2017-12-01Array

数组API

1 . instanceof

```
检测一个对象是否是数组;(用来对付复杂数据类型;)
// 简单数据类型 typeof ;
A instanceof B                // A是不是B造出来的;
eg : var arr = [1,2,3];
    console.log(arr instanceof Array);       //arr属不属于Array类型;
```

2 . Array.isArray( )

```
Array.isArray(参数); // 判断参数是不是数组,返回布尔值;
eg : var arr = [1,2,3];
    var num = 123;
    console.log(Array.isArray(arr));        //true
    console.log(Array.isArray(num));        //false
```

3 . toString( )

```
数组.toString(); // 把数组变成字符串,去除了[],内容用逗号链接;
eg : var arr = ["aaa","bbb","ccc"];
     console.log(arr.toString());           //返回 aaa,bbb,ccc
```

4 . valueOf( )

```
数组.valueOf();  //返回数组本身;
eg : var arr = ["aaa","bbb","ccc"];
    console.log(arr.valueOf());          //返回数组本身  ["aaa","bbb","ccc"]
```

5 . join(参数)

```
数组.join(参数);  // 数组中的元素可以按照参数进行链接变成一个字符串;
eg : console.log(arr.join());            //和toString()一样用逗号链接
    console.log(arr.join());             //和toString()一样用逗号链接
    console.log(arr.join("|"));          //用参数链接
    console.log(arr.join("&"));          //用参数链接
    console.log(arr.join(" "));          //如果是空格, 真的用空格链接
    console.log(arr.join(""));           //空字符是无缝连接
```

数组的添加和删除

1 . push() 和 pop()

```
1. 数组.push()         //在数组的最末尾添加元素;
2. 数组.pop()          //不需要参数;在数组的最末尾删除一项;
eg : var arr = [1,2,3];
    var aaa = arr.push("abc");       //在数组的最末尾添加一个元素;
    console.log(arr);        //元素被修改了
    console.log(aaa);        //返回值是数组的长度;
    aaa = arr.pop();         //不需要参数;在数组的最末尾删除一项;
```

```
    console.log(arr);        //元素被修改了
    console.log(aaa);        //被删除的那一项
```

2 . unshift() 和 shift()

```
1. 数组.unshift()          //在数组的最前面添加一个元素;
2. 数组.shift()        //不需要参数;在数组的最前面删除一项;
eg : var arr = [1,2,3];
    aaa = arr.unshift("abc");        //在数组的最前面添加一个元素;
    console.log(arr);        //元素被修改了
    console.log(aaa);        //返回值是数组的长度;
    aaa = arr.shift();       //不需要参数;在数组的最前面删除一项;
    console.log(arr);        //元素被修改了
    console.log(aaa);        //被删除的那一项
```

数组元素的排序

1 . reverse()

```
reverse()     //翻转数组
eg : var arr1 = [1,2,3,4,5];
    var aaa = arr1.reverse();        // [5,4,3,2,1]
```

2 . sort( )

```
sort()       // 数组中元素排序;(默认：从小到大)
 //  默认：按照首个字符的Unicode编码排序;如果第一个相同那么就比较第二个...
 eg : var arr = [4,5,1,3,2,7,6];
    var aaa =arr.sort();
    console.log(aaa);           // [1, 2, 3, 4, 5, 6, 7]
    console.log(aaa === arr);        // true 原数组被排序了(冒泡排序)
    //默认还可以排列字母;
    var arr2 = ["c","e","d","a","b"];
    var bbb = arr2.sort();
    console.log(bbb);          // ["a", "b", "c", "d", "e"]
    console.log(bbb===arr2);         // true 原数组被排序了(冒泡排序)
sort() //数值大小排序方法,需要借助回调函数;
eg : var arr = [4,5,1,13,2,7,6];
     //回调函数里面返回值如果是：参数1-参数2;升幂;     参数2-参数1;降幂;
    arr.sort(function (a,b) {
        return a-b; //升序
        //return b-a; //降序
        //return b.value-a.value; //按照元素value属性的大小排序;
    });
    console.log(arr); // [1, 2, 4, 5, 6, 7, 13]

sort( ) 底层原理
var aaa = bubbleSort([1,12,3], function (a,b) {
//        return a-b;//实参：array[j]-array[j+1];
        return b-a;//实参：array[j+1]-array[j];
    });
    console.log(aaa);
    function bubbleSort(array,fn){
        //外循环控制轮数，内循环控制次数，都是元素个数-1;
        for(var i=0;i<array.length-1;i++){
```

```
            for(var j=0;j<array.length-1-i;j++){//次数优化，多比较一轮，少比较一次；
                //满足条件交换位置；
                // if(array[j]>array[j+1]){//大于升幂排序；否则降幂；
                //a-b>0  和  a>b是一个意思；
                //b-a>0  和  a<b是一个意思；
                // if(array[j]-array[j+1]>0){//升幂排序
                //if(array[j+1]-array[j]>0){//降幂排序
                //把两个变量送到一个函数中；
                if(fn(array[j],array[j+1])>0){
                    var temp = array[j];
                    array[j] = array[j+1];
                    array[j+1] = temp;
                }
            }
        }
        //返回数组
        return array;
    }
```

数组元素的操作

1 . concat( )

```
数组1.concat(数组2); // 链接两个数组;
eg : var arr1 = [1,2,3];
    var arr2 = ["a","b","c"];
    var arr3 = arr1.concat(arr2);
    console.log(arr3)            //    [1, 2, 3, "a", "b", "c"]
```

2 . slice( )

```
数组.slice(开始索引值，结束索引值);      //数组截取;
eg : var arr = [1, 2, 3, "a", "b", "c"];
  console.log(arr.slice(3));              //从索引值为3截取到最后;["a", "b", "c"]
  console.log(arr.slice(0,3));            //包左不包右;[1, 2, 3]
  console.log(arr.slice(-2));             //负数是后几个;["b", "c"]
  console.log(arr.slice(3,0));            //如果前面的比后面的大，那么就是[];[]
  console.log(arr);                       //原数组不被修改;[1, 2, 3, "a", "b", "c"]
```

3 . splice( )

```
数组.splice(开始索引值，删除几个，替换内容1，替换内容2，...);  // 替换和删除;
eg : var arr = [1,2,3,4,5,6,"a", "b", "c"]
    arr.splice(5);          //从索引值为3截取到最后;(删除)
    console.log(arr);       // [1, 2, 3, 4, 5]
    arr.splice(1,2);        //(删除指定个数)从索引为1的开始删除2个
    console.log(arr);       //[1, 4, 5]
```

4 . indexOf / lastIndexOf

```
数组.indexOf(元素);       // 给元素，查索引(从前往后)
数组.lastIndexOf(元素);   // 给元素，查索引(从后往前)
eg : var arr = ["a","b","c","d","c","b","b"];
    console.log(arr.indexOf("b"));            // 1 查到以后立刻返回
    console.log(arr.lastIndexOf("b"));        // 6 找到以后立刻返回
```

```
    console.log(arr.indexOf("xxx"));          // -1;  查不到就返回-1;
```

数组的迭代

1 . every()

```
对数组中每一项运行回调函数，如果都返回true，every返回true，
如果有一项返回false，则停止遍历 every返回false；不写默认返回false
像保镖失误一次，游戏结束！！！
eg : var arr = [111,222,333,444,555];
    arr.every(function (a,b,c) {
        console.log(a);          //元素
        console.log(b);          //索引值
        console.log(c);          //数组本身;
        console.log("-----");            //数组本身;
        //数组中元素赋值：c[b] = 值;       a=有时候无法赋值;
        return true;
    });
//every返回一个bool值，全部是true才是true；有一个是false，结果就是false
eg : var bool = arr.every(function (element, index, array) {
        //判断：我们定义所有元素都大于200;
        //if(element > 100){
        if(element > 200){
            return true;
        }else{
            return false;
        }
    })
    alert(bool); //false
```

2 . filter()

```
//   对数组中每一项运行回调函数，该函数返回结果是true的项组成的新数组
//     新数组是有老数组中的元素组成的，return为ture的项;
eg : var arr = [111,222,333,444,555];
    var newArr = arr.filter(function (element, index, array) {
        //只要是奇数，就组成数组;(数组中辨别元素)
        if(element%2 === 0){
            return true;
        }else{
            return false;
        }
    })
    console.log(newArr); // [222, 444]
```

3 . forEach()

```
// 和for循环一样；没有返回值;
eg : var arr = [111,222,333,444,555];
    var sum = 0;
    var aaa = arr.forEach(function (element,index,array) {
        console.log(element);        // 输出数组中的每一个元素
        console.log(index);          // 数组元素对应的索引值
        console.log(array);          // 数组本身 [111, 222, 333, 444, 555]
        sum += element;              //数组中元素求和;
```

```
    });
    console.log(sum);        // 数组元素加起来的和
    console.log(aaa);        //undefined; 没有返回值 所以返回undefined
```

4 . map()

```
//  对数组中每一项运行回调函数，返回该函数的结果组成的新数组
//    return什么新数组中就有什么; 不return返回undefined; 对数组二次加工
eg : var arr = [111,222,333,444,555];
    var newArr = arr.map(function (element, index, array) {
        if(index == 2){
            return element;          // 这里return了 所以下面返回的值是333
        }
        return element*100;          // 返回的元素值都乘上100后的值
    })
    console.log(newArr);             // [11100, 22200, 333, 44400, 55500]
```

5 . some()

```
//对数组中每一项运行回调函数，如果该函数对某一项返回true，则some返回true; 像杀手，有一个成功，就胜利了！！！
eg : var arr = [111,222,333,444,555];
    var bool = arr.some(function (ele,i,array) {
        //判断：数组中有3的倍数
        if(ele%3 == 0){
            return true;
        }
        return false;
    })
    alert(bool); //true ; 有一个成功就是true
```

清空数组

```
1. arr.length = 0; // (不好，伪数组无法清空)
2. arr.splice(0); // 伪数组没有这个方法;
3. arr = [];      // 可以操作伪数组; (推荐!)
```

数组案例

数组的去重

```
 var arr = ["鸣人","鸣人","佐助","佐助","小樱","小樱"];
  //  方法1:   思路：定义一个新数组，遍历老数组，判断，如果新数组里面没有老数组的元素就添加，否则就不添加;
    var newArr = [];
    //遍历老数组
    arr.forEach(function (ele,index,array) {
        //检测老数组中的元素，如果新数组中存在就不添加了，不存在才添加;
        if(newArr.indexOf(ele) === -1){//不存在就添加;(去新数组中查找元素索引值，如果为-1就是没有)
            newArr.push(ele);
        }
    });
    console.log(newArr); // ["鸣人", "佐助", "小樱"]
```