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\*\*，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 = “\*\*”
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：\*\*.onclick = test   
• 通过事件添加进行绑定：addEventListener(\*\*, ‘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

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

*复制代码*

　　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

*复制代码*

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

*复制代码*

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？

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

*复制代码*

　　执行完后foo的值为111，foo的类型为String。

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. }

　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. numberArray.reverse(); // 5,1,4,2,6,3
3. numberArray.sort(function(a,b){  //6,5,4,3,2,1
4. return b-a;
5. })

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);

　　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. }

*复制代码*

　　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. }

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);
5. 谈谈This对象的理解。
6. this是js的一个关键字，随着函数使用场合不同，this的值会发生变化。
7. 但是有一个总原则，那就是this指的是调用函数的那个对象。

this一般情况下：是全局对象Global。 作为方法调用，那么this就是指这个对象

什么是闭包（closure），为什么要用它？

<http://www.ruanyifeng.com/blog/2009/08/learning_javascript_closures.html>

闭包就是能够读取其他函数内部变量的函数。

由于在Javascript语言中，只有函数内部的子函数才能读取局部变量，因此可以把闭包简单理解成"定义在一个函数内部的函数"。

所以，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。

**闭包的用途**

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。

怎么来理解这句话呢？请看下面的代码。

　　function f1(){

　　　　var n=999;

　　　　nAdd=function(){n+=1}

　　　　function f2(){  
　　　　　　alert(n);  
　　　　}

　　　　return f2;

　　}

　　var result=f1();

　　result(); // 999

　　nAdd();

　　result(); // 1000

在这段代码中，result实际上就是闭包f2函数。它一共运行了两次，第一次的值是999，第二次的值是1000。这证明了，函数f1中的局部变量n一直保存在内存中，并没有在f1调用后被自动清除。

为什么会这样呢？原因就在于f1是f2的父函数，而f2被赋给了一个全局变量，这导致f2始终在内存中，而f2的存在依赖于f1，因此f1也始终在内存中，不会在调用结束后，被垃圾回收机制（garbage collection）回收。

这段代码中另一个值得注意的地方，就是"nAdd=function(){n+=1}"这一行，首先在nAdd前面没有使用var关键字，因此nAdd是一个全局变量，而不是局部变量。其次，nAdd的值是一个匿名函数（anonymous function），而这个匿名函数本身也是一个闭包，所以nAdd相当于是一个setter，可以在函数外部对函数内部的局部变量进行操作。

* new操作符具体干了什么呢?
* 1、创建一个空对象，并且 this 变量引用该对象，同时还继承了该函数的原型。
* 2、属性和方法被加入到 this 引用的对象中。
* 3、新创建的对象由 this 所引用，并且最后隐式的返回 this 。
* var obj = {};
* obj.\_\_proto\_\_ = Base.prototype;

Base.call(obj);

## 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

## .call() 和 .apply() 的区别？

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

注意：js 中的函数其实是对象，函数名是对 Function 对象的引用。

function add(a,b)

{

alert(a+b);

}

function sub(a,b)

{

alert(a-b);

}

add.call(sub,3,1);

## 函数调用的三种方式: obj.myFunc(); myFunc.call(obj,arg); myFunc.apply(obj,[arg1,arg2..]);



## Jquery与jQuery UI 有啥区别？

* \*jQuery是一个js库，主要提供的功能是选择器，属性修改和事件绑定等等。
* \*jQuery UI则是在jQuery的基础上，利用jQuery的扩展性，设计的插件。

提供了一些常用的界面元素，诸如对话框、拖动行为、改变大小行为等等

## 针对 jQuery 的优化方法？

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

\*频繁操作的DOM，先缓存起来再操作。用Jquery的链式调用更好。

比如：var str=$("a").attr("href");

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

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

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

## offsetWidth/offsetHeight,clientWidth/clientHeight与scrollWidth/scrollHeight的区别

* offsetWidth/offsetHeight返回值包含**content + padding + border**，效果与e.getBoundingClientRect()相同
* clientWidth/clientHeight返回值只包含**content + padding**，如果有滚动条，也**不包含滚动条**
* scrollWidth/scrollHeight返回值包含**content + padding + 溢出内容的尺寸**

## mouseover/mouseout与mouseenter/mouseleave的区别与联系

1. mouseover/mouseout是标准事件，**所有浏览器都支持**；mouseenter/mouseleave是IE5.5引入的特有事件后来被DOM3标准采纳，现代标准浏览器也支持
2. mouseover/mouseout是**冒泡**事件；mouseenter/mouseleave**不冒泡**。需要为**多个元素监听鼠标移入/出事件时，推荐mouseover/mouseout托管，提高性能**
3. 标准事件模型中event.target表示发生移入/出的元素,**vent.relatedTarget**对应移出/如元素；在老IE中event.srcElement表示发生移入/出的元素，**event.toElement**表示移出的目标元素，**event.fromElement**表示移入时的来源元素

## sessionStorage,localStorage,cookie区别

1. 都会在浏览器端保存，有大小限制，同源限制
2. cookie会在请求时发送到服务器，作为会话标识，服务器可修改cookie；web storage不会发送到服务器
3. cookie有path概念，子路径可以访问父路径cookie，父路径不能访问子路径cookie
4. 有效期：cookie在设置的有效期内有效，默认为浏览器关闭；sessionStorage在窗口关闭前有效，localStorage长期有效，直到用户删除
5. 共享：sessionStorage不能共享，localStorage在同源文档之间共享，cookie在同源且符合path规则的文档之间共享
6. localStorage的修改会促发其他文档窗口的update事件
7. cookie有secure属性要求HTTPS传输
8. 浏览器不能保存超过300个cookie，单个服务器不能超过20个，每个cookie不能超过4k。web storage大小支持能达到5M

## 什么闭包,闭包有什么用

**闭包是在某个作用域内定义的函数，它可以访问这个作用域内的所有变量**。闭包作用域链通常包括三个部分：

1. 函数本身作用域。
2. 闭包定义时的作用域。
3. 全局作用域。

闭包常见用途：

1. 创建特权方法用于访问控制
2. 事件处理程序及回调

## 客户端存储localStorage和sessionStorage

* localStorage有效期为永久，sessionStorage有效期为顶层窗口关闭前
* 同源文档可以读取并修改localStorage数据，sessionStorage只允许同一个窗口下的文档访问，如通过iframe引入的同源文档。
* Storage对象通常被当做普通javascript对象使用：**通过设置属性来存取字符串值**，也可以通过**setItem(key, value)设置**，**getItem(key)读取**，**removeItem(key)删除**，**clear()删除所有数据**，**length表示已存储的数据项数目**，**key(index)返回对应索引的key**

## javascript有哪些方法定义对象

1. 对象字面量： var obj = {};
2. 构造函数： var obj = new Object();
3. Object.create(): var obj = Object.create(Object.prototype);

## ===运算符判断相等的流程是怎样的

1. 如果两个值不是相同类型，它们不相等
2. 如果两个值都是null或者都是undefined，它们相等
3. 如果两个值都是布尔类型true或者都是false，它们相等
4. 如果其中有一个是**NaN**，它们不相等
5. 如果都是数值型并且数值相等，他们相等， -0等于0
6. 如果他们都是字符串并且在相同位置包含相同的16位值，他它们相等；如果在长度或者内容上不等，它们不相等；两个字符串显示结果相同但是编码不同==和===都认为他们不相等
7. 如果他们指向相同对象、数组、函数，它们相等；如果指向不同对象，他们不相等

## ==运算符判断相等的流程是怎样的

1. 如果两个值类型相同，按照===比较方法进行比较
2. 如果类型不同，使用如下规则进行比较
   1. 如果其中一个值是null，另一个是undefined，它们相等
   2. 如果一个值是**数字**另一个是**字符串**，将**字符串转换为数字**进行比较
   3. 如果有布尔类型，将**true转换为1，false转换为0**，然后用==规则继续比较
   4. 如果一个值是对象，另一个是数字或字符串，将对象转换为原始值然后用==规则继续比较
   5. **其他所有情况都认为不相等**

## 评价一下三种方法实现继承的优缺点,并改进

function Shape() {}

function Rect() {}

// 方法1

Rect.prototype = new Shape();

// 方法2

Rect.prototype = Shape.prototype;

// 方法3

Rect.prototype = Object.create(Shape.prototype);

Rect.prototype.area = function () {

// do something

};

方法1：

1. 优点：正确设置原型链实现继承
2. 优点：父类实例属性得到继承，原型链查找效率提高，也能为一些属性提供合理的默认值
3. 缺点：父类实例属性为引用类型时，不恰当地修改会导致所有子类被修改
4. 缺点：创建父类实例作为子类原型时，可能无法确定构造函数需要的合理参数，这样提供的参数继承给子类没有实际意义，当子类需要这些参数时应该在构造函数中进行初始化和设置
5. 总结：继承应该是继承方法而不是属性，为子类设置父类实例属性应该是通过在子类构造函数中调用父类构造函数进行初始化

方法2：

1. 优点：正确设置原型链实现继承
2. 缺点：父类构造函数原型与子类相同。修改子类原型添加方法会修改父类

方法3：

1. 优点：正确设置原型链且避免方法1.2中的缺点
2. 缺点：ES5方法需要注意兼容性

改进：

1. 所有三种方法应该在子类构造函数中调用父类构造函数实现实例属性初始化

function Rect() {

Shape.call(this);

}

1. 用新创建的对象替代子类默认原型，设置Rect.prototype.constructor = Rect;保证一致性
2. 第三种方法的polyfill：

function create(obj) {

if (Object.create) {

return Object.create(obj);

}

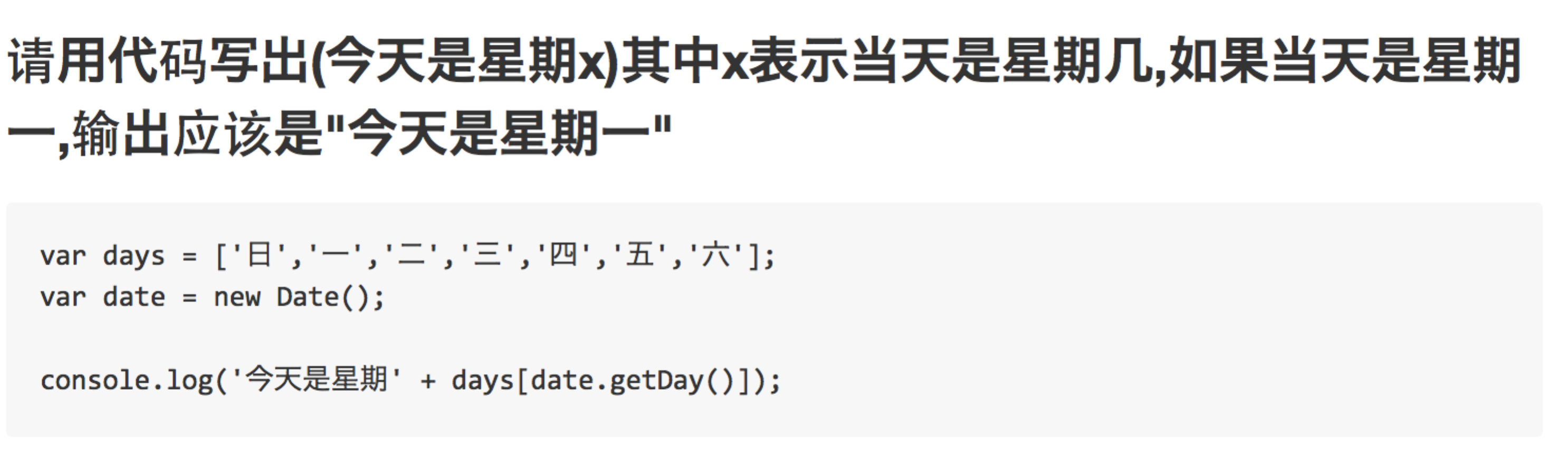
function f() {};

f.prototype = obj;

return new f();

}

# $javascript编程部分



## parseFloat的用法

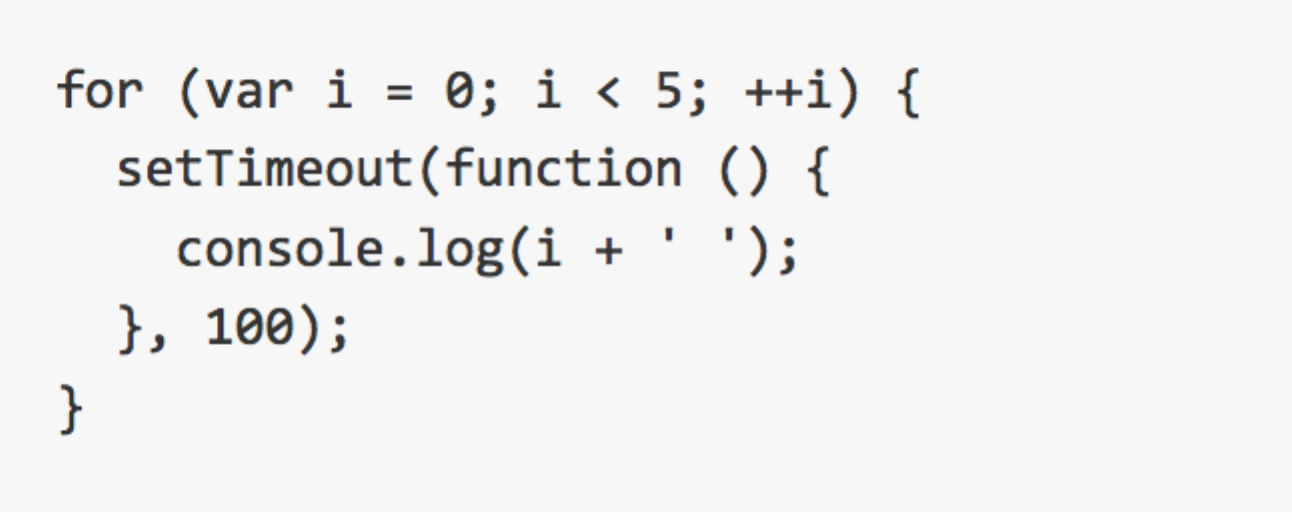


## appendChild的使用方法

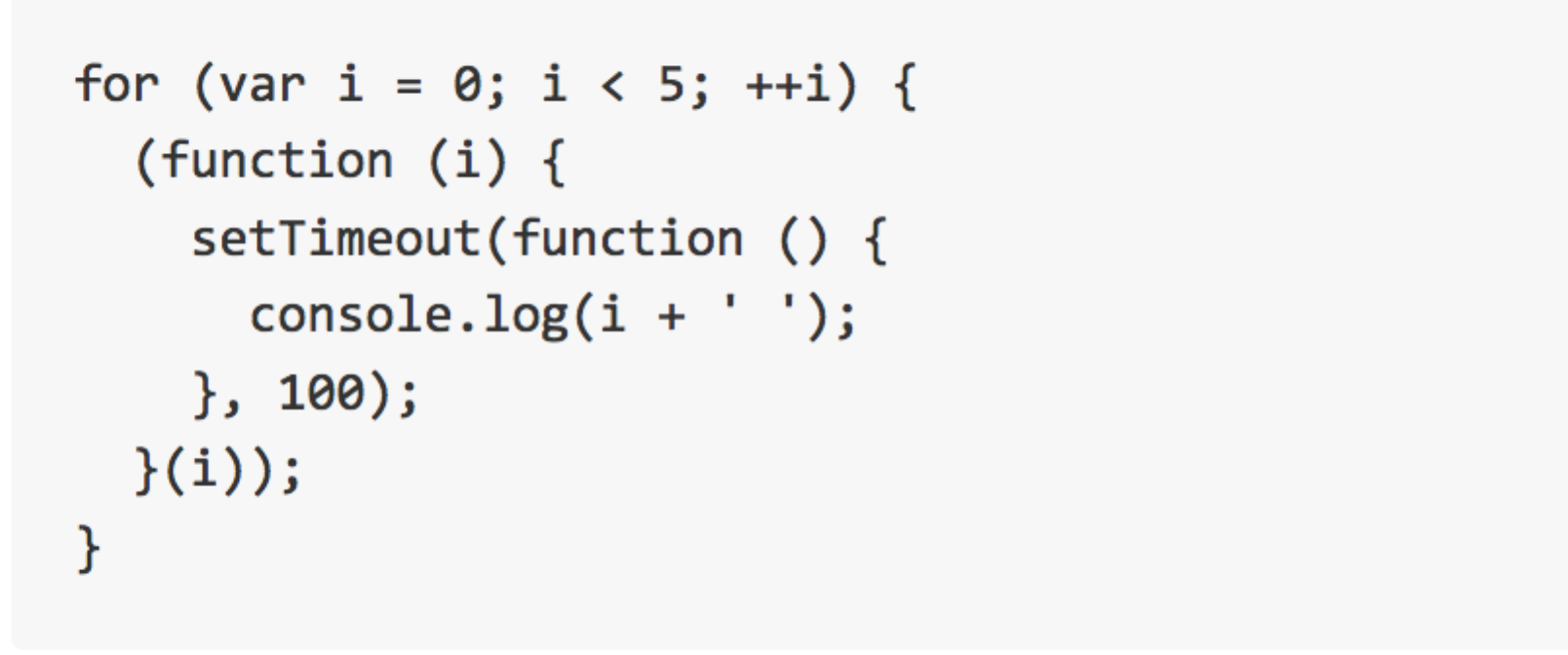


## 关于settimeout的理解

### 下面这段代码想要循环延时输出结果0 1 2 3 4,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



不能输出正确结果，因为循环中setTimeout接受的参数函数通过闭包访问变量i。javascript运行环境为单线程，setTimeout注册的函数需要等待线程空闲才能执行，此时for循环已经结束，i值为5.五个定时输出都是5  
修改方法：将setTimeout放在函数立即调用表达式中，将i值作为参数传递给包裹函数，创建新闭包



## js中绑定事件

## 

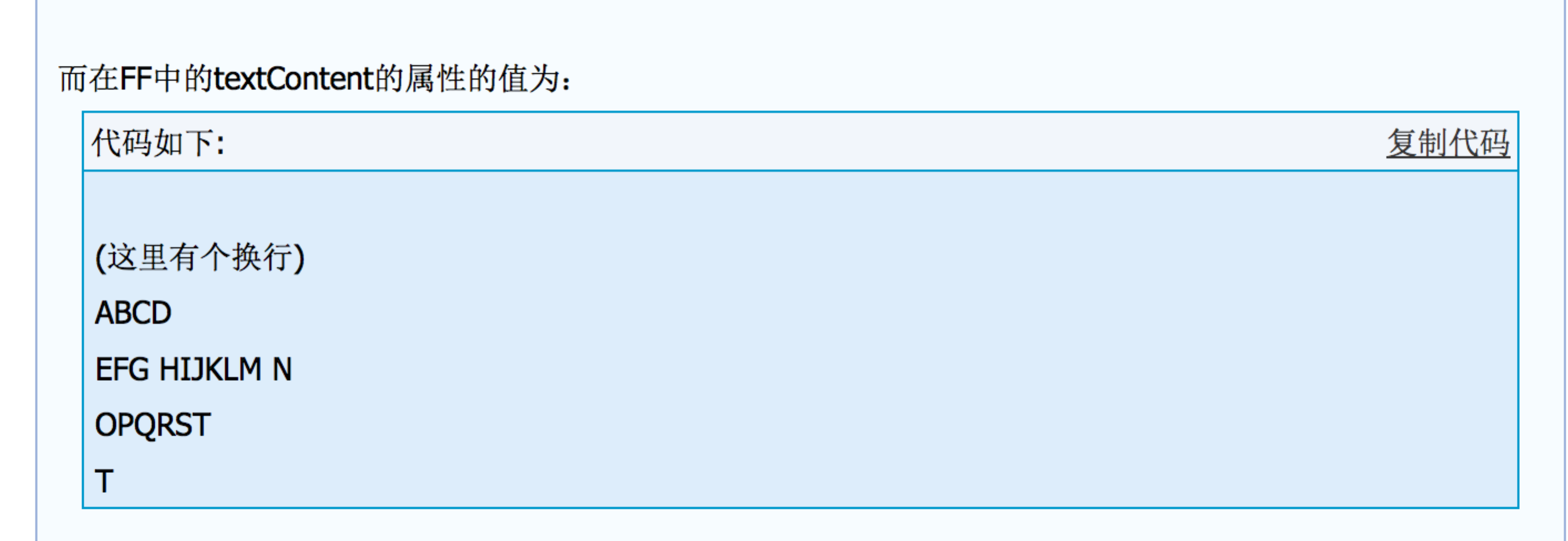


## XML DOM insertBefore() 方法



## javascript textContent与innerText的异同分析





## 如何判断一个对象是否为数组

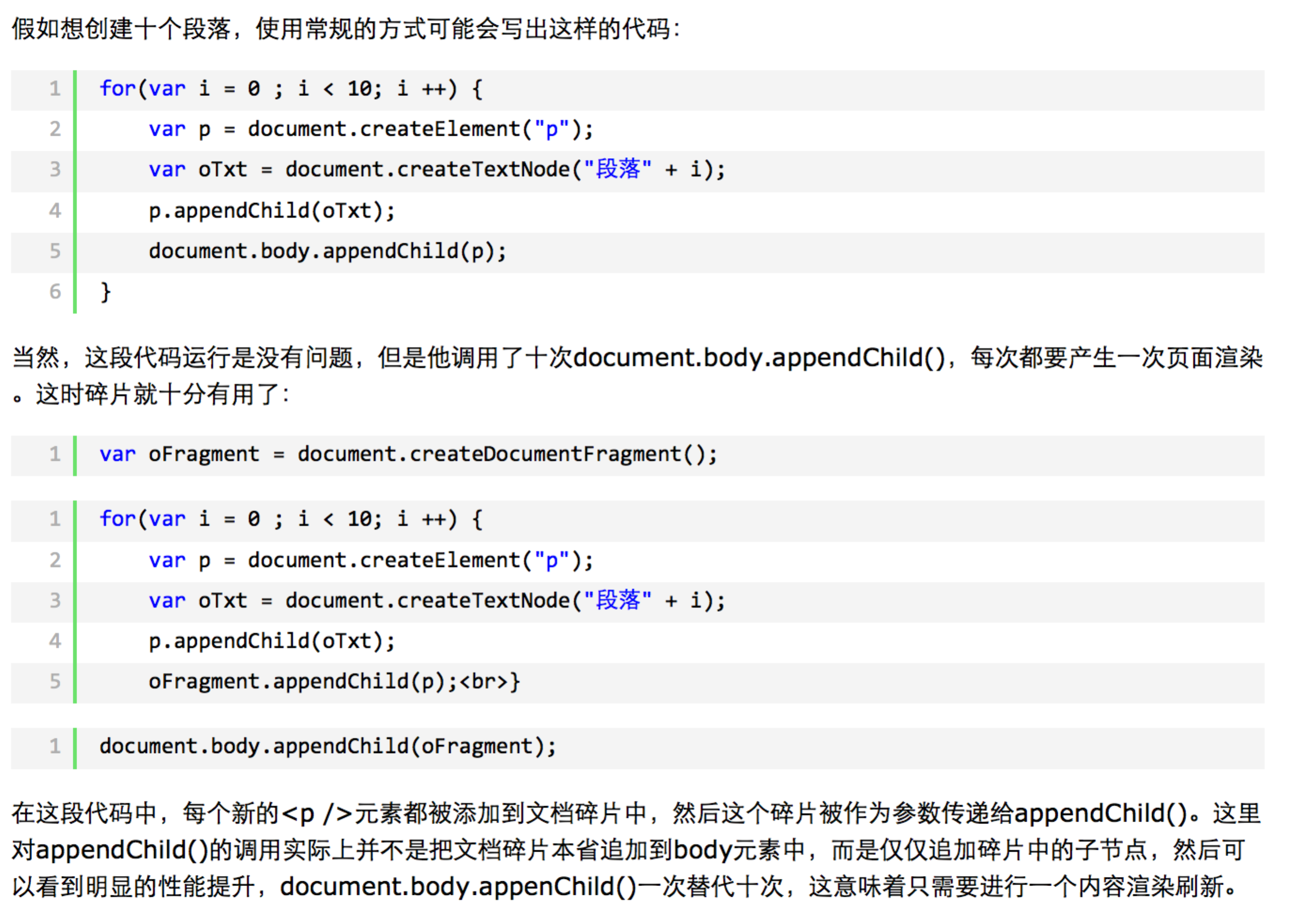
如果浏览器支持Array.isArray()可以直接判断否则需进行必要判断



## 如何判断一个对象是否为函数



## createDocumentFragment方法，当大量需要改变的时候，应先把改动输入到碎片。



## 编写一个函数将列表子元素顺序反转



# 写出3个使用this的典型应用

/\*  
事件： 如onclick  this->发生事件的对象  
构造函数          this->new 出来的object  
call/apply        改变this\*/

如何深度克隆  
/\*var arr = [1,2,43];  
var json = {a:6,b:4,c:[1,2,3]};  
var str = 'sdfsdf';  
  
var json2 = clone(json);  
  
alert(json['c'])  
function clone(obj){  
        var oNew = new obj.constructor(obj.valueOf());  
        if(obj.constructor == Object){  
                for(var i in obj){  
                        oNew[i] = obj[i];  
                        if(typeof(oNew[i]) == 'object'){  
                                clone(oNew[i]);  
                        }  
                }  
        }  
        return oNew;  
}\*/