reduce 方法（升序）

语法：

array1.reduce(callbackfn[, initialValue])

|  |  |
| --- | --- |
| **参数** | **定义** |
| *array1* | 必需。一个数组对象。 |
| *callbackfn* | 必需。一个接受最多四个参数的函数。对于数组中的每个元素，**reduce** 方法都会调用 *callbackfn* 函数一次。 |
| *initialValue* | 可选。如果指定 *initialValue*，则它将用作初始值来启动累积。第一次调用 *callbackfn* 函数会将此值作为参数而非数组值提供 |

返回值：

        通过最后一次调用回调函数获得的累积结果。

异常：

        当满足下列任一条件时，将引发 **TypeError** 异常：

* *callbackfn* 参数不是函数对象。
* 数组不包含元素，且未提供 *initialValue*。

回调函数语法：

    function callbackfn(previousValue, currentValue, currentIndex, array1)

    可使用最多四个参数来声明回调函数。

    下表列出了回调函数参数。

|  |  |
| --- | --- |
| **回调参数** | **定义** |
| *previousValue* | 通过上一次调用回调函数获得的值。如果向 **reduce** 方法提供 *initialValue*，则在首次调用函数时，*previousValue* 为 *initialValue*。 |
| *currentValue* | 当前数组元素的值。 |
| *currentIndex* | 当前数组元素的数字索引。 |
| *array1* | 包含该元素的数组对象。 |

第一次调用回调函数

在第一次调用回调函数时，作为参数提供的值取决于 **reduce** 方法是否具有 *initialValue* 参数。

如果向 reduce 方法提供 *initialValue*：

* *previousValue* 参数为 *initialValue*。
* *currentValue* 参数是数组中的第一个元素的值。

如果未提供 *initialValue*：

* *previousValue* 参数是数组中的第一个元素的值。
* *currentValue* 参数是数组中的第二个元素的值。

修改数组对象

数组对象可由回调函数修改。

下表描述了在 **reduce** 方法启动后修改数组对象所获得的结果。

|  |  |
| --- | --- |
| **reduce 方法启动后的条件** | **元素是否传递给回调函数** |
| 在数组的原始长度之外添加元素。 | 否。 |
| 添加元素以填充数组中缺少的元素。 | 是，如果该索引尚未传递给回调函数。 |
| 元素被更改。 | 是，如果该元素尚未传递给回调函数。 |
| 从数组中删除元素。 | 否，除非该元素已传递给回调函数。 |

实例：

1.下面的示例将数组值连接成字符串，各个值用“::”分隔开。由于未向 **reduce** 方法提供初始值，第一次调用回调函数时会将“abc”作为 *previousValue* 参数并将“def”作为 *currentValue* 参数。

1. function appendCurrent (previousValue, currentValue) {
2. return previousValue + "::" + currentValue;
3. }
4. var elements = ["abc", "def", 123, 456];
5. var result = elements.reduce(appendCurrent);
6. document.write(result);
7. // Output:
8. //  abc::def::123::456

2.下面的示例向数组添加舍入后的值。使用初始值 0 调用 **reduce** 方法。

1. function addRounded (previousValue, currentValue) {
2. return previousValue + Math.round(currentValue);
3. }
4. var numbers = [10.9, 15.4, 0.5];
5. var result = numbers.reduce(addRounded, 0);
6. document.write (result);
7. // Output: 27

3.下面的示例向数组中添加值。 *currentIndex* 和 *array1* 参数用于回调函数

1. function addDigitValue(previousValue, currentDigit, currentIndex, array) {
2. var exponent = (array.length - 1) - currentIndex;
3. var digitValue = currentDigit \* Math.pow(10, exponent);
4. return previousValue + digitValue;
5. }
6. var digits = [4, 1, 2, 5];
7. var result = digits.reduce(addDigitValue, 0);
8. document.write (result);
9. // Output: 4125

此题分析：

    首先赋予了初始值0，那么currentDigit就是从4开始的，调用方法四次，这样可以把四次方法调用的参数都写出来：(0,4,0,array)、(4,1,1,array)、(1,2,2,array)、(2,5,3,array)，再一次进行计算，由于初始值是0，所有只需要计算出每个方法的返回值最后相加即可。array.length始终为4，则四次计算的值分别为4000+100+20+5=4125

reduceRight 方法（降序）

 reduceRight的语法以及回调函数的规则和reduce方法是一样的，区别就是在与reduce是升序，即角标从0开始，而reduceRight是降序，即角标从arr.length-1开始。如果有初始值，则从最后一个数开始计算，如果没有初始值，则previousValue参数是数组中最后一个元素的值，currentValue是数组中倒数第二个元素的值。

示例：

1.下面的示例获取数组中值为 1 到 10 之间的元素。提供给 **reduceRight** 方法的初始值是一个空数组。

1. function Process2(previousArray, currentValue) {
2. var nextArray;
3. if (currentValue >= 1 && currentValue <= 10)
4. nextArray = previousArray.concat(currentValue);
5. else
6. nextArray = previousArray;
7. return nextArray;
8. }
9. var numbers = [20, 1, -5, 6, 50, 3];
10. var emptyArray = new Array();
11. var resultArray = numbers.reduceRight(Process2, emptyArray);
12. document.write("result array=" + resultArray);
13. // Output:
14. //  result array=3,6,1

2.**reduceRight** 方法可应用于字符串。下面的示例演示如何使用此方法反转字符串中的字符。

1. function AppendToArray(previousValue, currentValue) {
2. return previousValue + currentValue;
3. }
4. var word = "retupmoc";
5. var result = [].reduceRight.call(word, AppendToArray, "the ");
6. // var result = Array.prototype.reduceRight.call(word, AppendToArray, "the ");
7. document.write(result);
8. // Output:
9. // the computer

这里可以直接使用空数组调用reduceRight方法，并且使用call方法将参数引入。也可以是直接使用原型链的方式进行调用，即Array.prototype.reduceRight.call(word,

AppendToArray, "the ");