**Шпаргалка по методам массива:**

* ***Для добавления/удаления элементов:***
  + push (...items) – добавляет элементы в конец,
  + pop() – извлекает элемент с конца,
  + shift() – извлекает элемент с начала,
  + unshift(...items) – добавляет элементы в начало.
  + splice(pos, deleteCount, ...items) – начиная с индекса pos, удаляет deleteCount элементов и вставляет items.
  + slice(start, end) – создаёт новый массив, копируя в него элементы с позиции start до end (не включая end).
  + concat(...items) – возвращает новый массив: копирует все члены текущего массива и добавляет к нему items. Если какой-то из items является массивом, тогда берутся его элементы.
* ***Для поиска среди элементов:***
  + indexOf/lastIndexOf(item, pos) – ищет item, начиная с позиции pos, и возвращает его индекс или -1, если ничего не найдено.
  + includes(value) – возвращает true, если в массиве имеется элемент value, в противном случае false.
  + find/filter(func) – фильтрует элементы через функцию и отдаёт первое/все значения, при прохождении которых через функцию возвращается true.
  + findIndex похож на find, но возвращает индекс вместо значения.
* ***Для перебора элементов:***
  + forEach(func) – вызывает func для каждого элемента. Ничего не возвращает.
* ***Для преобразования массива:***
  + map(func) – создаёт новый массив из результатов вызова func для каждого элемента.
  + sort(func) – сортирует массив «на месте», а потом возвращает его.
  + reverse() – «на месте» меняет порядок следования элементов на противоположный и возвращает изменённый массив.
  + split/join – преобразует строку в массив и обратно.
  + reduce(func, initial) – вычисляет одно значение на основе всего массива, вызывая func для каждого элемента и передавая промежуточный результат между вызовами.
* Дополнительно:
  + Array.isArray(arr) проверяет, является ли arr массивом.

Обратите внимание, что методы sort, reverse и splice изменяют исходный массив.

**Работа с массивами в JavaScript**

Массивы — чрезвычайно полезны для структурированного хранения и передачи большого количества данных. Однако для удобной работы с такой громоздкой конструкцией требуются специальные методы. Рассмотрим некоторые из них.

**[Объявление](https://learn.javascript.ru/array" \l "obyavlenie)**

Существует два варианта синтаксиса для создания пустого массива:

let arr = new Array();

let arr = [];

Практически всегда используется второй вариант синтаксиса. В скобках мы можем указать начальные значения элементов:

let fruits = ["Яблоко", "Апельсин", "Слива"];

Элементы массива нумеруются, начиная с нуля.

Мы можем получить элемент, указав его номер в квадратных скобках:

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits[0] ); // Яблоко

alert( fruits[1] ); // Апельсин

alert( fruits[2] ); // Слива

Мы можем заменить элемент:

fruits[2] = 'Груша'; // теперь ["Яблоко", "Апельсин", "Груша"]

…Или добавить новый к существующему массиву:

fruits[3] = 'Лимон'; // теперь ["Яблоко", "Апельсин", "Груша", "Лимон"]

**length**

Общее число элементов массива содержится в его свойстве length:

let fruits = ["Яблоко", "Апельсин", "Слива"];

alert( fruits.length ); // 3

Свойство length автоматически обновляется при изменении массива. Если быть точными, это не количество элементов массива, а наибольший цифровой индекс плюс один.

Например, единственный элемент, имеющий большой индекс, даёт большую длину:

let fruits = [];

fruits[123] = "Яблоко";

alert( fruits.length ); // 124 остальные undefind

Обратите внимание, что обычно мы не используем массивы таким образом.

Ещё один интересный факт о свойстве length – его можно перезаписать.

Если мы вручную увеличим его, ничего интересного не произойдёт. Зато, если мы уменьшим его, массив станет короче. Этот процесс необратим, как мы можем понять из примера:

let arr = [1, 2, 3, 4, 5];

arr.length = 2; // укорачиваем до 2 элементов

alert( arr ); // [1, 2]

arr.length = 5; // возвращаем length как было

alert( arr[3] ); // undefined: значения не восстановились

Таким образом, самый простой способ очистить массив – это arr.length = 0;.

В массиве могут храниться элементы любого типа.

Например:

// разные типы значений

let arr = [ 'Яблоко', { name: 'Джон' }, true, function() { alert('привет'); } ];

// получить элемент с индексом 1(объект) и затем показать его свойство

alert( arr[1].name ); // Джон

// получить элемент с индексом 3(функция) и выполнить её

arr[3](); // привет

**Методы, работающие с концом массива:**

**pop**

Удаляет последний элемент из массива и возвращает его:

let fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.pop() ); // удаляем "Груша" и выводим его

alert( fruits ); // Яблоко, Апельсин

**push**

Добавляет элемент в конец массива:

let fruits = ["Яблоко", "Апельсин"];

fruits.push("Груша");

alert( fruits ); // Яблоко, Апельсин, Груша

Вызов fruits.push(...) равнозначен fruits[fruits.length] = ....

**Методы, работающие с началом массива:**

**shift**

Удаляет из массива первый элемент и возвращает его:

let fruits = ["Яблоко", "Апельсин", "Груша"];

alert( fruits.shift() ); // удаляем Яблоко и выводим его

alert( fruits ); // Апельсин, Груша

**unshift**

Добавляет элемент в начало массива:

let fruits = ["Апельсин", "Груша"];

fruits.unshift('Яблоко');

alert( fruits ); // Яблоко, Апельсин, Груша

Методы push и unshift могут добавлять сразу несколько элементов:

let fruits = ["Яблоко"];

fruits.push("Апельсин", "Груша");

fruits.unshift("Ананас", "Лимон");

// ["Ананас", "Лимон", "Яблоко", "Апельсин", "Груша"]

alert( fruits );

## Удаляет все элементы массива

|  |  |
| --- | --- |
|  | delete arr[index] |

Или

Arr.length=0;

## Удалить deleteCountэлементов, начиная с номера index, а затем вставить elem1, ..., elemNна их место. Возвращает массив из удалённых элементов.

## arr.splice(index[, deleteCount, elem1, ..., elemN])

## создает новый массив, равный arr и добавляет указанные в скобках элементы

## arr.concat(value1, value2, … valueN)

## arr.concat(arr2)

## Где этот элемент?

## array.indexOf(element)

Возвращает позицию (индекс) элемента в массиве.  
Если в данном массиве такого элемента нет, вернет -1.  
Если в массиве несколько элементов с таким значением, вернет индекс первого найденного.  
Можно указать начальную точку для поиска, как в предыдущем методе. array.indexOf(element,1)

## Перебор

В массивах перебор элементов осуществляется обычным циклом или циклом for-of:

## for (var i=0; i<arr.length; i++){...}

## 

## for (let v of arr) { ... }

## Вызвать функцию для каждого элемента массива:

|  |  |
| --- | --- |
|  | arr.forEach(callback(item, i, arr)[, this]) |

в функцию передаются — элемент массива, его номер и сам массив.  
Вторым аргументом forEachможет принимать контекст вызова функции callback.

|  |  |
| --- | --- |
|  | arr.every(callback(item, i, arr)[, this]) |

возвращает true, если вызов callbackвернёт trueдля каждого элемента arr

|  |  |
| --- | --- |
|  | arr.some(callback(item, i, arr)[, this])  возвращает true, если вызов callbackвернёт trueхотя бы для одного элемента arr Из массива в строку и обратноlet string = 'some interesting text';let arr = string.split(' '); |

первый аргумент — разделитель (если пустая строка, разобьет по буквам)  
второй аргумент — ограничение на количество элементов в массиве

let arr = ['some', 'interesting', 'text'];

let string = arr.join(' ');

new Array(4).join("ля") // повторит разделитель ля 4 раза, массив будет пустым