Array.prototype.reduce()

reduce() 方法对累加器和数组中的每个元素（从左到右）应用一个函数，将其减少为单个值。

```
var total = [0, 1, 2, 3].reduce(function(sum, value) {
  return sum + value;
}, 0);
// total is 6

var flattened = [[0, 1], [2, 3], [4, 5]].reduce(function(a, b) {
  return a.concat(b);
}, []);
// flattened is [0, 1, 2, 3, 4, 5]
```

语法

```
arr.reduce(callback[, initialValue])
```

参数

callback

执行数组中每个值的函数，包含四个参数：

accumulator

累加器累加回调的返回值; 它是上一次调用回调时返回的累积值，或 **initialValue** （如下所示）。

currentValue

数组中正在处理的元素。

currentIndex

数组中正在处理的当前元素的索引。 如果提供了 **initialValue** ，则索引号为0，否则为索引为1。

array

调用 reduce 的数组

initialValue

[可选] 用作第一个调用 callback 的第一个参数的值。 如果没有提供初始值，则将使用数组中的第一个元素。 在没有初始值的空数组上调用 reduce 将报错。

返回值

函数累计处理的结果

描述

reduce 为数组中的每一个元素依次执行 callback 函数，不包括数组中被删除或从未被赋值的元素，接受四个参数：

- accumulator
- currentValue
- currentIndex
- array

回调函数第一次执行时， accumulator 和 currentValue 的取值有两种情况：调用 reduce 时提供 initialValue ， accumulator 取值为 initialValue ， currentValue 取数组中的第一个值；没有提供 initialValue ， accumulator 取数组中的第一个值， currentValue 取数组中的第二个值。

注意：如果没有提供 initialValue ，reduce 会从索引1的地方开始执行 callback 方法，跳过第一个索引。如果提供 initialValue ，从索引0开始。

如果数组为空且没有提供 initialValue ，会抛出 TypeError 。如果数组仅有一个元素（无论位置如何）并且没有提供 initialValue ， 或者有提供 initialValue 但是数组为空，那么此唯一值将被返回并且 callback 不会被执行。

提供初始值通常更安全，正如下面的例子，如果没有提供 initialValue ，则可能有三种输出：

```
var maxCallback = ( pre, cur ) => Math.max( pre.x, cur.x );
var maxCallback2 = ( max, cur ) => Math.max( max, cur );

// reduce() without initialValue
[ { x: 22 }, { x: 42 } ].reduce( maxCallback ); // 42
[ { x: 22 }            ].reduce( maxCallback ); // { x: 22 }
[                      ].reduce( maxCallback ); // TypeError
```

```
8
9  // map/reduce; better solution, also works for empty arrays
10 [ { x: 22 }, { x: 42 } ].map( el => el.x )
11                        .reduce( maxCallback2, -Infinity );
```

reduce如何运行

假如运行下段代码：

```
1 [0, 1, 2, 3, 4].reduce(function(accumulator, currentValue, currentIndex,
2   return accumulator + currentValue;
3 });
```

callback 被调用四次，每次调用的参数和返回值如下表：

callback	accumulator	currentValue	currentIndex	array	return value
first call	0	1	1	[0, 1, 2, 3, 4]	1
second call	1	2	2	[0, 1, 2, 3, 4]	3
third call	3	3	3	[0, 1, 2, 3, 4]	6
fourth call	6	4	4	[0, 1, 2, 3, 4]	10

由 reduce 返回的值将是上次回调调用的值（10）。

你同样可以使用箭头函数的形式，下面的代码会输出跟前面一样的结果

您还可以提供Arrow Function 代替完整功能。 下面的代码将产生与上面的代码中相同的输出：

```
1 [0, 1, 2, 3, 4].reduce((prev, curr) => prev + curr );
```

如果你打算提供一个初始值作为 reduce 方法的第二个参数，以下是运行过程及结果：

```
[0, 1, 2, 3, 4].reduce(( accumulator , currentValue, currentIndex, array) => {
```

callback	accumulator	currentValue	currentIndex	array	return value
first call	10	0	0	[0, 1, 2, 3, 4]	10
second call	10	1	1	[0, 1, 2, 3, 4]	11
third call	11	2	2	[0, 1, 2, 3, 4]	13
fourth call	13	3	3	[0, 1, 2, 3, 4]	16
fifth call	16	4	4	[0, 1, 2, 3, 4]	20

这种情况下 reduce 返回的值是 20 。

例子

数组里所有值的和

```
var sum = [0, 1, 2, 3].reduce(function (a, b) {
  return a + b;
}, 0);
// sum is 6
```

你也可以写成箭头函数的形式：

```
var total = [ 0, 1, 2, 3 ].reduce(
  ( acc, cur ) => acc + cur,
  0
);
```

将二维数组转化为一维

```
var flattened = [[0, 1], [2, 3], [4, 5]].reduce(
  function(a, b) {
    return a.concat(b);
  },
  []
);
// flattened is [0, 1, 2, 3, 4, 5]
```

你也可以写成箭头函数的形式：

```
var flattened = [[0, 1], [2, 3], [4, 5]].reduce(
  ( acc, cur ) => acc.concat(cur),
  []
);
```

计算数组中每个元素出现的次数

```
var names = ['Alice', 'Bob', 'Tiff', 'Bruce', 'Alice'];

var countedNames = names.reduce(function (allNames, name) {
  if (name in allNames) {
    allNames[name]++;
  }
  else {
    allNames[name] = 1;
  }
  return allNames;
}, {});
// countedNames is:
// { 'Alice': 2, 'Bob': 1, 'Tiff': 1, 'Bruce': 1 }
```

使用扩展运算符和initialValue绑定包含在对象数组中的数组

```
// friends - an array of objects
// where object field "books" - list of favorite books
var friends = [{
  name: 'Anna',
  books: ['Bible', 'Harry Potter'],
  age: 21
```

```
}, {
  name: 'Bob',
  books: ['War and peace', 'Romeo and Juliet'],
  age: 26
}, {
  name: 'Alice',
  books: ['The Lord of the Rings', 'The Shining'],
  age: 18
}];

// allbooks - list which will contain all friends' books +
// additional list contained in initialValue
var allbooks = friends.reduce(function(prev, curr) {
  return [...prev, ...curr.books];
}, ['Alphabet']);

// allbooks = [
//   'Alphabet', 'Bible', 'Harry Potter', 'War and peace',
//   'Romeo and Juliet', 'The Lord of the Rings',
//   'The Shining'
// ]
```

Polyfill

```
// Production steps of ECMA-262, Edition 5, 15.4.4.21
// Reference: http://es5.github.io/#x15.4.4.21
// https://tc39.github.io/ecma262/#sec-array.prototype.reduce
if (!Array.prototype.reduce) {
  Object.defineProperty(Array.prototype, 'reduce', {
    value: function(callback /*, initialValue*/) {
      if (this === null) {
        throw new TypeError( 'Array.prototype.reduce ' +
          'called on null or undefined' );
      }
      if (typeof callback !== 'function') {
        throw new TypeError( callback +
          ' is not a function');
      }

      // 1. Let O be ? ToObject(this value).
      var o = Object(this);

```

```
      // 2. Let len be ? ToLength(? Get(O, "length")).
      var len = o.length >>> 0;

      // Steps 3, 4, 5, 6, 7
      var k = 0;
      var value;

      if (arguments.length >= 2) {
        value = arguments[1];
      } else {
        while (k < len && !(k in o)) {
          k++;
        }

        // 3. If len is 0 and initialValue is not present,
        //    throw a TypeError exception.
        if (k >= len) {
          throw new TypeError( 'Reduce of empty array ' +
            'with no initial value' );
        }
        value = o[k++];
      }

      // 8. Repeat, while k < len
      while (k < len) {
        // a. Let Pk be ! ToString(k).
        // b. Let kPresent be ? HasProperty(O, Pk).
        // c. If kPresent is true, then
        //    i.  Let kValue be ? Get(O, Pk).
        //    ii. Let accumulator be ? Call(
        //          callbackfn, undefined,
        //          « accumulator, kValue, k, O »).
        if (k in o) {
          value = callback(value, o[k], k, o);
        }

        // d. Increase k by 1.
        k++;
      }

      // 9. Return accumulator.
      return value;
    }
```

```
  });
}
```

如果您需要兼容不支持 `Object.defineProperty` 的JavaScript引擎，那么最好不要 polyfill `Array.prototype` 方法，因为你无法使其成为不可枚举的。

规范

Specification	Status	Comment
ECMAScript 5.1 (ECMA-262) Array.prototype.reduce	ST Standard	Initial definition. Implemented in JavaScript 1.8.
ECMAScript 2015 (6th Edition, ECMA-262) Array.prototype.reduce	ST Standard	
ECMAScript Latest Draft (ECMA-262) Array.prototype.reduce	LS Living Standard	

浏览器兼容性

Desktop Mobile

Feature	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari
Basic Support	(Yes)	(Yes)	3	9	10.5	4

相关链接

- `Array.prototype.reduceRight()`