## 1：数组

定义：数组是一个可以存储一组或一系列相关数据的容器。

为什么要使用数组？

1：为了解决大量相关数据的存储和使用的问题。

2：模拟真实的世界（班级、军队）。

## 2：创建数组

1：通过构造函数的方式来创建。 var a=new Array();

1.直接赋值

var a=new Array(数据1，数据2，…);

--------------------------------------------------------------

var a=new Array(数值)

如果括号中只有一个元素，并且这个元素是数值类型的，那么他就是指定数组的长度。 并且它的值都是undefined。

**数组的属性:length 属性** （获取整个数组的长度）。

2.声明以后再赋值

var a=new Array(); a[0]=1; a[1]=2; a[2]=3;

2：隐形声明的方式。 var a=[];

1.直接赋值:

var a=[1,2,3,4];

2.声明以后再赋值:

var a=[]; a[0]=1; a[1]=2; a[2]=3;

**JS数组可以存储任何类型的值。**

## 3：访问数组

通过数组的(中括号)下标访问。

数组下标从0开始，他的最大值，是length-1。有8个元素的话，则长度是8，如果要访问最后一个元素，则访问7。因为下标从0开始，01234567。

## 4：遍历数组

1：for循环。

2：while循环。

3：for in循环。

for in的作用：

1：用于数组的遍历。

2：用于对象属性的遍历。

## 5：数组分类

1：按照下标的类型

1.下标是数字的类型叫做(索引数组)

2.下标是字符串类型叫做(关联数组)，必须通过对象模拟的形式才能实现，一般不用。

2：按照维度来分类

1.一维数组

2.二维数组，通过对象模拟的形式才能实现。

声明二维数组:

var arr=[[1,2,3],[4,5,6]]; alert(arr[1][1]);

## 应用：

1：山上有一口缸可以装50升水，现在有15升水。老和尚叫小和尚下山挑水，每次可以挑5升。问：小和尚要挑几次水才可以把水缸挑满？通过编程解决这个问题。

2：定义一个含有30个整型元素的数组，按顺序分别赋予从2开始的偶数；然后按顺序每五个数求出一个平均值，放在另一个数组中并输出。试编程。

3：通过循环按行顺序为一个5×5的二维数组a赋1到25的自然数，然后输出该数组的左下半三角。试编程。

## 1：数组属性

1.length

设置或返回数组元素的数目。

2.constructor

返回构造函数的引用。

## 2：数组方法

1. arr.push(数组元素......) 参数之间要用逗号隔开。

向数组的末尾添加新的元素，返回值是新数组的长度。

可以一次添加多个元素。

2.arr.unshift(数组元素.....)

向数组的开头加入新的元素，返回值是新数组的长度。

可以一次添加多个元素。

3. arr.pop()

删除数组的最后一个元素，返回删除的元素。

4. arr.shift()

删除数组的第一个元素，返回删除的元素。

5.万能的添加删除函数

arr.splice(index,数量,添加的元素.....) 前俩个元素必须有

(1)index 从何处开始添加或删除，必须是数值类型(数组的下标)

(2)数量 规定了删除的个数，如果是0，则不删除。

(3)需要添加的元素，可以当作替换的元素。

如果有删除的元素，返回删除的元素。

6.str.split(分隔符)

将字符串分割为数组。

7.arr.join([分隔符])

把数组元素按照指定分隔符组合成一个字符串，如果没有指定分隔符，默认是用“,”分割，返回结果就是组合成的字符串。

8.arr.slice()

数组的分割。

从截取指定的开始位置，到结束位置(不包括)的元素。如果不指定结束位置，则从指定的开始位置，取到结尾(数组的下标)。

支持负数(-1开头) 返回的是新数组。 **不改动原来的数组。**

9. arr.concat()

连接两个或更多的数组，并返回新数组，对原数组没有任何影响。

10.arr.reverse()

数组翻转方法。改变原数组

11.arr.sort()

对数组进行排序，如果没有参数，则按照字母的编码进行排序，如果要按照其他的顺序来排序，要提供一个函数。

回调函数会提供两个参数(a,b)。

a<b a在b前。

a=b 顺序按照原样输出。

a>b b在a前。

默认排序例子：

['d', 'c', 'b', 'a'].sort()

// ['a', 'b', 'c', 'd']

[4, 3, 2, 1].sort()

// [1, 2, 3, 4]

[11, 101].sort()

// [101, 11]

[10111, 1101, 111].sort()

// [10111, 1101, 111]

如果想让sort方法按照自定义方式排序，可以传入一个函数作为参数，表示按照自定义方法进行排序。该函数本身接受两个参数，表示进行比较的两个元素。如果返回值大于0，表示第一个元素排在第二个元素后面；其他情况下，都是第一个元素排在第二个元素前面。

[10111, 1101, 111].sort(function (a, b) {

return a - b;

}) // [111, 1101, 10111]

[ { name: "张三", age: 30 }, { name: "李四", age: 24 }, { name: "王五", age: 28 } ].sort(function (o1, o2) {

return o1.age - o2.age;

}) // [ { name: "李四", age: 24 }, { name: "王五", age: 28 }, { name: "张三", age: 30 } ]

## 3：排序算法

1：冒泡排序

思路分析：在要排序的一组数中，对当前还未排好的序列，从前往后对相邻的两个数依次进行比较和调整，让较大的数往下沉，较小的往上冒。

即：每当两相邻的数比较后发现它们的排序与排序要求相反时，就将它们互换。

function bubbleSort (arr) {

// 控制冒泡的次数  
 for (var i = 1;i<arr.length;i++) {

// 控制每轮冒出一个数需要比较的次数  
 for (var j = 0; j < arr.length-i; j++) {  
 if (arr[j] > arr[j+1]) {  
 var temp=arr[j];  
 arr[j]=arr[j+1];  
 arr[j+1]=temp;  
 }  
 }  
 }  
 return arr;  
}

2：选择排序

思路分析：选择排序和冒泡排序类似，也是依次对相邻的数进行两两比较。不同之处在于，它不是每比较一次就调换位置，而是一轮比较完毕，找到最大值(或者最小值)之后，将其放在正确的位置，其他数的位置不变。

function selectionSort(arr){

var len = arr.length,min;

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

// 将当前位置设为最小值

min = i;

// 检查数组其余部分是否更小

for (j=i+1; j < len; j++){

if (arr[j] < arr[min]){ min = j; }

}

// 如果当前位置不是最小值，将其换为最小值

if (i != min){

var temp = arr[i];

arr[i] = arr[min]; arr[min] = temp;

}

} return arr;

}

3：插入排序（扩展）

思路分析：在要排序的一组数中，假设前面的数已经是排好顺序的，现在要把第n个数插到前面的有序数中，使得这n个数也是排好顺序的。如此反复循环，直到全部排好顺序。

function insertSort(arr) {

var len=arr.length;

for(var i=1; i < len; i++) {

var tmp = arr[i];

//内层循环控制，比较并插入

for(var j = i-1;j >= 0; j--) {

if(tmp < arr[j]) {

//发现插入的元素要小，交换位置，将后边的元素与前面的元素互换

arr[j+1] = arr[j];

arr[j] = tmp;

}

}

}

return arr;

}

4：快速排序（扩展）

选择一个基准元素，通常选择第一个元素或者最后一个元素。通过一趟扫描，将待排序列分成两部分，一部分比基准元素小，一部分大于等于基准元素。此时基准元素在其排好序后的正确位置，然后在用同样的方法递归的排序划分的两部分。

function quickSort (arr) {

if (arr.length <= 1) { return arr; }

var pivotIndex = Math.floor(arr.length / 2);

var pivot = arr.splice(pivotIndex, 1)[0];

var left = [];

var right = [];

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

if (arr[i] < pivot) {

left.push(arr[i]);

} else {

right.push(arr[i]);

}

}

return quickSort(left).concat([pivot], quickSort(right));

}

## 4：值传递和引用传递

1：值传递

定义：调用函数时将实际参数复制一份传递到函数中，这样在函数中如果对参数进行修改，将不会影响到实际参数。

2：引用传递

定义：调用函数时将实际参数的地址传递到函数中，这样在函数中如果对参数所进行的修改，将影响到实际参数。

## 应用：

1：数组的冒泡排序。

2：数组的选择排序。

3：编写函数map(arr) 把数组中的每一位数字都增加30%。

4：编写函数has(arr , 60) 判断数组中是否存在60这个元素，返回布尔类型。

## 综合应用：

1：编写函数norepeat(arr) 将数组的重复元素去掉，并返回新的数组。

2：有一个**从小到大排好序**的数组。现输入一个数，要求按原来的规律将它插入数组中。

3：以下是某班级一次考试的成绩表。请计算总成绩，并按总成绩排名。统计各单科成绩第一名，输出其成绩与学号。

