## 1、javascript的typeof返回哪些数据类型

object number function boolean underfind string

typeof isNaN

**检测数组的几种方式：**

Array.isArray(); es5

toString.call([]);//”[object Array]”

var arr=[];

arr.**constructor**;//Array

instanceof Array

## 2、传统事件绑定和符合W3C标准的事件绑定有什么区别？

<div onclick=””>123</div>

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

**符合W3C标准的事件绑定的方式 addEventListener/attachEvent**

A、非IE浏览器：

addEventListenerww

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前缀**

addEventListener(“click”,function(){},true);//此时的事件就是在**事件捕获阶段**执行

第三个参数：代表是否在捕获阶段执行，默认值是false

addEventListener(“click”,function(e){

})

addEventListener(“click”,function(){},false) 事件在冒泡阶段执行

B、IE浏览器

ie9开始，ie11 edge：addEventListener

ie9以前 ie8：attachEvent/detachEvent

1. 进行事件类型传参需要带上on前缀

dom1.attachEvent(“onclick”,functioin(){});

1. 这种方式只支持事件冒泡，不支持事件捕获

## 3、IE和标准下有哪些兼容性的写法

ev = ev || window.event 🡺获取触发事件的对象

var target = ev.srcElement||ev.target 🡪获取事件的源对象

document.documentElement.clientWidth || document.body.clientWidth

## 4、call和apply的区别

call和apply相同点：

都是为了用一个本不属于一个对象的方法，让这个对象去执行

toString.call([],1,2,3)

toString.apply([],[1,2,3])

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

Object.apply(this,arguments)

区别：call第二个参数开始接受一个参数列表

apply第二个参数开始接受一个参数数组

## 5、Javascript如何实现继承？

原型链继承，借用构造函数继承，原型+构造函数组合继承，寄生式继承

## 6、Javascript创建对象的几种方式？

第一种：对象字面量、Object构造函数

第二种：构造函数

第三种：纯构造函数

第四种：空构造函数+原型

第五种：混合构造函数+原型

第六种：动态原型

第七种：寄生构造函数

第八种：Object.create()——ES5 ie9+

以下是寄生构造函数模式：

——其实是工厂模式的变种，只不过是放在其他构造函数中创建对象

代码如下：

function SpecialArray(){  
 var values = [];  
 values.push.apply(values, arguments);  
 values.toPipedString = function(){  
 return this.join("|");  
 };  
 return values;  
}  
var *colors* = new SpecialArray("red", "blue", "green");  
alert(*colors*.toPipedString()); //"red|blue|green"

## 7、JavaScript this指针、闭包、作用域

this：指向调用上下文

闭包：内层作用域可以访问外层作用域的变量

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

## 7、如何阻止事件冒泡和默认事件

e. stopPropagation();//标准浏览器阻止事件冒泡

event.canceBubble=true;//ie9之前

阻止默认事件：

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

return false

e.preventDefault();

## 8、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

协议：http,ftp https

主机名；localhost 127.0.0.1

端口名：80:

127.0.0.1:1011

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

**JSONP**的实现原理：通过动态创建script标签





## 9、编写一个数组去重的方法

[1,3,1,5,2,3,7]=>[1,3,5,2,7]

方法1：

1. 先创建一个空数组，用来保存最终的结果
2. 循环原数组中的每个元素
3. 再对每个元素进行二次循环，判断是否有与之相同的元素，如果没有，将把这个元素放到新数组中
4. 返回这个新数组

方法2：

**数组的indexOf()**方法返回给定元素能找在数组中找到的第一个索引值，否则返回-1。

Array.prototype.unique2 = function()

{

var n = []; //一个新的临时数组

for(var i = 0; i < this.length; i++) //遍历当前数组

{

//如果当前数组的第i已经保存进了临时数组，那么跳过，

//否则把当前项push到临时数组里面

if (n.indexOf(this[i]) == -1) n.push(this[i]);

}

return n;

}

## 10、JavaScript是一门什么样的语言，它有哪些特点？

没有标准答案。

运行环境：浏览器中的JS引擎（v8.。。 ff:splidemonkey）

运行在node.js：v8引擎上

语言特性：面向对象（创建对象的多种方式，继承的多种方式、原型链），动态语言

//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，或者说是一个弱类型语言  
//静态语言：c# java c c++  
//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，

int c;//c就是整数类型//我们把这种声明一个变量就确定了它的数据类型这样的语言称之为静态语言，也称之为强类型语言  
// 而且在任何时候都不可以改变他的数据类型

## 11、如何检测数组的数据类型

方法一、判断其是否具有“数组性质”，如slice()方法。可自己给该变量定义slice方法，故有时会失效

方法二、obj instanceof Array 在某些IE版本中不正确

方法三、toString.call([]);//”[object Array]”

方法四、方法一二皆有漏洞，在ECMA Script5中定义了新方法Array.isArray(), 保证其兼容性，最好的方法如下：

if(typeof Array.isArray==="undefined"){

  Array.isArray = function(arg){

        return Object.prototype.toString.call(arg)==="[object Array]"

    };

}

## 12、希望获取到页面中所有的checkbox怎么做

|  |  |
| --- | --- |
|  | var domList = document.getElementsByTagName(‘input’)  var checkBoxList = [];//返回的所有的checkbox  var len = domList.length;　　//缓存到局部变量  while (len--) {　　0  　　if (domList[len].type == ‘checkbox’) {      　　checkBoxList.push(domList[len]);  　　}  } |

## 13、Javascript的事件流模型都有什么？ DOM事件流

“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

“DOM事件流”：三个阶段：事件捕捉，目标阶段，事件冒泡

## 14、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” |

**undefined**会在以下三种情况下产生：

a、一个变量定义了却没有被赋值

b、想要获取一个对象上不存在的属性或者方法:

c、一个数组中没有被赋值的元素

## 15、看代码给答案。

|  |  |
| --- | --- |
|  | var a = new Object();  a.value = 1;  b = a;  b.value = 2;  alert(a.value);//2 |

答案：2（考察引用数据类型细节）

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

1) 实现对该数组的倒排，反转，输出[5,1,4,2,6,3]

reverse()

2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

冒泡排序

## 17、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 //今年：2016  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); |

## 18、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’);

## 19、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**

答案：if(!foo) foo = bar; //如果foo存在，值不变，否则把bar的值赋给foo。

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。

注意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**

## 20、看下列代码，将会输出什么?( 变量声明提前

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

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

## 21、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}。

## 22、

### a、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。

原因：setTimeout会在js引擎空闲的时候再执行

Javascript事件处理器在线程空闲之前不会运行。追问，

### b、如果让a中代码输出1 2 3？

|  |  |
| --- | --- |
|  | for(var i=1;i<=3;i++){     setTimeout((function(a){  //改成立即执行函数         console.log(a);     })(i),0);  }; |

## 23、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+$/,"");  //\s匹配**空白字符**：回车、换行、制表符tab 空格   }  }   // test the function  var str = " \t\n test string ".trim();  alert(str == "test string"); // alerts "true" |

## 24、Javascript中callee和caller的作用？

arguments.callee：获得当前函数的引用

caller是返回一个对函数的引用，该函数调用了当前函数；如果不是由其他函数调用的，那么返回值就是null

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。

那么问题来了？

**如果一对兔子每月生一对兔子；一对新生兔，从第二个月起就开始生兔子；假定每对兔子都是一雌一雄，试问一对兔子，第n个月能繁殖成多少对兔子？（使用callee完成）**

## 25、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 = [];(**这里是一个数字0**)C、var obj = //; D、var obj = {};

## 26、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 **undefined**

B、var a=‘123’; b=NaN //字符串

C、var a =undefined , b =NaN

D、var a=NaN , b='undefined' //”NaNundefined”

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

A、34 B、35 C、36 D、37

//21+4+10

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

## 29、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\_\_\_\_\_\_ 。

## 30、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++; //一开始：a=12

//程序执行时：13+21+30+13：细心

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);

## 31、写出程序运行的结果？

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

k = i + j;}

结果：10

for(i=0, j=0; i<10, j<6; i++, j++){  
 //终止条件：j=5 i=5  
 k = i + j;//k=10  
}  
//结果：10

## 32、阅读以下代码，请分析出结果

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

arr[4]='z';//[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);

弹出提示对话框：z,,5,3,1,z,,5,3,1

**考点：**reverse 方法1、颠倒数组中元素的位置，

2、并返回该数组的引用。

## 33、给一个按钮绑定点击事件的函数，确认用户是否退出当前页面，确认之后关闭窗口

confirm

window.close()

## 34、定义一个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。

<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>

## 35、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()

## 36、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

Document.getElementById 根据元素id查找元素

Document.getElementByName 根据元素name查找元素

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

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

## 38、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：

function(){}:只能自己执行自己

第三种（函数对象方式）：了解

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

## 41、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

## 42、内存泄露是什么？

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。

## 43、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)]){ //str.charAt(i) =”a”  
                json[str.charAt(i)] = 1; //json[“a”]=1  
        }else{  
                json[str.charAt(i)]++; //让a键的值+1，也就是这个字符多出现了一次  
        }  
};  
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');

## 44、写一个获取非行间样式的函数 //内嵌、外联，行内

function getStyle(obj,attr,value)

{

if(!value)

{

**if(obj.currentStyle)//ie**

**{**

**return obj.currentStyle(attr);**

**}**

**else{//标准浏览器**

**obj.getComputedStyle(attr,false);**

**}**

}

else

{

obj.style[attr] = value;

}

}

## 45、解释jsonp的原理

动态创建script标签，给请求的地址中添加一个get参数，这个参数代表回调函数，也就是希望服务器处理完我的请求之后，在前端执行这个对调函数

## 46、javascript的本地对象，内置对象和宿主对象

本地对象为array obj regexp等可以new实例化

Array RegExp String Boolean 。。。

内置对象为global、Math 等不可以实例化的

关于global对象我们无法在浏览器中发现她的存在，因为他的属性和方法都被绑定在了window对象中

每一个宿主对象他的实现都是取决于不同的浏览器的，这样就产生了浏览器兼容性问题

宿主为浏览器自带的document,window 等

## 47、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 jion() 转换为字符串  var str = '12345678';  str = str.split('').reverse().**join**(''); |

## 48、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

## 49、window.location.hash 返回的是什么？

答：锚点 ， 返回值：#love ；

//url:http://www.sina.com/getage?#age

这时就返回”#age”

## 50、window.location.reload() 作用？

    答：刷新当前页面。

## 51、看题做答：

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

## 52、下面输出多少？

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

## 53、下面代码输出什么

function changeObjectProperty (o) {

//var o;

//o=CSSer;

    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);  //” http://www.csser.com/”

## 54、观看代码，有什么输出

|  |  |
| --- | --- |
|  | 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，  }, 0);  a = 66;  console.log(a); //先打印全局变量a的值：66 再执行setTimeout里面的局部变量 |

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 。

## 55、看输出结果

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

## 56、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 |

## 57、观察以下代码的输出

|  |  |
| --- | --- |
|  | 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 |

## 58、结果是什么？

|  |  |
| --- | --- |
|  | function foo(){      foo.a = function(){alert(1)};      this.a = function(){alert(2)}; //obj.a      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 |

## 59、分析代码结果

var a = 5;

function test(){

   var a;

  a = 0;

    alert(a);

    alert(this.a); //没有定义 a这个属性

        alert(a)

}

test(); // 0, 5, 0

var obj=new test(); // 0, undefined, 0 //由于类它自身没有属性a， 所以是undefined

## 60、结果是

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

技巧：双向非操作可以把字符串和数字转换为布尔值。

## 61、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

|  |  |
| --- | --- |
|  | var reg = /^[a-zA-Z\_][a-zA-Z0-9\_]{4,19}/, |

## 62、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；

## 63、JS中的简单继承

——>**call方法**！

|  |  |
| --- | --- |
|  | //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(); |

## 64、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。

最佳实现：jquery 1.7之后：就推荐使用on() off()

## 65、简述link和import的区别？

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。

## 66、看下列代码输出什么？

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

执行完后foo的值为111，foo的类型为Number。

## 67、看下列代码,输出什么？

var a = new Object();

a.value = 1;

b = a;

b.value = 2;

alert(a.value);

执行完后输出结果为2

## 68、你如何优化自己的代码？

代码重用：声明函数把整个过程封装起来；把他们弄成一些对象

避免全局变量（命名空间，封闭空间，模块化mvc..） angularjs

拆分函数避免函数过于臃肿：单一职责原则：条理更加清晰，**更好的维护**

**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

内存管理，尤其是闭包中的变量释放

变量的命名规范：尤其是在实际项目中

$div：代表一个jquery对象

dom：代表一个原生的dom对象

希望一个函数是构造函数：函数的首字母都要大写

变量一定要**语义化**：

## 69、请描述出下列代码运行的结果

function d(){

console.log(this);

}

d();//window

## 70、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");

## 71、请用原生js实现jquery的get\post功能，以及跨域情况下

ajax

先创建一个HttpRequest()

再设置他的请求头：请求的数据类型：json/xml

再设置他的onreadstatechange

最后再提交这个请求

## 72、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（雪碧图、精灵图），

//2、合理的设置缓存 localStorage cookie

//3、资源合并、压缩 html代码的最小化，压缩体积：就是减少带宽

// 一些小的js文件合并成大的js文件

//将外部的js文件置底

## 73、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

## 74、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

异步是为了进行局部刷新，提升用户体验

2005 年诞生ajax

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。

## 75、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

//如果说是整个项目中都适用的功能，那么可以进行扩展

## 76、什么是三元表达式？“三元”表示什么意思？

? :

因为运算符会涉及3个表达式

var a=3>2?5:2;

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

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

## 78、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一般用于简单的数据查询，严格要求不是那么高的场景

## 79、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。

## 80、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？ |

## 81、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 |

## 82、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*

*jquery mobile：是jquery自己出的支持移动端网页开发，不过非常笨重，但是功能非常强大*

*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(去哪儿前端总监，作者：司徒正美)：MV\*框架，适合用于单页应用开发(SPA)*

## 83、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）

**Object.keys**——IE9+ ES5

或者使用**for…in**并过滤出继承的属性

for(o in obj){

if(obj.hasOwnproperty(o)){//判断o不是继承来的属性

//把o这个属性放入到一个数组中

}

}

## 84、下面这段代码想要循环昝输出结果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

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

setTimeout(function(){

console.log(i+’’);

},100\*i);

}

## 85、解释下这个css选择器什么发生什么？

[role=nav]>ul a:not([href^-mailto]){}

## 86、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

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

## 87、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

D: link方式的样式的权重高于@import的权重

import诞生于CSS2.1规范，任何支持CSS2.1以上版本的浏览器都支持Import

link方式的样式跟import没有权重高低这么个说法，只有书写顺序的不同才会决定样式

## 88、变量的命名规范以及命名推荐

## 89、三种弹窗的单词以及三种弹窗的功能

alert

confirm

prompt

## 91、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  
 }  
});

## 90、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。

## 92、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*);  
}

## 93、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

## 94、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。

## 95、《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}/ 010 021 020 025 028 0755：深圳

## 96、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>

## 97、下面代码会有什么样的输出

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());

答案：1 undefined

## 98、下面代码的输出：

(function test(){

var a=b=5; **//b是一个隐式的全局变量**

alert(typeof a);

alert(typeof b);

})();

alert(typeof a);

alert(typeof b);

//number number undefined number

## 99、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 5

i=7 6

i=8 7

i=9 8

## 100、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 {//b=NaN

console.log(!b \* 2 + "2" - 0 + 4); //1\*2+”2”-0+4

}

答案：26

考点：**乘号运算符的运算规则**：

1. 如果其中一个数是NaN，则结果就是NaN

2、如果乘积超过了ECMAScript设定的范围，那么就会返回Infinity、-Infinity

3、**如果参与乘法运算的某个操作数不是数值，js引擎会先调用Number()将这个数变成一个数值类型，**

比如：空字符串就会变成0 布尔值true就会变成1，空数组[]也会变成0，undefined转换结果则是变成了NaN

## 101、以下代码会有什么输出

<script>

var a = 1;

</script>

<script>

var a;

var b = a \* 0;

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

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

} else {

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

}

</script>

答案：6

## 102、观察输出结果

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

## 103、输出结果：

var t = 10;

function test(test){

var t = test++;

}test(t);

console.log(t);

答案：10

## 104、输出结果：

var t = 10;

function test(test){

t = test++;//这一步全局变量t的值等于改变前的test的值

}test(t);

console.log(t);

答案：10

## 105、输出结果：

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

## 106、输出结果：

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

## 107:、输出结果：

<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

## 108、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

## 109、☆☆☆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

## 110、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

## 111、程序中捕获异常的方法？

window.error=function(){};

try{}catch(){}finally{}

## 112、给字符串扩展一个兼容所有浏览器的清除前后的空格的方法

if (!String.prototype.trim) {

String.prototype.trim = function() {

return this.replace(/^\s+/, "").replace(/\s+$/,"");

} }

## 113、下列控制台都输出什么

function setName(){

name="张三";

}

setName();

console.log(name);

答案："张三"

## 114、

//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

## 115、

var b=2;

function test2(){

window.b=3;

console.log(b);

}

test2();

答案：3

## 116、

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();

## 117：

var arr = [];

arr[0] = 'a';

arr[1] = 'b';

arr[10] = 'c';

alert(arr.length); //答案：11

console.log(arr[5]); //答案：undefined

## 118：

var a=1;

console.log(a++); //答案：1

console.log(++a); //答案：3

## 119：

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

## 120、

parseInt(3.14); //3

parseFloat("3asdf"); //3

parseInt("1.23abc456");

**parseInt(true);// "true"=> NaN**

//对一个非数字或者一个非字符串类型进行数据转换的时候，会先把这个数据转换为一个字符串类型，然后再转换为数值类型

考点：考察非数字转换为整数的熟练度

## 121、

//考点：函数声明提前

function bar() {

return foo;

foo = 10;

function foo() {}

//var foo = 11;

}

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

## 122、

考点：函数声明提前

var foo = 1;

function bar() {

foo = 10;

return;

function foo() {}

}

bar();

alert(foo);//答案：1

## 123、

console.log**(a);//是一个函数**

var a = 3;

function a(){}

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

## 124、

//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);

## 125、

function foo(a) {

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

}

foo(1, 2, 3);

## 126、

bar();//报错 因为bar这个名称只能在函数内部被访问

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

## 127、

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";

}

## 128、jQuery 的 slideUp动画 ，如果目标元素是被外部事件驱动, 当鼠标快速地连续触发外部元素事件, 动画会滞后的反复执行，该如何处理呢?

先stop(true,true)后slideUp()

示例代码参考：J:\代码,PPT,笔记,电子书\面试题\面试题\_jquery\_slideup.html

关于stop()参考：J:\代码,PPT,笔记,电子书\面试题\面试题\_jquery.png

## 129、Ajax 是什么? 如何创建一个Ajax？

Ajax并不算是一种新的技术，全称是asynchronous javascript and xml，可以说是已有技术的组合，主要用来实现客户端与服务器端的异步通信效果，实现页面的局部刷新，早期的浏览器并不能原生支持ajax，可以使用隐藏帧（iframe）方式变相实现异步效果，后来的浏览器提供了对ajax的原生支持

使用ajax原生方式发送请求主要通过XMLHttpRequest(标准浏览器)、ActiveXObject(IE浏览器)对象实现异步通信效果

基本步骤：

var xhr =null;//创建对象

if(window.XMLHttpRequest){

xhr = new XMLHttpRequest();

}else{

xhr = new ActiveXObject("Microsoft.XMLHTTP");

}

xhr.open(“方式”,”地址”,”标志位”);//初始化请求

xhr.setRequestHeader(“”,””);//设置http头信息

xhr.onreadystatechange =function(){}//指定回调函数

xhr.send();//发送请求

## 130、同步和异步的区别?

同步：阻塞的

-张三叫李四去吃饭，李四一直忙得不停，张三一直等着，直到李四忙完两个人一块去吃饭

=浏览器向服务器请求数据，服务器比较忙，浏览器一直等着（页面白屏），直到服务器返回数据，浏览器才能显示页面

异步：非阻塞的

-张三叫李四去吃饭，李四在忙，张三说了一声然后自己就去吃饭了，李四忙完后自己去吃

=浏览器向服务器请求数据，服务器比较忙，浏览器可以自如的干原来的事情（显示页面），服务器返回数据的时候通知浏览器一声，浏览器把返回的数据再渲染到页面，局部更新

## 131、GET和POST的区别，何时使用POST？

GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符，有的浏览器是8000个字符

POST：一般用于修改服务器上的资源，对所发送的信息没有限制

在以下情况中，请使用 POST 请求：

1. 无法使用缓存文件（更新服务器上的文件或数据库）

2. 向服务器发送大量数据（POST 没有数据量限制）

3. 发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

## 132、ajax的缺点

1、ajax不支持浏览器back按钮。

2、安全问题 AJAX暴露了与服务器交互的细节。

3、对搜索引擎的支持比较弱。//不会执行你的js脚本，只会操作你的网页源代码

4、破坏了程序的异常机制。

5、无法跨域请求，解决方式：jsonp

## 133、http常见的状态码有那些？分别代表是什么意思？

200 OK //客户端请求成功

400 Bad Request //客户端请求有语法错误，不能被服务器所理解

403 Forbidden //服务器收到请求，但是拒绝提供服务

404 Not Found //请求资源不存在，输入了错误的URL

500 Internal Server Error //服务器发生不可预期的错误

503 Server Unavailable //服务器当前不能处理客户端的请求，一段时间后可能恢复正常

## 134、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 arguments.callee**

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

if else while for 无法声明函数

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

## 135、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 。

## 136、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

## 137、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。

## 138、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

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

## 140、javascript对象的几种创建方式？

1. 工厂模式

2. 构造函数模式

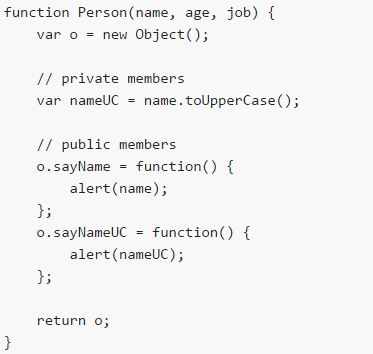
3. 原型模式

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



## 141、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

## 142、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

## 143、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

## 144、模块化怎么做？

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

立即执行函数,不暴露私有成员

var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){

　　　　　　//...

　　　　};

　　　　var m2 = function(){

　　　　　　//...

　　　　};

　　　　return {

　　　　　　m1 : m1,

　　　　　　m2 : m2

　　　　};

　　})();