JavaScript知识点整理

原创 2017年09月24日 20:59:46 标签：javascript 582

1.JavaScript的定义
JavaScript是一种专门为与网页交互而设计的脚本语言。有下列三部分组成
①ECMAScript，提供核心语言功能
②文档对象模型（DOM），提供访问与操作网页内容的方法与接口
③浏览器对象模型（BOM），提供与浏览器交互的方法与接口

2.在HTML中使用javascript
`<script>`元素
属性：async（异步），表示应该立即下载脚本；async = “async”
defer（延迟），表示脚本可以延迟到文档完全被解析和显示之后再执行,defer = “defer”
只要不存在async和defer属性，浏览器都会按照script元素在页面出现的先后顺序依次进行解析。

3.ECMAScript基本数据类型
Undefined、Null、Boolean、String、Number、Object

4.typeof操作符
返回值（字符串）：
“undefined”,”boolean”,”string”,”object”,”function”,”number”

```
> typeof undefined
"undefined"
> typeof null
"object"
> typeof 5
"number"
> typeof '5'
"string"
> typeof {}
"object"
> typeof []
"object"
> typeof true
"boolean"
> typeof 0
"number"
> typeof NaN
"number"
```

```
> typeof function(){}
"function"
```

4.Boolean()转化规则
调用Boolean()函数可以将任何数据类型转化为boolean类型。其转换规则如下：

Boolean	true	false
String	任何非空字符串	空字符串(“”)
Number	任何非零实数	0和NaN
Object	任何对象	null
Undefined	不适用	undefined

注意：

```
> Boolean("")
false
> Boolean("  ")
true
> Boolean("    ")
true
```

字符串内容为空格 不等于 空字符串

5.Number()转化规则

Boolean	true转为1；false转为0
String	只包含数字，返回对应十进制数值；空字符串返回0；包含其他非数字字符，返回NaN
Number	简单的传入与返回
Undefined	返回NaN
Object	调用对象的valueOf()方法，依照前面规则返回

6.parseInt()转化规则
parseInt()函数在转换字符串时，忽略字符串前的空格，如果第一个字符不是数字字符或负号，返回NaN；如果是数字字符，继续解析字符串，直到遇到非数字字符，停止解析，返回已解析部分对应的数值。

```
> parseInt('AF')
NaN
> parseInt('AF',16)
175
> parseInt('  ')
NaN
> parseInt('123blue')
123
> parseInt('123.50.5')
123
> parseInt('-123.50.5')
-123
```

注意：parseInt()可以传入第二个参数，作为指定第一个参数的数制。

7.位操作符
25 = 11001(二进制);
3= 11(二进制)

与	&	25 & 3 = 1
或	|	25 | 3 = 27
非	~	~25 = -26
异或	^	25 ^ 3 = 26
左移	‹‹	25 ‹‹ 3 = 200
右移	››	25 ›› 3 = 3
无符号右移	›››	25 ›››3 = 3

8.基本类型值在内存中占据固定大小的空间，因此被保存在栈内存中。引用类型的值是对象，保存在堆内存中。

9.instanceof操作符
确定一个值是哪种引用类型

```
> [] instanceof Array
true
> [] instanceof Object
true
> [] instanceof RegExp
false
> 5 instanceof Object
false
> {} instanceof Array
Uncaught SyntaxError: Unexpected token instanceof
> null instanceof Array
false
> null instanceof Object
false
> var a = {};a instanceof Array
false
```

10.延长作用域链
①try-catch语句的catch块
②with语句

11.垃圾收集
javascript具有自动垃圾收集机制
①离开作用域的值将被自动标记为回收，因此将在垃圾收集期间被删除
②“标记清除”是目前主流的垃圾收集算法，算法思想是给当前不使用的值加上标记，然后再回收其内存
③另一种垃圾收集算法是“引用计数”，这种算法的思想是跟踪记录所有值被引用的次数，当引用次数为0，回收内存。javascript引擎目前不再使用这种算法。
④当代码中存在循环引用现象，“引用计数”算法会导致问题。
⑤解除变量的引用不仅有助于消除循环引用现象，而且对垃圾收集也有好处。为了确保有效地回收内存，应该及时解除不再使用的全局对象，全局对象属性以及循环引用变量的引用。

12.因为引用计数产生的内存泄漏，在ES6中的解决办法是什么？
ES6 考虑到了这一点，推出了两种新的数据结构：WeakSet 和 WeakMap。它们对于值的引用都是不计入垃圾回收机制的，所以名字里面才会有一个”Weak”，表示这是弱引用。

13.渐进增强和优雅降级
渐进增强（Progressive Enhancement）：一开始就针对低版本浏览器进行构建页面，完成基本的功能，然后再针对高级浏览器进行效果、交互、追加功能达到更好的体验。
优雅降级（Graceful Degradation）：一开始就构建站点的完整功能，然后针对浏览器测试和修复。比如一开始使用 CSS3 的特性构建了一个应用，然后逐步针对各大浏览器进行 hack 使其可以在低版本浏览器上正常浏览。

在传统软件开发中，经常会提到向上兼容和向下兼容的概念。渐进增强相当于向上兼容，而优雅降级相当于向下兼容。

14.事件委托和性能优化
对“事件处理程序过多”问题的解决方案就是事件委托。事件委托利用了事件冒泡，只指定一个事件处理程序，就可以管理某一类型的所有事件。例如：

```
<ul id="myul">
 <li id="go">go somewhere</li>
 <li id="do">do something</li>
 <li id="del">delete something</li>
</ul>

<script>
  var myul = document.getElementById('myul');
  EventUtil.addHandler(myul,'click',function(event){
   event = EventUtil.getEvent(event);
   var target = EventUtil.getTarget(event);
   switch(target.id){
    case 'go':doSomething...;break;
    case 'do':doSomething...;break;
```

```
    case 'del':doSomething...;break;
  }
 });
</script>
```

如果可行的话，也可以考虑为document对象添加一个事件处理程序，用以处理页面上发生的某种特定类型的事件。这样做与传统做法相比较有如下优点：
①document对象很快就可以访问，而且可以在页面生命周期的任何时间点上为他添加事件处理程序。
②在页面设置事件处理程序所需时间更少。只添加一个事件处理程序所需的DOM引用更少，所花的时间也更少。
③整个页面占用的内存空间更少，能够提升整体性能。
最适合采用事件委托技术的事件包括click、mousedown、mouseup、keydown、keyup和keypress。

15.给定一个a值和一个b值，如何在不使用任何辅助空间的情况下，交换a和b的值？

```
> var a = 1; var b = 2;
  a = a + b;b = a - b; a = a - b ;console.log(a,b);
  2 1
  undefined
> var a = 1; var b = 2;
  a = a ^ b;b = a ^ b; a = a ^ b ;console.log(a,b);
  2 1
  undefined
>
```

16.for-in的缺点
for-in是一种精准的迭代语句，可以用来枚举对象的属性。
①ECMAScript对象的属性没有顺序，因此通过for-in循环输出的属性名的顺序是不可预测的。
②如果要迭代的对象的变量值为null或undefined，for-in语句会抛出错误。ECMAScript5已经更正这一行为，不再抛出错误，而是不执行循环体。为了最大限度地保证兼容性，建议在使用for-in循环之前，先检测确认该对象不是为null或者undefined。

```
> var obj = {
   "name":"jack",
   "age":18
  };
  if(obj){
   for(var item in obj){
    console.log(item,obj[item]);
   }
  }
  name jack
  age 18
```

17.数组去重问题
方法1：

```
> var a = [5,0,5,2,0,6];
  var b = [];
  for(var i=0;i<a.length;i++){
   if(b.indexOf(a[i])===-1){
    b.push(a[i]);
   }
  }
  console.log(b);
  ▶ [5, 0, 2, 6]
```

方法2：

```
> var a = [5,0,5,2,0,6];
  var b = Array.from(new Set(a));
  console.log(b);
  ▶ [5, 0, 2, 6]
```

方法3：

```
var a = [5,0,5,2,0,6];
var b = [...new Set(a)];
console.log(b);
▶ [5, 0, 2, 6]
```

18.手写实现斐波那契序列

```
//Fibonacci数列，从第三项开始，每一项是前面两项之和。0、1、1、2、3、5、8、13、21、34、……
//求下标为index的Fibonacci数值
function GetNthFibonacci(index){
    if(index<=0){return 0;}
    if(index<=1){return 1;}
    var first = 1;
    var second = 1;
    var third = 0;
    for(var i=2;i<index;i++){
        third = first + second;
        first = second;
        second = third;
    }
    return third;
};
```

19.冒泡和捕获

事件冒泡：IE的事件流叫事件冒泡，即事件开始时由最具体的元素接收，然后逐级向上传播到较为不具体的节点。

事件捕获：不太具体的节点更早接收到事件，而最具体的节点最后接收到事件。事件捕获的用意在于在事件到达预定目标之前捕获它。

“DOM2级事件”规定的事件流包括三个阶段：事件捕获阶段、处于目标阶段、事件冒泡阶段。

20.阻止冒泡的方法，阻止默认事件的方法(兼容IE)

阻止冒泡的方法：w3c的方法是e.stopPropagation()，IE则是使用e.cancelBubble = true；

阻止默认事件的方法：w3c的方法是e.preventDefault()，IE则是使用e.returnValue = false;

21.js操作获取和设置cookie

```
/*cookie的三种操作：读取，写入，删除*/
var CookieUtil = {
  get:function(name){
    var cookieName = encodeURIComponent(name) + "=",
        cookieStart = document.cookie.indexOf(cookieName),
        cookieValue = null;
    if (cookieStart > -1) {
      var cookieEnd = document.cookie.indexOf(";",cookieStart);
      if (cookieEnd == -1) {
        cookieEnd = document.cookie.length;
      }
      cookieValue = decodeURIComponent(document.cookie.substring(cookieStart + cookieName.l
ength,cookieEnd));
    }
    return cookieValue;
  },
  set:function(name,value,expires,path,domain,secure){
    var cookieText = encodeURIComponent(name) + "=" + encodeURIComponent(value);
    if (expires instanceof Date) {
      cookieText += ";expires=" + expires.toGMTString();
    }
    if (path) {
      cookieText += ";path=" + path;
    }
    if (domain) {
      cookieText += ";domain=" + domain;
    }
    if (secure) {
      cookieText += ";secure";
    }
    document.cookie = cookieText;
  },
```

```
  unset:function(name,path,domain,secure){
    this.set(name,"",new Date(0),path,domain,secure);
  }
};
```

22.js对象的深度克隆代码实现

```
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{
        return Obj;
    }
}
```

23.WEB应用从服务器主动推送Data到客户端有那些方式？
html5 websocket
WebSocket 通过 Flash
XHR长时间连接
XHR Multipart Streaming
不可见的Iframe
<script>标签的长时间连接(可跨域)

24.JavaScript 原型，原型链？有什么特点？
我们创建的每个函数都有一个prototype（原型）属性，这个属性是一个指针，指向一个对象，而这个对象的用途是包含可以由特定类型的所有实例共享的属性和方法。
原型链是由一些用来继承和共享属性的对象组成的（有限的）对象链

25.写一个通用的事件侦听器函数

```
// 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.removeEnentListener) {
            element.removeEnentListener(type, handler, false);
        } else if (element.detachEvent) {
            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;
    }
};
```

26.js单体内置对象

Global对象、Math对象

顶层函数（全局函数）

函数	描述
decodeURI()	解码某个编码的 URI。
decodeURIComponent()	解码一个编码的 URI 组件。
encodeURI()	把字符串编码为 URI。
encodeURIComponent()	把字符串编码为 URI 组件。
escape()	对字符串进行编码。
eval()	计算 JavaScript 字符串，并把它作为脚本代码来执行。
getClass()	返回一个 JavaObject 的 JavaClass。
isFinite()	检查某个值是否为有穷大的数。
isNaN()	检查某个值是否是数字。
Number()	把对象的值转换为数字。
parseFloat()	解析一个字符串并返回一个浮点数。
parseInt()	解析一个字符串并返回一个整数。
String()	把对象的值转换为字符串。
unescape()	对由 escape() 编码的字符串进行解码。

顶层属性（全局属性）

方法	描述
Infinity	代表正的无穷大的数值。
java	代表 java.* 包层级的一个 JavaPackage。
NaN	指示某个值是不是数字值。
Packages	根 JavaPackage 对象。
undefined	指示未定义的值。

Math 对象方法

方法	描述
abs(x)	返回数的绝对值。
acos(x)	返回数的反余弦值。
asin(x)	返回数的反正弦值。
atan(x)	以介于 -PI/2 与 PI/2 弧度之间的数值来返回 x 的反正切值。
atan2(y,x)	返回从 x 轴到点 (x,y) 的角度（介于 -PI/2 与 PI/2 弧度之间）。
ceil(x)	对数进行上舍入。
cos(x)	返回数的余弦。
exp(x)	返回 e 的指数。
floor(x)	对数进行下舍入。
log(x)	返回数的自然对数（底为e）。
max(x,y)	返回 x 和 y 中的最高值。
min(x,y)	返回 x 和 y 中的最低值。
pow(x,y)	返回 x 的 y 次幂。
random()	返回 0 ~ 1 之间的随机数。
round(x)	把数四舍五入为最接近的整数。
sin(x)	返回数的正弦。
sqrt(x)	返回数的平方根。
tan(x)	返回角的正切。
toSource()	返回该对象的源代码。
valueOf()	返回 Math 对象的原始值。

Math 对象属性

属性	描述
E	返回算术常量 e，即自然对数的底数（约等于2.718）。
LN2	返回 2 的自然对数（约等于0.693）。
LN10	返回 10 的自然对数（约等于2.302）。
LOG2E	返回以 2 为底的 e 的对数（约等于 1.414）。
LOG10E	返回以 10 为底的 e 的对数（约等于0.434）。
PI	返回圆周率（约等于3.14159）。
SQRT1_2	返回返回 2 的平方根的倒数（约等于 0.707）。
SQRT2	返回 2 的平方根（约等于 1.414）。

27.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）

28.GET和POST的区别，何时使用POST？
GET：一般用于信息获取，使用URL传递参数，对所发送信息的数量也有限制，一般在2000个字符
POST：一般用于修改服务器上的资源，对所发送的信息没有限制

GET方式需要使用 Request.QueryString 来取得变量的值
POST方式通过 Request.Form 来获取变量的值
也就是说 Get 是通过地址栏来传值，而 Post 是通过提交表单来传值。

在以下情况中，请使用 POST 请求：
无法使用缓存文件（更新服务器上的文件或数据库）
向服务器发送大量数据（POST 没有数据量限制）
发送包含未知字符的用户输入时，POST 比 GET 更稳定也更可靠

29.什么是 “use strict”; ? 使用它的好处和坏处分别是什么？
ECMAscript 5添加了第二种运行模式：”严格模式”（strict mode）。顾名思义，这种模式使得Javascript在更严格的条件下运行。

设立”严格模式”的目的，主要有以下几个：
1. 消除Javascript语法的一些不合理、不严谨之处，减少一些怪异行为;
2. 消除代码运行的一些不安全之处，保证代码运行的安全；
3. 提高编译器效率，增加运行速度；
4. 为未来新版本的Javascript做好铺垫。
注：经过测试 IE6,7,8,9 均不支持严格模式。

缺点：
现在网站的 JS 都会进行压缩，一些文件用了严格模式，而另一些没有。这时这些本来是严格模式的文件，被 merge 后，这个串就到了文件的中间，不仅没有指示严格模式，反而在压缩后浪费了字节。

30.javascript里面的继承怎么实现，如何避免原型链上面的对象共享?
用构造函数和原型链的混合模式去实现继承，避免对象共享可以参考经典的extend()函数，很多前端框架都有封装的，就是用一个空函数当做中间变量

31.javascript继承的 6 种方法
原型链 继承
借用构造函数 继承
组合继承(原型+借用构造)
原型式 继承
寄生式 继承
寄生组合式 继承

32.javascript对象的几种创建方式
工厂模式
构造函数模式
原型模式
组合使用构造函数模式和原型模式
动态原型模式
寄生构造函数模式
稳妥构造函数模式

33.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。
会引起内存泄漏的操作：
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）

34.call()/apply()的作用？
改变函数运行的作用域。
apply方法接收两个参数：一个是函数运行的作用域，另一个是参数数组。
call方法作用与apply方法相同，区别在于接收参数方式不同，在使用call方法时，传递给函数的参数需逐个列举出来。

35.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

36.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
```

1. 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
```

1. 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
```