DAY06 数组的方法和数组的排序

2023年4月25日 18:02

冒泡排序

- 先遍历数组，让挨着的两个进行比较，如果前一个比后一个大，那么就把两个换个位置
- 数组遍历一遍以后，那么最后一个数字就是最大的那个了
- 然后进行第二遍的遍历，还是按照之前的规则，第二大的数字就会跑到倒数第二的位置
- 以此类推，最后就会按照顺序把数组排好了

```
var arr = [14, 5, 11, 7, 15, 2];
// 继续编写循环，控制比较多少遍(6个数字比较多少轮最终排好)
for (var i = 0; i < arr.length - 1; i++) {//-1:6个数字比较5轮
  // 先遍历数组,开始第一轮
  for (var j = 0; j < arr.length; j++) {
    // 让挨着(arr[j] arr[j+1])的两个进行比较
    if (arr[j] > arr[j + 1]) {
      // 如果前一个比后一个大，那么就把两个换个位置
      var temp = arr[j];
      arr[j] = arr[j + 1];
      arr[j + 1] = temp;
    }
  }
}
console.log(arr);
```

```
优化上面的代码
var arr = [14, 5, 11, 7, 15, 2];
for (var i = 0; i < arr.length - 1; i++) {//比较的轮数
  for (var j = 0; j < arr.length - 1 - i; j++) {//每一轮的比较，-1表示6个数字仅需要两两比较5次  -i:每一轮都会排好一个数字，下次两两比较次数也要减少。
    if (arr[j] > arr[j + 1]) {
      var temp = arr[j];
      arr[j] = arr[j + 1];
      arr[j + 1] = temp;
    }
  }
}
console.log(arr);
```

选择排序

- 先假定数组中的第 0 个就是最小的数字的索引
- 然后遍历数组，只要有一个数字比我小，那么就替换之前记录的索引
- 直到数组遍历结束后，就能找到最小的那个索引，然后让最小的索引换到第 0 个的位置
- 再来第二趟遍历，假定第 1 个是最小的数字的索引

- 在遍历一次数组，找到比我小的那个数字的索引
- 遍历结束后换个位置
- 依次类推，也可以把数组排序好

```
var arr = [1, 5, 11, 7, 15, 2];
for (var i = 0; i < arr.length; i++) {
  var minIndex = i;//假设的最小值的索引
  for (var j = i + 1; j < arr.length; j++) {
    if (arr[minIndex] > arr[j]) {//当前的值比假设的最小值值还要小，当前的值就是最小值。
      minIndex = j;
    }
  }
  // 走到这里，表示上面的for循环已经得到了第一遍比较的最小值的索引。
  // 进行交换位置
  if (i !== minIndex) {//这里的条件是优化，如果假设的最小值真的就是最小值，无需交换的。
    var temp = arr[i];
    arr[i] = arr[minIndex];
    arr[minIndex] = temp;
  }
}
console.log(arr);
```

实例方法(原型方法)

实例方法是ES提供的，提供给实例对象使用的。

必须先拿到实例对象，使用方法

方法主要依赖的就是记忆，多使用这些方法，采取一些合适的方式去记住它的功能，参数配置，返回值...

栈方法和队列方法

一.栈方法(后进先出)

1.push() 可以接收任意数量的参数，把它们逐个添加到数组末尾，并返回修改后数组的长度，改变原数组

```
// var arr = ['zhangsan', 'lisi'];
// console.log(arr.push('wangwu', 'zhaoliu', 'sunqi'));//5,改变后的数组的擦汗高难度
// console.log(arr);//['zhangsan', 'lisi', 'wangwu', 'zhaoliu', 'sunqi']
```

2.pop() 从数组末尾移除最后一项，减少数组的length值，然后返回移除的项，改变原数组

```
// var arr = ['zhangsan', 'lisi', 'wangwu', 'zhaoliu', 'sunqi'];
// console.log(arr.pop());//sunqi  返回移除的项
// console.log(arr.pop());//zhaoliu  返回移除的项
// console.log(arr);
```

二.队列方法(先进先出)

1.unshift()可以接收任意数量的参数，把它们逐个添加到数组前面，并返回修改后数组的长度，改变原数组。

```
// var arr = [1, 2, 3];
// console.log(arr.unshift(0));//4
// console.log(arr);//[0,1,2,3]
```

2.shift() 从数组前面移除一项，减少数组的length值，然后返回移除的项，改变原数组。

```
// var arr = ['zhangsan', 'lisi', 'wangwu', 'zhaoliu', 'sunqi', 'wangba'];
// for (var i = 0, len = arr.length; i < len; i++) {
//   arr.shift();
// }
// console.log(arr);
```

重排序方法

1.reverse() 方法会反转数组项的顺序，改变原数组

```
// var arr = [1, 2, 3, 4, 5];
// console.log(arr.reverse());//[5, 4, 3, 2, 1]
// console.log(arr);//[5, 4, 3, 2, 1]
```

2.sort() 方法按字母升序排列数组项

unicode编码

0-9 48-57

A-Z 65-90

a-z 97-122

```
// var arr = ['liuyi', 'chener', 'zhangsan', 'lisi'];
// console.log(arr.sort());//['chener', 'lisi', 'liuyi', 'zhangsan']
```

如果按其他方式而非字母表顺序进行数组排列必须给sort方法传递一个比较函数做参数，改变原数组排序数字,利用作者提供的比较函数做方法的参数

```
var arr = [9, 6, 3000, 100, 4, 7, 5, 20, 8];
// function compare1(value1, value2) {//升序
//   return value1 - value2;
// }
// function compare2(value1, value2) {//降序
//   return value2 - value1;
// }
// console.log(arr.sort(compare1));
// console.log(arr.sort(compare2));
```

推荐简洁的写法

```
console.log(arr.sort(function (a, b) { return a - b }));//[4, 5, 6, 7, 8, 9, 20, 100, 3000]
console.log(arr.sort(function (a, b) { return b - a }));//[3000, 100, 20, 9, 8,
```

```
7, 6, 5, 4]
```

操作方法和转换方法

一.操作方法

1.concat() 方法可以基于当前数组中的所有项创建一个新数组，参数可以是数组项或者数组，不改变原数组

```
// var arr1 = [1, 2];
// var arr2 = [3, 4];
// var arr3 = [5, 6];
// var newArr = arr1.concat(arr2, arr3);
// console.log(newArr);// [1, 2, 3, 4, 5, 6]
// var arr = [];
// arr.push(arr1, arr2, arr3);
// console.log(arr);//[[1,2],[3,4],[5,6]]
```

2.slice() 方法它能够基于当前数组中的一或多项创建一个新数组, 可以接受一或两个参数，即要返回项的起始和结束位置，不包括结束位置，不改变原数组

```
// var arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];
// console.log(arr.slice());//全部截取  ['a', 'b', 'c', 'd', 'e', 'f', 'g']
// console.log(arr.slice(2));//从索引为2的位置开始截取 ['c', 'd', 'e', 'f', 'g']
// console.log(arr.slice(2, 5));//从索引2的位置开始截取到索引5位置,但不包括5这个位置['c', 'd', 'e']
```

了解特殊情况

负数索引从后面往前数位置，-1开始，截取依然还是从左往由

```
// console.log(arr.slice(2, -2));// ['c', 'd', 'e']
// console.log(arr.slice(-6, -2));// ['b','c', 'd', 'e']
// console.log(arr.slice(-60, -20));// [] 超过索引值，没有截取
```

3.splice() 方法对数组进行删除、插入、替换，是最强大的数组方法，返回值是数组，改变原数组。

3.1.删除

第一个参数删除的起始位置，第二个参数删除的长度，返回值是被删除的数组项组成的数组

```
// var arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];
// console.log(arr.splice(1, 3));//['b', 'c', 'd']
// console.log(arr);//['a', 'e', 'f', 'g']
```

3.2.插入

根据删除演变而来，删除0项，添加几项

第一个参数删除的起始位置，第二个参数删除的长度，返回值是被删除的数组项组成的数组

从第三个参数开始，就是要插入的数组项

```
// var arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];
```

```
// console.log(arr.splice(3, 0, 'hehe', 'haha'));//[]
// console.log(arr);//['a', 'b', 'c', 'hehe', 'haha', 'd', 'e', 'f', 'g']
```

3.3.替换

根据删除演变而来，删除几项，添加几项

```
// var arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];
// console.log(arr.splice(2, 3, 'zhangsan', 'lisi', 'wangwu'));//[ 'c', 'd', 'e']
// console.log(arr);//['a', 'b','zhangsan', 'lisi', 'wangwu','f', 'g']
```

二.转换方法

1.join()方法，将数组转换成对应的字符串，参数就是连接符，不改变原数组

将数组项拼接成一个字符串，可以设定用什么连接符进行拼接。

```
var arr = ['a', 'b', 'c', 'd', 'e', 'f', 'g'];
console.log(arr.join('#'));//a#b#c#d#e#f#g
console.log(arr.join('%'));//a%b%c%d%e%f%g
console.log(arr.join());//a,b,c,d,e,f,g    注意使用默认逗号进行拼接
console.log(arr.join(''));//abcdefg    使用较多
console.log(arr.join(' '));//a b c d e f g
console.log(arr);//['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

位置方法

indexOf() lastIndexOf()返回要查找的数组项在数组中的索引位置，没找到的情况下返回-1。

两个参数：要查找的数组项 和 表示查找起点位置的索引（可选的）

indexOf() 方法从数组的开头开始向后查找。

lastIndexOf() 方法则从数组的末尾开始向前查找。

```
var arr = ['a', 'b', 'c', 'd', 'a', 'b', 'c', 'd'];
console.log(arr.indexOf('c'));//2    左往右，找到输出对应的索引
console.log(arr.indexOf('c'));//2    左往右，找到输出对应的索引
console.log(arr.indexOf('c', 3));//6    索引3位置开始查找c数组项
```

特殊：负数从后往前数，-1开始

```
var arr = ['a', 'b', 'c', 'd', 'a', 'b', 'c', 'd'];
console.log(arr.indexOf('c', -3));//6 负数从后往前数位置
console.log(arr.lastIndexOf('c'));//6
console.log(arr.indexOf('f'));//-1
console.log(arr.indexOf('g'));//-1
console.log(arr.lastIndexOf('g'));//-1
```

数组去重

将数组中的重复项去掉

1.思路 - 利用indexOf

准备一个新数组 []

查看待去重的数组项是否存在新数组中。 利用indexOf输出的-1进行判断

存在，不添加进新数组

不存在，添加进新数组

```
// var arr = [1, 2, 3, 4, 4, 4, 4, 3, 2, 1];
// var newArr = [];
// for (var i = 0; i < arr.length; i++) {
//   if (newArr.indexOf(arr[i]) === -1) {//不满足条件，不存在，添加进新数组
//     newArr.push(arr[i])
//   }
// }
// console.log(arr);
// console.log(newArr);
```

迭代方法

迭代的两层函数

是重复反馈过程的活动(遍历)，其目的通常是为了逼近所需目标或结果

上一次结果作为下一次的初始值。

新的5个迭代方法(ES5新增的方法,前面学习的方法都是ES5之前的方法)，自带遍历。

参数

第一个参数：函数(做参数的函数又有三个参数)

第二个参数：可选的参数，表示this指向(暂时忽略)

1.forEach() 对数组中的每一项运行给定函数,这个方法没有返回值(取代for循环遍历数组)。

```
// var arr = [100, 200, 300, 400, 500, 600, 700, 800, 900];
// arr.forEach(function (value, index, array) {//value:数组项的值，index:数组项的索引，array数组本身
//   console.log(value, index, array);
// });
```

2.map() 对数组中的每一项运行给定函数，返回每次函数调用的结果组成的数组，返回就是数组.(不考虑true或者false)

映射：一一对应,每一个数组项一定对应一个结果

```
// var arr = [100, 200, 300, 400, 500, 600, 700, 800, 900];
// var result = arr.map(function (value, index, array) {//value:数组项的值，index:数组项的索引，array数组本身
//   // return value / 100;
//   // return value >= 500;
//   return '' + value;
// });
// console.log(result);
```

3.filter() 对数组中的每一项运行给定函数，返回该函数结果为 true 的项组成的数组。

过滤，筛选，根据条件得到需要的数组项

```
// var arr = [100, 200, 300, 400, 500, 600, 700, 800, 900];
// var result = arr.filter(function (value, index, array) {//value:数组项的值,
index:数组项的索引, array数组本身
//   return value >= 500
// });
// console.log(result);//[500, 600, 700, 800, 900]
```

4.some()对数组中的每一项运行给定函数，如果该函数对任意一项返回 true ，则返回 true。

```
// var arr = [100, 200, 300, 400, 500, 600, 700, 800, 900];
// var result = arr.some(function (value) {
//   return value >= 900;
// });
// console.log(result);
```

此方法比较适合找存在感

```
// var arr = [100, 200, 300, '400', 500, 600, 700, 800, 900];
// var result = arr.some(function (value) {
//   return typeof value === 'string';
// });
// console.log(result);
```

5.every()对数组中的每一项运行给定函数，如果该函数对每一项都返回 true,则返回 true 。

```
// var arr = [100, 200, 300, 400, 500, 600, 700, 800, 900];
// var result = arr.every(function (value) {
//   return typeof value === 'number';
// });
// console.log(result);
```

归并方法(统计)

即每一次迭代得到的结果会作为下一次迭代的初始值，这个方法都会迭代数组的所有项，然后构建一个最终返回的值。

类似于求和的含义

1.reduce() 方法，对数组中的每一项运行给定函数，从数组的第一项开始，逐个遍历到最后。

两个参数：每一项上调用的函数和（可选的）作为归并的初始值。

调用的函数接收4个参数：前一个值、当前值、项的索引和数组对象。

1.1.reduce只有一个函数做参数的情况

```
// var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
// var result = arr.reduce(function (prev, curr, index, array) {//前一个值(迭代的结
果)、当前值、数组项的索引和数组对象。
//   console.log(prev + '|' + curr);
//   return prev + curr;
//   // prev: 第一次指向数组的第一项值1，后面就指向前面迭代的结果
//   // curr: 指向数组从第二项开始的值2-10
// });
```

```
// console.log(result);//55
```

1.2.reduce有两个参数

每一项上调用的函数和（可选的）作为归并的初始值。

```
// var arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
// var result = arr.reduce(function (prev, curr) {
//   return prev + curr;
//   // prev: 第一次reduce第二个参数1000，后面就指向前面迭代的结果
//   // curr: 指向数组的每一项1-10
// }, 1000);//1000迭代的初始值 prev第一次指向这里
// console.log(result); //1055
```