重排序方法

数组中已经存在两个可以直接用来重排序的方法：reverse()和 sort()。有读者可能猜到了，reverse()方法会反转数组项的顺序。请看下面这个例子。

```
var values = [1, 2, 3, 4, 5];
values.reverse();
alert(values);         //5,4,3,2,1
```

在默认情况下，sort()方法按升序排列数组项——即最小的值位于最前面，最大的值排在最后面。为了实现排序，sort()方法会调用每个数组项的 toString()转型方法，然后比较得到的字符串，以确定如何排序。即使数组中的每一项都是数值，sort()方法比较的也是字符串，如下所示。

```
var values = [0, 1, 5, 10, 15];
values.sort();
alert(values);      //0,1,10,15,5
```

可见，即使例子中值的顺序没有问题，但 sort()方法也会根据测试字符串的结果改变原来的顺序。因为数值 5 虽然小于 10，但在进行字符串比较时，"10"则位于"5"的前面，于是数组的顺序就被修改了。不用说，这种排序方式在很多情况下都不是最佳方案。因此 sort()方法可以接收一个比较函数作为参数，以便我们指定哪个值位于哪个值的前面。

比较函数接收两个参数，如果第一个参数应该位于第二个之前则返回一个负数，如果两个参数相等则返回 0，如果第一个参数应该位于第二个之后则返回一个正数。以下就是一个简单的比较函数：

比较函数接收两个参数，如果第一个参数应该位于第二个之前则返回一个负数，如果两个参数相等则返回 0，如果第一个参数应该位于第二个之后则返回一个正数。

```
function compare(value1, value2) {
    if (value1 < value2) {
        return -1;
    } else if (value1 > value2) {
        return 1;
    } else {
        return 0;
    }
}
```

ArrayTypeExample15.htm

这个比较函数可以适用于大多数数据类型，只要将其作为参数传递给 sort()方法即可，如下面这个例子所示。

```
var values = [0, 1, 5, 10, 15];
values.sort(compare);
alert(values);   //0,1,5,10,15
```

在将比较函数传递到 sort()方法之后，数值仍然保持了正确的升序。当然，也可以通过比较函数产生降序排序的结果，只要交换比较函数返回的值即可。

```
function compare(value1, value2) {
    if (value1 < value2) {
        return 1;
    } else if (value1 > value2) {
        return -1;
    } else {
        return 0;
    }
}

var values = [0, 1, 5, 10, 15];
values.sort(compare);
alert(values);    // 15,10,5,1,0
```

在这个修改后的例子中，比较函数在第一个值应该位于第二个之后的情况下返回 1，而在第一个值应该在第二个之前的情况下返回–1。交换返回值的意思是让更大的值排位更靠前，也就是对数组按照降序排序。当然，如果只想反转数组原来的顺序，使用 reverse()方法要更快一些。

> reverse()和 sort()方法的返回值是经过排序之后的数组。

对于数值类型或者其 valueOf()方法会返回数值类型的对象类型，可以使用一个更简单的比较函数。这个函数只要用第二个值减第一个值即可。

```
function compare(value1, value2){
    return value2 - value1;
}
```


由于比较函数通过返回一个小于零、等于零或大于零的值来影响排序结果，因此减法操作就可以适当地处理所有这些情况。