Метод **fill()** заполняет все элементы массива от начального до конечного индексов одним значением.

const array1 = [1, 2, 3, 4];

// fill with 0 from position 2 until position 4

console.log(array1.fill(0, 2, 4));

// expected output: [1, 2, 0, 0]

// fill with 5 from position 1

console.log(array1.fill(5, 1));

// expected output: [1, 5, 5, 5]

console.log(array1.fill(6));

// expected output: [6, 6, 6, 6]

Метод **filter()** создаёт новый массив со всеми элементами, прошедшими проверку, задаваемую в передаваемой функции.

const words = ['spray', 'limit', 'elite', 'exuberant', 'destruction', 'present'];

const result = words.filter(word => word.length > 6);

console.log(result);

// expected output: Array ["exuberant", "destruction", "present"]

Метод **forEach()** выполняет указанную функцию один раз для каждого элемента в массиве.

const array1 = ['a', 'b', 'c'];

array1.forEach(element => console.log(element));

// expected output: "a"

// expected output: "b"

// expected output: "c"

Метод **includes()** определяет, содержит ли массив определённый элемент, возвращая в зависимости от этого true или false.

const array1 = [1, 2, 3];

console.log(array1.includes(2));

// expected output: true

const pets = ['cat', 'dog', 'bat'];

console.log(pets.includes('cat'));

// expected output: true

console.log(pets.includes('at'));

// expected output: false

Метод **indexOf()** возвращает первый индекс, по которому данный элемент может быть найден в массиве или -1, если такого индекса нет.

Метод indexOf() сравнивает искомый элемент searchElement с элементами в массиве, используя [строгое сравнение (en-US)](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators#using_the_equality_operators) (тот же метод используется оператором ===, тройное равно).

**searchElement**

Искомый элемент в массиве.

**fromIndex**

Индекс, с которого начинать поиск. Если индекс больше или равен длине массива, возвращается -1, что означает, что массив даже не просматривается. Если индекс является отрицательным числом, он трактуется как смещение с конца массива. Обратите внимание: если индекс отрицателен, массив всё равно просматривается от начала к концу. Если рассчитанный индекс оказывается меньше 0, поиск ведётся по всему массиву. Значение по умолчанию равно 0, что означает, что просматривается весь массив.

В следующем примере indexOf() используется для поиска значений в массиве.

var array = [2, 5, 9];

array.indexOf(2); // 0

array.indexOf(7); // -1

array.indexOf(9, 2); // 2

array.indexOf(2, -1); // -1

array.indexOf(2, -3); // 0

Метод **join()** объединяет все элементы массива (или [массивоподобного объекта](https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Indexed_collections#working_with_array-like_objects)) в строку. Преобразует все элементы массива в строки и объединяет их в одну большую строку. Элемент массива с типом undefined или null преобразуется в пустую строку.

const elements = ['Fire', 'Air', 'Water'];

console.log(elements.join());

// expected output: "Fire,Air,Water"

console.log(elements.join(''));

// expected output: "FireAirWater"

console.log(elements.join('-'));

// expected output: "Fire-Air-Water"

Метод **keys()** возвращает новый **итератор массива** **Array Iterator**, содержащий ключи каждого индекса в массиве.

Метод **map()** создаёт новый массив с результатом вызова указанной функции для каждого элемента массива. Метод map вызывает переданную функцию callback один раз для каждого элемента, в порядке их появления и конструирует новый массив из результатов её вызова.

Метод **pop()** удаляет **последний** элемент из массива и возвращает его значение. Если вы вызовете pop() на пустом массиве, он вернёт значение [undefined](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/undefined).

var myFish = ['ангел', 'клоун', 'мандарин', 'хирург'];

console.log(myFish); // ['ангел', 'клоун', 'мандарин', 'хирург']

var popped = myFish.pop();

console.log(myFish); // ['ангел', 'клоун', 'мандарин']

console.log(popped); // 'хирург'

Метод **push()** добавляет один или более элементов в конец массива и возвращает новую длину массива. Метод push не является привязанным к типу; этот метод может быть [вызван](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/call) или [применён](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/apply) к объектам, напоминающим массив. Метод опирается на свойство length для определения места вставки значений. Если свойство length не может быть преобразовано в число, будет использовать индекс 0. Сюда входит случай несуществования свойства length, в этом случае оно также будет создано.

Единственными родными массивоподобными объектами являются [строки](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/String), хотя к ним он не может быть применён, поскольку строки являются неизменяемыми.

[Пример: слияние двух массивов](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/push#example_merging_two_arrays)

В этом примере используется функция [apply()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/apply) для помещения всех элементов из второго массива в первый.

var vegetables = ['пастернак', 'картошка'];

var moreVegs = ['сельдерей', 'свёкла'];

// Сливает второй массив с первым

// Эквивалентно вызову vegetables.push('сельдерей', 'свёкла');

Array.prototype.push.apply(vegetables, moreVegs);

console.log(vegetables); // ['пастернак', 'картошка', 'сельдерей', 'свёкла']

### [Пример: добавление элементов в массив](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/push#example_adding_elements_to_an_array)

Следующий код создаёт массив sports, содержащий два элемента, а затем добавляет к нему ещё два элемента. Переменная total будет содержать новую длину массива.

var sports = ['футбол', 'бейсбол'];

var total = sports.push('американский футбол', 'плавание');

console.log(sports); // ['футбол', 'бейсбол', 'американский футбол', 'плавание']

console.log(total); // 4

Метод **reduce()** применяет функцию **reducer** к каждому элементу массива (слева-направо), возвращая одно результирующее значение.

### [Суммирование всех значений в массиве](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce#example_sum_up_all_values_within_an_array)

var total = [0, 1, 2, 3].reduce(function(a, b) {

return a + b;

});

// total == 6

### [Суммирование значений в массиве объектов](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce#%D1%81%D1%83%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_%D0%B7%D0%BD%D0%B0%D1%87%D0%B5%D0%BD%D0%B8%D0%B9_%D0%B2_%D0%BC%D0%B0%D1%81%D1%81%D0%B8%D0%B2%D0%B5_%D0%BE%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BE%D0%B2)

Чтобы суммировать значения, содержащиеся в массиве объектов, вы **должны** указать initialValue, чтобы каждый элемент смог пройти через callback.

var initialValue = 0;

var sum = [{x: 1}, {x:2}, {x:3}].reduce(function (accumulator, currentValue) {

return accumulator + currentValue.x;

}, initialValue)

// sum == 6

Тоже самое, но со стрелочной функцией:

var initialValue = 0;

var sum = [{x: 1}, {x:2}, {x:3}].reduce(

(accumulator, currentValue) => accumulator + currentValue.x,

initialValue

);

// sum == 6

Copy to Clipboard

[Разворачивание массива массивов](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/Reduce#example_flatten_an_array_of_arrays)

var flattened = [[0, 1], [2, 3], [4, 5]].reduce(function(a, b) {

return a.concat(b);

});

// flattened равен [0, 1, 2, 3, 4, 5]

Метод **reverse()** на месте обращает порядок следования элементов массива. Первый элемент массива становится последним, а последний — первым.

const array1 = ['one', 'two', 'three'];

console.log('array1:', array1);

// expected output: "array1:" Array ["one", "two", "three"]

const reversed = array1.reverse();

console.log('reversed:', reversed);

// expected output: "reversed:" Array ["three", "two", "one"]

// Careful: reverse is destructive -- it changes the original array.

console.log('array1:', array1);

// expected output: "array1:" Array ["three", "two", "one"]

Метод **shift()** удаляет **первый** элемент из массива и возвращает его значение. Этот метод изменяет длину массива.

var myFish = ['ангел', 'клоун', 'мандарин', 'хирург'];

console.log('myFish до: ' + myFish);

//myFish до: ангел,клоун,мандарин,хирург

var shifted = myFish.shift();

console.log('myFish после: ' + myFish);

//myFish после: клоун,мандарин,хирург

console.log('Удалён этот элемент: ' + shifted);

//Удалён этот элемент: ангел

Метод **unshift()** добавляет один или более элементов в начало массива и возвращает новую длину массива. Метод unshift вставляет переданные значения в начало массивоподобного объекта.

Метод unshift не является привязанным к типу; этот метод может быть [вызван](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/call) или [применён](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Function/apply) к объектам, напоминающим массив. Объекты, не содержащие свойство length, отражающее последний элемент в серии последовательных числовых, начинающихся с нуля, свойств, могут повести себя неправильным образом.

var arr = [1, 2];

arr.unshift(0); // результат вызова равен 3, новой длине массива

// arr равен [0, 1, 2]

arr.unshift(-2, -1); // = 5

// arr равен [-2, -1, 0, 1, 2]

arr.unshift([-3]);

// arr равен[[-3], -2, -1, 0, 1, 2]

Метод **sort()** на месте сортирует элементы массива и возвращает отсортированный массив. Сортировка не обязательно [устойчива](https://ru.wikipedia.org/wiki/%D0%A3%D1%81%D1%82%D0%BE%D0%B9%D1%87%D0%B8%D0%B2%D0%B0%D1%8F_%D1%81%D0%BE%D1%80%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%BA%D0%B0) ([англ.](https://en.wikipedia.org/wiki/Sorting_algorithm#Stability)). Порядок сортировки по умолчанию соответствует порядку кодовых точек Unicode. Отсортированный массив. Важно, что копия массива не создаётся - массив сортируется [*на месте*](https://en.wikipedia.org/wiki/In-place_algorithm).

Для числового сравнения, вместо строкового, функция сравнения может просто вычитать b из a. Следующая функция будет сортировать массив по возрастанию:

function compareNumbers(a, b) {

return a - b;

}

Метод sort можно удобно использовать с [функциональными выражениями](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/function) (и [замыканиями](https://developer.mozilla.org/ru/docs/Web/JavaScript/Closures)):

var numbers = [4, 2, 5, 1, 3];

numbers.sort(function(a, b) {

return a - b;

});

console.log(numbers); // [1, 2, 3, 4, 5]

### [Пример: сортировка c помощью map](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/sort#example_sorting_maps)

Функция сравнения (compareFunction) может вызываться несколько раз для каждого элемента в массиве. В зависимости от природы функции сравнения, это может привести к высоким расходам ресурсов. Чем более сложна функция сравнения и чем больше элементов требуется отсортировать, тем разумнее использовать [map](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array/map) для сортировки. Идея состоит в том, чтобы обойти массив один раз, чтобы извлечь фактические значения, используемые для сортировки, во временный массив, отсортировать временный массив, а затем обойти временный массив для получения правильного порядка.

// массив для сортировки

var list = ['Дельта', 'альфа', 'ЧАРЛИ', 'браво'];

// временный массив содержит объекты с позицией и значением сортировки

var mapped = list.map(function(el, i) {

return { index: i, value: el.toLowerCase() };

});

// сортируем массив, содержащий уменьшенные значения

mapped.sort(function(a, b) {

if (a.value > b.value) {

return 1; }

if (a.value < b.value) {

return -1; }

return 0;

});

// контейнер для результа

var result = mapped.map(function(el) {

return list[el.index];

});

Метод **toString()** возвращает строковое представление указанного массива и его элементов. Объект [Array](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array) переопределяет метод toString объекта [Object](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Object). Для объектов класса Array, метод toString соединяет массив и возвращает одну строку, содержащую каждый элемент массива, разделённый запятыми. Например, следующий код создаёт массив и использует метод toString для преобразования массива в строку.

var monthNames = ['Янв', 'Фев', 'Мар', 'Апр'];

var myVar = monthNames.toString(); // присваивает 'Янв,Фев,Мар,Апр' переменной myVar.

Copy to Clipboard

JavaScript вызывает метод toString автоматически, когда массив представляется текстовым значением или когда массив находится в контексте конкатенации строк.