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：

```
<span style="font-family: verdana, geneva;">if(typeof  Array.isArray==="undefined")
{
    Array.isArray =  function(arg){
                return  Object.prototype.toString.call(arg)==="[object Array]"
        };
}
</span>
```

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)

```
<span style="font-family: verdana, geneva;">document.getElementById(“ID”).value
</span>
```

4.希望获取到页面中所有的 checkbox 怎么做？(不使用第三方框架)

```
<span style="font-family: verdana, geneva;">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]);
    }
}
```

```
</span>
```

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)

```
<span style="font-family: verdana, geneva;">var  dom =
document.getElementById(“ID”);
dom.innerHTML = “xxxx”;
dom.style.color = “#000”;
</span>
```

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：
- 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。

```
<span style="font-family: verdana, geneva;">var  a;
alert(typeof  a);  // undefined
alert(b);  // 报错</span>
```

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。

```
<span style="font-family: verdana, geneva;">var  a =  null;
alert(typeof  a);  //object
</span>
```

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。

```
<span style="font-family: verdana, geneva;">var  undefined;
undefined ==  null;  // true
1 ==  true;      // true
2 ==  true;      // false
0 ==  false;    // true
0 ==  '';          // true
NaN == NaN;    // false
[] ==  false;  // true
[] == ![];      // true</span>
```

- 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？

```
<span style="font-family: verdana, geneva;">var  foo =  "11"+2-"1";
console.log(foo);
console.log(typeof  foo);</span>
```

执行完后 foo 的值为 111，foo 的类型为 Number。

```
<span style="font-family: verdana, geneva;">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
console.log(foo);
console.log(typeof  foo);</span>
```

执行完后 foo 的值为 1121(此处是字符串拼接)，foo 的类型为 String。

11.看代码给答案。

```
<span style="font-family: verdana, geneva;">var  a =  new  Object();
a.value = 1;
b = a;
b.value = 2;
alert(a.value);
</span>
```

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(" "))

那么问题来了，**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"。**

```
<span style="font-family: verdana, geneva;">function  combo(msg){
        var  arr = msg.split("-");
        var  len = arr.length;
        //将 arr.length 存储在一个局部变量可以提高 for 循环效率
        for(var  i=1;i<len;i++){
                arr[i]=arr[i].charAt(0).toUpperCase()+arr[i].substr(1,arr[i].length-1);
        }
        msg=arr.join("");
        return  msg;
}</span>
```

(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]

```
<span style="font-family: verdana, geneva;">var  numberArray = [3,6,2,4,1,5];
numberArray.reverse();  // 5,1,4,2,6,3
numberArray.sort(function(a,b){    //6,5,4,3,2,1
      return  b-a;
})
</span>
```

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

```
<span style="font-family: verdana, geneva;">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);
</span>
```

15.将字符串"<tr><td>{id}</td><td>{id}</td><td>{name}</td></tr>"中的{id}替换成 10，{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}</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

```
<span style="font-family: verdana, geneva;">function  escapeHtml(str) {
```

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

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)

```
<span style="font-family: verdana, geneva;">var  foo = 1;
function(){
        console.log(foo);
        var  foo = 2;
        console.log(foo);
} </span>
```

答案：输出 undefined 和 2。上面代码相当于：

```
<span style="font-family: verdana, geneva;">var  foo = 1;
function(){
        var  foo;
        console.log(foo);  //undefined
        foo = 2;
        console.log(foo);  // 2;
} </span>
```

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。

```
<span style="font-family: verdana, geneva;">var  iArray = [];
funtion getRandom(istart, iend){
                var  iChoice = istart - iend +1;
                return  Math.floor(Math.random() * iChoice + istart;
}
for(var  i=0; i<10; i++){
                iArray.push(getRandom(10,100));
}
iArray.sort();</span>
```

20.把两个数组合并，并删除第二个元素。

```
<span style="font-family: verdana, geneva;">var  array1 = ['a','b','c'];
var  bArray = ['d','e','f'];
var  cArray = array1.concat(bArray);
cArray.splice(1,1);
</span>
```

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：

```
function  serilizeUrl(url) {
        var  result = {};
        url = url.split("?")[1];
        var  map = url.split("&");
        for(var  i = 0, len = map.length; i < len; i++) {
                result[map[i].split("=")[0]] = map[i].split("=")[1];
        }
        return  result;
}
```

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：

```
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。

```
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。那么问题来了，**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
```

```
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

```
// 方法一：
Object.prototype.clone =  function() {
        var  o =  this.constructor === Array ? [] : {};
        for  (var  e  in  this) {
                o[e] =  typeof  this[e] ==="object"?this[e].clone():this[e];
        }
        return  o;
}

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

2.如何消除一个数组里面重复的元素？

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

function  deRepeat() {
        var  newArr = [];
        var  obj = {};
        var  index = 0;
        var  l = arr.length;
        for  (var  i = 0; i < l; i++) {
                if  (obj[arr[i]] == undefined) {
                        obj[arr[i]] = 1;
                        newArr[index++] = arr[i];
                }  else  if  (obj[arr[i]] == 1)
                        continue;
        }
        return  newArr;

}
var  newArr2 = deRepeat(arr);
alert(newArr2);  //输出 1,2,3,4,5,6,9,25
```

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：

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

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

答案：

```
function  MadDog() {
        this.yelp =  function() {
                    var  self =  this;
                    setInterval(function() {
                                self.wow();
                    }, 500);
          }
}
MadDog.prototype =  new  Dog();

```

```
//for test
var  dog =  new  Dog();
dog.yelp();
var  madDog =  new  MadDog();
madDog.yelp();
```

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

```
<ul  id=”test”>
        <li>这是第一条</li>
        <li>这是第二条</li>
        <li>这是第三条</li>
</ul>
```

答案：

```
// 方法一：
var  lis=document.getElementById('2223').getElementsByTagName('li');
for(var  i=0;i<3;i++)
{
        lis[i].index=i;
        lis[i].onclick=function(){
                alert(this.index);
        };
}

//方法二：
var  lis=document.getElementById('2223').getElementsByTagName('li');
for(var  i=0;i<3;i++)
{
        lis[i].index=i;
        lis[i].onclick=(function(a){
                return  function() {
                        alert(a);
                }
        })(i);
}
```

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。

答案：（过长，点击打开）

View Code

6.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：

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

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'

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

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：

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

如果要传入多个参数呢？显然上面的方法不能满足要求，所以更好的方法是：

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

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：

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

        console.log.apply(console, args);
};
```

10.对作用域上下文和 this 的理解，看下列代码：

```
var  User = {
        count: 1,

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

问两处 console 输出什么？为什么？

答案是 1 和 undefined。

func 是在 winodw 的上下文中被执行的，所以会访问不到 count 属性。

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：

```
Function.prototype.bind = Function.prototype.bind ||  function(context) {
        var  self =  this;

        return  function() {
                return  self.apply(context, arguments);
        };
}

var  func = User.getCount.bind(User);
console.log(func());
```

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。

```
/*
  * 传递函数给 whenReady()
  * 当文档解析完毕且为操作准备就绪时，函数作为 document 的方法调用
  */
var  whenReady = (function() {  //这个函数返回 whenReady()函数
        var  funcs = [];  //当获得事件时，要运行的函数
        var  ready =  false;  //当触发事件处理程序时,切换为 true

        //当文档就绪时,调用事件处理程序
        function  handler(e) {
                if  (ready)  return;  //确保事件处理程序只完整运行一次

                //如果发生 onreadystatechange 事件，但其状态不是 complete 的话,
那么文档尚未准备好
                if  (e.type ===  'onreadystatechange'  &&
document.readyState !==  'complete') {
                        return;
                }

                //运行所有注册函数
                //注意每次都要计算 funcs.length
                //以防这些函数的调用可能会导致注册更多的函数
                for  (var  i = 0; i < funcs.length; i++) {
                        funcs[i].call(document);
                }
                //事件处理函数完整执行,切换 ready 状态, 并移除所有函数
                ready =  true;
                funcs =  null;
        }
        //为接收到的任何事件注册处理程序
        if  (document.addEventListener) {
                document.addEventListener('DOMContentLoaded',
handler,  false);
                document.addEventListener('readystatechange',
handler,  false);  //IE9+
                window.addEventListener('load', handler,  false);
        }  else  if  (document.attachEvent) {
                document.attachEvent('onreadystatechange', handler);
                window.attachEvent('onload', handler);
        }
        //返回 whenReady()函数
```

```
        return  function  whenReady(fn) {
                if  (ready) {
                        fn.call(document);
                }  else  {
                        funcs.push(fn);
                }
        }
})();
```

如果上述代码十分难懂，下面这个简化版：

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

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

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

13.

请实现下面功能（只实现tips组件部分）

1. 用户第一次进来时显示，同一天访问该页面不显示tip提示
2. 用户点击"我知道了"此后访问该页面不再显示tip提醒。


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

        var  exp =  new  Date();
        exp.setTime(exp.getTime() + days * 24 * 60 * 60 * 1000);  //设置过期时间为 days 天

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

function  getCookie(name) {
        var  result =  "";
        var  myCookie =  ""  + document.cookie +  ";";
        var  searchName =  "+name+"  =  ";
        var startOfCookie = myCookie.indexOf(searchName);
        var endOfCookie;
        if (satrtOfCookie != -1) {
                startOfcookie += searchName.length;
                endOfCookie = myCookie.indexOf(";", startOfCookie);
                result = (myCookie.substring(startOfCookie, endOfCookie));
        }
        return result;
}
(function() {
        var oTips = document.getElementById('tips'); //假设 tips 的 id 为 tips

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

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

                        if (tips == "never_show_again") return false;
                },
                    hideTip: function(bNever) {
                        if (bNever) setcookie('tips', 'never_show_again', 365);
                        oTips.style.display = "none"; //隐藏

                },
                    showTip: function() {
                        oTips.style.display = "inline";  //显示，假设 tips 为行级元素

```

```
                },
                    init:  function() {
                        var  _this =  this;
                        if  (this.check()) {
                                _this.showTip();
                                setcookie('tips',  'show', 1);
                        }
                        oTips.onclick =  function() {
                                _this.hideTip(true);
                        };
                }
        };
        page.init();
})();
```

14.说出以下函数的作用是？空白区域应该填写什么？

```
//define
(function(window) {
        function  fn(str) {
                this.str = str;
        }

        fn.prototype.format =  function() {
                var  arg = ______;
                return  this.str.replace(_____,  function(a, b) {
                        return  arg[b] ||  "";
                });
        }
        window.fn = fn;
})(window);

//use
(function() {
        var  t =  new  fn('<p><a href="{0}">{1}</a><span>{2}</span></p>');
        console.log(t.format('http://www.alibaba.com',  'Alibaba',  'Welcome'));
})();
```

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

第一个空是：**arguments**

第二个空是：**/\{(\d+)\}/ig**

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。

16.讲解原生 Js 实现 ajax 的原理。

Ajax 的全称是 Asynchronous JavaScript and XML，其中，Asynchronous 是异步的意思，它有别于传统 web 开发中采用的同步的方式。

Ajax 的原理简单来说通过 XmlHttpRequest 对象来向服务器发异步请求，从服务器获得数据，然后用 javascript 来操作 DOM 而更新页面。

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

XMLHttpRequest 这个对象的属性有：

- onreadystatechang　　每次状态改变所触发事件的事件处理程序。
- responseText　　从服务器进程返回数据的字符串形式。
- responseXML　　从服务器进程返回的 DOM 兼容的文档数据对象。
- status　　从服务器返回的数字代码，比如常见的 404（未找到）和 200（已就绪）
- status Text　　伴随状态码的字符串信息
- readyState　　对象状态值
  - 0 (未初始化) 对象已建立，但是尚未初始化（尚未调用 open 方法）
    - 1 (初始化) 对象已建立，尚未调用 send 方法
    - 2 (发送数据) send 方法已调用，但是当前的状态及 http 头未知
    - 3 (数据传送中) 已接收部分数据，因为响应及 http 头不全，这时通过 responseBody 和 responseText 获取部分数据会出现错误，
    - 4 (完成) 数据接收完毕,此时可以通过通过 responseXml 和 responseText 获取完整的回应数据

下面简单封装一个函数：（略长，点击打开）

View Code

上述代码大致表述了 ajax 的过程，释义自行 google，问题未完，**那么知道什么是 Jsonp 和 pjax 吗？**

答案：

Jsonp：(JSON with Padding)是一种跨域请求方式。主要原理是利用了 script 标签可以跨域请求的特点，由其 src 属性发送请求到服务器，服务器返回 js 代码，网页端接受响应，然后就直接执行了，这和通过 script 标签

引用外部文件的原理是一样的。JSONP 由两部分组成：回调函数和数据，回调函数一般是由网页端控制，作为参数发往服务器端，服务器端把该函数和数据拼成字符串返回。

pjax：pjax 是一种基于 ajax+history.pushState 的新技术，该技术可以无刷新改变页面的内容，并且可以改变页面的 URL。（关键点：可以实现 ajax 无法实现的后退功能）pjax 是 `ajax` + `pushState` 的封装，同时支持本地存储、动画等多种功能。目前支持 jquery、qwrap、kissy 等多种版本。