

## 18、在Javascript中什么是伪数组？如何将伪数组转化为标准数组？

伪数组（类数组）：无法直接调用数组方法或期望length属性有什么特殊的行为，但仍可以对真正数组遍历方法来遍历它们。典型的是函数的argument参数，还有像调用getElementsByTagName,document.childNodes之类的,它们都返回NodeList对象都属于伪数组。可以使用Array.prototype.slice.call(fakeArray)将数组转化为真正的Array对象。

假设接第八题题干，我们要给每个log方法添加一个”(app)”前缀，比如’hello world!’ ->’(app)hello world!’。方法如下：

|  |
| --- |
| function log(){        var args = Array.prototype.slice.call(arguments);  //为了使用unshift数组方法，将argument转化为真正的数组        args.unshift('(app)');        console.log.apply(console, args);      }; |

## 19、对作用域上下文和this的理解，看下列代码：

|  |
| --- |
| var User = {    count: 1,    getCount: function() {      return this.count;    }  };  console.log(User.getCount());  // what?  var func = User.getCount;  console.log(func());  // what? |

问两处console输出什么？为什么？

答案是1和undefined。

func是在winodw的上下文中被执行的，所以会访问不到count属性。

继续追问，那么如何确保Uesr总是能访问到func的上下文，即正确返回1。正确的方法是使用Function.prototype.bind。兼容各个浏览器完整代码如下：

|  |
| --- |
| Function.prototype.bind = Function.prototype.bind || function(context){     var self = this;     return function(){        return self.apply(context, arguments);     };  }  var func = User.getCount.bind(User);  console.log(func()); |

## 20、原生JS的window.onload与Jquery的$(document).ready(function(){})有什么不同？如何用原生JS实现Jq的ready方法？

window.onload()方法是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。

|  |
| --- |
| /\*   \* 传递函数给whenReady()   \* 当文档解析完毕且为操作准备就绪时，函数作为document的方法调用   \*/  var whenReady = (function() {               //这个函数返回whenReady()函数      var funcs = [];             //当获得事件时，要运行的函数      var ready = false;          //当触发事件处理程序时,切换为true      //当文档就绪时,调用事件处理程序      function handler(e) {          if(ready) return;       //确保事件处理程序只完整运行一次          //如果发生onreadystatechange事件，但其状态不是complete的话,那么文档尚未准备好          if(e.type === 'onreadystatechange' && document.readyState !== 'complete') {              return;          }          //运行所有注册函数          //注意每次都要计算funcs.length          //以防这些函数的调用可能会导致注册更多的函数          for(var i=0; i<funcs.length; i++) {              funcs[i].call(document);          }          //事件处理函数完整执行,切换ready状态, 并移除所有函数          ready = true;          funcs = null;      }      //为接收到的任何事件注册处理程序      if(document.addEventListener) {          document.addEventListener('DOMContentLoaded', handler, false);          document.addEventListener('readystatechange', handler, false);            //IE9+          window.addEventListener('load', handler, false);      }else if(document.attachEvent) {          document.attachEvent('onreadystatechange', handler);          window.attachEvent('onload', handler);      }      //返回whenReady()函数      return function whenReady(fn) {          if(ready) { fn.call(document); }          else { funcs.push(fn); }      }  })(); |

如果上述代码十分难懂，下面这个简化版：

|  |
| --- |
| function ready(fn){      if(document.addEventListener) {//标准浏览器          document.addEventListener('DOMContentLoaded', function() {              //注销事件, 避免反复触发              document.removeEventListener('DOMContentLoaded',arguments.callee, false);              fn();//执行函数          }, false);      }else if(document.attachEvent) {//IE          document.attachEvent('onreadystatechange', function() {              if(document.readyState == 'complete') {                  document.detachEvent('onreadystatechange', arguments.callee);                  fn();//函数执行              }          });      }  }; |

## 21、（设计题）想实现一个对页面某个节点的拖曳？如何做？（使用原生JS）

回答出概念即可，下面是几个要点

1. 给需要拖拽的节点绑定mousedown, mousemove, mouseup事件
2. mousedown事件触发后，开始拖拽
3. mousemove时，需要通过event.clientX和clientY获取拖拽位置，并实时更新位置
4. mouseup时，拖拽结束
5. 需要注意浏览器边界的情况

## 22、请实现如下功能

[](http://jbcdn2.b0.upaiyun.com/2014/10/a2e3bf5eab2f56af1010d2d0153ed19d.png)

|  |
| --- |
| function setcookie(name,value,days){  //给cookie增加一个时间变量  　　var exp = new Date();  　　exp.setTime(exp.getTime() + days\*24\*60\*60\*1000); //设置过期时间为days天  　　document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();  }  function getCookie(name){  　　var result = "";  　　var myCookie = ""+document.cookie+";";  　　var searchName = "+name+"=";  　　var startOfCookie = myCookie.indexOf(searchName);  　　var endOfCookie;  　　if(satrtOfCookie != -1){  　　　　startOfcookie += searchName.length;  　　　　endOfCookie = myCookie.indexOf(";",startOfCookie);  　　　　result = (myCookie.substring(startOfCookie,endOfCookie));  　　}  　　return result;  }  (function(){  　　var oTips = document.getElementById('tips');//假设tips的id为tips  　　var page = {  　　check: function(){//检查tips的cookie是否存在并且允许显示  　　　　var tips = getCookie('tips');  　　　　if(!tips || tips == 'show') return true;//tips的cookie不存在  　　　　if(tips == "never\_show\_again") return false;  　　},  　　hideTip: function(bNever){  　　　　if(bNever) setcookie('tips', 'never\_show\_again', 365);  　　　　oTips.style.display = "none";//隐藏  　　},  　　showTip: function(){  　　oTips.style.display = "inline";//显示，假设tips为行级元素  　　},  　　init: function(){  　　　　var \_this = this;  　　　　if(this.check()){  　　　　\_this.showTip();  　　　　setcookie('tips', 'show', 1);  　　}  　　oTips.onclick = function(){  　　　　\_this.hideTip(true);  　　};  　　}  　　};    page.init();  })(); |

## 23、说出以下函数的作用是？空白区域应该填写什么？

|  |
| --- |
| //define  (function(window){      function fn(str){          this.str=str;      }        fn.prototype.format = function(){          var arg = \_\_\_\_\_\_;          return this.str.replace(\_\_\_\_\_,function(a,b){               return arg[b]||"";        });      }      window.fn = fn;  })(window);    //use  (function(){      var t = new fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');      console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));  })(); |

答案：访函数的作用是使用format函数将函数的参数替换掉{0}这样的内容，返回一个格式化后的结果：

第一个空是：arguments

第二个空是：/\{(\d+)\}/ig

## Javascript作用域链?

理解变量和函数的访问范围和生命周期，全局作用域与局部作用域的区别，JavaScript中没有块作用域，函数的嵌套形成不同层次的作用域，嵌套的层次形成链式形式，通过作用域链查找属性的规则需要深入理解。

## 谈谈this对象的理解。

理解不同形式的函数调用方式下的this指向，理解事件函数、定时函数中的this指向，函数的调用形式决定了this的指向。

## 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个步骤，一次解析成js语句，一次执行）

## 关于事件，IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

[1].在IE中,事件对象是作为一个全局变量来保存和维护的.所有的浏览器事件,不管是用户触发的，还是其他事件,都会更新window.event对象.所以在代码中，只要调用window.event就可以获取事件对象， 再event.srcElement就可以取得触发事件的元素进行进一步处理.

[2].在FireFox中，事件对象却不是全局对象，一般情况下，是现场发生，现场使用，FireFox把事件对象自动传给事件处理程序.

关于事件的兼容性处理要熟练掌握，事件对象具体哪些属性存在兼容性问题，IE与标准事件模型事件冒泡与事件捕获的支持要理解

## 什么是闭包（closure），为什么要用它？

简单的理解是函数的嵌套形成闭包，闭包包括函数本身已经它的外部作用域

使用闭包可以形成独立的空间，延长变量的生命周期，报存中间状态值

## 29、javascript 代码中的"use strict";是什么意思 ? 使用它区别是什么？

意思是使用严格模式，使用严格模式，一些不规范的语法将不再支持

# 严格模式

链接：<http://www.ruanyifeng.com/blog/2013/01/javascript_strict_mode.html>

**全局变量显式声明**

**静态绑定**

**禁止使用with语句**

**eval中定义的变量都是局部变量**

**禁止this关键字指向全局对象**

**禁止在函数内部遍历调用栈**

严格模式下无法删除变量。只有configurable设置为true的对象属性，才能被删除

正常模式下，对一个对象的只读属性进行赋值，不会报错，只会默默地失败。严格模式下，将报错。

严格模式下，对一个使用getter方法读取的属性进行赋值，会报错。

严格模式下，对禁止扩展的对象添加新属性，会报错。

严格模式下，删除一个不可删除的属性，会报错。

正常模式下，如果对象有多个重名属性，最后赋值的那个属性会覆盖前面的值。严格模式下，这属于语法错误。

正常模式下，如果函数有多个重名的参数，可以用arguments[i]读取。严格模式下，这属于语法错误。

正常模式下，整数的第一位如果是0，表示这是八进制数，比如0100等于十进制的64。严格模式禁止这种表示法，整数第一位为0，将报错。

**不允许对arguments赋值**

**arguments不再追踪参数的变化**

**禁止使用arguments.callee**

严格模式只允许在全局作用域或函数作用域的顶层声明函数。也就是说，不允许在非函数的代码块内声明函数

严格模式新增了一些保留字：implements, interface, let, package, private, protected, public, static, yield。

## 30、如何判断一个对象是否属于某个类(严格来说在ES6之前，js没有类的概念)？

instanceof constructor

## 31、new操作符具体干了什么呢?

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 。

## 32、用原生JavaScript的实现过什么功能吗？

主要考察原生js的实践经验

## 33、Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

HasOwnProperty

## 34、对JSON的了解？

轻量级数据交互格式，可以形成复杂的嵌套格式，解析非常方便

## 35、js延迟加载的方式有哪些？

方案一：<script>标签的async="async"属性（详细参见：script标签的async属性）

方案二：<script>标签的defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用AJAX得到脚本内容，然后通过eval\_r(xmlhttp.responseText)来运行脚本）

方案五：iframe方式

## 36、模块化开发怎么做？

理解模块化开发模式：浏览器端requirejs，seajs；服务器端nodejs；ES6模块化；fis、webpack等前端整体模块化解决方案；grunt、gulp等前端工作流的使用

## 37、AMD（Modules/Asynchronous-Definition）、CMD（Common Module Definition）规范区别？

理解这两种规范的差异，主要通过requirejs与seajs的对比，理解模块的定义与引用方式

的差异以及这两种规范的设计原则

参考链接1：<https://www.zhihu.com/question/20351507/answer/14859415>

参考链接2：<https://github.com/seajs/seajs/issues/277>

1、对于依赖的模块，AMD 是提前执行，CMD 是延迟执行。不过 RequireJS 从 2.0 开始，也改成可以延迟执行（根据写法不同，处理方式不同）。CMD 推崇 as lazy as possible.

2、CMD 推崇依赖就近，AMD 推崇依赖前置。

3. AMD 的 API 默认是一个当多个用，CMD 的 API 严格区分，推崇职责单一。比如 AMD 里，require 分全局 require 和局部 require，都叫 require。CMD 里，没有全局 require，而是根据模块系统的完备性，提供 seajs.use 来实现模块系统的加载启动。CMD 里，每个 API 都简单纯粹。

## 38、requireJS的核心原理是什么？（如何动态加载的？如何避免多次加载的？如何 缓存的？）

核心是js的加载模块，通过正则匹配模块以及模块的依赖关系，保证文件加载的先后顺序，根据文件的路径对加载过的文件做了缓存

## 39、让你自己设计实现一个requireJS，你会怎么做？

核心是实现js的加载模块，维护js的依赖关系，控制好文件加载的先后顺序

## 40、谈一谈你对ECMAScript6的了解？

ES6新的语法糖，类，模块化等新特性

关于ES6参考链接：<http://es6.ruanyifeng.com/>

1. [ECMAScript 6简介](http://es6.ruanyifeng.com/" \l "docs/intro)
2. [let和const命令](http://es6.ruanyifeng.com/" \l "docs/let)
3. [变量的解构赋值](http://es6.ruanyifeng.com/" \l "docs/destructuring)
4. [字符串的扩展](http://es6.ruanyifeng.com/" \l "docs/string)
5. [正则的扩展](http://es6.ruanyifeng.com/" \l "docs/regex)
6. [数值的扩展](http://es6.ruanyifeng.com/" \l "docs/number)
7. [数组的扩展](http://es6.ruanyifeng.com/" \l "docs/array)
8. [函数的扩展](http://es6.ruanyifeng.com/" \l "docs/function)
9. [对象的扩展](http://es6.ruanyifeng.com/" \l "docs/object)
10. [Symbol](http://es6.ruanyifeng.com/" \l "docs/symbol)
11. [Proxy和Reflect](http://es6.ruanyifeng.com/" \l "docs/proxy)
12. [二进制数组](http://es6.ruanyifeng.com/" \l "docs/arraybuffer)
13. [Set和Map数据结构](http://es6.ruanyifeng.com/" \l "docs/set-map)
14. [Iterator和for...of循环](http://es6.ruanyifeng.com/" \l "docs/iterator)
15. [Generator函数](http://es6.ruanyifeng.com/" \l "docs/generator)
16. [Promise对象](http://es6.ruanyifeng.com/" \l "docs/promise)
17. [异步操作和Async函数](http://es6.ruanyifeng.com/" \l "docs/async)
18. [Class](http://es6.ruanyifeng.com/" \l "docs/class)
19. [Decorator](http://es6.ruanyifeng.com/" \l "docs/decorator)
20. [Module](http://es6.ruanyifeng.com/" \l "docs/module)

## 41、ECMAScript6 怎么写class么，为什么会出现class这种东西?

class Point {

constructor(x, y) {

this.x = x;

this.y = y;

}

toString() {

return '('+this.x+', '+this.y+')';

}

}

## 42、异步加载的方式有哪些？

方案一：<script>标签的async="async"属性（详细参见：script标签的async属性）

方案二：<script>标签的defer="defer"属性

方案三：动态创建<script>标签

方案四：AJAX eval（使用AJAX得到脚本内容，然后通过eval\_r(xmlhttp.responseText)来运行脚本）

方案五：iframe方式

## 43、documen.write和 innerHTML的区别?

document.write是重写整个document, 写入内容是字符串的html

innerHTML是HTMLElement的属性，是一个元素的内部html内容

## 44、DOM操作——怎样添加、移除、移动、复制、创建和查找节点?

（1）创建新节点

createDocumentFragment() //创建一个DOM片段

createElement\_x() //创建一个具体的元素

createTextNode() //创建一个文本节点

（2）添加、移除、替换、插入

appendChild()

removeChild()

replaceChild()

insertBefore()

（3）查找

getElementsByTagName() //通过标签名称

getElementsByName() //通过元素的Name属性的值

getElementById() //通过元素Id，唯一性

## 45、call() 和 apply() 的含义和区别？

apply的参数是数组形式，call的参数是单个的值，除此之外在使用上没有差别，重点理解这两个函数调用的this改变

## 46、数组和对象有哪些原生方法，列举一下？

Array.concat( ) 连接数组

Array.join( ) 将数组元素连接起来以构建一个字符串

Array.length 数组的大小

Array.pop( ) 删除并返回数组的最后一个元素

Array.push( ) 给数组添加元素

Array.reverse( ) 颠倒数组中元素的顺序

Array.shift( ) 将元素移出数组

Array.slice( ) 返回数组的一部分

Array.sort( ) 对数组元素进行排序

Array.splice( ) 插入、删除或替换数组的元素

Array.toLocaleString( ) 把数组转换成局部字符串

Array.toString( ) 将数组转换成一个字符串

Array.unshift( ) 在数组头部插入一个元素

**Object对象的常用方法**

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

## 47、JS 怎么实现一个类。怎么实例化这个类

严格来讲js中并没有类的概念，不过js中的函数可以作为构造函数来使用，通过new来实例化，其实函数本身也是一个对象。

## 48、JavaScript中的作用域与变量声明提升？

理解JavaScript的预解析机制，js的运行主要分两个阶段：js的预解析和运行，预解析阶段所有的变量声明和函数定义都会提前，但是变量的赋值不会提前

## 49、如何编写高性能的Javascript？

使用 DocumentFragment 优化多次 append

通过模板元素 clone ，替代 createElement

使用一次 innerHTML 赋值代替构建 dom 元素

使用 firstChild 和 nextSibling 代替 childNodes 遍历 dom 元素

使用 Array 做为 StringBuffer ，代替字符串拼接的操作

将循环控制量保存到局部变量

顺序无关的遍历时，用 while 替代 for

将条件分支，按可能性顺序从高到低排列

在同一条件子的多（ >2 ）条件分支时，使用 switch 优于 if

使用三目运算符替代条件分支

需要不断执行的时候，优先考虑使用 setInterval

## 50、那些操作会造成内存泄漏？

闭包，死循环

## 51、javascript对象的几种创建方式？

1. 工厂模式

2. 构造函数模式

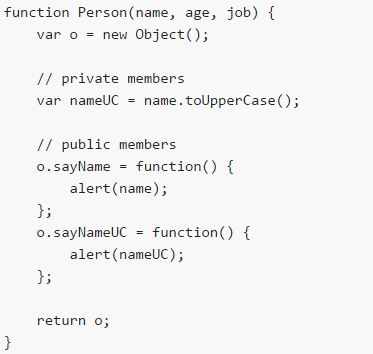
3. 原型模式

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



## 52、javascript继承的 6 种方法？

1. 原型链继承

2. 借用构造函数继承

3. 组合继承(原型+借用构造)

4. 原型式继承

5. 寄生式继承

6. 寄生组合式继承

## 53、eval是做什么的？

1. 它的功能是把对应的字符串解析成JS代码并运行

2. 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）

## 54、JavaScript 原型，原型链 ? 有什么特点？

1. 原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为 null 的话，我们就称之为原型链

2. 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

## 55、事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为

2. 事件处理机制：IE是事件冒泡、firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件

3. ev.stopPropagation();

注意旧ie的方法：ev.cancelBubble = true;

## 56、简述一下Sass、Less，且说明区别？

他们是动态的样式语言，是CSS预处理器,CSS上的一种抽象层。他们是一种特殊的语法/语言而编译成CSS。

变量符不一样，less是@，而Sass是$;

Sass支持条件语句，可以使用if{}else{},for{}循环等等。而Less不支持;

Sass是基于Ruby的，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

## 57、关于javascript中apply()和call()方法的区别？

相同点:两个方法产生的作用是完全一样的

不同点:方法传递的参数不同

Object.call(this,obj1,obj2,obj3)

Object.apply(this,arguments)

apply()接收两个参数，一个是函数运行的作用域(this)，另一个是参数数组。

call()方法第一个参数与apply()方法相同，但传递给函数的参数必须列举出来。

## 58、简述一下JS中的闭包？

闭包用的多的两个作用：读取函数内部的变量值；让这些变量值始终保存着(在内存中)。

同时需要注意的是：闭包慎用，不滥用，不乱用，由于函数内部的变量都被保存在内存中，会导致内存消耗大。

## 59、说说你对this的理解？

在JavaScript中，this通常指向的是我们正在执行的函数本身，或者是，指向该函数所属的对象。

全局的this → 指向的是Window

~~函数中的this → 指向的是函数所在的对象~~ 错误答案

对象中的this → 指向其本身

事件中this → 指向事件对象

## 60、分别阐述split(),slice(),splice(),join()？

join()用于把数组中的所有元素拼接起来放入一个字符串。所带的参数为分割字符串的分隔符，默认是以逗号分开。归属于Array

split()即把字符串分离开，以数组方式存储。归属于Stringstring

slice() 方法可从已有的数组中返回选定的元素。该方法并不会修改数组，而是返回一个子数组。如果想删除数组中的一段元素，应该使用方法 Array.splice()

splice() 方法向/从数组中添加/删除项目，然后返回被删除的项目。返回的是含有被删除的元素的数组。

## 61、事件委托是什么？

让利用事件冒泡的原理，让自己的所触发的事件，让他的父元素代替执行！

## 62、如何阻止事件冒泡和默认事件？

阻止浏览器的默认行为

window.event?window.event.returnValue=false:e.preventDefault();

停止事件冒泡

window.event?window.event.cancelBubble=true:e.stopPropagation();

原生JavaScript中，return false;只阻止默认行为，不阻止冒泡，jQuery中的return false;既阻止默认行为，又阻止冒泡

## 63、添加 删除 替换 插入到某个接点的方法？

obj.appendChidl()

obj.removeChild()

obj.replaceChild()

obj.innersetBefore()

## 64、你用过require.js吗？它有什么特性？

（1）实现js文件的异步加载，避免网页失去响应；

（2）管理模块之间的依赖性，便于代码的编写和维护。

## 65、谈一下JS中的递归函数，并且用递归简单实现阶乘？

递归即是程序在执行过程中不断调用自身的编程技巧，当然也必须要有一个明确的结束条件，不然就会陷入死循环。

## 66、请用正则表达式写一个简单的邮箱验证。

/^[a-zA-Z0-9\_-]+@[a-zA-Z0-9\_-]+(\.[a-zA-Z0-9\_-]+)+$/;

## 67、简述一下你对web性能优化的方案？

1、尽量减少 HTTP 请求

2、使用浏览器缓存

3、使用压缩组件

4、图片、JS的预载入

5、将脚本放在底部

6、将样式文件放在页面顶部

7、使用外部的JS和CSS

8、精简代码

## 68、在JS中有哪些会被隐式转换为false

Undefined、null、布尔值false、NaN、零、空字符串

## 69、定时器setInterval有一个有名函数fn1，setInterval（fn1,500）与setInterval（fn1(),500）有什么区别？

第一个是重复执行每500毫秒执行一次，后面一个只执行一次。

## 70、外部JS文件出现中文字符，会出现什么问题，怎么解决？

会出现乱码，加charset=”GB2312”;

另一种解决方式：网页文件和外部JS文件都是UTF8编码

## 71、谈谈浏览器的内核，并且说一下什么是内核？

Trident (['traɪd(ə)nt])--IE，Gecko (['gekəʊ])--Firefox, Presto (['prestəʊ])--opera,webkit—谷歌和Safari

浏览器内核又可以分成两部分：渲染引擎和 JS 引擎。它负责取得网页的内容（HTML、XML、图像等等）、整理讯息（例如加入 CSS 等），以及计算网页的显示方式，然后会输出至显示器或打印机。JS 引擎则是解析 Javascript 语言，执行 javascript 语言来实现网页的动态效果。

## 72、JavaScript原型，原型链 ? 有什么特点？

\* 原型对象也是普通的对象，是对象一个自带隐式的 \_\_proto\_\_ 属性，原型也有可能有自己的原型，如果一个原型对象的原型不为null的话，我们就称之为原型链。

\* 原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链。

\* JavaScript的数据对象有那些属性值？

　　writable：这个属性的值是否可以改。

　　configurable：这个属性的配置是否可以删除，修改。

　　enumerable：这个属性是否能在for…in循环中遍历出来或在Object.keys中列举出来。

　　value：属性值。

\* 当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，就会查找他的Prototype对象是否有这个属性。

function clone(proto) {

　　function Dummy() { }

　　Dummy.prototype = proto;

　　Dummy.prototype.constructor = Dummy;

　　return new Dummy(); //等价于Object.create(Person);

}

function object(old) {

function F() {};

F.prototype = old;

return new F();

}

var newObj = object(oldObject);

## 73、写一个通用的事件侦听器函数

`// event(事件)工具集，来源：https://github.com/markyun

markyun.Event = {

// 页面加载完成后

readyEvent : function(fn) {

if (fn==null) {

fn=document;

}

var oldonload = window.onload;

if (typeof window.onload != 'function') {

window.onload = fn;

} else {

window.onload = function() {

oldonload();

fn();

};

}

},

// 视能力分别使用dom0||dom2||IE方式 来绑定事件

// 参数： 操作的元素,事件名称 ,事件处理程序

addEvent : function(element, type, handler) {

if (element.addEventListener) {

//事件类型、需要执行的函数、是否捕捉

element.addEventListener(type, handler, false);

} else if (element.attachEvent) {

element.attachEvent('on' + type, function() {

handler.call(element);

});

} else {

element['on' + type] = handler;

}

},

// 移除事件

removeEvent : function(element, type, handler) {

if (element.removeEnentListener) {

element.removeEnentListener(type, handler, false);

} else if (element.datachEvent) {

element.detachEvent('on' + type, handler);

} else {

element['on' + type] = null;

}

},

// 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

stopPropagation : function(ev) {

if (ev.stopPropagation) {

ev.stopPropagation();

} else {

ev.cancelBubble = true;

}

},

// 取消事件的默认行为

preventDefault : function(event) {

if (event.preventDefault) {

event.preventDefault();

} else {

event.returnValue = false;

}

},

// 获取事件目标

getTarget : function(event) {

return event.target || event.srcElement;

},

// 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

getEvent : function(e) {

var ev = e || window.event;

if (!ev) {

var c = this.getEvent.caller;

while (c) {

ev = c.arguments[0];

if (ev && Event == ev.constructor) {

break;

}

c = c.caller;

}

}

return ev;

}

};

## 74、事件、IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

2. 事件处理机制：IE是事件冒泡、火狐是 事件捕获；

3. ev.stopPropagation();

## 75、什么是闭包（closure），为什么要用？

执行say667()后,say667()闭包内部变量会存在,而闭包内部函数的内部变量不会存在.使得Javascript的垃圾回收机制GC不会收回say667()所占用的资源，因为say667()的内部函数的执行需要依赖say667()中的变量。这是对闭包作用的非常直白的描述.

function say667() {

// Local variable that ends up within closure

var num = 666;

var sayAlert = function() { alert(num); }

num++;

return sayAlert;

}

var sayAlert = say667();

sayAlert()//执行结果应该弹出的667

## 76、如何判断一个对象是否属于某个类？

使用instanceof （待完善）

if(a instanceof Person){

alert('yes');

}

## 77、new操作符具体干了什么呢?

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 。

var obj = {};

obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

## 78、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

{'age':'12', 'name':'back'}

## 79、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

## 80、模块化怎么做？

立即执行函数,不暴露私有成员

1. 使用字面量实现命名空间(YUI)：

Itcast.common.dom={};

Itcast.common.css={};

Itcast.common.event={};

2、使用闭包

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();

## 81、异步加载的方式

(1) defer，只支持IE

(2) async：

(3) 创建script，插入到DOM中，加载完毕后callBack

documen.write和 innerHTML的区别

document.write只能重绘整个页面

innerHTML可以重绘页面的一部分

## 82、告诉我答案是多少？

(function(x){

delete x;

alert(x);

})(1+5);

函数参数无法delete删除，delete只能删除通过for in访问的属性。

当然，删除失败也不会报错，所以代码运行会弹出“6”。

## 83、JS中的call()和apply()方法的区别？

例子中用 add 来替换 sub，add.call(sub,3,1) == add(3,1) ，所以运行结果为：alert(4);

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b){

alert(a+b);

}

function sub(a,b){

alert(a-b);

}

add.call(sub,3,1);

## 84、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

## 85、jquery 中如何将数组转化为json字符串，然后再转化回来？

jQuery中没有提供这个功能，所以你需要先编写两个jQuery的扩展：

$.fn.stringifyArray = function(array) {

return JSON.stringify(array)

}

$.fn.parseArray = function(array) {

return JSON.parse(array)

}

然后调用：

$("").stringifyArray(array)

## 86、JavaScript中的作用域与变量声明提升？

其他部分

（HTTP、正则、优化、重构、响应式、移动端、团队协作、SEO、UED、职业生涯）

\*基于Class的选择性的性能相对于Id选择器开销很大，因为需遍历所有DOM元素。

\*频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。

比如：var str=$("a").attr("href");

\*for (var i = size; i < arr.length; i++) {}

for 循环每一次循环都查找了数组 (arr) 的.length 属性，在开始循环的时候设置一个变量来存储这个数字，可以让循环跑得更快：

for (var i = size, length = arr.length; i < length; i++) {}

## 87、前端开发的优化问题。

参考资料：J:\代码,PPT,笔记,电子书\面试题\雅虎14条优化规则.docx

（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。

## 88、http状态码有那些？分别代表是什么意思？

100-199 用于指定客户端应相应的某些动作。

200-299 用于表示请求成功。

300-399 用于已经移动的文件并且常被包含在定位头信息中指定新的地址信息。

400-499 用于指出客户端的错误。

400 语义有误，当前请求无法被服务器理解。

401 当前请求需要用户验证

403 服务器已经理解请求，但是拒绝执行它。

500-599 用于支持服务器错误。

503 – 服务不可用

## 89、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）

要熟悉前后端的通信流程，最好把动态网站的背后细节也介绍一遍