

[Công nghệ ▾](#)[Tài liệu tham khảo & hướng dẫn ▾](#)[Phản hồi ▾](#)[Đăng nhập !\[\]\(d66ff64371a51729ac8c1cdaa685ba6f_img.jpg\)](#)

Array.prototype.reduce()

Phương thức **reduce()** dùng để thực thi một hàm lên từng phần tử của mảng (từ trái sang phải) với một biến tích lũy để thu về một giá trị duy nhất.

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

JavaScript Demo: Array.reduce()



```
1 const array1 = [1, 2, 3, 4];
2 const reducer = (accumulator, currentValue) => accumulator + currentValue;
3
4 // 1 + 2 + 3 + 4
5 console.log(array1.reduce(reducer));
6 // expected output: 10
7
8 // 5 + 1 + 2 + 3 + 4
9 console.log(array1.reduce(reducer, 5));
10 // expected output: 15
11
```

Run ›

Reset

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

Các tham số



callback

Hàm dùng để thực thi với từng phần tử (element) của mảng, nhận vào 04 tham số:

accumulator

Biến tích lũy, truyền giá trị trả về của mỗi lần gọi callback; nó là giá trị tích lũy được trả về trong lần gọi callback trước, hoặc giá trị của tham số `initialValue`, nếu được cung cấp (xem bên dưới).

currentValue

Phần tử trong mảng hiện tại đang được xử lý.

currentIndex | Optional

Chỉ mục (index) của phần tử đang được xử lý. Bắt đầu tại 0, nếu giá trị `initialValue` được cung cấp, và tại 1 nếu không có `initialValue`.

array | Optional

Mảng đang được gọi với `reduce()`.

initialValue | Optional

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

Mô tả



`reduce()` thực thi hàm callback lên từng phần tử đang tồn tại trong mảng, bỏ qua những lỗ trống không giá trị, và nhận vào 04 tham số:

- `accumulator`
- `currentValue`
- `currentIndex`
- `array`

Trong lần đầu tiên callback được gọi, `accumulator` and `currentValue` có thể có một trong hai giá trị. Nếu tham số `initialValue` được cung cấp cho `reduce()`, thì `accumulator` sẽ bằng `initialValue`, và `currentValue` sẽ bằng phần tử đầu tiên của mảng. Nếu không có `initialValue`, `accumulator` sẽ bằng phần tử đầu tiên của mảng, và `currentValue` sẽ bằng phần tử thứ hai.

Ghi chú: Nếu `initialValue` không được cung cấp, `reduce()` sẽ thực thi callback bắt đầu từ index 1, bỏ qua index đầu tiên. Nếu `initialValue` được cung cấp, index sẽ bắt đầu từ 0.

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

Sẽ an toàn hơn nếu giá trị ban đầu được cung cấp, bởi vì có thể xảy ra lỗi nếu không có `initialValue` như ở ví dụ sau:



```
1  var maxCallback = ( acc, cur ) => Math.max( acc.x, cur.x );
2  var maxCallback2 = ( max, cur ) => Math.max( max, cur );
3
4  // reduce() không có initialValue
5  [ { x: 22 }, { x: 42 } ].reduce( maxCallback ); // 42
6  [ { x: 22 } ].reduce( maxCallback ); // { x: 22 }
7  [ ].reduce( maxCallback ); // TypeError
8
9  // map/reduce; giải pháp hay hơn, và nó có thể áp dụng được cho mảng rỗng hoặc lớn hơn
10 [ { x: 22 }, { x: 42 } ].map( el => el.x )
11   .reduce( maxCallback2, -Infinity );
```

Cách `reduce()` làm việc

Giả sử có một đoạn code với `reduce()` được hiện thực như sau:

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

callback	accumulator	currentValue	currentIndex	array	giá trị trả về
lần gọi thứ nhất	0	1	1	[0, 1, 2, 3, 4]	1
lần gọi thứ hai	1	2	2	[0, 1, 2, 3, 4]	3
lần gọi thứ ba	3	3	3	[0, 1, 2, 3, 4]	6
lần gọi thứ tư	6	4	4	[0, 1, 2, 3, 4]	10

Giá trị trả về cho `reduce()` chính là giá trị trả về của lần gọi callback cuối cùng (10).

Bạn cũng có thể cung cấp một hàm mũi tên **Arrow Function** thay vì một hàm đầy đủ. Đoạn code sau đây sẽ cho kết quả giống như đoạn code ở trên:

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

Nếu bạn cung cấp giá trị `initialValue` cho tham số thứ hai của hàm `reduce()`, thì kết quả sẽ như bên dưới:

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

callback	accumulator	currentValue	currentIndex	array	giá trị trả về
lần gọi thứ nhất	10	0	0	[0, 1, 2, 3, 4]	10
lần gọi thứ hai	10	1	1	[0, 1, 2, 3, 4]	11
lần gọi thứ ba	11	2	2	[0, 1, 2, 3, 4]	13
lần gọi thứ tư	13	3	3	[0, 1, 2, 3, 4]	16
lần gọi thứ năm	16	4	4	[0, 1, 2, 3, 4]	20

Giá trị trả về cho `reduce()` lần này sẽ là 20.

Ví dụ

Tính tổng của tất cả các phần tử của mảng 

```
1 | var sum = [0, 1, 2, 3].reduce(function (accumulator, currentValue) {
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
1 var total = [ 0, 1, 2, 3 ].reduce(  
2   ( accumulator, currentValue ) => accumulator + currentValue,  
3   0  
4 );
```



Tính tổng các giá trị bên trong một mảng các object [↗](#)

Để tính tổng các giá trị nằm bên trong các phần tử là object, bạn **phải** cung cấp một giá trị ban đầu để từng phần tử đều được callback chạy qua (và accumulator luôn luôn là giá trị kiểu số):

```
1 var initialValue = 0;  
2 var sum = [{x: 1}, {x: 2}, {x: 3}].reduce(function (accumulator, currentValue) {  
3   return accumulator + currentValue.x;  
4 }, initialValue)  
5  
6 console.log(sum) // logs 6
```

Tương tự, viết bằng arrow function:

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát


```
6  
7 console.log(sum) // logs 6
```



Trải phẳng một mảng chứa nhiều mảng con [🔗](#)

```
1 var flattened = [[0, 1], [2, 3], [4, 5]].reduce(  
2   function(accumulator, currentValue) {  
3     return accumulator.concat(currentValue);  
4   },  
5   []  
6 );  
7 // flattened is [0, 1, 2, 3, 4, 5]
```

Tương tự, viết bằng arrow function:

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

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
4   if (name in allNames) {  
5       allNames[name]++;  
6   }  
7   else {  
8       allNames[name] = 1;  
9   }  
10  return allNames;  
11 }, {}));  
12 // countedNames is:  
13 // { 'Alice': 2, 'Bob': 1, 'Tiff': 1, 'Bruce': 1 }
```



Nhóm các đối tượng theo giá trị property nào đó [🔗](#)

```
1  var people = [  
2    { name: 'Alice', age: 21 },  
3    { name: 'Max', age: 20 },  
4    { name: 'Jane', age: 20 }  
5  ];  
6  
7  function groupBy(objectArray, property) {
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
14     return acc;  
15   }, {}));  
16 }  
17  
18 var groupedPeople = groupBy(people, 'age');  
19 // groupedPeople is:  
20 // {  
21 //   20: [  
22 //     { name: 'Max', age: 20 },  
23 //     { name: 'Jane', age: 20 }  
24 //   ],  
25 //   21: [{ name: 'Alice', age: 21 }]  
26 // }
```



Ghép các mảng con bên trong các object sử dụng toán tử spread và initialValue [↗](#)

```
1 // friends - an array of objects  
2 // where object field "books" - list of favorite books  
3 var friends = [{  
4   name: 'Anna',  
5   books: ['The Hobbit', 'The Lord of the Rings', 'The Silmarillion']  
6 }];
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
11 }, {  
12   name: 'Alice',  
13   books: ['The Lord of the Rings', 'The Shining'],  
14   age: 18  
15 }];  
16  
17 // allbooks - list which will contain all friends' books +  
18 // additional list contained in initialValue  
19 var allbooks = friends.reduce(function(accumulator, currentValue) {  
20   return [...accumulator, ...currentValue.books];  
21 }, ['Alphabet']);  
22  
23 // allbooks = [  
24 //   'Alphabet', 'Bible', 'Harry Potter', 'War and peace',  
25 //   'Romeo and Juliet', 'The Lord of the Rings',  
26 //   'The Shining'  
27 // ]
```



Xóa các phần tử bị trùng trong mảng 

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
7     return accumulator;  
8 }, []);  
9 console.log(result); //[1,2,3,4,5]
```



Chạy các Promise theo trình tự

```
1  /**  
2   * Runs promises from array of functions that can return promises  
3   * in chained manner  
4   *  
5   * @param {array} arr - promise arr  
6   * @return {Object} promise object  
7   */  
8  function runPromiseInSequence(arr, input) {  
9      return arr.reduce(  
10         (promiseChain, currentFunction) => promiseChain.then(currentFunction),  
11         Promise.resolve(input)  
12     );  
13 }  
14
```

Khảo sát MDN

Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
21
22 // promise function 2
23 function p2(a) {
24   return new Promise((resolve, reject) => {
25     resolve(a * 2);
26   });
27 }
28
29 // function 3 - will be wrapped in a resolved promise by .then()
30 function f3(a) {
31   return a * 3;
32 }
33
34 // promise function 4
35 function p4(a) {
36   return new Promise((resolve, reject) => {
37     resolve(a * 4);
38   });
39 }
40
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
1 // Building-blocks to use for composition
2 const double = x => x + x;
3 const triple = x => 3 * x;
4 const quadruple = x => 4 * x;
5
6 // Function composition enabling pipe functionality
7 const pipe = (...functions) => input => functions.reduce(
8   (acc, fn) => fn(acc),
9   input
10 );
11
12 // Composed functions for multiplication of specific values
13 const multiply6 = pipe(double, triple);
14 const multiply9 = pipe(triple, triple);
15 const multiply16 = pipe(quadruple, quadruple);
16 const multiply24 = pipe(double, triple, quadruple);
17
18 // Usage
19 multiply6(6); // 36
20 multiply9(9); // 81
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
1  if (!Array.prototype.mapUsingReduce) {
2      Array.prototype.mapUsingReduce = function(callback, thisArg) {
3          return this.reduce(function(mappedArray, currentValue, index, array) {
4              mappedArray[index] = callback.call(thisArg, currentValue, index, array);
5              return mappedArray;
6          }, []);
7      };
8  }
9
10 [1, 2, , 3].mapUsingReduce(
11     (currentValue, index, array) => currentValue + index + array.length
12 ); // [5, 7, , 10]
```

Polyfill

```
1  // Production steps of ECMA-262, Edition 5, 15.4.4.21
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát


```
8      throw new TypeError( Array.prototype.reduce +
9        'called on null or undefined' );
10    }
11    if (typeof callback !== 'function') {
12      throw new TypeError( callback +
13        ' is not a function' );
14    }
15
16    // 1. Let O be ? ToObject(this value).
17    var o = Object(this);
18
19    // 2. Let len be ? ToLength(? Get(0, "length")).
20    var len = o.length >>> 0;
21
22    // Steps 3, 4, 5, 6, 7
23    var k = 0;
24    var value;
25
26    if (arguments.length >= 2) {
27      value = arguments[1];
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
35     if (k >= len) {  
36         throw new TypeError( 'Reduce of empty array ' +  
37             'with no initial value' );  
38     }  
39     value = o[k++];  
40 }  
41  
42 // 8. Repeat, while k < len  
43 while (k < len) {  
44     // a. Let Pk be ! ToString(k).  
45     // b. Let kPresent be ? HasProperty(O, Pk).  
46     // c. If kPresent is true, then  
47     //     i. Let kValue be ? Get(O, Pk).  
48     //     ii. Let accumulator be ? Call(  
49     //         callbackfn, undefined,  
50     //         « accumulator, kValue, k, O »).  
51     if (k in o) {  
52         value = callback(value, o[k], k, o);  
53     }  
54  
55     // d. Let next = ? ToString( ? Next(O) ).  
56     // e. If next is not NaN, go to step 8.
```

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

```
62 |     });
63 | }
```

Nếu bạn thực sự cần chức năng này trên những engine JavaScript không hỗ trợ `Object.defineProperty()`, bạn không nên thêm polyfill này vào `Array.prototype` bởi vì không có cách nào làm cho nó *không-duyệt-qua* (non-enumerable) được (property mới sẽ xuất hiện trong các vòng lặp for).

Đặc tả

Đặc tả	Trạng thái	Ghi chú
ECMAScript 5.1 (ECMA-262) The definition of 'Array.prototype.reduce()' in that specification.	<div>ST</div> Standard	Định nghĩa lần đầu. Hiện thực trong JavaScript 1.8.
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'Array.prototype.reduce()' in that	<div>ST</div>	

Khảo sát MDN 

Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát

Tương thích trình duyệt

[Update compatibility data on GitHub](#)

reduce

Chrome	Yes
Edge	12
Firefox	3
IE	9
Opera	10.5
Safari	4
WebView Android	Yes
Chrome Android	Yes
Edge Mobile	Yes
Firefox Android	4
Opera Android	Yes

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát



Xem thêm

- `Array.prototype.reduceRight()`

Khảo sát MDN



Hãy giúp chúng tôi hiểu 10 nhu cầu hàng đầu của các nhà phát triển Web và nhà thiết kế.

Tham gia khảo sát