前端面试题汇总

JS 基础

1. javascript 的 typeof 返回哪些数据类型

```
 alert(typeof [1, 2]); //object
alert(typeof 'leipeng'); //string
var i = true;
alert(typeof i); //boolean
alert(typeof 1); //number
var a;
alert(typeof a); //undefined
function a(){;};
alert(typeof a) //function
```

2. 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（== ,!!）

3. 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

4. 数组方法 pop() push() unshift() shift()

Push()尾部添加 pop()尾部删除

Unshift()头部添加 shift()头部删除

5. 事件绑定和普通事件有什么区别

普通添加事件的方法：

```
var btn = document.getElementById("hello");
btn.onclick = function(){
    alert(1);
}
btn.onclick = function(){
    alert(2);
}
```

执行上面的代码只会 alert 2

事件绑定方式添加事件：

```
var btn = document.getElementById("hello");
btn.addEventListener("click",function(){
    alert(1);
},false);
btn.addEventListener("click",function(){
    alert(2);
},false);
```

执行上面的代码会先 alert 1 再 alert 2

普通添加事件的方法不支持添加多个事件，最下面的事件会覆盖上面的，而事件绑定（addEventListener）方式添加事件可以添加多个。

addEventListener 不兼容低版本 IE

普通事件无法取消

addEventLisntener 还支持事件冒泡+事件捕获

6. IE 和 DOM 事件流的区别

1.执行顺序不一样、

2.参数不一样

3.事件加不加 on

4.this 指向问题

7. IE 和标准下有哪些兼容性的写法

Var ev = ev || window.event

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

Var target = ev.srcElement||ev.target

8. call 和 apply 的区别

call 方法:

语法：call(thisObj，Object1,Object2...)

定义：调用一个对象的一个方法，以另一个对象替换当前对象。

说明：

call 方法可以用来代替另一个对象调用一个方法。call 方法可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。

如果没有提供 thisObj 参数，那么 Global 对象被用作 thisObj。

apply 方法：

语法：apply(thisObj，[argArray])

定义：应用某一对象的一个方法，用另一个对象替换当前对象。

说明：

如果 argArray 不是一个有效的数组或者不是 arguments 对象，那么将导致一个 TypeError。

如果没有提供 argArray 和 thisObj 任何一个参数，那么 Global 对象将被用作 thisObj，并且无法被传递任何参数。

9. b 继承 a 的方法

```
function A( age, name ){
  this.age = age;
  this.name = name;
}

A.prototype.show = function(){
  alert('父级方法');
}

function B(age,name,job){
  A.apply( this, arguments );
  this.job = job;
}

B.prototype = new A();
var b = new A(14,'侠客行');
var a = new B(15,'狼侠','侠客');
```

10. 如何阻止事件冒泡和默认事件

canceBubble()只支持 IE,return false,stopPropagation()

11. 添加 删除 替换 插入到某个接点的方法

obj.appendChid()

obj.insertBefore()

obj.replaceChild()

obj.removeChild()

12. 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

内置对象为 onload Math 等不可以实例化的

宿主为浏览器自带的 document,window 等

13. window.onload 和 document ready 的区别

window.onload 是在 dom 文档树加载完和所有文件加载完之后执行一个函数 Document.ready 原生种没有这个方法，jquery 中有 $().ready(function),在 dom 文档树加载完之后执行一个函数（注意，这里面的文档树加载完不代表全部文件加载完）。

$(document).ready 要比 window.onload 先执行

window.onload 只能出来一次，$(document).ready 可以出现多次

14. ”==”和“===”的不同

前者会自动转换类型 只比较值

后者不会 既比较值又比较类型

15. 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、和端口号的组合

16. JavaScript 是一门什么样的语言，它有哪些特点？

没有标准答案。 基于对象的脚本语言

javaScript 一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型。它的解释器被称为 JavaScript 引擎，为浏览器的一部分，广泛用于客户端的脚本语言，

最早是在 HTML 网页上使用，用来给 HTML 网页增加动态功能。JavaScript 兼容于 ECMA 标准，因此也称为 ECMAScript。

基本特点

1．是一种解释性脚本语言（代码不进行预编译）。

2．主要用来向 HTML（标准通用标记语言下的一个应用）页面添加交互行为。

3．可以直接嵌入 HTML 页面，但写成单独的 js 文件有利于结构和行为的分离。

4．跨平台特性，在绝大多数浏览器的支持下，可以在多种平台下运行（如 Windows、Linux、Mac、Android、iOS 等）。

17. JavaScript 的数据类型都有什么？

基本数据类型：String,boolean,Number,Undefined, Null

引用数据类型：Object(Array,Date,RegExp,Function)

那么问题来了，如何判断某变量是否为数组数据类型？

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.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]"
    };
}
```

18. 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
```

19. 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--) {      //使用 while 的效率会比 for 循环更高
    if (domList[len].type ==  ‘checkbox’ ) {
        checkBoxList.push(domList[len]);
    }
}
```

20. 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”
```

21. 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？

直接在 DOM 里绑定事件：

在 JS 里通过 onclick 绑定：xxx.onclick = test

通过事件添加进行绑定：addEventListener(xxx, ‘click’, test)

那么问题来了，Javascript 的事件流模型都有什么？

“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

22. 看下列代码输出为何？解释原因。

```
var a;
alert(typeof a); // undefined
alert(b); // 报错
```

解释：Undefined 是一个只有一个值的数据类型，这个值就是“undefined”，在使用 var 声明变量但并未对其赋值进行初始化时，这个变量的值就是 undefined。而 b 由于未声明将报错。注意未申明的变量和声明了未赋值的是不一样的。

23. 看下列代码,输出什么？解释原因。

```
var a = null;
alert(typeof a); //object
```

解释：null 是一个只有一个值的数据类型，这个值就是 null。表示一个空指针对象，所以用 typeof 检测会返回“object”。

24. 看下列代码,输出什么？解释原因。

```
var undefined;
undefined == null; // true
1 == true;  // true
2 == true;  // false
0 == false;  // true
0 == '';  // true
NaN == NaN;  // false
[] == false; // true
[] == ![];  // true
```

- undefined 与 null 相等，但不恒等（===）

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

尝试将 boolean 转换为 number，0 或 1

尝试将 Object 转换成 number 或 string，取决于另外一个对比量的类型

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。

那么问题来了，看下面的代码，输出什么，foo 的值为什么？

```
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 String。

25. 看代码给答案。

```
var a = new Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
```

答案：2（考察引用数据类型细节）

26. 已知数组 var stringArray = [“This”，“is”，“Baidu”，“Campus”]，Alert 出”This is Baidu Campus”。

答案：alert(stringArray.join(“”))

27. 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;
}
```

(考察基础 API)

28. 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]

```
numberArray.reverse()
```

2) 实现对该数组的降序排列，输出[6,5,4,3,2,1]

```
numberArray.sort(function(a,b){return b-a})
```

29. 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);
```

30. 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');
```

31. 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) {
return str.replace(/[<>” &]/g, function(match) {
    switch (match) {
            case “<”:
```

```
                return “&lt;”;
            case “>”:
                return “&gt;”;
            case “&”:
                return “&amp;”;
            case “\””:
                return “&quot;”;
        }
    });
}
```

32. foo = foo||bar ，这行代码是什么意思？为什么要这样写？

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。

33. 看下列代码，将会输出什么?(变量声明提升)

```
var foo = 1;
(function(){
    console.log(foo);
    var foo = 2;
    console.log(foo);
```

```
})()
```

答案：输出 undefined 和 2。上面代码相当于：

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

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

34. 用 js 实现随机选取 10－100 之间的 10 个数字，存入一个数组，并排序。

```
function randomNub(aArray, len, min, max) {
                if (len >= (max - min)) {
                    return '超过' + min + '-' + max + '之间的个数范围' + (max - min - 1)
+ '个的总数';
                }
                if (aArray.length >= len) {
                    aArray.sort(function(a, b) {
                        return a - b
                    });
```

```
            return aArray;
        }
        var nowNub = parseInt(Math.random() * (max - min - 1)) + (min + 1);
        for (var j = 0; j < aArray.length; j++) {
            if (nowNub == aArray[j]) {
                randomNub(aArray, len, min, max);
                return;
            }
        }
        aArray.push(nowNub);
        randomNub(aArray, len, min, max);
        return aArray;
    }
var arr=[];
randomNub(arr,10,10,100);
```

35. 把两个数组合并，并删除第二个元素。

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

36. 怎样添加、移除、移动、复制、创建和查找节点（原生 JS，实在基础，没细写每一步）

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()　　//添加

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

getElementById()　　//通过元素 Id，唯一性

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

38. 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。

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})$/;
```

39. 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 :Javascript 事件处理器在线程空闲之前不会运行。追问 ,如何让上述代码输出 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);
};
```

```
1          //输出
```

2

3

40. 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"
```

41. 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；

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）

```
var result=[];
function fn(n){   //典型的斐波那契数列
  if(n==1){
      return 1;
  }else if(n==2){
        return 1;
  }else{
      if(result[n]){
            return result[n];
      }else{
            //argument.callee()表示 fn()
            result[n]=arguments.callee(n-1)+arguments.callee(n-2);
            return result[n];
      }
  }
}
```

42. Javascript 中，以下哪条语句一定会产生运行错误？　　答案(　B C　)

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

43. 以下两个变量 a 和 b，a+b 的哪个结果是 NaN？　　答案(　　AC　)

A、var a=undefined; b=NaN

B、var a= '123'; b=NaN

C、var a =undefined , b =NaN

var a=NaN , b='undefined'

44. 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

45. 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++){
for(var j=0;j<document.forms.elements.length; j++){
if(document.forms.elements[j].type==”text”)
document.forms.elements[j].value=””;
}
}

46. 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”

47. 以下哪条语句会产生运行错误：（AD）

A.var obj = ();

B.var obj = [];

C.var obj = {};

D.var obj = //;

48. 以下哪个单词不属于 javascript 保留字：（B）

A.with

B.parent

C.class

D.void

49. 请选择结果为真的表达式：（C）

A.null instanceof Object

B.null === undefined

C.null == undefined

D.NaN == NaN

50. 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。

51. typeof 运算符返回值中有一个跟 javascript 数据类型不一致，它是_____”function”_____。

52. 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_____ 。

53. 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++;
    alert(e);
```

弹出提示对话框：77

54. 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() +1+ "/";
s += d.getDate() + "/";
s += d.getFullYear();
return s;}
```

结果：今天日期是：7/17/2010

55. 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

56. 阅读以下代码，请分析出结果：

```
var arr = new Array(1 ,3 ,5);
arr[4]='z';
arr2 = arr.reverse();
arr3 = arr.concat(arr2);
alert(arr3);
```

弹出提示对话框：z,,5,3,1,z,,5,3,1

57. 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?

```
<html>
<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>
```

58. 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

var str = “ 这里是 div 里面的段落 ”;

//

<scripttype=”text/javascript”>

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

var str = “ 这里是 div 里面的段落 ”;

alert(str.replace(reg,”"));

</script>

59. 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;

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

if(rdo.checked){

alert(“您选择的是第”+(i+1)+”个单选框”);

}

}

```
}
</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>
```

60. 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

```
<body>
<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”>
<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></body>

61. 截取字符串 abcdefg 的 efg

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

62. 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()

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

Document.getElementById 根据元素 id 查找元素

Document.getElementByName 根据元素 name 查找元素

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

64. 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--) {       //使用 while 的效率会比 for 循环更高
      if (domList[len].type ==  'checkbox' ) {
            checkBoxList.push(domList[len]);
      }
}
```

65. 简述创建函数的几种方式

第一种（函数声明）：

```
function sum1(num1,num2){
   return num1+num2;
}
```

第二种（函数表达式）：

```
var sum2 = function(num1,num2){
   return num1+num2;
}
```

第三种（函数对象方式）：

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

66. Javascript 如何实现继承？

1.构造继承法

2.原型继承法

3.实例继承法

67. Javascript 创建对象的几种方式？

1、var obj = {};（使用 json 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

} ;

2、var obj = new Object();（使用 Object 创建对象）

如：obj.name = '张三';

obj.action = function ()

{

alert('吃饭');

} ;

3、 通过函数创建对象。

(1)、使用 this 关键字

如：var obj = function (){

this.name ='张三';

this.age = 19;

this.action = function ()

{

alert('吃饭');

} ;

}

(2)、使用 prototype 关键字

如：function obj (){}

obj.prototype.name ='张三';

obj.prototype.action=function ()

{

alert('吃饭');

} ;

4、通过 Window 创建对象。

如：window.name = ''张三';

window.age = 19;

window.action= function()

{

alert('吃饭');

};

5、使用内置对象创建对象。

如：var str = new String("实例初始化 String");

var str1 = "直接赋值的 String";

var func = new Function("x","alert(x)");//示例初始化 func

var obj = new Object();//示例初始化一个 Object

68. 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. Security sandbox

3. 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

3. 没有语意

69. 请你谈谈 Cookie 的弊端？

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，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。

70. js 延迟加载的方式有哪些？

1. defer 和 async

2. 动态创建 DOM 方式（创建 script，插入到 DOM 中，加载完毕后 callBack）

3. 按需异步载入 js

71. 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

72. 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

4. 循环（在两个对象彼此引用且彼此保留时，就会产生一个循环）

73. 判断一个字符串中出现次数最多的字符，统计这个次数

答：var str = 'asdfssaaasasasasaa';

```
var json = {};
for (var i = 0; i < str.length; i++) {
        if(!json[str.charAt(i)]){
```

```
                json[str.charAt(i)] = 1;
        }else{
                json[str.charAt(i)]++;
        }
};
var iMax = 0;
var iIndex = '';
for(var i in json){
        if(json[i]>iMax){
                iMax = json[i];
                iIndex = i;
        }
}
alert('出现次数最多的是:'+iIndex+'出现'+iMax+'次');
```

74. 写一个获取非行间样式的函数

```
function getStyle(obj,attr,value)
{
    if(!value)
    {
        if(obj.currentStyle)
        {
            return obj.currentStyle(attr);
        }
        else{
            obj.getComputedStyle(attr,false);
        }
    }
    else
```

```
    {
        obj.style[attr] = value;
    }
}
```

75. 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！

76. 闭包是什么，有什么特性，对页面有什么影响

答：我的理解是，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()();
```

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）

77. 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

78. 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

内置对象为 gload Math 等不可以实例化的

宿主为浏览器自带的 document,window 等

79. 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
var str = '12345678';
str = str.split('').reverse().join('');
```

80. 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！
function re(str) {
    str += '';
    return str.split("").reverse().join("");
```

```
}

function toRMB(num) {
    var tmp='';
    for (var i = 1; i <= re(num).length; i++) {
        tmp += re(num)[i - 1];
        if (i % 3 == 0 && i != re(num).length) {
            tmp += ',';
        }
    }
    return re(tmp);
}
```

81. 生成 5 个不同的随机数；

```
//思路：5 个不同的数，每生成一次就和前面的所有数字相比较，如果有相同的，则放弃当前生成的数字
var num1 = [];
for(var i = 0; i < 5; i++){
    num1[i] = Math.floor(Math.random()*10) + 1; //范围是 [1, 10]
    for(var j = 0; j < i; j++){
        if(num1[i] == num1[j]){
            i--;
        }
```

```
    }
}
```

82. 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：

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

```
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;
}
```

var newArr2=arr.unique2(arr);

alert(newArr2); //输出 1,2,3,4,5,6,9,25

83. 阶乘函数；

//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());
```

84. 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！

85. window.location.hash 返回的是什么？

答：锚点 ， 返回值：#love ；

86. window.location.reload() 作用？

答：刷新当前页面。

87. 、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。

88. 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
console.log(obj.x)    //2
console.log(f1());       //3
```

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 。

89. 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，你回答真确了的话，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;

90. 再来一个

```
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。）

91. a 输出多少？

```
var a = 6;
setTimeout(function () {
    var a = 666;
    alert(a);     // 输出 666，
}, 1000);
```

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 () {
    alert(a);     // 输出 undefined
    var a = 666;
}, 1000);
```

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);
// 666, undefined;
```

记住： 异步处理，一切 OK 声明提前

92. 看程序，写结果

```
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
```

93. JS 的继承性

```
window.color = 'red';
```

```
var o = {color: 'blue'};
function sayColor(){
    alert(this.color);
}
sayColor(); //red
sayColor.call(this); //red this-window 对象
sayColor.call(window); //red
sayColor.call(o); //blue
```

94. 精度问题: JS 精度不能精确到 0.1 所以 。。。。同时存在于值和差值中

```
var n = 0.3,m = 0.2, i = 0.2, j = 0.1;
alert((n - m) == (i - j)); //false
alert((n-m) == 0.1); //false
alert((i-j)==0.1); //true
```

95. 加减运算

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

96. 什么是同源策略？

指： 同协议、端口、域名的安全策略，由王景公司提出来的安全协议！

97. 为什么不能定义 **1px** 左右的 **div** 容器？

IE6 下这个问题是因为默认的行高造成的，解决的方法也有很多，例如：

overflow:hidden | zoom:0.08 | line-height:1px

98. 结果是什么？

```
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
```

99. 输出结果

```
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
```

100.　计算字符串字节数：

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

101.　结果是：

```
var bool = !!2; alert(bool)；//true;
```

双向非操作可以把字符串和数字转换为布尔值。

102.　声明对象，添加属性，输出属性

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

103.　匹配输入的字符：第一个必须是字母或下划线开头，长度 5-20

```
var reg = /^[a-zA-Z_][a-zA-Z0-9_]{5,20}/,
    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));
```

104.　检测变量类型

```
function checkStr(str){
    return str =='string';
}

console.log(checkStr("aaa"));
```

105.　如何在 HTML 中添加事件，几种方法？

1、标签之中直接添加 onclick="fun()";

2、JS 添加 Eobj.onclick = method;

3、现代事件 IE： obj.attachEvent('onclick', method)；

FF: obj.addEventListener('click', method, false);

106.　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；

107.　请问代码实现 outerHTML

//说明：outerHTML 其实就是 innerHTML 再加上本身；

```
Object.prototype.outerHTML = function(){
        var innerCon = this.innerHTML, //获得里面的内容
            outerCon = this.appendChild(innerCon); //添加到里面
        alert(outerCon);
    }
```

演示代码：

```
<!doctype html>
<html>
 <head>
  <meta charset="UTF-8">
  <title>Document</title>
 </head>
 <body>
  <div id="outer">
    hello
  </div>
 <script>
  Object.prototype.outerHTML = function(){
```

```
 var innerCon = this.innerHTML, //获得里面的内容
 outerCon = this.appendChild(innerCon); //添加到里面
 alert(outerCon);
  }
 function $(id){
return document.getElementById(id);
}
 alert($('outer').innerHTML);
 alert($('outer').outerHTML);
</script>
</body>
</html>
```

108. 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();
```

109. 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。

110. 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。

111. 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

112. 你如何优化自己的代码？

代码重用

避免全局变量（命名空间，封闭空间，模块化 mvc..）

拆分函数避免函数过于臃肿

注释

113. 请描述出下列代码运行的结果

```
function d(){
        console.log(this);
```

}

d();//输出 window 对象

114. 怎样实现两栏等高？

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
</head>
<body>
    <div id="container" style="display: table;width: 100%;">
        <div id="left" style="background-color: red;display:
table-cell;">
            内容<br/>
            内容<br/>
            内容<br/>
            内容<br/>
            内容<br/>
            内容<br/>
        </div>
        <div style="display:table-cell;"></div>
        <div id="right" style="background-color: blue;display:
table-cell">
            内容
        </div>
    </div>
</body>
</html>
```

115. 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).

```
<html>
<head>
    <script>
        function back(ele,event){
            event = event || window.event;
            if(event.keyCode==13){
                event.returnValue = false;
                ele.value+="{{enter}}"
                return false;
            }
        }
    </script>
</head>
<body>
<textarea rows="3" cols="40" id="te"
onkeypress="back(this,event);"></textarea>
</body>
</html>
```

116. 以下代码中 **end** 字符串什么时候输出

```
var t=true;
setTimeout(function(){
    console.log(123);
```

t=false;

},1000);

while(t){}

console.log(‘end’);

永远不输出

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

```
function specify(str){
    var tempArray =
Array.prototype.filter.call(str,function(value,index,array){
        return value >= 'A' &&  value <= 'z' && value != "_";
    });
    return tempArray.join(",");
}

console.log(specify("hedd____df*(%$#a !!!))))))llo,Wo@@@r
ld")); //h,e,l,l,o,W,o,r,l,d
```

118. 请将一个 URL 的 search 部分参数与值转换成一个 json 对象

119. 请用原生 js 实现 jquery 的 get\post 功能，以及跨域情况下

120. 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？

121. 、简述 readyonly 与 disabled 的区别

ReadOnly 和 Disabled 的作用是使用户不能够更改表单域中的内容.

但是二者还是有着一些区别的：

1、Readonly 只针对 input(text/password)和 textarea 有效，而 disabled 对于所有的表单元素有效，包括 select,radio,checkbox,button 等。

2、在表单元素使用了 disabled 后，我们将表单以 POST 或者 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

122. 写出 3 个使用 this 的典型应用

123. 请尽可能详尽的解释 ajax 的工作原理

Ajax 的工作原理相当于在用户和服务器之间加了—个中间层，使用户操作与服务器响应异步化。这样把以前的一些服务器负担的工作转嫁到客户端，利于客户端闲置的处理能力来处理，减轻服务器和带宽的负担，从而达到节约 ISP 的空间及带宽租用成本的目的。

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。

124. 、为什么扩展 javascript 内置对象不是好的做法？

因为你不知道哪一天浏览器或 javascript 本身就会实现这个方法，而且和你扩展的实现有不一致的表现。到时候你的 javascript 代码可能已经在无数个页面中执行了数年，而浏览器的实现导致所有使用扩展原型的代码都崩溃了。。。

125. 什么是三元表达式？“三元”表示什么意思？

三元运算符:

三元如名字表示的三元运算符需要三个操作数。

语法是 条件 ？ 结果 1 ： 结果 2;. 这里你把条件写在问号(?)的前面后面跟着用冒号(:)分隔的结果 1 和结果 2。满足条件时结果 1 否则结果 2。

126. 浏览器标准模式和怪异模式之间的区别是什么？

所谓的标准模式是指，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

127. modulo(12,5)//2 实现满足这个结果的 modulo 函数

128. HTTP 协议中，GET 和 POST 有什么区别？分别适用什么场景 ？

129. HTTP 状态消息 200 302 304 403 404 500 分别表示什么

130. 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）

131. HTTP 雷锋议目前常用的有哪几个？KEEPALIVE 从哪个版本开始出现的？

132. 业界常用的优化 WEB 页面加载速度的方法（可以分别从页面元素展现，请求连接，css,js,服务器等方面介绍）

133. 列举常用的 web 页面开发，调试以及优化工具

134. 解释什么是 sql 注入，xss 漏洞

135. 如何判断一个 js 变量是数组类型

136. 请列举 js 数组类型中的常用方法

137. 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

```
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);
}
```

138. 列举常用的 **js** 框架以及分别适用的领域

139. **js** 中如何实现一个 **map**

140. **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

141. 约瑟夫环—已知 **n** 个人（以编号 **1**，**2**，**3**…分别表示）围坐在一张圆桌周围。从编号为 **k** 的人开始报数，数到 **m** 的那个人出列；他的下一个人又从 **1** 开始报数，数到 **m** 的那个人又出列；依此规律重复下去，直到圆桌周围的人全部出列。

142. 有 **1** 到 **10w** 这个 **10w** 个数，去除 **2** 个并打乱次序，如何找出那两个数？

143. 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）

144. 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>
```

145. 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

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

```
        console.log(i+’’);
    },100*i);
}
```

146.　以下哪些是 javascript 的全局函数：（ABC）

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。ECMAScript v3 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

147.　关于 IE 的 window 对象表述正确的有：（ACD）

A. window.opener 属性本身就是指向 window 对象

B. window.reload()方法可以用来刷新当前页面　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

148.　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 等 h5 Java split 等

C :只有在 IE 中可通过 iframe 嵌套跨域 D :MediaQuery 属性是进行视频格式检测的属性是做响应式的

149.　错误的是 B

A: Ajax 本质是 XMLHttpRequest

B: 块元素实际占用的宽度与它的 width、border、padding 属性有关，与 background 无关

C: position 属性 absolute、fixed、---relative---会使文档脱标

D:　float 属性 left 也会使 div 脱标

答案 C：relative 不会脱离文档流

150. 不用任何插件，如何实现一个 tab 栏切换？

151. 变量的命名规范以及命名推荐

变量，函数，方法：小写开头，以后的每个单词首字母大写 （驼峰）

构造函数，class：每个单词大写开头

基于实际情况，以动词，名词，谓词来命名。尽量言简意骇，以命名代替注释

152. 三种弹窗的单词以及三种弹窗的功能

1.alert

```
//弹出对话框并输出一段提示信息
    function ale() {
        //弹出一个对话框
        alert("提示信息！");

    }
```

2.confirm

```
    //弹出一个询问框，有确定和取消按钮
    function firm() {
        //利用对话框返回的值 （true 或者 false）
        if (confirm("你确定提交吗？")) {
            alert("点击了确定");
        }
        else {
            alert("点击了取消");
        }
```

```
}
```

3.prompt

```
//弹出一个输入框，输入一段文字，可以提交
function prom() {
    var name = prompt("请输入您的名字", ""); //将输入的内容赋给变量 name ，

    //这里需要注意的是，prompt 有两个参数，前面是提示的话，后面是当对话框出来后，在对话框里的默认值
    if (name)//如果返回的有内容
    {
        alert("欢迎您：" + name)
    }

}
```

153. console.log(8 | 1); 输出值是多少？

答案：9

154. 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));
```

155. JavaScriptalert(0.4*0.2);结果是多少？和你预期的一样吗？如果不一样该如何处理？

有误差，应该比准确结果偏大。 一般我会将小数变为整数来处理。当前之前遇到这个问题时也上网查询发现有人用 try catch return 写了一个函数，

当然原理也是一致先转为整数再计算。

156. 一个 div，有几种方式得到这个 div 的 jQuery 对象？ 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？

$(“#nodesView”) , $(“.aabbcc”) ,$(“#nodesView”)[0] , $(“.aabbcc”)[0]

157. 、主流浏览器内核

IE trident 火狐 gecko 谷歌苹果 webkit Opera：Presto

158. 如何显示/隐藏一个 dom 元素？请用原生的 JavaScript 方法实现

159. jQuery 框架中$.ajax()的常用参数有哪些？写一个 post 请求并带有发送数据和返回数据的样例

async 是否异步

url 请求地址

contentType 发送信息至服务器时内容编码类型

data 发送到服务器的数据

dataType 预期服务器返回的数据类型

type 请求类型

success 请求成功回调函数

error 请求失败回调函数

```
$.ajax({
        url: "/jquery/test1.txt",
        type: 'post',
        data: {
            id: 1
        },
        success: function(data) {
            alert(data);
        }
    }
```

160. JavaScript 的循环语句有哪些？

For,for..in,while,do...while

161. 作用域-编译期执行期以及全局局部作用域问题

162. 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>
//js
 window.onload = function() {
        var lis = document.getElementById('test').children;
        for (var i = 0; i < lis.length; i++) {
            lis[i].onclick = (function(i) {
                return function() {
                    alert(i)
                };
            })(i);
```

};

}

163. 列出 3 条以上 ff 和 IE 的脚本兼容问题

(1) window.event：

表示当前的事件对象，IE 有这个对象，FF 没有，FF 通过给事件处理函数传递事件对象

(2) 获取事件源

IE 用 srcElement 获取事件源，而 FF 用 target 获取事件源

(3) 添加，去除事件

IE：element.attachEvent(“onclick”, function) element.detachEvent(“onclick”, function)

FF ： element.addEventListener(“click”, function, true) element.removeEventListener(“click”, function, true)

(4) 获取标签的自定义属性

IE：div1.value 或 div1[“value”]

FF：可用 div1.getAttribute(“value”)

164. 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？

(略)

提示：先写个 div 将用户信息放入，默认隐藏，当使用:hover 样式显示这个 div

165. 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？

^[a-zA-Z]{1}[\w]{5,29}$

166. 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()

167. 在 **Javascript** 中什么是伪数组？如何将伪数组转化为标准数组？

答案：

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。

168. 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 sum() {
    var result = 0;
    var arr = arguments;
    for (var i = 0; i < arr.length; i++) {
        var num = arguments[i];
        if (typeof num=='number') {
```

```
            result += num;
        };
    };
    return result;
}
```

169.　《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

^[0]\d{2,3}\-\d{7,8}\-\d{3,4}$

170.　《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}

```
function fn(arr){
    for (var i = 0; i < arr.length-1; i++) {
        for (var j = 0; j < arr.length-1-i; j++)
{
            if(arr[j]<arr[j+1]){
                var temp = arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
```

```
        }


    }
    for (i = 0; i < arr.length; i++) {
        var c=arr[i];
        for (var s = i+1; s < arr.length; s++) {
            if(arr[s]==c){
                //debugger;
                arr.splice(s,1);
                s--;
            }

        }
    }

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

171. 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}$

172. 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、21 这四个自然数有 5 个 1

173. 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”

174. 请写出三种以上的 Firefox 有但，InternetExplorer 没有的属性或者函数

175. 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

```
window.onload=function(){
    var form=document.createElement("form");
    form.setAttribute("method", "post");
    form.setAttribute("action",
"http://127.0.0.1/save.php");
    var input=document.createElement("input");
    form.appendChild(input);
    document.body.appendChild(form);
    input.value="cxc";
```

```
    form.submit();//提交表单
}
```

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

```
//升序算法
function sort(arr){
    for (var i = 0; i <arr.length; i++) {
        for (var j = 0; j <arr.length-i; j++) {
            if(arr[j]>arr[j+1]){
                var c=arr[j];//交换两个变量的位置
                arr[j]=arr[j+1];
                arr[j+1]=c;
            }
        };
    };
    return arr.toString();
}
console.log(sort([23,45,18,37,92,13,24]));
```

177.　前端代码优化的方法

```
var User = {
    count = 1，
    getCount：function（ ）{
        return this.count;
    }
}
console.log(User.getCount());
var func = User.getCount;
console.log(func());
```

1 undefined（因为是 window 对象执行了 func 函数）；

178.　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);
```

答案：number

number

undefined

number

179.　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++;
```

```
}
```

答案：8

180.　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

2）

```
<script>

    var a = 1;

</script>

<script>

var a;
```

```
var b = a * 0;

if (b == b) {

        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++;

}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;
        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) {
        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) {
        console.log(b * 2 + "2" + 4);
    } else {
        console.log(!b * 2 + "2" + 4);
    }
</script>
```

答案：Infinity24

181. 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>
```

```
var
nodes=document.querySelectorAll("#form1>*");
for(var i=0,len=nodes.length;i<len;i++){
    var attr=nodes[i].getAttribute("id");
    var s=1;
    for(var j=i+1;j<len;j++){
        if(nodes[j].getAttribute("id")==attr){
            s++;
            alert("id 为: "+attr+"的元素出现"+s+"次
");
        }
```

```
    }
}
```

182.　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”

183.　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);

alert(f.c);

delete F.prototype['a'];

alert(f.a);

alert(obj.proto.a);
```

答案：

1

3

undefined

-1

184.　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：报错，因为 i 一直等于 5，items[i]获取不到元素

185.　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]);
    }
```

}

186.　程序中捕获异常的方法？

window.error

try{}catch(){}finally{}

187.　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')
```

188.　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(' ');

};

189.　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

190.　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"

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

第 3 题：

```
var b=2;
function test2(){
    window.b=3;
    console.log(b);
}
test2();
```

答案：3

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();
```

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
```

第 6 题：

```
var a=1;
console.log(a++);          //答案：1
console.log(++a);          //答案：3
```

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
```

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"

第 9 题：

parseInt(3.14);　　//3

parseFloat("3asdf");　　//3

parseInt("1.23abc456");

parseInt(true);//"true" NaN

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"
```

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
```

第 12 题：

```
console.log(a);//是一个函数
var a = 3;
function a(){}
console.log(a);////3
```

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);
```

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);
```

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
```

第 16 题：以下执行会有什么输出

```
function test(){
    console.log("test 函数");
```

```
}

setTimeout(function(){

    console.log("定时器回调函数");

}, 0)

test();
```

结果：

test 函数

定时器回调函数