Методы массивов

Добавление/удаление элементов

`arr.push(...items)`	Добавляет один или более элементов **в конец массива** и возвращает новую длину массива
`arr.pop()`	Удаляет **последний** элемент из массива и возвращает его значение
`arr.unshift(...items)`	Добавляет один или более элементов **в начало массива** и возвращает новую длину массива.
`arr.shift()`	Удаляет **первый** элемент из массива и возвращает его значение. Этот метод изменяет длину массива.

Добавление/удаление элементов

`arr.splice(index[, deleteCount, elem1, ..., elemN])`	Изменяет содержимое массива, **удаляя** существующие элементы и/или **добавляя** новые, **изменяя** значения на новые
`arr.slice([start], [end])`	Возвращает новый массив, содержащий **копию** части исходного массива.
`arr.concat(arg1, arg2...)`	Создает новый массив, соединяя два или больше массивов(и дополнительные значения) в один и возвращает результат

Задания (используя splice)

1. Создать функцию образовать массив случайных чисел
2. Создать функцию подсчета количества нулей.
3. Если есть нулевые элементы, удалить их
4. увеличить все четные элементы на 1
5. удалить встречающиеся числа первого массива из второго массива

С помощью функции образовать массив случайных чисел

```
function getArray(n) {
    let arr = [];
    for (let i = 0; i < n; i++)
        arr.push(Math.floor(Math.random() * 100));
    return arr;
}

let drr = getArray(10) ;
```

```
//Создать функцию подсчета количества нулей.


 function countNull(arr) {
    let k = 0;
    arr.forEach((elem) => elem == 0 ? k++ : k);
    return k;
}
let k0 = countNull(drr);
```

Поиск в массиве

`arr.indexOf(item, from)`	Ищет элемент `item`, начиная с индекса `from`, и **возвращает первый** индекс, на котором был найден искомый элемент, в противном случае `-1`.
`arr.includes(item, from)`	Ищет элемент `item`, начиная с индекса `from`, и **возвращает `true`**, если поиск успешен, то есть определяет, содержит ли массив определённый элемент. Выявляет NaN, undefined
`arr.find(function(item, index, array){}`	**Возвращает значение первого найденного в массиве элемента**, которое удовлетворяет заданному условию.
`arr.findIndex(function(item, index, array){}`	**Возвращает индекс в массиве**, если элемент удовлетворяет условию проверяющей функции. В противном случае возвращает -1.

Поиск в массиве

`arr.filter(function(item, index, array){}`	Возвращает массив из всех подходящих элементов по условию
`arr.every(function(item, index, array){}`	Проверяет, все ли элементы соответствуют определенному условию.В метод every() в качестве параметра передается функция, которая представляет условие. Эта функция в качестве параметра принимает элемент и возвращает true (если элемент соответствует условию) или false (если не соответствует). Если хотя бы один элемент не соответствует условию, то метод every() возвращает значение false.
`arr.some(function(item, index, array){}`	Проверяет, соответствует ли хотя бы один элемент условию. И в этом случае метод some() возвращает true. Если элементов, соответствующих условию, в массиве нет, то возвращается значение false

Преобразование массива

`arr.map(function(item, index, array) {}`	Он вызывает функцию для каждого элемента массива и **возвращает новый массив результатов** выполнения этой функции.
`arr.reduce(function(accumulator, item, index, array) { }, [initial]);`	Функция применяется по очереди ко всем элементам массива и «переносит» свой результат на следующий вызов. ● `accumulator` – результат предыдущего вызова этой функции, равен `initial` при первом вызове (если передан `initial`), ● `item` – очередной элемент массива, ● `index` – его индекс, ● `array` – сам массив.

//проверить наличие элементов

```
let disksSize = ['500Gb', '1Tb', '2Tb'];

let index = disksSize.indexOf('1Tb');  // 1

let result= disksSize.includes('1Tb');  // true

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

let q = [1, 2, 4, 7, , NaN, 0];

console.log(`q=`, q);

if (q.includes(NaN)) console.log("есть NaN"); //есть NaN

if (q.includes(undefined)) console.log("есть undefined"); //есть undefined
```

```
// получаем первый элемент, который больше 10
let numbers = [1, 2, 3, 5, 8, 13, 21, 34];
let found = num.find(function(elem, i){
      return elem>10;
});

//или

let found = numbers.find(elem => elem > 10 );
console.log(found); // 13
```

//вывести элементы по условию (>10)

```
let num = [1, 2, 3, 5, 8, 13, 21, 34];
let d = num.filter(function(elem, i){
        return elem>10;
});
```

или

```
 let d = num.filter(elem => elem>10); //13,21,34
```

//проверить, что все элементы положительные

```
let numbers = [ 1, -12, 8, -4, 25, 42 ];
let res= numbers.every(n => n > 0);
console.log(res); // false
```

//Если хотя бы один элемент не соответствует условию, то метод every() возвращает значение false.

——----------

//проверить, что есть положительные элементы

```
let numbers = [ 1, -12, 8, -4, 25, 42 ];
let res= numbers.some(n => n > 0);
console.log(res); // true
```

//Вычислить квадраты чисел массива, увеличить на 2

```
let numbers = [ 1, 2, 3, 4, 5, 6];
const squaresNum = numbers.map(n => n * n);
console.log(squaresNum);   //  [1, 4, 9, 16, 25, 36]
```

//Функция, которая передается в метод map() получает текущий перебираемый элемент, выполняет над ним операции и возвращает некоторое значение. Это значение затем попадает в результирующий массив squaresNum

Метод reduce

Метод reduce сворачивает массив к одному значению (редуцирует). К примеру, с помощью reduce можно легко найти сумму элементов массива (то есть массив сведется к одному значению - к сумме элементов).
Первым параметром метод reduce получает функцию, которая последовательно выполнится для каждого элемента массива, начиная с первого.
В эту функцию можно передавать 4 параметра. Если эти параметры есть (они не обязательны), то в первый автоматически попадет промежуточный результат, во второй попадет элемент массива, в третий - его номер в массиве (индекс), а в четвертый - сам массив.
Промежуточный результат - это переменная, в которую будет накапливаться то значение, которое вернет метод reduce, когда переберет все элементы массива. К примеру, туда последовательно можно накапливать сумму элементов массива: сначала положить первый элемент, при следующем проходе цикла уже сумму первого элемента и второго, при следующем проходе - сумму первого, второго и третьего. И так, пока массив не закончится.
Функция, которую принимает reduce, должна возвращать новое значение промежуточного результата.
Вторым параметром метода reduce указывается начальное значение промежуточного результата. Если его не указать, то оно будет равно первому элементу массива, а обработка элементов начнется со второго элемента

//Найдем сумму элементов массива:

```
let arr = [1, 2, 3, 4, 5, 6];
let result = arr.reduce(function(sum, elem) {
      return sum + elem;
}, 0);
console.log(result);

// или
let result = arr.reduce((sum, elem) => sum + elem, 0);
```

//Найдем сумму неотрицательных чисел массива

```
let arr = [1, -2, -3, 4, 5, -6];
let result = arr.reduce(function(sum, elem) {
        if (elem >= 0) {
                return sum + elem;
        } else {
                return sum;
        }
});
console.log(result);
```

Задание 1 - Создать массив из 10 случайных чисел и написать несколько функций для работы с ним.

1. Функция принимает массив и выводит его на экран.
2. Функция принимает массив и выводит только четные элементы.
3. Функция принимает массив и возвращает сумму всех элементов массива.
4. Функция принимает массив и возвращает его максимальный элемент.
5. Функция добавления нового элемента в массив по указанному индексу.
6. Функция удаления элемента из массива по указанному индексу.

Задание 2 - Создать еще один массив из 5 случайных чисел и написать следующие функции.

1. Функция принимает 2 массива и возвращает новый массив, в котором собраны все элементы из двух массивов без повторений.

2.Функция принимает 2 массива и возвращает новый массив, в котором собраны общие элементы (то есть элементы, которые встречаются и в первом и во втором массивах) без повторений.

3. Функция принимает 2 массива и возвращает новый массив, в котором собраны все элементы из первого массива, которых нет во втором массиве.