**Метод length()**

Возвращает количество элементов в массиве. Длина - это встроенное свойство массива, которое возвращает количество элементов в массиве. Это числовое значение, представляющее длину или размер массива.

Синтаксис

arrayName.length();

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

console.log(myArray.length);

// Output: 5

**Метод reverse()**

Изменяет порядок элементов в массиве на обратный.reverse() - это встроенный метод, который изменяет порядок элементов в массиве на обратный. Этот метод работает непосредственно с массивом и не создает новый массив. Он изменяет исходный массив и возвращает ссылку на перевернутый массив.

Синтаксис

arrayName.reverse();

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

console.log(myArray);

myArray.reverse();

console.log(myArray);

// Output:

// [1, 2, 3, 4, 5]

// [5, 4, 3, 2, 1]

**Метод sort()**

Сортирует элементы массива по возрастанию или в указанном порядке. Sort() - это встроенный метод, который используется для сортировки элементов массива в порядке возрастания. Метод sort() сортирует элементы массива по месту и возвращает ссылку на отсортированный массив.

Синтаксис

arrayName.sort(compareFunction);

JavaScript

Пример

const fruits = ['banana', 'apple', 'orange', 'grape'];

console.log(fruits);

fruits.sort();

console.log(fruits);

// Output:

// ["banana", "apple", "orange", "grape"]

// ["apple", "banana", "grape", "orange"]

**Метод fill()**

Изменяет все элементы в массиве со статическим значением с начального индекса на конечный индекс.fill() - это встроенный метод, используемый для изменения значений всех элементов в массиве со статическим значением с начального индекса на конечный индекс. Этот метод изменяет исходный массив, а также возвращает ссылку на измененный массив.

Синтаксис

arrayName.fill(value, start, end);

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

console.log(myArray);

myArray.fill(0, 1, 4);

console.log(myArray);

// Output:

// [1, 2, 3, 4, 5]

// [1, 0 0, 0, 5]

**Метод from()**

Создает новый массив из массивоподобного или итерируемого объекта. Метод from() - это встроенный метод, который используется для создания нового массива из массивоподобного или итерируемого объекта. Массивоподобным или итерируемым объектом может быть любой объект со свойством length или объект, реализующий итерируемый протокол. Этот метод возвращает новый массив с элементами, скопированными из исходного массивоподобного или итерируемого объекта.

Синтаксис

Array.from(arrayLike, mapFn, thisArg);

JavaScript

Пример

const myArrayLikeObject = {length: 3, 0: 'apple', 1: 'banana', 2: 'orange'};

const myArray = Array.from(myArrayLikeObject);

console.log(myArray);

// Output: ["apple", "banana", "orange"]

**Метод join()**

Объединяет все элементы массива в строку, используя указанный разделитель. Метод join() - это встроенный метод, который используется для объединения всех элементов массива в строку с использованием указанного разделителя. Этот метод возвращает новую строку, которая создается в результате объединения всех элементов массива с указанным разделителем.

Синтаксис

arrayName.join(separator);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

const myString = myArray.join(', ');

console.log(myString);

// Output: "apple, banana, orange"

JavaScript

В этом примере мы создаем массив myArray с тремя элементами и используем метод join() для объединения всех элементов массива с указанным разделителем, метод join() возвращает новую строку myString, которая содержит элементы массива, объединенные с разделителем. Наконец, мы записываем новую строку myString в консоль. Метод join() также может принимать необязательный параметр-разделитель, который определяет символ или строку, используемые для разделения элементов массива в результирующей строке. Если разделитель не указан, в качестве разделителя по умолчанию используется запятая.

**Метод toString()**

Возвращает строку, представляющую указанный массив и его элементы. Метод toString() - это встроенный метод, который используется для возврата строки, представляющей указанный массив и его элементы. Метод преобразует каждый элемент массива в строку и объединяет их вместе с запятыми между ними. Он возвращает новую строку, которая представляет результирующую строку всех элементов массива.

Синтаксис

arrayName.toString();

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

const myString = myArray.toString();

console.log(myString);

// Output: "apple,banana,orange"

**Метод pop()**

Удаляет последний элемент из массива и возвращает этот элемент. Метод pop() - это встроенный метод, который используется для удаления последнего элемента из массива и возврата этого элемента. Этот метод изменяет исходный массив, удаляя последний элемент, и возвращает удаленный элемент.

Синтаксис

arrayName.pop();

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

const lastElement = myArray.pop();

console.log(myArray);

console.log(lastElement);

// Output:

// ["apple", "banana"]

// "orange"

**Метод forEach()**

Выполняет предоставленную функцию один раз для каждого элемента массива. Метод forEach() - это встроенный метод, который выполняет предоставленную функцию один раз для каждого элемента массива. Метод forEach() - это функция более высокого порядка, которая принимает функцию обратного вызова в качестве своего аргумента. Функция обратного вызова выполняется для каждого элемента массива, при этом текущий элемент, индекс элемента и весь массив передаются в качестве аргументов.

Синтаксис

arrayName.forEach(callback(currentValue, index, array));

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

myArray.forEach(function(element, index) {

console.log(`${element} is at index ${index}`);

});

// Output:

// "apple is at index 0"

// "banana is at index 1"

// "orange is at index 2"

**Метод shift()**

Удаляет первый элемент из массива и возвращает этот элемент, сдвигая все последующие элементы на более низкий индекс. Метод shift() - это встроенный метод, который используется для удаления первого элемента из массива и возврата этого элемента, сдвигая все последующие элементы на более низкий индекс. Этот метод изменяет исходный массив, удаляя первый элемент, и возвращает удаленный элемент.

Синтаксис

arrayName.shift();

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

const firstElement = myArray.shift();

console.log(myArray);

console.log(firstElement);

// Output:

// ["banana", "orange"]

// "apple"

**Метод** copyWithin()  **копирайтинга ()**

Копирует последовательность элементов внутри массива в другую позицию в том же массиве. Метод copyWithin() - это встроенный метод, который используется для копирования последовательности элементов внутри массива в другую позицию в том же исходном массиве путем копирования части массива в другое местоположение в том же массиве. copywithin(), метод принимает три аргумента.

targetIndex: Индекс, с которого следует начинать копирование элементов. Этот аргумент является обязательным.

startIndex: Индекс, с которого следует начинать копирование элементов. Этот аргумент является необязательным и по умолчанию равен 0, если не указан.

endIndex: Индекс, по которому следует прекратить копирование элементов. Этот аргумент является необязательным и по умолчанию равен длине массива, если он не указан.

Синтаксис

arrayName.copyWithin(target, start, end);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange', 'mango', 'watermelon'];

myArray.copyWithin(1, 3, 5);

console.log(myArray);

// Output:

// ["apple", "mango", "watermelon", "mango", "watermelon"]

**Метод push()**

Добавляет один или несколько элементов в конец массива и возвращает новую длину массива. Метод push() - это встроенный метод, используемый для добавления одного или нескольких элементов в конец массива и возвращающий новую длину массива. Этот метод изменяет исходный массив, добавляя указанные элементы, и возвращает новую длину результирующего массива.

Синтаксис

arrayName.push(element1, element2, ..., elementN);

JavaScript

Пример

const myArray = ['apple', 'banana'];

const newLength = myArray.push('orange', 'mango');

console.log(myArray);

console.log(newLength);

// Output:

// ["apple", "banana", "orange", "mango"]

// 4

**Метод unshift()**

Добавляет один или несколько элементов в начало массива и возвращает новую длину массива. метод unshift() - это встроенный метод, который используется для добавления одного или нескольких элементов в начало массива и возвращает новую длину массива. Этот метод изменяет исходный массив, добавляя указанные элементы в начале, и возвращает новую длину результирующего массива.

Синтаксис

arrayName.unshift(element1, element2, ..., elementN);

JavaScript

Пример

const myArray = ['apple', 'banana'];

const newLength = myArray.unshift('orange', 'mango');

console.log(myArray);

console.log(newLength);

// Output:

// ["orange", "mango", "apple", "banana"]

// 4

**Метод concat()**

Возвращает новый массив путем объединения двух или более массивов. Метод concat() - это встроенный метод, который используется для объединения двух или более массивов в новый массив. Метод concat() создает новый массив, который включает все элементы исходных массивов в том порядке, в котором они были указаны.

Синтаксис

arrayName.concat(array1, array2, ..., arrayN);

JavaScript

Пример

const arr1 = [1, 2, 3];

const arr2 = [4, 5, 6];

const arr3 = [7, 8, 9];

const mergedArr = arr1.concat(arr2, arr3);

console.log(mergedArr);

// Output:

// [1, 2, 3, 4, 5, 6, 7, 8, 9]

**Метод some()**

Проверяет, передает ли хотя бы один элемент в массиве заданное условие (функция обратного вызова). Метод some() - это встроенный метод, который используется для проверки, удовлетворяет ли хотя бы один элемент в массиве заданному условию (указанному в функции обратного вызова). Метод some() возвращает логическое значение (true или false) в зависимости от того, удовлетворяет ли какой-либо элемент в массиве условию, указанному в функции обратного вызова.

Синтаксис

arrayName.some(callback(currentValue, index, array));

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

const hasEven = myArray.some(function(element) {

return element % 2 === 0;});

console.log(hasEven);

// Output:

// true

**Способ соединения ()**

Изменяет содержимое массива путем удаления, замены или добавления элементов на место. Метод splice() - это встроенный метод, который используется для изменения содержимого массива путем удаления, замены или добавления элементов на место. Метод splice() может удалить один или несколько элементов из массива, заменить существующие элементы новыми элементами или добавить новые элементы в массив.

Метод splice(): принимает три основных аргумента-

startIndex: Индекс, с которого следует начать изменение массива. Этот аргумент является обязательным.

deleteCount: Количество элементов, которые необходимо удалить из массива. Если этот аргумент равен 0, никакие элементы не удаляются. Этот аргумент необязателен.

newElement1, newElement2, ...: Элементы для добавления в массив, начиная с startIndex. Эти аргументы необязательны.

Синтаксис

arrayName.splice(start, deleteCount, item1, item2, ...);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange'];

myArray.splice(1, 1, 'mango', 'pineapple');

console.log(myArray);

// Output:

// ["apple", "mango", "pineapple", "orange"]

**Метод flat()**

Создает новый массив со всеми элементами вложенного массива, рекурсивно объединенными до указанной глубины. Метод flat() - это встроенный метод, который используется для создания нового массива со всеми элементами вложенного массива, рекурсивно объединенными до указанной глубины. Метод flat() преобразует многомерный массив в одномерный массив путем объединения всех элементов подмассива. Метод flat() может принимать необязательный аргумент depth, который определяет максимальную глубину подмассивов, которые должны быть сглажены. Если аргумент глубины не указан, метод flat() выравнивает все уровни вложенных массивов.

Синтаксис

arrayName.flat(depth);

JavaScript

Пример

const nestedArray = [1, 2, [3, 4, [5, 6]]];

const flattenedArray = nestedArray.flat(2);

console.log(flattenedArray);

// Output:

// [1, 2, 3, 4, 5, 6]

**Метод lastIndexOf()**

Возвращает последний индекс, по которому указанный элемент найден в массиве. Метод lastIndexOf() - это встроенный метод, который используется для возврата последнего индекса (позиции), по которому указанный элемент найден в массиве. Метод lastIndexOf() выполняет поиск по массиву справа налево (от конца к началу) до тех пор, пока не найдет указанный элемент. Метод lastIndexOf() может принимать необязательный аргумент fromIndex, который указывает позицию, с которой следует начинать поиск в массиве справа налево. Если аргумент fromIndex не указан, метод lastIndexOf() выполняет поиск по всему массиву.

Синтаксис

arrayName.lastIndexOf(searchElement, fromIndex);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange', 'banana'];

const lastIndex = myArray.lastIndexOf('banana');

console.log(lastIndex);

// Output:

// 3

**Метод every()**

Проверяет, все ли элементы в массиве передают заданное условие (функция обратного вызова). метод every() - это встроенный метод, который используется для проверки того, все ли элементы в массиве соответствуют заданному условию (указанному в функции обратного вызова). Метод every() возвращает логическое значение (true или false) в зависимости от того, удовлетворяют ли все элементы в массиве условию, указанному в функции обратного вызова.

Синтаксис

array.every(callback(currentValue, index, array);

JavaScript

Пример

const myArray = [2, 4, 6, 8, 10];

const areEven = myArray.every(function(element) {

return element % 2 === 0;

});

console.log(areEven);

// Output:

// true

**Метод slice()**

Возвращает мелкую копию части массива в новый массив. метод slice() - это встроенный метод, который используется для возврата мелкой копии части массива в новый массив. Метод slice() создает новый массив, который включает указанный диапазон элементов из исходного массива без изменения исходного массива.

Метод slice() принимает два основных аргумента:

startIndex: Индекс, с которого начинается нарезка массива. Этот аргумент является необязательным и по умолчанию равен 0.

endIndex: Индекс, по которому завершается нарезка массива (элемент с этим индексом не включен в нарезанный массив). Этот аргумент необязателен и по умолчанию равен концу массива.

Синтаксис

array.slice(start, end);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange', 'mango', 'pineapple'];

const slicedArray = myArray.slice(1, 4);

console.log(slicedArray);

// Output:

// ["banana", "orange", "mango"]

**Метод find()**

Возвращает значение первого элемента в массиве, который удовлетворяет предоставленной функции тестирования. Метод find() - это встроенный метод, который используется для возврата значения первого элемента в массиве, удовлетворяющего предоставленной функции тестирования. Метод find() просматривает массив слева направо, пока не найдет элемент, удовлетворяющий тестирующей функции, и возвращает значение этого элемента.

Метод find() принимает один основной аргумент

функция тестирования: Функция, которая проверяет каждый элемент в массиве, чтобы увидеть, удовлетворяет ли он определенному условию. Эта функция должна возвращать логическое значение (true или false).

Синтаксис

array.find(callback(element, index, array));

JavaScript

Пример

const myArray = [2, 4, 6, 8, 10];

const foundElement = myArray.find(function(element) {

return element > 5;});

console.log(foundElement);

// Output:

// 6

**Метод includes() включает в себя**

Определяет, включает ли массив определенный элемент, возвращающий значение true или false. Метод includes() - это встроенный метод, который используется для определения, включает ли массив определенный элемент, возвращая логическое значение (true или false) в зависимости от того, найден ли элемент в массиве.

Метод includes() принимает два аргумента

элемент поиска: Элемент для поиска в массиве.

startIndex: Индекс, с которого начинается поиск в массиве (необязательно).

Синтаксис

array.includes(searchElement, fromIndex);

JavaScript

Пример

const myArray = ['apple', 'banana', 'orange', 'mango', 'pineapple'];

const hasPineapple = myArray.includes('pineapple');

console.log(hasPineapple);

// Output:

// true

**Метод reduce()**

Применяет функцию к накопителю и каждому элементу массива (слева направо), чтобы свести его к одному значению. Метод reduce() - это встроенный метод, который используется для применения функции к накопителю и каждому элементу массива слева направо, чтобы уменьшить его до одного значения. Метод reduce() может использоваться для суммирования значений в массиве, объединения строк или выполнения любой другой агрегации или вычисления в массиве.

Метод reduce() принимает два основных аргумента,

функция reducer: Функция, применяемая к аккумулятору и каждому элементу массива. Эта функция должна принимать два аргумента: аккумулятор и текущий элемент массива.

initialValue: Начальное значение накопителя (необязательно). Если этот аргумент не указан, метод запустится с первым элементом массива в качестве значения.

Синтаксис

array.reduce(callback(accumulator, currentValue, index, array), initialValue);

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

const sum = myArray.reduce(function(accumulator, currentValue) {

return accumulator + currentValue;});

console.log(sum);

// Output:

// 15

**32. метод filter()**

Создает новый массив со всеми элементами, которые передают заданное условие (функция обратного вызова). Метод filter() - это встроенный метод массива, который создает новый массив со всеми элементами, которые передают предоставленное условие (функция обратного вызова). Метод filter() выполняет итерацию по массиву и возвращает новый массив, содержащий только те элементы, которые удовлетворяют условию, указанному в функции обратного вызова.

Метод filter() принимает один основной аргумент,

функция обратного вызова: Функция, которая проверяет каждый элемент массива на соответствие заданному условию. Если элемент удовлетворяет условию, он включается в новый массив. Эта функция должна возвращать логическое значение (true или false).

Синтаксис

array.filter(callback(element, index, array));

JavaScript

Пример

const myArray = [1, 2, 3, 4, 5];

const result = myArray.filter(function(element) {

return element % 2 === 0;});

console.log(result);

// Output:

// [2, 4]

**34. метод map()**

Описание: Создает новый массив с результатами вызова предоставленной функции для каждого элемента массива. Метод map() - это встроенный метод, который используется для создания нового массива с результатами вызова предоставленной функции для каждого элемента массива. Метод map() применяет предоставленную функцию к каждому элементу массива и возвращает новый массив с результирующими значениями.

Метод map() принимает один основной аргумент,

функция обратного вызова: Функция для вызова каждого элемента в массиве. Эта функция должна возвращать измененное значение для каждого элемента.

Синтаксис

array.map(callback(element, index, array));

JavaScript

Пример

const myArray = [1, 2, 3];

const result = myArray.map(function(element) {

return element + 1;});

console.log(result);

// Output:

// [2, 3, 4]