**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。**

**答：**

var iArray = [];

function getRandom(istart, iend){

        var iChoice = iend - istart +1;

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

}

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

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

}

iArray.sort();

**var *arr***=[];  
**for**(**i**=0;**i**<10;**i**++){  
 **var *r***=**Math**.floor(**Math**.random()\*91+10);  
 ***arr***.push(***r***);  
}  
alert(***arr***);

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

**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);

**var *arr1***=[**'a'**,**'b'**,**'c'**];  
**var *arr2***=[1,2,3,4];  
**var *arr3***=***arr1***.concat(***arr2***)  
***arr3***.splice(1,1);  
alert(***arr3***);

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

**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）**

**答：**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

**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：

1 function serilizeUrl(url) {

2 var result = {};

3 url = url.split("?")[1];

4 var map = url.split("&");

5 for(var i = 0, len = map.length; i < len; i++) {

6 result[map[i].split("=")[0]] = map[i].split("=")[1];

7 }

8 return result;

9 }

**看下面代码，给出输出结果。**

|  |  |
| --- | --- |
| 1  2  3  4  5 | for(var i=1;i<=3;i++){    setTimeout(function(){        console.log(i);    },0);  }; |

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

答案：4 4 4。

原因：Javascript事件处理器在线程空闲之前不会运行。那么问题来了，**如何让上述代码输出1 2 3？**

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

2 setTimeout((function(a){ //改成立即执行函数

3 console.log(a);

4 })(i),0);

5 };

6

7 1 //输出

8 2

9 3

**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：

1 if (!String.prototype.trim) {

2 String.prototype.trim = function() {

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

4 }

5 }

6

7 // test the function

8 var str = " \t\n test string ".trim();

9 alert(str == "test string"); // alerts "true"

Javascript中callee和caller的作用？

答案：

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）

1 var result=[];

2 function fn(n){ //典型的斐波那契数列

3 if(n==1){

4 return 1;

5 }else if(n==2){

6 return 1;

7 }else{

8 if(result[n]){

9 return result[n];

10 }else{

11 //argument.callee()表示fn()

12 result[n]=arguments.callee(n-1)+arguments.callee(n-2);

13 return result[n];

14 }

15 }

16 }

**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

1 // 方法一：

2 Object.prototype.clone = function(){

3 var o = this.constructor === Array ? [] : {};

4 for(var e in this){

5 o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];

6 }

7 return o;

8 }

9

10 //方法二：

11 /\*\*

12 \* 克隆一个对象

13 \* @param Obj

14 \* @returns

15 \*/

16 function clone(Obj) {

17 var buf;

18 if (Obj instanceof Array) {

19 buf = []; //创建一个空的数组

20 var i = Obj.length;

21 while (i--) {

22 buf[i] = clone(Obj[i]);

23 }

24 return buf;

25 }else if (Obj instanceof Object){

26 buf = {}; //创建一个空对象

27 for (var k in Obj) { //为这个对象添加新的属性

28 buf[k] = clone(Obj[k]);

29 }

30 return buf;

31 }else{ //普通变量直接赋值

32 return Obj;

33 }

34 }

**如何消除一个数组里面重复的元素？**

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

2 function deRepeat(){

3 var newArr=[];

4 var obj={};

5 var index=0;

6 var l=arr.length;

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

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

9 {

10 obj[arr[i]]=1;

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

12 }

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

14 continue;

15 }

16 return newArr;

17

18 }

19 var newArr2=deRepeat(arr);

20 alert(newArr2); //输出1,2,3,4,5,6,9,25

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

**下面这个ul，如何点击每一列的时候alert其index?（**闭包）

|  |  |
| --- | --- |
| 1  2  3  4  5 | <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> |

答案：

1 // 方法一：

2 var lis=document.getElementById('2223').getElementsByTagName('li');

3 for(var i=0;i<3;i++)

4 {

5 lis[i].index=i;

6 lis[i].onclick=function(){

7 alert(this.index);

8 };

9 }

10

11 //方法二：

12 var lis=document.getElementById('2223').getElementsByTagName('li');

13 for(var i=0;i<3;i++)

14 {

15 lis[i].index=i;

16 lis[i].onclick=(function(a){

17 return function() {

18 alert(a);

19 }

20 })(i);

21 }

**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}\*/

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** query (selector) {
3. **var** reg = /^(#)?(\.)?(\w+)$/img;
4. **var** regResult = reg.exec(selector);
5. **var** result = [];
7. //如果是id选择器
8. **if**(regResult[1] && regResult[3]) {
9. **if**(**typeof** document.querySelector === "function") {
10. result.push(document.querySelector(regResult[0]));  //  用 result[result.length] = ..速度更快
11. }
12. **else** {
13. result.push(document.getElementById(regResult[3]));
14. }
15. }
16. //如果是class选择器
17. **else** **if**(regResult[2] && regResult[3]) {
18. **if**(**typeof** document.getElementsByClassName === 'function') {
19. **var** doms = document.getElementsByClassName(regResult[3]);
20. **if**(doms) {
21. result = converToArray(doms);
22. }
23. } **else** {  //如果不支持getElementsByClassName函数
24. **var** allDoms = document.getElementsByTagName("\*") ;
25. **for**(**var** i = 0, len = allDoms.length; i < len; i++) {
26. **if**(allDoms[i].className.search(**new** RegExp(regResult[2])) > -1) {
27. result.push(allDoms[i]);
28. }
29. }
30. }
31. }
32. //如果是标签选择器
33. **else** **if**(regResult[3]) {
34. **var** doms = document.getElementsByTagName(regResult[3].toLowerCase());
35. **if**(doms) {
36. result = converToArray(doms);
37. }
38. }
39. **return** result;
40. }
42. **function** converToArray(nodes){
43. **var** array = **null**;
44. **try**{
45. array = Array.prototype.slice.call(nodes,0);//针对非IE浏览器
46. }**catch**(ex){
47. array = **new** Array();
48. **for**( **var** i = 0 ,len = nodes.length; i < len ; i++ ) {
49. array.push(nodes[i])
50. }
51. }
52. **return** array;
53. }

**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);

      });

   }

}

评价：

* 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：

1 function addEvent(elem, type, handler){

2 　　if(elem.addEventListener){

3 　　　　elem.addEventListener(type, handler, false);

4 　　}else if(elem.attachEvent){

5 　　　　elem['temp' + type + handler] = handler;

6 　　　　elem[type + handler] = function(){

7 　　　　elem['temp' + type + handler].apply(elem);

8 　　};

9 　　elem.attachEvent('on' + type, elem[type + handler]);

10 }else{

11 　　elem['on' + type] = handler;

12 　　}

13 }

**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’**

1 String.prototype.spacify = function(){

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

3 };

接着上述答题，那么问题来了

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Javscript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。（函数声明提升）

**初级Javascript：**

**1.JavaScript是一门什么样的语言，它有哪些特点？**

没有标准答案。

**2.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：

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

2 {

3 Array.isArray = function(arg){

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

5 };

6 }

**3.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)**

1 document.getElementById(“ID”).value

**4.希望获取到页面中所有的checkbox怎么做？(不使用第三方框架)**

1 var domList = document.getElementsByTagName(‘input’)

2 var checkBoxList = [];

3 var len = domList.length;　　//缓存到局部变量

4 while (len--) {　　//使用while的效率会比for循环更高

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

6 　　checkBoxList.push(domList[len]);

7 　　}

8 }

**5.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)**

1 var dom = document.getElementById(“ID”);

2 dom.innerHTML = “xxxx”

3 dom.style.color = “#000”

**6.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：<div onclick=”test()”></div>
* 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

**7.什么是Ajax和JSON，它们的优缺点。**

Ajax是异步JavaScript和XML，用于在Web页面中实现异步数据交互。

优点：

* 可以使得页面不重载全部内容的情况下加载局部内容，降低数据传输量
* 避免用户不断刷新或者跳转页面，提高用户体验

缺点：

* 对搜索引擎不友好（
* 要实现ajax下的前后退功能成本较大
* 可能造成请求数的增加
* 跨域问题限制

JSON是一种轻量级的数据交换格式，ECMA的一个子集

优点：轻量级、易于人的阅读和编写，便于机器（JavaScript）解析，支持复合数据类型（数组、对象、字符串、数字）

**8.看下列代码输出为何？解释原因。**

1 var a;

2 alert(typeof a); // undefined

3 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。

**9.看下列代码,输出什么？解释原因。**

1 var a = null;

2 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”。

**10.看下列代码,输出什么？解释原因。**

1 var undefined;

2 undefined == null; // true

3 1 == true;   // true

4 2 == true;   // false

5 0 == false;  // true

6 0 == '';     // true

7 NaN == NaN;  // false

8 [] == false; // true

9 [] == ![];   // 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的类型为什么？

1 var foo = "11"+2-"1";

2 console.log(foo);

3 console.log(typeof foo);

执行完后foo的值为111，foo的类型为Number。

1 var foo = "11"+2+"1"; //体会加一个字符串'1' 和 减去一个字符串'1'的不同

2 console.log(foo);

3 console.log(typeof foo);

执行完后foo的值为113，foo的类型为String。

**11.看代码给答案。**

1 var a = new Object();

2 a.value = 1;

3 b = a;

4 b.value = 2;

5 alert(a.value);

答案：2（考察引用数据类型细节）

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

答案：alert(stringArray.join(“”))

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”。

1 function combo(msg){

2 var arr=msg.split("-");

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

4 arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);

5 }

6 msg=arr.join("");

7 return msg;

8 }

(考察基础API)

**13.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]

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

2

3 numberArray.reverse(); // 5,1,4,2,6,3

4

5 numberArray.sort(function(a,b){ //6,5,4,3,2,1

6 return b-a;

7 })

**14.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**

1 var d = new Date();

2 // 获取年，getFullYear()返回4位的数字

3 var year = d.getFullYear();

4 // 获取月，月份比较特殊，0是1月，11是12月

5 var month = d.getMonth() + 1;

6 // 变成两位

7 month = month < 10 ? '0' + month : month;

8 // 获取日

9 var day = d.getDate();

10 day = day < 10 ? '0' + day : day;

11 alert(year + '-' + month + '-' + day);

**15.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’);

**16.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**

1 function escapeHtml(str) {

2 return str.replace(/[<>”&]/g, function(match) {

3 switch (match) {

4 case “<”:

5 return “&lt;”;

6 case “>”:

7 return “&gt;”;

8 case “&”:

9 return “&amp;”;

10 case “\””:

11 return “&quot;”;

12 }

13 });

14 }

**17.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。

**18.看下列代码，将会输出什么?(变量声明提升)**

1 var foo = 1;

2 function(){

3 console.log(foo);

4 var foo = 2;

5 console.log(foo);

6 }

答案：输出undefined 和 2。上面代码相当于：

1 var foo = 1;

2 function(){

3 var foo;

4 console.log(foo); //undefined

5 foo = 2;

6 console.log(foo); // 2;

7 }

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。

**19.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。**

1 var iArray = [];

2 funtion getRandom(istart, iend){

3         var iChoice = istart - iend +1;

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

5 }

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

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

8 }

9 iArray.sort();

**20.把两个数组合并，并删除第二个元素。**

1 var array1 = ['a','b','c'];

2 var bArray = ['d','e','f'];

3 var cArray = array1.concat(bArray);

4 cArray.splice(1,1);

**21.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

**22.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：

1 function serilizeUrl(url) {

2 var result = {};

3 url = url.split("?")[1];

4 var map = url.split("&");

5 for(var i = 0, len = map.length; i < len; i++) {

6 result[map[i].split("=")[0]] = map[i].split("=")[1];

7 }

8 return result;

9 }

**23.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：

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

**24.看下面代码，给出输出结果。**

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

2 setTimeout(function(){

3 console.log(i);

4 },0);

5 };

答案：4 4 4。

原因：Javascript事件处理器在线程空闲之前不会运行。那么问题来了，**如何让上述代码输出1 2 3？**

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

2 setTimeout((function(a){ //改成立即执行函数

3 console.log(a);

4 })(i),0);

5 };

6

7 1 //输出

8 2

9 3

**25.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：

1 if (!String.prototype.trim) {

2 String.prototype.trim = function() {

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

4 }

5 }

6

7 // test the function

8 var str = " \t\n test string ".trim();

9 alert(str == "test string"); // alerts "true"

**26.Javascript中callee和caller的作用？**

答案：

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）

1 var result=[];

2 function fn(n){ //典型的斐波那契数列

3 if(n==1){

4 return 1;

5 }else if(n==2){

6 return 1;

7 }else{

8 if(result[n]){

9 return result[n];

10 }else{

11 //argument.callee()表示fn()

12 result[n]=arguments.callee(n-1)+arguments.callee(n-2);

13 return result[n];

14 }

15 }

16 }

**中级Javascript：**

**1.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

1 // 方法一：

2 Object.prototype.clone = function(){

3 var o = this.constructor === Array ? [] : {};

4 for(var e in this){

5 o[e] = typeof this[e] === "object" ? this[e].clone() : this[e];

6 }

7 return o;

8 }

9

10 //方法二：

11 /\*\*

12 \* 克隆一个对象

13 \* @param Obj

14 \* @returns

15 \*/

16 function clone(Obj) {

17 var buf;

18 if (Obj instanceof Array) {

19 buf = []; //创建一个空的数组

20 var i = Obj.length;

21 while (i--) {

22 buf[i] = clone(Obj[i]);

23 }

24 return buf;

25 }else if (Obj instanceof Object){

26 buf = {}; //创建一个空对象

27 for (var k in Obj) { //为这个对象添加新的属性

28 buf[k] = clone(Obj[k]);

29 }

30 return buf;

31 }else{ //普通变量直接赋值

32 return Obj;

33 }

34 }

**2.如何消除一个数组里面重复的元素？**

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

2 function deRepeat(){

3 var newArr=[];

4 var obj={};

5 var index=0;

6 var l=arr.length;

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

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

9 {

10 obj[arr[i]]=1;

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

12 }

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

14 continue;

15 }

16 return newArr;

17

18 }

19 var newArr2=deRepeat(arr);

20 alert(newArr2); //输出1,2,3,4,5,6,9,25

**3.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：**

1 function Dog() {

2 this.wow = function() {

3 alert(’Wow’);

4 }

5 this.yelp = function() {

6 this.wow();

7 }

8 }

**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）

答案：

1 function MadDog() {

2 this.yelp = function() {

3 var self = this;

4 setInterval(function() {

5 self.wow();

6 }, 500);

7 }

8 }

9 MadDog.prototype = new Dog();

10

11 //for test

12 var dog = new Dog();

13 dog.yelp();

14 var madDog = new MadDog();

15 madDog.yelp();

**4.下面这个ul，如何点击每一列的时候alert其index?（闭包）**

1 <ul id=”test”>

2 <li>这是第一条</li>

3 <li>这是第二条</li>

4 <li>这是第三条</li>

5 </ul>

答案：

1 // 方法一：

2 var lis=document.getElementById('2223').getElementsByTagName('li');

3 for(var i=0;i<3;i++)

4 {

5 lis[i].index=i;

6 lis[i].onclick=function(){

7 alert(this.index);

8 };

9 }

10

11 //方法二：

12 var lis=document.getElementById('2223').getElementsByTagName('li');

13 for(var i=0;i<3;i++)

14 {

15 lis[i].index=i;

16 lis[i].onclick=(function(a){

17 return function() {

18 alert(a);

19 }

20 })(i);

21 }

5.**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}\*/**

答案：（过长，点击打开）

**6.请评价以下代码并给出改进意见。**

1 if(window.addEventListener){

2 var addListener = function(el,type,listener,useCapture){

3 el.addEventListener(type,listener,useCapture);

4 };

5 }

6 else if(document.all){

7 addListener = function(el,type,listener){

8 el.attachEvent("on"+type,function(){

9 listener.apply(el);

10 });

11 }

12 }

评价：

* 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：

1 function addEvent(elem, type, handler){

2 　　if(elem.addEventListener){

3 　　　　elem.addEventListener(type, handler, false);

4 　　}else if(elem.attachEvent){

5 　　　　elem['temp' + type + handler] = handler;

6 　　　　elem[type + handler] = function(){

7 　　　　elem['temp' + type + handler].apply(elem);

8 　　};

9 　　elem.attachEvent('on' + type, elem[type + handler]);

10 }else{

11 　　elem['on' + type] = handler;

12 　　}

13 }

**7.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’**

1 String.prototype.spacify = function(){

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

3 };

接着上述答题，那么问题来了

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Javscript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。（函数声明提升）

**8.定义一个log方法，让它可以代理console.log的方法。**

可行的方法一：

1 function log(msg)　{

2 console.log(msg);

3 }

4

5 log("hello world!") // hello world!

6

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：

1 function log(){

2 console.log.apply(console, arguments);

3 };

那么问题来了，**apply和call方法的异同**？

答案：

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]) 。

**9.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。

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：

1 function log(){

2 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

3 args.unshift('(app)');

4

5 console.log.apply(console, args);

6 };

**10.对作用域上下文和this的理解，看下列代码：**

1 var User = {

2 count: 1,

3

4 getCount: function() {

5 return this.count;

6 }

7 };

8

9 console.log(User.getCount()); // what?

10

11 var func = User.getCount;

12 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。

那么问题来了，**如何确保Uesr总是能访问到func的上下文，即正确返回1**。

答案：正确的方法是使用Function.prototype.bind。兼容各个浏览器完整代码如下：

1 Function.prototype.bind = Function.prototype.bind || function(context){

2 var self = this;

3

4 return function(){

5 return self.apply(context, arguments);

6 };

7 }

8

9 var func = User.getCount.bind(User);

10 console.log(func());

**11.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。

1 /\*

2 \* 传递函数给whenReady()

3 \* 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

4 \*/

5 var whenReady = (function() { //这个函数返回whenReady()函数

6 var funcs = []; //当获得事件时，要运行的函数

7 var ready = false; //当触发事件处理程序时,切换为true

8

9 //当文档就绪时,调用事件处理程序

10 function handler(e) {

11 if(ready) return; //确保事件处理程序只完整运行一次

12

13 //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

14 if(e.type === 'onreadystatechange' && document.readyState !== 'complete') {

15 return;

16 }

17

18 //运行所有注册函数

19 //注意每次都要计算funcs.length

20 //以防这些函数的调用可能会导致注册更多的函数

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

22 funcs[i].call(document);

23 }

24 //事件处理函数完整执行,切换ready状态, 并移除所有函数

25 ready = true;

26 funcs = null;

27 }

28 //为接收到的任何事件注册处理程序

29 if(document.addEventListener) {

30 document.addEventListener('DOMContentLoaded', handler, false);

31 document.addEventListener('readystatechange', handler, false); //IE9+

32 window.addEventListener('load', handler, false);

33 }else if(document.attachEvent) {

34 document.attachEvent('onreadystatechange', handler);

35 window.attachEvent('onload', handler);

36 }

37 //返回whenReady()函数

38 return function whenReady(fn) {

39 if(ready) { fn.call(document); }

40 else { funcs.push(fn); }

41 }

42 })();

如果上述代码十分难懂，下面这个简化版：

1 function ready(fn){

2 if(document.addEventListener) { //标准浏览器

3 document.addEventListener('DOMContentLoaded', function() {

4 //注销事件, 避免反复触发

5 document.removeEventListener('DOMContentLoaded',arguments.callee, false);

6 fn(); //执行函数

7 }, false);

8 }else if(document.attachEvent) { //IE

9 document.attachEvent('onreadystatechange', function() {

10 if(document.readyState == 'complete') {

11 document.detachEvent('onreadystatechange', arguments.callee);

12 fn(); //函数执行

13 }

14 });

15 }

16 };

**12.（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

**13.**

****

1 function setcookie(name,value,days){ //给cookie增加一个时间变量

2 　　var exp = new Date();

3 　　exp.setTime(exp.getTime() + days\*24\*60\*60\*1000); //设置过期时间为days天

4 　　document.cookie = name + "="+ escape (value) + ";expires=" + exp.toGMTString();

5 }

6 function getCookie(name){

7 　　var result = "";

8 　　var myCookie = ""+document.cookie+";";

9 　　var searchName = "+name+"=";

10 　　var startOfCookie = myCookie.indexOf(searchName);

11 　　var endOfCookie;

12 　　if(satrtOfCookie != -1){

13 　　　　startOfcookie += searchName.length;

14 　　　　endOfCookie = myCookie.indexOf(";",startOfCookie);

15 　　　　result = (myCookie.substring(startOfCookie,endOfCookie));

16 　　}

17 　　return result;

18 }

19 (function(){

20 　　var oTips = document.getElementById('tips');//假设tips的id为tips

21 　　var page = {

22 　　check: function(){//检查tips的cookie是否存在并且允许显示

23 　　　　var tips = getCookie('tips');

24 　　　　if(!tips || tips == 'show') return true;//tips的cookie不存在

25 　　　　if(tips == "never\_show\_again") return false;

26 　　},

27 　　hideTip: function(bNever){

28 　　　　if(bNever) setcookie('tips', 'never\_show\_again', 365);

29 　　　　oTips.style.display = "none";//隐藏

30 　　},

31 　　showTip: function(){

32 　　oTips.style.display = "inline";//显示，假设tips为行级元素

33 　　},

34 　　init: function(){

35 　　　　var \_this = this;

36 　　　　if(this.check()){

37 　　　　\_this.showTip();

38 　　　　setcookie('tips', 'show', 1);

39 　　}

40 　　oTips.onclick = function(){

41 　　　　\_this.hideTip(true);

42 　　};

43 　　}

44 　　};

45 page.init();

46 })();

**14.说出以下函数的作用是？空白区域应该填写什么？**

1 //define

2 (function(window){

3 function fn(str){

4 this.str=str;

5 }

6

7 fn.prototype.format = function(){

8 var arg = \_\_\_\_\_\_;

9 return this.str.replace(\_\_\_\_\_,function(a,b){

10 return arg[b]||"";

11 });

12 }

13 window.fn = fn;

14 })(window);

15

16 //use

17 (function(){

18 var t = new fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');

19 console.log(t.format('http://www.alibaba.com','Alibaba','Welcome'));

20 })();

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

**15.用面向对象的Javascript来介绍一下自己。（没答案哦亲，自己试试吧）**

答案： 对象或者Json都是不错的选择哦

[各大互联网公司2014前端笔试面试题–HTML,CSS篇](http://www.codeceo.com/article/html-css-interview.html)

**Html篇：**

**1.你做的页面在哪些流览器测试过？这些浏览器的内核分别是什么?**

* IE: trident内核
* Firefox：gecko内核
* Safari:webkit内核
* Opera:以前是presto内核，Opera现已改用Google Chrome的Blink内核
* Chrome:Blink(基于webkit，[Google与Opera Software共同开发](http://baike.baidu.com/view/1369399.htm#1_5))

**2.每个HTML文件里开头都有个很重要的东西，Doctype，知道这是干什么的吗？**

答案：<!DOCTYPE> 声明位于文档中的最前面的位置，处于 <html> 标签之前。此标签可告知浏览器文档使用哪种 HTML 或 XHTML 规范。（重点：告诉浏览器按照何种规范解析页面）

**3.Quirks模式是什么？它和Standards模式有什么区别**

答案：

从IE6开始，引入了Standards模式，标准模式中，浏览器尝试给符合标准的文档在规范上的正确处理达到在指定浏览器中的程度。

在IE6之前CSS还不够成熟，所以IE5等之前的浏览器对CSS的支持很差， IE6将对CSS提供更好的支持，然而这时的问题就来了，因为有很多页面是基于旧的布局方式写的，而如果IE6 支持CSS则将令这些页面显示不正常，如何在即保证不破坏现有页面，又提供新的渲染机制呢？

在写程序时我们也会经常遇到这样的问题，如何保证原来的接口不变，又提供更强大的功能，尤其是新功能不兼容旧功能时。遇到这种问题时的一个常见做法是增加参数和分支，即当某个参数为真时，我们就使用新功能，而如果这个参数 不为真时，就使用旧功能，这样就能不破坏原有的程序，又提供新功能。IE6也是类似这样做的，它将DTD当成了这个“参数”，因为以前的页面大家都不会去写DTD，所以IE6就假定 如果写了DTD，就意味着这个页面将采用对CSS支持更好的布局，而如果没有，则采用兼容之前的布局方式。这就是Quirks模式（怪癖模式，诡异模式，怪异模式）。

区别：

总体会有布局、样式解析和脚本执行三个方面的区别。

盒模型：在W3C标准中，如果设置一个元素的宽度和高度，指的是元素内容的宽度和高度，而在Quirks 模式下，IE的宽度和高度还包含了padding和border。



设置行内元素的高宽：在Standards模式下，给<span>等行内元素设置wdith和height都不会生效，而在quirks模式下，则会生效。

设置百分比的高度：在standards模式下，一个元素的高度是由其包含的内容来决定的，如果父元素没有设置百分比的高度，子元素设置一个百分比的高度是无效的用margin:0 auto设置水平居中：使用margin:0 auto在standards模式下可以使元素水平居中，但在quirks模式下却会失效。

（还有很多，答出什么不重要，关键是看他答出的这些是不是自己经验遇到的，还是说都是看文章看的，甚至完全不知道。）

**4.div+css的布局较table布局有什么优点？**

* 改版的时候更方便 只要改css文件。
* 页面加载速度更快、结构化清晰、页面显示简洁。
* 表现与结构相分离。
* 易于优化（seo）搜索引擎更友好，排名更容易靠前。

**5.a：img的alt与title有何异同？b：strong与em的异同？**

答案：

a:

* alt(alt text):为不能显示图像、窗体或applets的用户代理（UA），alt属性用来指定替换文字。替换文字的语言由lang属性指定。(在IE浏览器下会在没有title时把alt当成 tool tip显示)
* title(tool tip):该属性为设置该属性的元素提供建议性的信息。

b:

* strong:粗体强调标签，强调，表示内容的重要性
* em:斜体强调标签，更强烈强调，表示内容的强调点

**6.你能描述一下渐进增强和优雅降级之间的不同吗?**

* 渐进增强 progressive enhancement：针对低版本浏览器进行构建页面，保证最基本的功能，然后再针对高级浏览器进行效果、交互等改进和追加功能达到更好的用户体验。
* 优雅降级 graceful degradation：一开始就构建完整的功能，然后再针对低版本浏览器进行兼容。

区别：优雅降级是从复杂的现状开始，并试图减少用户体验的供给，而渐进增强则是从一个非常基础的，能够起作用的版本开始，并不断扩充，以适应未来环境的需要。降级（功能衰减）意味着往回看；而渐进增强则意味着朝前看，同时保证其根基处于安全地带。

“优雅降级”观点

“优雅降级”观点认为应该针对那些最高级、最完善的浏览器来设计网站。而将那些被认为“过时”或有功能缺失的浏览器下的测试工作安排在开发周期的最后阶段，并把测试对象限定为主流浏览器（如 IE、Mozilla 等）的前一个版本。

在这种设计范例下，旧版的浏览器被认为仅能提供“简陋却无妨 (poor, but passable)” 的浏览体验。你可以做一些小的调整来适应某个特定的浏览器。但由于它们并非我们所关注的焦点，因此除了修复较大的错误之外，其它的差异将被直接忽略。

“渐进增强”观点

“渐进增强”观点则认为应关注于内容本身。

内容是我们建立网站的诱因。有的网站展示它，有的则收集它，有的寻求，有的操作，还有的网站甚至会包含以上的种种，但相同点是它们全都涉及到内容。这使得“渐进增强”成为一种更为合理的设计范例。这也是它立即被 Yahoo! 所采纳并用以构建其“分级式浏览器支持 (Graded Browser Support)”策略的原因所在。

**那么问题了。现在产品经理看到IE6,7,8网页效果相对高版本现代浏览器少了很多圆角，阴影（CSS3），要求兼容（使用图片背景，放弃CSS3），你会如何说服他？**

**（自由发挥）**

**7.为什么利用多个域名来存储网站资源会更有效？**

* CDN缓存更方便
* 突破浏览器并发限制
* 节约cookie带宽
* 节约主域名的连接数，优化页面响应速度
* 防止不必要的安全问题

**8.请谈一下你对网页标准和标准制定机构重要性的理解。**

（无标准答案）网页标准和标准制定机构都是为了能让web发展的更‘健康’，开发者遵循统一的标准，降低开发难度，开发成本，SEO也会更好做，也不会因为滥用代码导致各种BUG、安全问题，最终提高网站易用性。

**9.请描述一下cookies，sessionStorage和localStorage的区别？**

sessionStorage用于本地存储一个会话（session）中的数据，这些数据只有在同一个会话中的页面才能访问并且当会话结束后数据也随之销毁。因此sessionStorage不是一种持久化的本地存储，仅仅是会话级别的存储。而localStorage用于持久化的本地存储，除非主动删除数据，否则数据是永远不会过期的。

**web storage和cookie的区别**

* Web Storage的概念和cookie相似，区别是它是为了更大容量存储设计的。Cookie的大小是受限的，并且每次你请求一个新的页面的时候Cookie都会被发送过去，这样无形中浪费了带宽，另外cookie还需要指定作用域，不可以跨域调用。
* 除此之外，Web Storage拥有setItem,getItem,removeItem,clear等方法，不像cookie需要前端开发者自己封装setCookie，getCookie。但是Cookie也是不可以或缺的：Cookie的作用是与服务器进行交互，作为HTTP规范的一部分而存在 ，而Web Storage仅仅是为了在本地“存储”数据而生。

**10.简述一下src与href的区别。**

答案：

src用于替换当前元素，href用于在当前文档和引用资源之间确立联系。

src是source的缩写，指向外部资源的位置，指向的内容将会嵌入到文档中当前标签所在位置；在请求src资源时会将其指向的资源下载并应用到文档内，例如js脚本，img图片和frame等元素。

<script src =”js.js”></script>

当浏览器解析到该元素时，会暂停其他资源的下载和处理，直到将该资源加载、编译、执行完毕，图片和框架等元素也如此，类似于将所指向资源嵌入当前标签内。这也是为什么将js脚本放在底部而不是头部。

href是Hypertext Reference的缩写，指向网络资源所在位置，建立和当前元素（锚点）或当前文档（链接）之间的链接，如果我们在文档中添加

<link href=”common.css” rel=”stylesheet”/>

那么浏览器会识别该文档为css文件，就会并行下载资源并且不会停止对当前文档的处理。这也是为什么建议使用link方式来加载css，而不是使用@import方式。

**11.知道的网页制作会用到的图片格式有哪些？**

答案：

png-8，png-24，jpeg，gif，svg。

但是上面的那些都不是面试官想要的最后答案。面试官希望听到是Webp,Apng。（是否有关注新技术，新鲜事物）

科普一下**Webp**：WebP格式，谷歌（google）开发的一种旨在加快图片加载速度的图片格式。图片压缩体积大约只有JPEG的2/3，并能节省大量的服务器带宽资源和数据空间。Facebook Ebay等知名网站已经开始测试并使用WebP格式。

在质量相同的情况下，[WebP](http://baike.baidu.com/view/4447461.htm)格式图像的体积要比JPEG格式图像小40%。

**Apng**：全称是**“Animated Portable Network Graphics”**, 是PNG的位图动画扩展，可以实现png格式的动态图片效果。04年诞生，但一直得不到各大浏览器厂商的支持，直到日前得到 iOS safari 8的支持，有望代替GIF成为下一代动态图标准。

**12.知道什么是微格式吗？谈谈理解。在前端构建中应该考虑微格式吗？**

答案：

微格式（Microformats）是一种让机器可读的语义化XHTML词汇的集合，是结构化数据的开放标准。是为特殊应用而制定的特殊格式。

优点：将智能数据添加到网页上，让网站内容在搜索引擎结果界面可以显示额外的提示。（应用范例：豆瓣，有兴趣自行google）

**13.在css/js代码上线之后开发人员经常会优化性能，从用户刷新网页开始，一次js请求一般情况下有哪些地方会有缓存处理？**

答案：dns缓存，cdn缓存，浏览器缓存，服务器缓存。

**14.一个页面上有大量的图片（大型电商网站），加载很慢，你有哪些方法优化这些图片的加载，给用户更好的体验。**

* 图片懒加载，在页面上的未可视区域可以添加一个滚动条事件，判断图片位置与浏览器顶端的距离与页面的距离，如果前者小于后者，优先加载。
* 如果为幻灯片、相册等，可以使用图片预加载技术，将当前展示图片的前一张和后一张优先下载。
* 如果图片为css图片，可以使用CSSsprite，SVGsprite，Iconfont、Base64等技术。
* 如果图片过大，可以使用特殊编码的图片，加载时会先加载一张压缩的特别厉害的缩略图，以提高用户体验。
* 如果图片展示区域小于图片的真实大小，则因在服务器端根据业务需要先行进行图片压缩，图片压缩后大小与展示一致。

**15.你如何理解HTML结构的语义化？**

* 去掉或样式丢失的时候能让页面呈现清晰的结构：

html本身是没有表现的，我们看到例如<h1>是粗体，字体大小2em，加粗；<strong>是加粗的，不要认为这是html的表现，这些其实html默认的css样式在起作用，所以去掉或样式丢失的时候能让页面呈现清晰的结构不是语义化的HTML结构的优点，但是浏览器都有有默认样式，默认样式的目的也是为了更好的表达html的语义，可以说浏览器的默认样式和语义化的HTML结构是不可分割的。

* 屏幕阅读器（如果访客有视障）会完全根据你的标记来“读”你的网页.

例如,如果你使用的含语义的标记,屏幕阅读器就会“逐个拼出”你的单词,而不是试着去对它完整发音.

* PDA、手机等设备可能无法像普通电脑的浏览器一样来渲染网页（通常是因为这些设备对CSS的支持较弱）

使用语义标记可以确保这些设备以一种有意义的方式来渲染网页.理想情况下,观看设备的任务是符合设备本身的条件来渲染网页.

语义标记为设备提供了所需的相关信息,就省去了你自己去考虑所有可能的显示情况（包括现有的或者将来新的设备）.例如,一部手机可以选择使一段标记了标题的文字以粗体显示.而掌上电脑可能会以比较大的字体来显示.无论哪种方式一旦你对文本标记为标题,您就可以确信读取设备将根据其自身的条件来合适地显示页面.

* 搜索引擎的爬虫也依赖于标记来确定上下文和各个关键字的权重

过去你可能还没有考虑搜索引擎的爬虫也是网站的“访客”,但现在它们他们实际上是极其宝贵的用户.没有他们的话,搜索引擎将无法索引你的网站,然后一般用户将很难过来访问.

* 你的页面是否对爬虫容易理解非常重要,因为爬虫很大程度上会忽略用于表现的标记,而只注重语义标记.

因此,如果页面文件的标题被标记,而不是,那么这个页面在搜索结果的位置可能会比较靠后.除了提升易用性外,语义标记有利于正确使用CSS和JavaScript,因为其本身提供了许多“钩钩”来应用页面的样式与行为.  
SEO主要还是靠你网站的内容和外部链接的。

* 便于团队开发和维护

W3C给我们定了一个很好的标准，在团队中大家都遵循这个标准，可以减少很多差异化的东西，方便开发和维护，提高开发效率，甚至[实现模块化开发](http://www.cssforest.org/blog/index.php?id=134)。

**16.谈谈以前端角度出发做好SEO需要考虑什么？**

* 了解搜索引擎如何抓取网页和如何索引网页

你需要知道一些搜索引擎的基本工作原理，各个搜索引擎之间的区别，搜索机器人（SE robot 或叫 web crawler）如何进行工作，搜索引擎如何对搜索结果进行排序等等。

* Meta标签优化

主要包括主题（Title)，网站描述(Description)，和关键词（Keywords）。还有一些其它的隐藏文字比如Author（作者），Category（目录），Language（编码语种）等。

* 如何选取关键词并在网页中放置关键词

搜索就得用关键词。关键词分析和选择是SEO最重要的工作之一。首先要给网站确定主关键词（一般在5个上下），然后针对这些关键词进行优化，包括关键词密度（Density），相关度（Relavancy），突出性（Prominency）等等。

* 了解主要的搜索引擎

虽然搜索引擎有很多，但是对网站流量起决定作用的就那么几个。比如英文的主要有Google，Yahoo，Bing等；中文的有百度，搜狗，有道等。不同的搜索引擎对页面的抓取和索引、排序的规则都不一样。还要了解各搜索门户和搜索引擎之间的关系，比如AOL网页搜索用的是Google的搜索技术，MSN用的是Bing的技术。

* 主要的互联网目录

Open Directory自身不是搜索引擎，而是一个大型的网站目录，他和搜索引擎的主要区别是网站内容的收集方式不同。目录是人工编辑的，主要收录网站主页；搜索引擎是自动收集的，除了主页外还抓取大量的内容页面。

* 按点击付费的搜索引擎

搜索引擎也需要生存，随着互联网商务的越来越成熟，收费的搜索引擎也开始大行其道。最典型的有Overture和百度，当然也包括Google的广告项目Google Adwords。越来越多的人通过搜索引擎的点击广告来定位商业网站，这里面也大有优化和排名的学问，你得学会用最少的广告投入获得最多的点击。

* 搜索引擎登录

网站做完了以后，别躺在那里等着客人从天而降。要让别人找到你，最简单的办法就是将网站提交（submit）到搜索引擎。如果你的是商业网站，主要的搜索引擎和目录都会要求你付费来获得收录（比如Yahoo要299美元），但是好消息是（至少到目前为止）最大的搜索引擎Google目前还是免费，而且它主宰着60％以上的搜索市场。

* 链接交换和链接广泛度（Link Popularity）

网页内容都是以超文本（Hypertext）的方式来互相链接的，网站之间也是如此。除了搜索引擎以外，人们也每天通过不同网站之间的链接来Surfing（“冲浪”）。其它网站到你的网站的链接越多，你也就会获得更多的访问量。更重要的是，你的网站的外部链接数越多，会被搜索引擎认为它的重要性越大，从而给你更高的排名。

* 合理的标签使用

**Css篇：**

**1.有哪项方式可以对一个DOM设置它的CSS样式？**

* 外部样式表，引入一个外部css文件
* 内部样式表，将css代码放在 <head> 标签内部
* 内联样式，将css样式直接定义在 HTML 元素内部

**2.CSS都有哪些选择器？**

* 派生选择器（用HTML标签申明）
* id选择器（用DOM的ID申明）
* 类选择器（用一个样式类名申明）
* 属性选择器（用DOM的属性申明，属于CSS2，IE6不支持，不常用，不知道就算了）

除了前3种基本选择器，还有一些扩展选择器，包括

* 后代选择器（利用空格间隔，比如div .a{  }）
* 群组选择器（利用逗号间隔，比如p,div,#a{  }）

那么问题来了，**CSS选择器的优先级是怎么样定义的？**

基本原则：

一般而言，选择器越特殊，它的优先级越高。也就是选择器指向的越准确，它的优先级就越高。

复杂的计算方法：

* 用1表示派生选择器的优先级
* 用10表示类选择器的优先级
* 用100标示ID选择器的优先级
  + div.test1 .span var 优先级 1+10 +10 +1
  + span#xxx .songs li 优先级1+100 + 10 + 1
  + #xxx li 优先级 100 +1

**那么问题来了，看下列代码，<p>标签内的文字是什么颜色的？。**

1 <style>

2 .classA{ color:blue;}

3

4 .classB{ color:red;}

5 </style>

6

7 <body>

8

9 <p class='classB classA'> 123 </p>

10

11 </body>

答案：red。与样式定义在文件中的先后顺序有关，即是后面的覆盖前面的，与在<p class=’classB classA’>中的先后关系无关。

3.**CSS中可以通过哪些属性定义，使得一个DOM元素不显示在浏览器可视范围内？**

最基本的：

设置display属性为none，或者设置visibility属性为hidden

技巧性：

设置宽高为0，设置透明度为0，设置z-index位置在-1000

4.**超链接访问过后hover样式就不出现的问题是什么？如何解决？**

答案：被点击访问过的超链接样式不在具有hover和active了,解决方法是改变CSS属性的排列顺序: L-V-H-A（link,visited,hover,active）

**5.什么是Css Hack？ie6,7,8的hack分别是什么？**

答案：针对不同的浏览器写不同的CSS code的过程，就是CSS hack。

示例如下：

1 #test {

2 width:300px;

3 height:300px;

4

5 background-color:blue; /\*firefox\*/

6 background-color:red\9; /\*all ie\*/

7 background-color:yellow\0; /\*ie8\*/

8 +background-color:pink; /\*ie7\*/

9 \_background-color:orange; /\*ie6\*/ }

10 :root #test { background-color:purple\9; } /\*ie9\*/

11 @media all and (min-width:0px){ #test {background-color:black\0;} } /\*opera\*/

12 @media screen and (-webkit-min-device-pixel-ratio:0){ #test {background-color:gray;} } /\*chrome and safari\*/

**6.请用Css写一个简单的幻灯片效果页面**

答案：知道是要用css3。使用animation动画实现一个简单的幻灯片效果。

1 /\*\*HTML\*\*/

2 div.ani

3

4 /\*\*css\*\*/

5 .ani{

6 width:480px;

7 height:320px;

8 margin:50px auto;

9 overflow: hidden;

10 box-shadow:0 0 5px rgba(0,0,0,1);

11 background-size: cover;

12 background-position: center;

13 -webkit-animation-name: "loops";

14 -webkit-animation-duration: 20s;

15 -webkit-animation-iteration-count: infinite;

16 }

17 @-webkit-keyframes "loops" {

18 0% {

19 background:url(http://d.hiphotos.baidu.com/image/w%3D400/sign=c01e6adca964034f0fcdc3069fc27980/e824b899a9014c08e5e38ca4087b02087af4f4d3.jpg) no-repeat;

20 }

21 25% {

22 background:url(http://b.hiphotos.baidu.com/image/w%3D400/sign=edee1572e9f81a4c2632edc9e72b6029/30adcbef76094b364d72bceba1cc7cd98c109dd0.jpg) no-repeat;

23 }

24 50% {

25 background:url(http://b.hiphotos.baidu.com/image/w%3D400/sign=937dace2552c11dfded1be2353266255/d8f9d72a6059252d258e7605369b033b5bb5b912.jpg) no-repeat;

26 }

27 75% {

28 background:url(http://g.hiphotos.baidu.com/image/w%3D400/sign=7d37500b8544ebf86d71653fe9f9d736/0df431adcbef76095d61f0972cdda3cc7cd99e4b.jpg) no-repeat;

29 }

30 100% {

31 background:url(http://c.hiphotos.baidu.com/image/w%3D400/sign=cfb239ceb0fb43161a1f7b7a10a54642/3b87e950352ac65ce2e73f76f9f2b21192138ad1.jpg) no-repeat;

32 }

33 }

**7.行内元素和块级元素的具体区别是什么？行内元素的padding和margin可设置吗？**

**块级元素(block)特性：**

* 总是独占一行，表现为另起一行开始，而且其后的元素也必须另起一行显示;
* 宽度(width)、高度(height)、内边距(padding)和外边距(margin)都可控制;

**内联元素(inline)特性：**

* 和相邻的内联元素在同一行;
* 宽度(width)、高度(height)、内边距的top/bottom(padding-top/padding-bottom)和外边距的top/bottom(margin-top/margin-bottom)都不可改变（也就是padding和margin的left和right是可以设置的），就是里面文字或图片的大小。

那么问题来了，**浏览器还有默认的天生inline-block元素（拥有内在尺寸，可设置高宽，但不会自动换行），有哪些**？

答案：<input> 、<img> 、<button> 、<texterea> 、<label>。

**8.什么是外边距重叠？重叠的结果是什么？**

答案：

外边距重叠就是margin-collapse。

在CSS当中，相邻的两个盒子（可能是兄弟关系也可能是祖先关系）的外边距可以结合成一个单独的外边距。这种合并外边距的方式被称为折叠，并且因而所结合成的外边距称为折叠外边距。

折叠结果遵循下列计算规则：

1. 两个相邻的外边距都是正数时，折叠结果是它们两者之间较大的值。
2. 两个相邻的外边距都是负数时，折叠结果是两者绝对值的较大值。
3. 两个外边距一正一负时，折叠结果是两者的相加的和。

**9.rgba()和opacity的透明效果有什么不同？**

答案：

rgba()和opacity都能实现透明效果，但最大的不同是opacity作用于元素，以及元素内的所有内容的透明度，

而rgba()只作用于元素的颜色或其背景色。（设置rgba透明的元素的子元素不会继承透明效果！）

**10.css中可以让文字在垂直和水平方向上重叠的两个属性是什么？**

答案：

垂直方向：line-height

水平方向：letter-spacing

**那么问题来了，关于letter-spacing的妙用知道有哪些么？**

答案:可以用于消除inline-block元素间的换行符空格间隙问题。

**11.如何垂直居中一个浮动元素？**

1 // 方法一：已知元素的高宽

2

3 #div1{

4 background-color:#6699FF;

5 width:200px;

6 height:200px;

7

8 position: absolute; //父元素需要相对定位

9 top: 50%;

10 left: 50%;

11 margin-top:-100px ; //二分之一的height，width

12 margin-left: -100px;

13 }

14

15 //方法二:未知元素的高宽

16

17 #div1{

18 width: 200px;

19 height: 200px;

20 background-color: #6699FF;

21

22 margin:auto;

23 position: absolute; //父元素需要相对定位

24 left: 0;

25 top: 0;

26 right: 0;

27 bottom: 0;

28 }

**那么问题来了，如何垂直居中一个<img>?**（用更简便的方法。）

1 #container //<img>的容器设置如下

2 {

3 display:table-cell;

4 text-align:center;

5 vertical-align:middle;

6 }

**12.px和em的区别。**

px和em都是长度单位，区别是，px的值是固定的，指定是多少就是多少，计算比较容易。em得值不是固定的，并且em会继承父级元素的字体大小。

浏览器的默认字体高都是16px。所以未经调整的浏览器都符合: 1em=16px。那么12px=0.75em, 10px=0.625em。

**13.描述一个”reset”的CSS文件并如何使用它。知道normalize.css吗？你了解他们的不同之处？**

重置样式非常多，凡是一个前端开发人员肯定有一个常用的重置CSS文件并知道如何使用它们。他们是盲目的在做还是知道为什么这么做呢？原因是不同的浏览器对一些元素有不同的默认样式，如果你不处理，在不同的浏览器下会存在必要的风险，或者更有戏剧性的性发生。

你可能会用[Normalize](http://necolas.github.io/normalize.css/)来代替你的重置样式文件。它没有重置所有的样式风格，但仅提供了一套合理的默认样式值。既能让众多浏览器达到一致和合理，但又不扰乱其他的东西（如粗体的标题）。

在这一方面，无法做每一个复位重置。它也确实有些超过一个重置，它处理了你永远都不用考虑的怪癖，像HTML的audio元素不一致或line-height不一致。

**14.Sass、LESS是什么？大家为什么要使用他们？**

他们是CSS预处理器。他是CSS上的一种抽象层。他们是一种特殊的语法/语言编译成CSS。

例如[Less](http://www.lesscss.org/)是一种动态样式语言. 将CSS赋予了动态语言的特性，如变量，继承，运算， 函数. LESS 既可以在客户端上运行 (支持IE 6+, Webkit, Firefox)，也可一在服务端运行 (借助 Node.js)。

为什么要使用它们？

* 结构清晰，便于扩展。
* 可以方便地屏蔽浏览器私有语法差异。这个不用多说，封装对浏览器语法差异的重复处理，减少无意义的机械劳动。
* 可以轻松实现多重继承。
* 完全兼容 CSS 代码，可以方便地应用到老项目中。LESS 只是在 CSS 语法上做了扩展，所以老的 CSS 代码也可以与 LESS 代码一同编译。

**15.display:none与visibility:hidden的区别是什么？**

* **display :**隐藏对应的元素但不挤占该元素原来的空间。
* **visibility:**隐藏对应的元素并且挤占该元素原来的空间。

即是，使用CSS display:none属性后，HTML元素（对象）的宽度、高度等各种属性值都将“丢失”;而使用visibility:hidden属性后，HTML元素（对象）仅仅是在视觉上看不见（完全透明），而它所占据的空间位置仍然存在。

**16.知道css有个content属性吗？有什么作用？有什么应用？**

答案：

知道。css的content属性专门应用在 before/after 伪元素上，用于来插入生成内容。

最常见的应用是利用伪类清除浮动。

1 //一种常见利用伪类清除浮动的代码

2 .clearfix:after {

3 content:"."; //这里利用到了content属性

4 display:block;

5 height:0;

6 visibility:hidden;

7 clear:both; }

8

9 .clearfix {

10 \*zoom:1;

11 }

after伪元素通过 content 在元素的后面生成了内容为一个点的块级元素，再利用clear:both清除浮动。

那么问题继续还有，**知道css计数器（序列数字字符自动递增）吗？如何通过css content属性实现css计数器？**

答案：css计数器是通过设置counter-reset 、counter-increment 两个属性 、及 counter()/counters()一个方法配合after / before 伪类实现。

# [JavaScript 编码实践](http://blog.csdn.net/cxl444905143/article/details/46859959)

分类： [JavaScript、CSS、HTML 理论知识](http://blog.csdn.net/cxl444905143/article/category/5644395)2015-07-13 11:06 5人阅读 [评论](http://blog.csdn.net/cxl444905143/article/details/46859959#comments)(0) [收藏](javascript:void(0);) [举报](http://blog.csdn.net/cxl444905143/article/details/46859959#report)

目录[(?)[+]](http://blog.csdn.net/cxl444905143/article/details/46859959)

## 以YYYY-MM-DD的格式输出今天的日期

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** Today () {
2. **var** d = **new** Date();
3. **var** year = d.getFullYear();  // 获取年，getFullYear()返回4位的数字
4. **var** month = d.getMonth() + 1;  // 获取月，月份比较特殊，0是1月，11是12月
5. month = month < 10 ? '0' + month : month;  // 变成两位
6. **var** day = d.getDate();  // 获取日
7. day = day < 10 ? '0' + day : day;
8. **return** year + '-' + month + '-' + day;
9. }

## 请写一个函数escapeHtml，将<, >, &, “进行转义

（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）

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** escapeHtml(str) {
2. **return** str.replace(/[<>"&]/g, **function**(match) {
3. **switch** (match) {
4. **case** "<":
5. **return** "<";
6. **case** ">":
7. **return** ">";
8. **case** "&":
9. **return** "&";
10. **case** "\"":
11. **return** """;
12. }
13. });
14. }

## 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。

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **var** iArray = [];
2. funtion getRandom(istart, iend){
3. **var** iChoice = istart - iend +1;
4. **return** Math.floor(Math.random() \* iChoice + istart;
5. }
6. **for**(**var** i=0; i<10; i++){
7. iArray.push(getRandom(10,100));
8. }
9. iArray.sort();

## 清除字符串前后的空格的函数，使用自带接口trim()，考虑兼容性

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **if** (!String.prototype.trim) {
2. String.prototype.trim = **function**() {
3. **return** **this**.replace(/^\s+/, "").replace(/\s+$/,"");
4. }
5. }
6. // test the function
7. **var** str = " \t\n test string ".trim();
8. alert(str == "test string"); // alerts "true"

## 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

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. //  方法一：
2. Object.prototype.clone = **function**(){
3. **var** o = **this**.constructor === Array ? [] : {};
4. **for**(**var** e **in** **this**){
5. o[e] = **typeof** **this**[e] === "object" ? **this**[e].clone() : **this**[e];
6. }
7. **return** o;
8. }
10. **function** clone (obj) {
11. **if** (obj **instanceof** Object) {
12. **return** obj.clone();
13. }
14. **return** obj;
15. }

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. //  方法二
2. **function** clone(obj) {
3. **var** buf;
4. **var** i;
5. **if** (obj **instanceof** Array) {
6. buf = [];
7. i = obj.length
8. **while**(i--) {
9. buf[i] = clone(obj[i]);
10. }
11. **return** buf;
12. } **else** **if** (obj **instanceof** Object) {
13. buf = {};
14. **for** (**var** e **in** obj) {
15. buf[e] = clone(obj[e]);
16. }
17. **return** buf;
18. } **else** {
19. **return** obj;
20. }
21. }

## 如何消除一个数组里面重复的元素？

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. //  方法一
2. **function** deRepeat(array){
3. **var** newArr = [];
4. **var** obj = {};
5. **var** index = 0;
6. **var** len = array.length;
7. **for**(**var** i = 0;i < l; i++){
8. **if**(obj[array[i]] == undefined) {
9. obj[array[i]] = **true**;
10. newArr[index++]=array[i];
11. }
12. **else** **if** (obj[array[i]] == **true**) {
13. **continue**;
14. }
15. }
16. **return** newArr;
17. }

方法二：

ES6 里使用 var newArr = new Set(array)

## 下面这个ul，如何点击每一列的时候alert其index?（闭包）

**[html]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **<ul** id=”test”**>**
2. **<li>**这是第一条**</li>**
3. **<li>**这是第二条**</li>**
4. **<li>**这是第三条**</li>**
5. **</ul>**

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. // 方法一：
2. **var** lis=document.getElementById('test').getElementsByTagName('li');
3. **for**(**var** i=0;i<3;i++)
4. {
5. lis[i].index=i;
6. lis[i].onclick=**function**(){
7. alert(**this**.index);
8. };
9. }
11. //方法二：
12. **var** lis=document.getElementById('test').getElementsByTagName('li');
13. **for**(**var** i=0;i<3;i++)
14. {
15. lis[i].onclick=(**function**(a){
16. **return** **function**() {
17. alert(a);
18. }
19. })(i);
20. }

## 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}\*/

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** query (selector) {
3. **var** reg = /^(#)?(\.)?(\w+)$/img;
4. **var** regResult = reg.exec(selector);
5. **var** result = [];
7. //如果是id选择器
8. **if**(regResult[1] && regResult[3]) {
9. **if**(**typeof** document.querySelector === "function") {
10. result.push(document.querySelector(regResult[0]));  //  用 result[result.length] = ..速度更快
11. }
12. **else** {
13. result.push(document.getElementById(regResult[3]));
14. }
15. }
16. //如果是class选择器
17. **else** **if**(regResult[2] && regResult[3]) {
18. **if**(**typeof** document.getElementsByClassName === 'function') {
19. **var** doms = document.getElementsByClassName(regResult[3]);
20. **if**(doms) {
21. result = converToArray(doms);
22. }
23. } **else** {  //如果不支持getElementsByClassName函数
24. **var** allDoms = document.getElementsByTagName("\*") ;
25. **for**(**var** i = 0, len = allDoms.length; i < len; i++) {
26. **if**(allDoms[i].className.search(**new** RegExp(regResult[2])) > -1) {
27. result.push(allDoms[i]);
28. }
29. }
30. }
31. }
32. //如果是标签选择器
33. **else** **if**(regResult[3]) {
34. **var** doms = document.getElementsByTagName(regResult[3].toLowerCase());
35. **if**(doms) {
36. result = converToArray(doms);
37. }
38. }
39. **return** result;
40. }
42. **function** converToArray(nodes){
43. **var** array = **null**;
44. **try**{
45. array = Array.prototype.slice.call(nodes,0);//针对非IE浏览器
46. }**catch**(ex){
47. array = **new** Array();
48. **for**( **var** i = 0 ,len = nodes.length; i < len ; i++ ) {
49. array.push(nodes[i])
50. }
51. }
52. **return** array;
53. }

## 如何用原生JS实现Jq的ready方法？

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** ready(fn){
2. **if**(document.addEventListener) {        //标准浏览器
3. document.addEventListener('DOMContentLoaded', **function**() {
4. //注销事件, 避免反复触发
5. document.removeEventListener('DOMContentLoaded',arguments.callee, **false**);
6. fn();            //执行函数
7. }, **false**);
8. }**else** **if**(document.attachEvent) {        //IE
9. document.attachEvent('onreadystatechange', **function**() {
10. **if**(document.readyState == 'complete') {
11. document.detachEvent('onreadystatechange', arguments.callee);
12. fn();        //函数执行
13. }
14. });
15. }
16. };

## 第一次显示 tips 实现



**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** setcookie(name,value,days){  //给cookie增加一个时间变量
2. **var** exp = **new** Date();
3. exp.setTime(exp.getTime() + days\*24\*60\*60\*1000); //设置过期时间为days天
4. document.cookie = name + "="+ escape(value) + ";expires=" + exp.toGMTString();
5. }
7. **function** getCookie(name){
8. **var** result = "";
9. **var** myCookie = document.cookie;
10. **var** searchName = "name" + "=";
11. **var** startOfCookie = myCookie.indexOf(searchName);
12. **var** endOfCookie;
13. **if**(satrtOfCookie != -1){
14. startOfcookie += searchName.length;
15. endOfCookie = myCookie.indexOf(";",startOfCookie);
16. result = myCookie.substring(startOfCookie,endOfCookie);
17. }
18. **return** result;
19. }
21. **var** dom = document.getElementById("test");
22. **var** page = {
23. init: **function** () {
24. **var** result = **this**.check();
25. **if** (result) {
26. **this**.showTip();
27. setCookie("tips", "showed", 1);
28. } **else** {
29. **this**.hideTip();
30. }
31. dom.onclick = **function** () {
32. page.hideTip();
34. };
35. },
36. check: **function** () {
37. **var** tips = getCookie("tips");
38. **if** (!tips) {   // 如果cookie过期了，这个cookie 的值是不存在了，会查询不到对应的name
39. **return** **true**;
40. } **else** **if** (tips == "never" || tips == "showed") {
41. **return** **false**;
42. }
43. },
44. showTip: **function** () {
45. dom.style.display = "inline-block";
46. },
47. hideTip: **function** () {
48. dom.style.display = "none";
49. setCookie("tips", "never", 365);
50. }
51. };
52. page.init();

### 用js实现如下功能，将给定的数字转化成千分位的格式 （待学习）

http://www.jb51.net/article/46300.htm

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **var** cuter2 = **function**( str ){//abcd
2. **return** str.replace( /\B(?=(?:\d{3})+$)/g, ',' );
3. };

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** spli(str){
2. **var** re = /(?=(?!\b)(\d{3})+$)/g;
3. **return** str.replace(re,',');
4. }

其中的(?=exp)用来匹配exp前面的位置，(?!exp)匹配后面跟的不是exp的位置

### 关于函数传参  值传递

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **var** ftn1 = **function**() {
2. console.log("test:ftn1");
3. };
5. **var** ftn2 = **function**() {
6. console.log("test:ftn2");
7. };
9. **function** ftn(f){
10. f();   // 运行结果：test:ftn1
11. <span style="white-space:pre">    </span>f = ftn2;
12. f();   // 运行结果：test:ftn2
13. }
14. ftn(ftn1);
16. console.log("====================华丽的分割线======================");
18. ftn1();// 运行结果：test:ftn1

执行 ftn(ftn1) 之前，栈内存中：

|  |  |
| --- | --- |
| **变量标识符** | **变量值** |
| ftn1 | &ftn1 |
| ftn2 | &ftn2 |

执行  ftn(ftn1) 之后， 执行  f = ftn2 之前，栈内存中：

|  |  |
| --- | --- |
| **变量标识符** | **变量值** |
| ftn1 | &ftn1 |
| ftn2 | &ftn2 |
| f | &ftn1 |

执行  f = ftn2 之后，栈内存中：

|  |  |
| --- | --- |
| **变量标识符** | **变量值** |
| ftn1 | &ftn1 |
| ftn2 | &ftn2 |
| f | &ftn2 |

### 生成随机的字母数字字符串

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **function** generateRandomAlphaNum(len) {
2. **var** rdmString = "";
3. **for**( ; rdmString.length < len; rdmString  += Math.random().toString(36).substr(2));
4. **return**  rdmString.substr(0, len);
5. }

Number 的 toString() 方法接受一个表示基数的参数，告诉它返回几进制数值的字符串形式。

例如：

**[javascript]** [view plaincopy](http://blog.csdn.net/cxl444905143/article/details/46859959)

1. **var** num = 10;
2. alert(num.toString(8)); // "12"