**Методы работы с массивами**

**Метод concat**

Метод **concat** сливает указанные массивы в один общий массив. Метод применяется к одному из массивов, а в параметрах метода передаются остальные массивы для слияния.

При этом метод не изменяет исходный массив, а возвращает новый.

**Синтаксис**

массив.concat(еще массив, и еще массив, и еще...)

**Пример 1**

Давайте сольем 3 массива в один с помощью метода **concat**:

let arr1 = [1, 2];

let arr2 = [3, 4];

let arr3 = [5, 6];

let result = arr1.concat(arr2, arr3);

console.log(result);

Результат выполнения кода:

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

**Пример 2**

А теперь давайте сольем два массива вместе:

let arr1 = [1, 2];

let arr2 = [3, 4];

let result = arr1.concat(arr2);

console.log(result);

Результат выполнения кода:

[1, 2, 3, 4]

**Пример 3**

Можно передавать не только массивы, но и конкретные значение:

let arr1 = [1, 2];

let arr2 = [3, 4];

let result = arr1.concat(arr2, 5, 6);

console.log(result);

Результат выполнения кода:

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

**Пример 4**

Можно применить метод к пустому массиву. В этом случае все сливаемые массивы передадутся параметрами метода:

let arr1 = [1, 2];

let arr2 = [3, 4];

let arr3 = [5, 6];

let result = [].concat(arr1, arr2, arr3);

console.log(result);

Результат выполнения кода:

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

**Метод reverse**

Метод **reverse** изменяет порядок элементов в массиве на обратный.

Метод *изменяет исходный массив* (он станет перевернутым) и возвращает также перевернутый массив (можно пользоваться и так, и так).

**Синтаксис**

массив.reverse();

**Пример** **1**

Давайте перевернем массив:

let arr = ['a', 'b', 'c'];

arr.reverse();

console.log(arr);

Результат выполнения кода:

['c', 'b', 'a']

**Пример** **2**

А теперь давайте запишем перевернутый массив в новую переменную:

let arr = ['a', 'b', 'c'];

let res = arr.reverse();

console.log(res);

Результат выполнения кода:

['c', 'b', 'a']

**Пример** **3**. **Применение**

Давайте перевернем символы строки в обратном порядке. Для этого разобьем строку в массив с помощью [split](http://code.mu/ru/javascript/manual/string/split/) по разделителю '' (этот разделитель положит каждый символ строки в отдельный элемент массива), перевернем этот массив с помощью **reverse** и затем сольем перевернутый массив обратно с помощью [join](http://code.mu/ru/javascript/manual/array/join/):

let str = '123456789';

let arr1 = str.split('');

let arr2 = arr1.reverse();

let result = arr2.join('');

console.log(result);

Результат выполнения кода:

'987654321'

**Пример 4. Применение**

Упростим решение предыдущей задачи - сольем все команды в цепочку:

let str = '123456789';

let result = str.split('').reverse().join('');

console.log(result);

Результат выполнения кода:

'987654321'

**Метод push**

Метод **push** добавляет неограниченное количество элементов в конец массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Синтаксис**

массив.push(элемент, элемент, элемент...)

**Пример 1**

Давайте в конец массива добавим еще два новых элемента:

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.push('1', '2');

console.log(arr);

Результат выполнения кода:

['a', 'b', 'c', 'd', 'e', '1', '2']

**Пример 2**

Добавим в массив два новых элемента и выведем новую длину массива:

let arr = ['a', 'b', 'c', 'd', 'e'];

let length = arr.push('1', '2');

console.log(length);

Результат выполнения кода:

7

**Пример 3. Применение**

Заполним массив числами от 1 до 9:

let arr = [];

for (let i = 1; i <= 9; i++) {

arr.push(i)

}

console.log(arr);

Результат выполнения кода:

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

**Метод unshift**

Метод **unshift** добавляет неограниченное количество новых элементов в начало массива. При этом исходный массив изменяется, а результатом возвращается новая длина массива.

**Синтаксис**

массив.unshift(элемент, элемент, элемент...);

**Пример 1**

Давайте в начало массива добавим еще два новых элемента и выведем измененный массив:

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.unshift('1', '2');

console.log(arr);

Результат выполнения кода:

['1', '2', 'a', 'b', 'c', 'd', 'e']

**Пример 2**

Давайте добавим два новых элемента и выведем новую длину массива:

let arr = ['a', 'b', 'c', 'd', 'e'];

let length = arr.unshift('1', '2');

console.log(length);

Результат выполнения кода:

7

**Метод shift**

Метод **shift** удаляет первый элемент из массива. При этом исходный массив изменяется, а результатом метода возвращается удаленный элемент.

**Синтаксис**

массив.shift();

**Пример 2**

Давайте удалим первый элемент из массива:

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.shift();

console.log(arr);

Результат выполнения кода:

['b', 'c', 'd', 'e']

**Пример 3**

Давайте удалим первый элемент из массива и выведем его на экран:

let arr = ['a', 'b', 'c', 'd', 'e'];

let elem = arr.shift();

console.log(elem);

Результат выполнения кода:

'a'

**Пример 3. Применение**

Пусть дан массив ['1', '2', '3', '4', '5', '6']. Давайте сделаем из него строку '16-25-34'. Для решения задачи используем комбинацию методов **shift**, **pop, push и join:**

let arr = ['1', '2', '3', '4', '5', '6'];

let result = [];

while (arr.length > 0) { // массив уменьшается в цикле пока не достигнет нуля

let first = arr.shift();

let last = arr.pop();

let str = first + last; // тут будет строка '16', потом '25', потом '34'

result.push(str);

}

// После цикла в result лежит массив ['16', '25', '34']. Сольем его в строку:

result = result.join('-');

console.log(result);

Результат выполнения кода:

'16-25-34'

**Метод pop**

Метод **pop** удаляет последний элемент из массива. При этом исходный массив изменяется, а результатом метода возвращается удаленный элемент.

**Синтаксис**

массив.pop()

**Пример 1**

Давайте удалим из массива последний элемент:

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.pop();

console.log(arr);

Результат выполнения кода:

['a', 'b', 'c', 'd']

**Пример 2**

Давайте выведем последний элемент, который был удален из исходного массива:

let arr = ['a', 'b', 'c', 'd', 'e'];

let del = arr.pop();

console.log(del);

Результат выполнения кода:

'e'

**Пример 3. Применение**

Пусть дан массив ['1', '2', '3', '4', '5', '6']. Давайте сделаем из него строку '16-25-34'. Для решения задачи используем комбинацию методов **pop**, **shift, push и join**:

let arr = ['1', '2', '3', '4', '5', '6'];

let result = [];

while (arr.length > 0) { // массив уменьшается в цикле пока не достигнет нуля

let first = arr.shift();

let last = arr.pop();

let str = first + last; // тут будет строка '16', потом '25', потом '34'

result.push(str);

}

// После цикла в result лежит массив ['16', '25', '34']. Сольем его в строку:

result = result.join('-');

console.log(result);

Результат выполнения кода:

'16-25-34'

**Метод slice**

Метод **slice** вырезает и возвращает указанную часть массива.

Первым параметром указывается номер элемента массива, с которого начинается вырезание, а вторым параметром - номер элемента, на котором закончится вырезание (при этом элемент с этим номером не включится в вырезанную часть).

Второй параметр не является обязательным. Если его не указать - подмассив возьмется с указанного в первом параметре элемента до конца массива.

Второй параметр также может принимать отрицательные значения. В этом случае отсчет элемента, на котором закончится обрезание, начинается с конца массива. Причем, последний элемент имеет номер **-1**, предпоследний **-2** и так далее.

К примеру, если написать **slice(1, -1)**, то вырезанная часть возьмется с первого элемента включительно по последний не включая его. Если вам нужно включить и последний элемент - нужно просто не задавать второй параметр для slice, вот так: **slice(1)**.

Метод не изменяет исходный массив.

Напоминаю, что в массиве нумерация элементов начинается с нуля.

**Синтаксис**

массив.slice(откуда отрезать, [докуда отрезать])

**Пример 1**

Давайте вырежем из массива элементы с нулевого по второй не включительно (второй не вырежется):

let arr = ['a', 'b', 'c', 'd', 'e'];

let sub = arr.slice(0, 2);

console.log(sub);

Результат выполнения кода:

['a', 'b']

**Пример 2**

Давайте вырежем с первого элемента до конца массива. Для этого второй параметр не задаем:

let arr = ['a', 'b', 'c', 'd', 'e'];

let sub = arr.slice(1);

console.log(sub);

Результат выполнения кода:

['b', 'c', 'd', 'e']

**Пример 3**

Давайте вырежем элементы со второго по предпоследний (-1 указывает на последний элемент и он не включится в извлеченную часть):

let arr = ['a', 'b', 'c', 'd', 'e'];

let sub = arr.slice(1, -1);

console.log(sub);

Результат выполнения кода:

['b', 'c', 'd']

Преимущество такого подхода в том, что вырезаться всегда будет часть массива, не включая последний элемент, **независимо** от размера массива.

**Метод splice**

Метод **splice** удаляет или добавляет элементы в массив. Можно только удалять элементы, только добавлять или делать и то и другое одновременно. Метод очень универсальный и сложный для понимания.

Метод изменяет сам массив и возвращает при этом массив удаленных элементов.

Первым параметром метод принимает номер элемента массива, который нужно удалить. Вторым параметром - сколько элементов массива следует удалить. Если его поставить в **0** - то элементы удалены не будут (только добавлены новые).

Дальше через запятую идут элементы, которые нужно добавить в массив (являются необязательными параметрами). Эти элементы добавятся вместо удаленных элементов массива.

Если удаления не было (когда второй параметр 0) - элементы вставятся в массив начиная с той позиции, которая указана первым параметром метода.

Первый параметр может иметь отрицательное значение. В этом случае отсчет позиции начнется не с начала массива, а с конца. Причем, последний элемент имеет номер **-1**, предпоследний **-2** и так далее.

**Синтаксис**

массив.splice(откуда удаляем, сколько элементов удаляем, [вставить элемент], [вставить элемент]...);

**Пример 1**

Давайте удалим элементы с первого, 3 штуки (это 'b', 'c' и 'd'):

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.splice(1, 3);

console.log(arr);

Результат выполнения кода:

['a', 'e']

**Пример 2**

Давайте выведем массив удаленных элементов:

let arr = ['a', 'b', 'c', 'd', 'e'];

let del = arr.splice(1, 3);

console.log(del);

Результат выполнения кода:

['b', 'c', 'd']

**Пример 3**

Давайте сначала удалим элемент с номером 2, а потом вместо него вставим еще 3 новых элемента ('1', '2' и '3'):

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.splice(2, 1, '1', '2', '3');

console.log(arr);

Результат выполнения кода:

['a', 'b', '1', '2', '3', 'd', 'e']

**Пример 4**

Давайте теперь ничего не будем удалять, но на позицию 2 (это значит перед элементом с номером 2), вставим еще 3 новых элемента ('1', '2' и '3'):

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.splice(2, 0, '1', '2', '3');

console.log(arr);

Результат выполнения кода:

['a', 'b', '1', '2', '3', 'c', 'd', 'e']

**Пример 5. Отрицательное значение**

Давайте удалим предпоследний элемент 'd' (его номер -2):

let arr = ['a', 'b', 'c', 'd', 'e'];

arr.splice(-2, 1);

console.log(arr);

Результат выполнения кода:

['a', 'b', 'c', 'e']

**Метод sort**

Метод **sort** производит сортировку массива в лексикографическом порядке.

Также можно указать параметром собственную функцию для сортировки, но это необязательно.

**Синтаксис**

массив.sort([функция])

**Пример 1**

Давайте посортируем массив:

let arr = ["Банан", "Апельсин", "Яблоко", "Манго"];

console.log(arr.sort());

Результат выполнения кода:

[Апельсин, Банан, Манго, Яблоко]

**Функция Object.keys**

Функция **Object.keys** позволяет ключи объекта в виде массива.

**Синтаксис**

Object.keys(объект);

**Примеры**

**Пример 1**

Давайте получим ключи объекта в виде массива:

var obj **=** {a: 1, b: 2, c: 3};

console.log(Object.keys(obj));

Результат выполнения кода:

['a', 'b', 'c']