# 做笔记

# **三、JS基础**

## 1.介绍js的基本数据类型。

  Undefined、Null、Boolean、Number、String、

  ECMAScript 2015 新增:Symbol(创建后独一无二且不可变的数据类型 )

## 2.介绍js有哪些内置对象？

         Object 是 JavaScript 中所有对象的父对象

         数据封装类对象：Object、Array、Boolean、Number 和 String

         其他对象：Function、Arguments、Math、Date、RegExp、Error

## 3.说几条写JavaScript的基本规范？

         1.不要在同一行声明多个变量。

         2.请使用 ===/!==来比较true/false或者数值

         3.使用对象字面量替代new Array这种形式

         4.不要使用全局函数。

         5.Switch语句必须带有default分支

         6.函数不应该有时候有返回值，有时候没有返回值。

         7.For循环必须使用大括号

         8.If语句必须使用大括号

         9.for-in循环中的变量 应该使用var关键字明确限定作用域，从而避免作用域污染。

## 4.JavaScript原型，原型链 ? 有什么特点？

         每个对象都会在其内部初始化一个属性，就是prototype(原型)，当我们访问一个对象的属性时，

         如果这个对象内部不存在这个属性，那么他就会去prototype里找这个属性，这个prototype又会有自己的prototype，

         于是就这样一直找下去，也就是我们平时所说的原型链的概念。

         关系：instance.constructor.prototype = instance.\_\_proto\_\_

         特点：

                   JavaScript对象是通过引用来传递的，我们创建的每个新对象实体中并没有一份属于自己的原型副本。

                   当我们修改原型时，与之相关的对象也会继承这一改变。

          当我们需要一个属性的时，Javascript引擎会先看当前对象中是否有这个属性， 如果没有的话，

        就会查找他的Prototype对象是否有这个属性，如此递推下去，一直检索到 Object 内建对象。

             function Func(){}

             Func.prototype.name = "Sean";

             Func.prototype.getInfo = function() {

               return this.name;

             }

             var person = new Func();//现在可以参考var person = Object.create(oldObject);

             console.log(person.getInfo());//它拥有了Func的属性和方法

             //"Sean"

             console.log(Func.prototype);

             // Func { name="Sean", getInfo=function()}

## 5.JavaScript有几种类型的值？你能画一下他们的内存图吗？

         栈：原始数据类型（Undefined，Null，Boolean，Number、String）

         堆：引用数据类型（对象、数组和函数）

         两种类型的区别是：存储位置不同；

         原始数据类型直接存储在栈(stack)中的简单数据段，占据空间小、大小固定，属于被频繁使用数据，所以放入栈中存储；

         引用数据类型存储在堆(heap)中的对象,占据空间大、大小不固定。如果存储在栈中，将会影响程序运行的性能；

         引用数据类型在栈中存储了指针，该指针指向堆中该实体的起始地址。当解释器寻找引用值时，

          会首先检索其在栈中的地址，取得地址后从堆中获得实体

## 6.如何将字符串转化为数字，例如'12.3b'?

         parseFloat('12.3b');

          正则表达式，'12.3b'.match(/(\d)+(\.)?(\d)+/g)[0] \* 1, 但是这个不太靠谱，提供一种思路而已。

## 7.如何将浮点数点左边的数每三位添加一个逗号，如12000000.11转化为『12,000,000.11』?

         function commafy(num){

             return num && num

                 .toString()

                 .replace(/(\d)(?=(\d{3})+\.)/g, function($1, $2){

                     return $2 + ',';

                 });

         }

## 8.如何实现数组的随机排序？

        方法一：javascript

                 var arr = [1,2,3,4,5,6,7,8,9,10];

             function randSort1(arr){

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

                                var rand = parseInt(Math.random()\*len);

                                var temp = arr[rand];

                                arr[rand] = arr[i];

                                arr[i] = temp;

                      }

                      return arr;

             }

             console.log(randSort1(arr));

         方法二：

                 var arr = [1,2,3,4,5,6,7,8,9,10];

        function randSort2(arr){

            var mixedArray = [];

            while(arr.length > 0){

                var randomIndex = parseInt(Math.random()\*arr.length);

                mixedArray.push(arr[randomIndex]);

                arr.splice(randomIndex, 1);

            }

            return mixedArray;

        }

        console.log(randSort2(arr));

         方法三：

                   var arr = [1,2,3,4,5,6,7,8,9,10];

        arr.sort(function(){

            return Math.random() - 0.5;

        })

        console.log(arr);

## 9.javascript创建对象的几种方式？

         javascript创建对象简单的说,无非就是使用内置对象或各种自定义对象，当然还可以用JSON；但写法有很多种，也能混合使用。

         1、对象字面量的方式

             person={firstname:"Mark",lastname:"Yun",age:25,eyecolor:"black"};

         2、用function来模拟无参的构造函数

             function Person(){}

             var person=new Person();//定义一个function，如果使用new"实例化",该function可以看作是一个Class

             person.name="Mark";

             person.age="25";

             person.work=function(){

             alert(person.name+" hello...");

             }

             person.work();

         3、用function来模拟参构造函数来实现（用this关键字定义构造的上下文属性）

             function Pet(name,age,hobby){

                this.name=name;//this作用域：当前对象

                this.age=age;

                this.hobby=hobby;

                this.eat=function(){

                   alert("我叫"+this.name+",我喜欢"+this.hobby+",是个程序员");

                }

             }

             var maidou =new Pet("麦兜",25,"coding");//实例化、创建对象

             maidou.eat();//调用eat方法

         4、用工厂方式来创建（内置对象）

             var wcDog =new Object();

             wcDog.name="旺财";

             wcDog.age=3;

             wcDog.work=function(){

                alert("我是"+wcDog.name+",汪汪汪......");

             }

             wcDog.work();

         5、用原型方式来创建

             function Dog(){

              }

             Dog.prototype.name="旺财";

             Dog.prototype.eat=function(){

              alert(this.name+"是个吃货");

             }

             var wangcai =new Dog();

             wangcai.eat();

         6、用混合方式来创建

             function Car(name,price){

               this.name=name;

               this.price=price;

             }

             Car.prototype.sell=function(){

                alert("我是"+this.name+"，我现在卖"+this.price+"万元");

             }

             var camry =new Car("凯美瑞",27);

             camry.sell();

## 10.谈谈This对象的理解

         this总是指向函数的直接调用者（而非间接调用者）；

         如果有new关键字，this指向new出来的那个对象；

         在事件中，this指向触发这个事件的对象，特殊的是，IE中的attachEvent中的this总是指向全局对象Window；

## 11.eval是做什么的？

         它的功能是把对应的字符串解析成JS代码并运行；

         应该避免使用eval，不安全，非常耗性能（2次，一次解析成js语句，一次执行）。

         由JSON字符串转换为JSON对象的时候可以用eval，var obj =eval('('+ str +')');

## 12.什么是window对象? 什么是document对象?

         window对象是指浏览器打开的窗口。

         document对象是Documentd对象（HTML 文档对象）的一个只读引用，window对象的一个属性。

## 13.null，undefined 的区别？

         null        表示一个对象是“没有值”的值，也就是值为“空”；

         undefined   表示一个变量声明了没有初始化(赋值)；

         undefined不是一个有效的JSON，而null是；

         undefined的类型(typeof)是undefined；

         null的类型(typeof)是object；

         Javascript将未赋值的变量默认值设为undefined；

         Javascript从来不会将变量设为null。它是用来让程序员表明某个用var声明的变量时没有值的。

         typeof undefined

             //"undefined"

             undefined :是一个表示"无"的原始值或者说表示"缺少值"，就是此处应该有一个值，但是还没有定义。当尝试读取时会返回 undefined；

             例如变量被声明了，但没有赋值时，就等于undefined

         typeof null

             //"object"

             null : 是一个对象(空对象, 没有任何属性和方法)；

             例如作为函数的参数，表示该函数的参数不是对象；

         注意：

             在验证null时，一定要使用　=== ，因为 == 无法分别 null 和　undefined

             null == undefined // true

             null === undefined // false

         再来一个例子：

             null

             Q：有张三这个人么？

             A：有！

             Q：张三有房子么？

             A：没有！

             undefined

             Q：有张三这个人么？

             A：有！

             Q: 张三有多少岁？

             A: 不知道（没有被告诉）

## 14.写一个通用的事件监听器函数。

         // event(事件)工具集，来源：github.com/markyun

    markyun.Event = {

        // 页面加载完成后

        readyEvent : function(fn) {

            if (fn==null) {

                fn=document;

            }

            var oldonload = window.onload;

            if (typeof window.onload != 'function') {

                window.onload = fn;

            } else {

                window.onload = function() {

                    oldonload();

                    fn();

                };

            }

        },

        // 视能力分别使用dom0||dom2||IE方式 来绑定事件

        // 参数： 操作的元素,事件名称 ,事件处理程序

        addEvent : function(element, type, handler) {

            if (element.addEventListener) {

                //事件类型、需要执行的函数、是否捕捉

                element.addEventListener(type, handler, false);

            } else if (element.attachEvent) {

                element.attachEvent('on' + type, function() {

                    handler.call(element);

                });

            } else {

                element['on' + type] = handler;

            }

        },

        // 移除事件

        removeEvent : function(element, type, handler) {

            if (element.removeEventListener) {

                element.removeEventListener(type, handler, false);

            } else if (element.datachEvent) {

                element.detachEvent('on' + type, handler);

            } else {

                element['on' + type] = null;

            }

        },

        // 阻止事件 (主要是事件冒泡，因为IE不支持事件捕获)

        stopPropagation : function(ev) {

            if (ev.stopPropagation) {

                ev.stopPropagation();

            } else {

                ev.cancelBubble = true;

            }

        },

        // 取消事件的默认行为

        preventDefault : function(event) {

            if (event.preventDefault) {

                event.preventDefault();

            } else {

                event.returnValue = false;

            }

        },

        // 获取事件目标

        getTarget : function(event) {

            return event.target || event.srcElement;

        },

        // 获取event对象的引用，取到事件的所有信息，确保随时能使用event；

        getEvent : function(e) {

            var ev = e || window.event;

            if (!ev) {

                var c = this.getEvent.caller;

                while (c) {

                    ev = c.arguments[0];

                    if (ev && Event == ev.constructor) {

                        break;

                    }

                    c = c.caller;

                }

            }

            return ev;

        }

    };

## 15.事件是？IE与火狐的事件机制有什么区别？ 如何阻止冒泡？

         1. 我们在网页中的某个操作（有的操作对应多个事件）。例如：当我们点击一个按钮就会产生一个事件。是可以被 JavaScript 侦测到的行为。

        2. 事件处理机制：IE是事件冒泡、Firefox同时支持两种事件模型，也就是：捕获型事件和冒泡型事件；

        3. ev.stopPropagation();（旧ie的方法 ev.cancelBubble = true;）

## 16.如何判断一个对象是否属于某个类？

         使用instanceof （待完善）

            if(a instanceof Person){

                alert('yes');

            }

## 17.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);

## 18.Javascript中，有一个函数，执行时对象查找时，永远不会去查找原型，这个函数是？

         hasOwnProperty

         javaScript中hasOwnProperty函数方法是返回一个布尔值，指出一个对象是否具有指定名称的属性。此方法无法检查该对象的原型链中是否具有该属性；该属性必须是对象本身的一个成员。

         使用方法：

         object.hasOwnProperty(proName)

         其中参数object是必选项。一个对象的实例。

         proName是必选项。一个属性名称的字符串值。

         如果 object 具有指定名称的属性，那么JavaScript中hasOwnProperty函数方法返回 true，反之则返回 false。

## 19.JSON 的了解？

         JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。

         它是基于JavaScript的一个子集。数据格式简单, 易于读写, 占用带宽小

         如：{"age":"12", "name":"back"}

         JSON字符串转换为JSON对象:

         var obj =eval('('+ str +')');

         var obj = str.parseJSON();

         var obj = JSON.parse(str);

         JSON对象转换为JSON字符串：

         var last=obj.toJSONString();

         var last=JSON.stringify(obj);

## 20.模块化开发怎么做？

         立即执行函数,不暴露私有成员

          var module1 = (function(){

    　　　　var \_count = 0;

    　　　　var m1 = function(){

    　　　　　　//...

    　　　　};

    　　　　var m2 = function(){

    　　　　　　//...

    　　　　};

    　　　　return {

    　　　　　　m1 : m1,

    　　　　　　m2 : m2

    　　　　};

    　　})();​​​​

## 21.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

## 22.DOM操作——怎样添加、移除、移动、复制、创建和查找节点?

         （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()

           removeChild()

           replaceChild()

           insertBefore() //在已有的子节点前插入一个新的子节点

         （3）查找

           getElementsByTagName()    //通过标签名称

           getElementsByName()    //通过元素的Name属性的值(IE容错能力较强，会得到一个数组，其中包括id等于name值的)

           getElementById()    //通过元素Id，唯一性

## 23..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);

## 24.那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。

         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。

         闭包、控制台日志、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）

## 25.用js实现千位分隔符?

         function commafy(num) {

             return num && num

                 .toString()

                 .replace(/(\d)(?=(\d{3})+\.)/g, function($0, $1) {

                     return $1 + ",";

                 });

         }

         console.log(commafy(1234567.90)); //1,234,567.90

## 26.检测浏览器版本版本有哪些方式？

         功能检测、userAgent特征检测

         比如：navigator.userAgent

         //"Mozilla/5.0 (Macintosh; Intel Mac OS X 10\_10\_2) AppleWebKit/537.36

       (KHTML, like Gecko) Chrome/41.0.2272.101 Safari/537.36"

## 27.列举IE与其他浏览器不一样的特性？

         事件不同之处：

             触发事件的元素被认为是目标（target）。而在 IE 中，目标包含在 event 对象的 srcElement 属性；

             获取字符代码、如果按键代表一个字符（shift、ctrl、alt除外），IE 的 keyCode 会返回字符代码（Unicode），

                       DOM 中按键的代码和字符是分离的，要获取字符代码，需要使用 charCode 属性；

           阻止某个事件的默认行为，IE 中阻止某个事件的默认行为，必须将 returnValue 属性设置为 false，Mozilla 中，

                    需要调用 preventDefault() 方法；

             停止事件冒泡，IE 中阻止事件进一步冒泡，需要设置 cancelBubble 为 true，Mozzilla 中，

                       需要调用 stopPropagation()；

## 28.什么叫优雅降级和渐进增强？

         优雅降级：Web站点在所有新式浏览器中都能正常工作，如果用户使用的是老式浏览器，

                   则代码会针对旧版本的IE进行降级处理了,使之在旧式浏览器上以某种形式降级体验却不至于完全不能用。

         如：border-shadow

         渐进增强：从被所有浏览器支持的基本功能开始，逐步地添加那些只有新版本浏览器才支持的功能,

         向页面增加不影响基础浏览器的额外样式和功能的。当浏览器支持时，它们会自动地呈现出来并发挥作用。

         如：默认使用flash上传，但如果浏览器支持 HTML5 的文件上传功能，则使用HTML5实现更好的体验；

## 29.请写出一下代码的运行结果

         var f= true

         if(f===true){

                   var a = 10

         }

         function fn(){

                   var b = 20

                   c = 30

         }

         fn()

         console.log(a)  //10

         console.log(b)  //报错 b is not defined ,所以不会输出b,c，如果再函数前面var b，则会出现undefined

         console.log(c)

## 30.输出今天的日期，以YYYY-MM-DD的方式，比如今天是2015年7月7日，则输出2015-07-07

         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;

         console.log(year + '-' + month + '-' + day)

## 31.看下列代码，将会输出什么(考察点：声明的变量提升，IIFE)

         var foo = 1       ;

         (function(){

                   console.log(foo)   //undefined

                   var foo = 2

                   console.log(foo)  //2

         })()

## 32.如何添加HTML事件，有几种方法？（至少两种方式）

         1）通过HTML元素属性

         <a href="###" onclick='do'>name</a>

         2)通过对象属性

              对象指的是DOM树里的对象，我们都知道，所有的html元素在DOM(文档对象类型)里都存在一个相应的DOM元素。

                   html结构:

                   <a href="###" id="n">name</a>

                   js：

                   document.getElementById('n').onclick = function(){语句}

         3)通过W3C监听方式（标准方式）或者IE专属的中间模型添加事件

                   W3C方式：elemennt.addEventListener(事件名,处理函数引用,true || false)

                            true表示捕获事件，false冒泡阶段捕获事件，一般我们设为false

                   IE模式: element.attachEvent( 'on' + 事件名，处理函数引用)

                   兼容的模型：

                            function bind(obj,eventName,fn){

                                     if(obj.addEventListener){

                                               //标准浏览器

                                               obj.addEventListener(eventName,fn，false)

                                     }else{

                                               //非标准浏览器(ie 678)

                                               obj.attachEvent('on'+eventName,function(){

                                                        fn.call(obj)

                                               })

                                     }

                            }

## 33.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

## 34.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（== ,!!）

## 35.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

## 36.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

map() : 遍历数组中的元素, 返回一个新数组(包含回调函数返回的数据)

filter():遍历数组中的元素, 返回一个新数组(包含回调函数返回true的元素)

## 37.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

## 38.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

## 39.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

call和apply的区别

功能一样, 都是将当前函数作为指定对象的方法执行, 即函数中的this是指定对象

call(thisObj，arg1,arg2...)   //将所有参数一个一个传递进去

apply(thisObj，[argArray])   //将所有参数放在数组中传入

## 40.Worker继承Person的方法

//使用构造函数+原型的组合模式

function Person( age, name ){

  this.age = age;

  this.name = name;

}

Person.prototype.show = function(){

  alert('父级方法');

}

function Worker(age,name,job){

  Person.apply( this, arguments );

  this.job = job;

}

Worker.prototype = new Person();

var Person = new Person(14,'张三 ');

var Worker = new Worker (25,'李四','程序员');

## 41.如何阻止事件冒泡和事件默认行为

//阻止事件冒泡

if(typeof ev.stopPropagation=='function') {  //标准的

         ev.stopPropagation();

} else { //非标准IE

         window.event.cancelBubble = true;

}

//阻止事件默认行为

return false

## 42.添加 删除 替换 插入到某个元素的方法

element.appendChild()

element.insertBefore()

element.replaceChild()

element.removeChild()

## 43.javascript的内置对象和宿主对象

内置对象为Object, Array, Function, Date, Math等

宿主为浏览器自带的window 等

## 44.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原生中没有这个方法，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

## 45.”==”和“===”的不同

前者会自动转换类型

后者不会

## 46.浏览器的同源策略

一段脚本(ajax)只能读取来自于同一来源的窗口和文档的属性，这里的同一来源指的是主机名、协议和端口号的组合

## 47.JavaScript是一门什么样的语言，它有哪些特点？

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。

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）。

## 48.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, 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]"

    };

}

## 49.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

## 50.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

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

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

    　　checkBoxList.push(domList[i]);

　　}

}

## 51.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”

## 52.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

## 53.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。

## 54.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”。

## 55.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。

## 56.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）

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

alert(stringArray.join(“”))

## 58.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;

}

## 59.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})

## 60.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);

## 61.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。

## 62.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);

## 63.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

## 64.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？

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})$/;

## 65.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：回调函数是在for结束之后才运行的。追问，如何让上述代码输出1 2 3？

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

   setTimeout((function(j){  //改成立即执行函数

console.log(j);

   })(i),0);

};

## 66.写一个function，清除字符串前后的空格。（兼容所有浏览器）

if (!String.prototype.trim) {

    String.prototype.trim = function() {

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

 }

}

//测试

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

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

## 67.Javascript中, 以下哪条语句一定会产生运行错误？

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

68.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'

## 69.var a=10; b=20; c=4;  ++b+c+a++ 以下哪个结果是正确的？

A 34   B、35  C、36  D、37     答案(  B  )

## 70.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=”";

}

}

## 71.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”

## 72.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 = //;

## 73.以下哪个单词不属于javascript保留字：（B）

A.withB.parentC.classD.void

**74.请选择结果为真的表达式：（C）**

A.null instanceof ObjectB.null === undefinedC.null == undefinedD.NaN == NaN

## 75.typeof运算符返回值中有一个跟javascript数据类型不一致，它是Array。

## 76.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 。

## 77.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

## 78.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：7/21/2016

## 79.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

## 80.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

## 81.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 = “<div>这里是div<p>里面的段落</p></div>”;

<scripttype=”text/javascript”>

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

var str = “<div>这里是div<p>里面的段落</p></div>”;

alert(str.replace(reg,”"));

</script>

## 82.截取字符串abcdefg的efg

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

## 83.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()

## 84.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

## 85.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");

## 86.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

## 87.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、​通过构造函数创建对象。

(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、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

## 88.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

## 89.请你谈谈Cookie的弊端？

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。

## 90.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​​​​

## 91.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

## 92.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）

## 93.判断一个字符串中出现次数最多的字符，统计这个次数

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');

## 94.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;

         }

}

## 95.事件委托是什么

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！

## 96.闭包是什么，有什么特性，对页面有什么影响

当内部函数使用了外部函数的局部变量时, 产生的一个对象(包含了所有使用了的变量)

         作用: 在函数执行完后, 局部变量还会存在

         function outer(){

    var num = 1;

    function inner(){

        var n = 2;

        alert(n + num);

    }

    return inner;

}

var r = outer();

r();

## 97.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

## 98.字符串反转，如将 '12345678' 变成 '87654321'

//思路：先将字符串转换为数组 split()，利用数组的反序函数 reverse()颠倒数组，再利用 jion() 转换为字符串

var str = '12345678';

str = str.split('').reverse().join('');

## 99.将数字 12345678 转化成 RMB形式 如： 12,345,678

//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);

}

## 100.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--;

        }

    }

}

## 101.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

## 102.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());

## 103.window.location.reload() 作用？

    刷新当前页面。

## 104.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;

## 105.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

## 106.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

## 107.加减运算

alert('5'+3); //53 string

alert('5'+'3'); //53 string

alert('5'-3); //2 number

alert('5'-'3'); //2 number

## 108.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

## 109.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

## 110.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

## 111.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。

## 112.声明对象，添加属性，输出属性

var obj = {

        name: 'leipeng',

        showName: function(){

            alert(this.name);

        }

    }

obj.showName();

## 113.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));

## 114.检测变量类型

function checkStr(str){

    return str =='string';

}

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

## 115.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、绑定事件  IE： obj.attachEvent('onclick', method)；

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

## 116.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；

## 117.JS中的简单继承 call方法

//定义个父母类，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();

## 118.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。

## 119.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。

## 120.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​​​​