

Array



- Array.isArray()
- valueOf() toString()
- push() pop()
- shift() unshift()
- join()
- concat()
- reverse()
- slice()
- splice()
- sort()
- map()
- forEach()
- filter()
- some() every()
- reduce() reduceRight()
- indexOf() lastIndexOf()
- Array.isArray()
- Array.isArray()



```
var arr = new Array(2);
arr.length // 2
arr // [ empty x 2 ]
```


 Array()
 
 2
 


 new
 

```
var arr = Array(2);
// 
var arr = new Array(2);
```


 new
 

Array()
 

```
// 
new Array() // []

// 
new Array(1) // [ empty ]
new Array(2) // [ empty x 2 ]

// 
new Array(3.2) // RangeError: Invalid array length
new Array(-3) // RangeError: Invalid array length

// 
new Array('abc') // ['abc']
new Array([1]) // [Array[1]]

// 
new Array(1, 2) // [1, 2]
new Array('a', 'b', 'c') // ['a', 'b', 'c']
```


 Array()
 

```
// bad
var arr = new Array(1, 2);
```

```
// good
var arr = [1, 2];
```

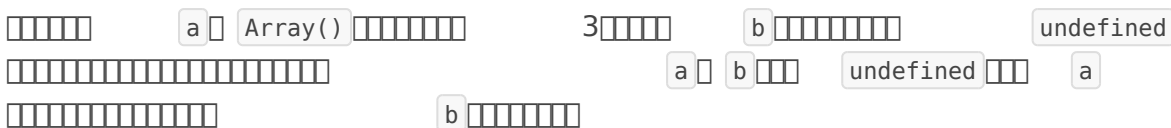


```
var a = new Array(3);
var b = [undefined, undefined, undefined];

a.length // 3
b.length // 3

a[0] // undefined
b[0] // undefined

0 in a // false
0 in b // true
```



Array.isArray()

Array.isArray  typeof 

```
var arr = [1, 2, 3];

typeof arr // "object"
Array.isArray(arr) // true
```

 typeof  Object  Array.isArray 



valueOf() → toString()

`valueOf`

valueOf 

value0f

```
var arr = [1, 2, 3];  
arr.valueOf() // [1, 2, 3]
```

`toString`

toString

```
var arr = [1, 2, 3];  
arr.toString() // "1,2,3"
```

```
var arr = [1, 2, 3, [4, 5, 6]];
arr.toString() // "1,2,3,4,5,6"
```

push() □ pop()

push

[illegible]

```
var arr = [];  
  
arr.push(1) // 1  
arr.push('a') // 2  
arr.push(true, {}) // 4  
arr // [1, 'a', true, {}]
```

--	--	--	--	--	--

push

[illegible]

```
var arr = ['a', 'b', 'c'];

arr.pop() // 'c'
arr // ['a', 'b']
```

--	--	--	--	--	--

pop

undefined

```
[].pop() // undefined
```

push() pop() “” stack()

```
var arr = [];  
arr.push(1, 2);  
arr.push(3);  
arr.pop();  
arr // [1, 2]
```

3

shift() unshift()

shift()

```
var a = ['a', 'b', 'c'];  
  
a.shift() // 'a'  
a // ['b', 'c']
```

shift()

shift()

```
var list = [1, 2, 3, 4];  
var item;  
  
while (item = list.shift()) {  
  console.log(item);  
}  
  
list // []
```

list.shift()

0

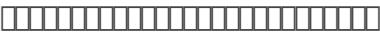
false

push() shift() “” queue()

unshift()

```
var a = ['a', 'b', 'c'];
```

```
a.unshift('x'); // 4
a // ['x', 'a', 'b', 'c']
```

unshift() 

```
var arr = [ 'c', 'd' ];
arr.unshift('a', 'b') // 4
arr // [ 'a', 'b', 'c', 'd' ]
```

join()

join()



```
var a = [1, 2, 3, 4];

a.join(' ') // '1 2 3 4'
a.join(' | ') // "1 | 2 | 3 | 4"
a.join() // "1,2,3,4"
```



undefined  null 

```
[undefined, null].join('#')
// '#'

['a',, 'b'].join('-')
// 'a--b'
```

 call 

```
Array.prototype.join.call('hello', '-')
// "h-e-l-l-o"

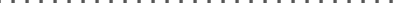
var obj = { 0: 'a', 1: 'b', length: 2 };
Array.prototype.join.call(obj, '-')
// 'a-b'
```

concat()

[illegible]

```
// [2, {a: 1}]
```

```
newArray[0].a // 2
```



reverse

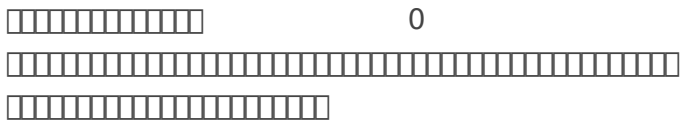
```
var a = ['a', 'b', 'c'];

a.reverse() // ["c", "b", "a"]
a // ["c", "b", "a"]
```

slice()

slice() 

```
arr.slice(start, end);
```



```
var a = ['a', 'b', 'c'];

a.slice(0) // ["a", "b", "c"]
a.slice(1) // ["b", "c"]
a.slice(1, 2) // ["b"]
a.slice(2, 6) // ["c"]
a.slice() // ["a", "b", "c"]
```



```
var a = ['a', 'b', 'c'];
a.slice(-2) // ["b", "c"]
a.slice(-2, -1) // ["b"]
```



```
var a = ['a', 'b', 'c'];
a.slice(4) // []
a.slice(2, 1) // []
```

slice() 

```
Array.prototype.slice.call({ 0: 'a', 1: 'b', length: 2 })  
// ['a', 'b']
```

```
Array.prototype.slice.call(document.querySelectorAll("div"));  
Array.prototype.slice.call(arguments);
```

|||||
|||||

call |||||

slice()

splice()

splice()

|||||
|||||

```
arr.splice(start, count, addElement1, addElement2, ...);
```

splice |||||

0

|||||
|

```
var a = ['a', 'b', 'c', 'd', 'e', 'f'];  
a.splice(4, 2) // ["e", "f"]  
a // ["a", "b", "c", "d"]
```

||||| 4 |||||

```
var a = ['a', 'b', 'c', 'd', 'e', 'f'];  
a.splice(4, 2, 1, 2) // ["e", "f"]  
a // ["a", "b", "c", "d", 1, 2]
```

|||||

|||||

```
var a = ['a', 'b', 'c', 'd', 'e', 'f'];  
a.splice(-4, 2) // ["c", "d"]
```

|||||

c |||||

|||||

splice |||||

0 |

```
var a = [1, 1, 1];  
  
a.splice(1, 0, 2) // []  
a // [1, 2, 1, 1]
```

[illegible]

```
var a = [1, 2, 3, 4];  
a.splice(2) // [3, 4]  
a // [1, 2]
```

sort()

sort

```
['d', 'c', 'b', 'a'].sort()
// ['a', 'b', 'c', 'd']

[4, 3, 2, 1].sort()
// [1, 2, 3, 4]

[11, 101].sort()
// [101, 11]

[10111, 1101, 111].sort()
// [10111, 1101, 111]
```

```
[4, 3, 2, 1].sort()
// [1, 2, 3, 4]

[11, 101].sort()
// [101, 11]

[10111, 1101, 111].sort()
// [10111, 1101, 111]
```

```
[11, 101].sort()
// [101, 11]

[10111, 1101, 111].sort()
// [10111, 1101, 111]
```

```
[10111, 1101, 111].sort()

// [10111, 1101, 111]
```

Diagram illustrating a sorting process. A top row of 15 empty boxes is followed by a `sort()` button. Below it is a long row of 25 empty boxes. At the bottom, two groups are shown: the first has two boxes with `101` in between, and the second has two boxes with `11` in between.

[illegible]

```
[10111, 1101, 111].sort(function (a, b) {  
    return a - b;  
})  
// [111, 1101, 10111]
```

[illegible][illegible]

numbers

map ()

```
[1, 2, 3].map(function(elem, index, arr) {
  return elem * index;
});
// [0, 2, 6]
```

□□□□□ [1, 2, 3] □□

this

--	--	--	--	--	--

```
var arr = ['a', 'b', 'c'];

[1, 2].map(function (e) {
  return this[e];
}, arr)

// ['b', 'c']
```

arr 

map()

```
var f = function (n) { return 'a' };

[1, undefined, 2].map(f) // ["a", "a", "a"]
[1, null, 2].map(f) // ["a", "a", "a"]
[1, , 2].map(f) // ["a", , "a"]
```

null 

forEach()

`forEach()`


```
function log(element, index, array) {
  console.log('[' + index + '] = ' + element);
}
```

```
[2, 5, 9].forEach(log);
// [0] = 2
// [1] = 5
// [2] = 9
```

```

[ ]
[ ]
[ ]
[ ]

```

```
forEach() [ ]
```

```
map()
```

```

[ ]
[ ]

```

```
forEach() [ ]
```

```
this [ ]
```

```
var out = [];

[1, 2, 3].forEach(function(elem) {
  this.push(elem * elem);
}, out);

out // [1, 4, 9]
```

```

[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]
[ ]

```

```
out [ ] forEach() [ ]
```

```
this [ ]
```

```
out [ ]
```

```

[ ]
[ ]

```

```
forEach()
```

```
[ ]
```

```
for [ ]
```

```
var arr = [1, 2, 3];

for (var i = 0; i < arr.length; i++) {
  if (arr[i] === 2) break;
  console.log(arr[i]);
}

// 1
```

```
[ ]
```

```
forEach() [ ]
```

```
forEach() [ ]
```

```
var log = function (n) {
  console.log(n + 1);
};
```


[illegible]

this

--	--	--	--	--	--	--	--	--

this

--	--	--	--	--	--

obj

--	--	--	--	--	--	--

some() $\square$ every()

[illegible]false ☐

true

false

```

    return elem >= 3;
  });
  // false

```

|||||
arr
|||||
3
|||||
false

|||||
some
||||
false
every
||||
true
|||||

```

function isEven(x) { return x % 2 === 0 }

[].some(isEven) // false
[].every(isEven) // true

```

some
every
|||||
this
||||

reduce() reduceRight()

reduce()
reduceRight()
reduce()

reduceRight()

```

[1, 2, 3, 4, 5].reduce(function (a, b) {
  console.log(a, b);
  return a + b;
})
// 1 2
// 3 3
// 6 4
// 10 5
// |||||15

```

reduce()
reduce()
n
n - 1

- a
1
b
2
- a
3
b
3
- a
6
b
4
- a
10
b
5

|||||
15

reduce()
reduceRight()

1.
2.
3.
4.

1

```
[1, 2, 3, 4, 5].reduce(function (  
  a,    //   
  b,    //   
  i,    //   
  arr //   
) {  
  // ... ..
```

reduce()

reduceRight()

```
[1, 2, 3, 4, 5].reduce(function (a, b) {  
  return a + b;  
}, 10);  
// 25
```

a

10

10

25

b

5

reduce()

```
function add(prev, cur) {  
  return prev + cur;  
}  
  
[].reduce(add)  
// TypeError: Reduce of empty array with no initial value  
[].reduce(add, 1)  
// 1
```

reduce()

reduceRight()

```
function subtract(prev, cur) {
  return prev - cur;
}

[3, 2, 1].reduce(subtract) // 0
[3, 2, 1].reduceRight(subtract) // -4
```

|||||
reduce()
|||||
3
2
1
reduceRight
|||||
1
2
3

|||||

||

```
function findLongest(entries) {
  return entries.reduce(function (longest, entry) {
    return entry.length > longest.length ? entry : longest;
  }, '');
}

findLongest(['aaa', 'bb', 'c']) // "aaa"
```

|||||
reduce()

 |||||

 |||||

indexOf() lastIndexOf()

indexOf
|||||
-1

```
var a = ['a', 'b', 'c'];

a.indexOf('b') // 1
a.indexOf('y') // -1
```

indexOf
|||||

```
['a', 'b', 'c'].indexOf('a', 1) // -1
```

|||||
1
a
-1

lastIndexOf
|||||
-1

```
var a = [2, 5, 9, 2];
a.lastIndexOf(2) // 3
a.lastIndexOf(7) // -1
```

□□□□□□□□□□

NaN □□□□□□□□□□□□□□□□

NaN □

```
[NaN].indexOf(NaN) // -1
[NaN].lastIndexOf(NaN) // -1
```

□□□□□□□□□□□□□□□□

=== □□□□□□

NaN □□□□□□□□□□

□□□□

□□□□□□□□□□□□□□□□□□□□□□

```
var users = [
  {name: 'tom', email: 'tom@example.com'},
  {name: 'peter', email: 'peter@example.com'}
];

users
.map(function (user) {
  return user.email;
})
.filter(function (email) {
  return /^t/.test(email);
})
.forEach(function (email) {
  console.log(email);
});
// "tom@example.com"
```

□□□□□□□□□□

Email □□□□□□□□□□□□□□□□

t □□□

Email

□□□□□□□□□□

□□□□

- Nicolas Bevacqua, [Fun with JavaScript Native Array Functions](#)

□□ #2

□ □□ □□ 2018-01-30 22:08:49 UTC

□ □□ □□ 2021-07-21 12:41:30 UTC