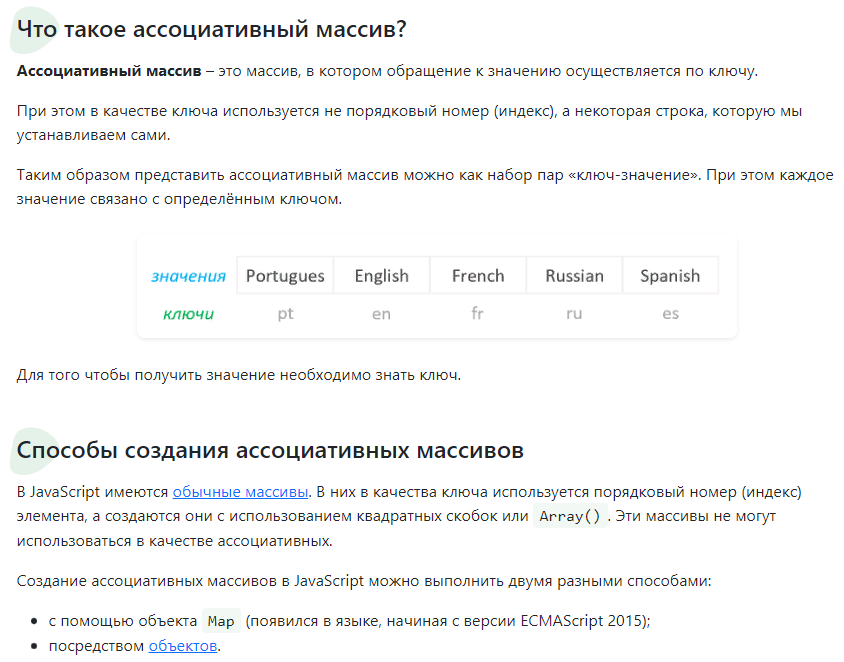
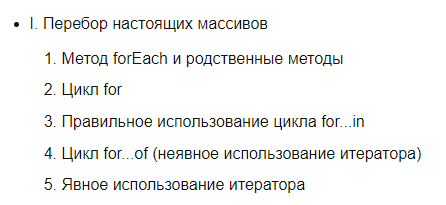
ассоциативный массив



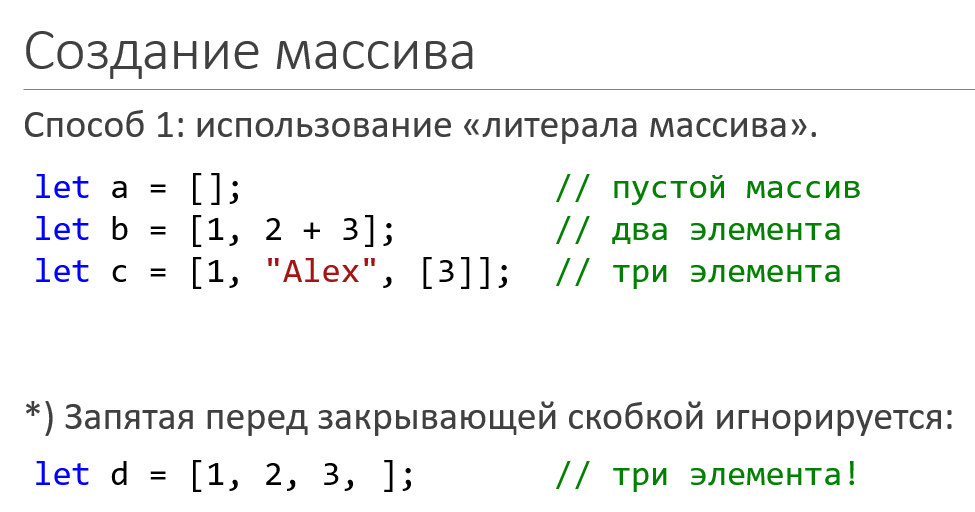
перебор массива

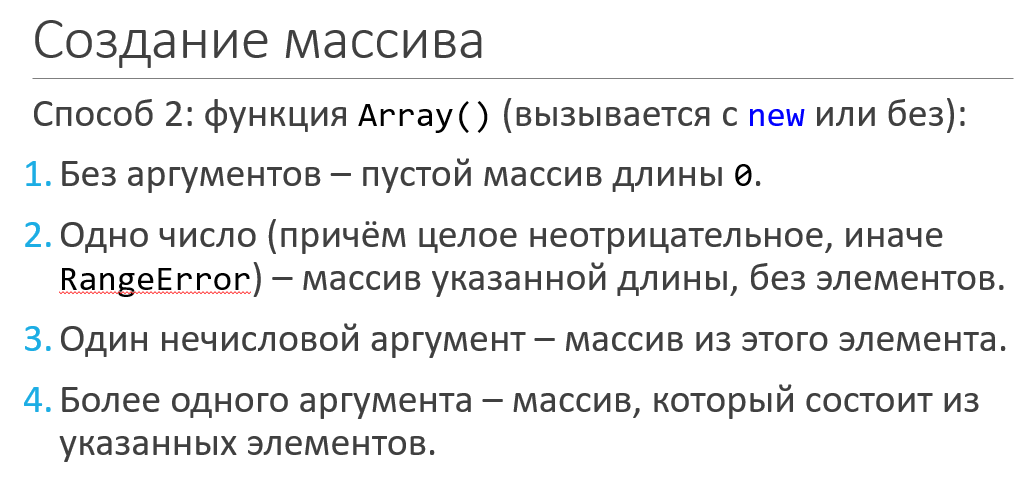


<https://doka.guide/js/array-foreach/>

**по лабе**

как создать массив

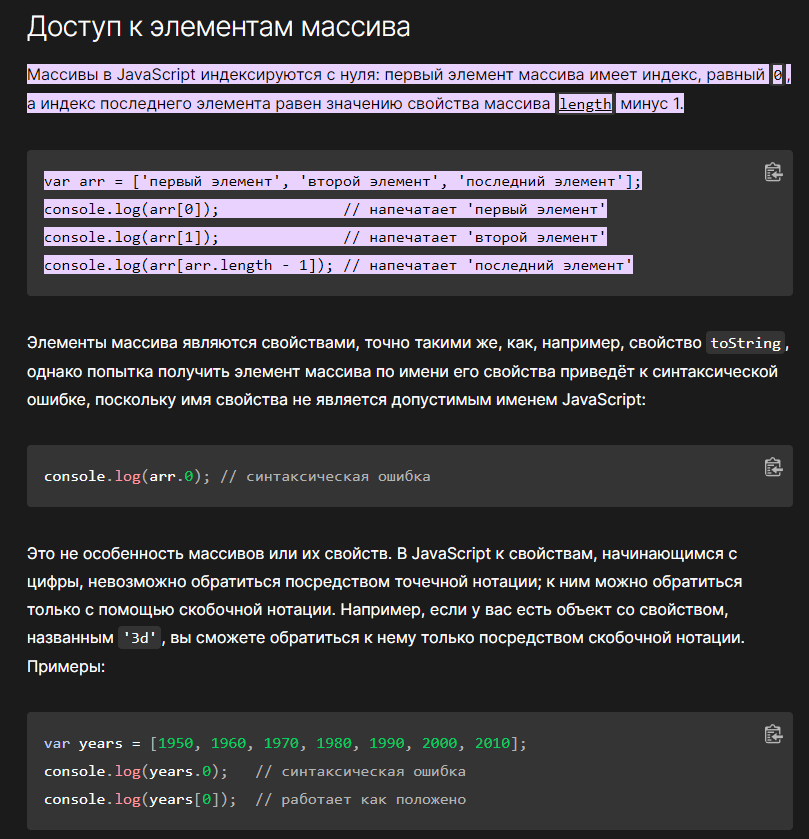




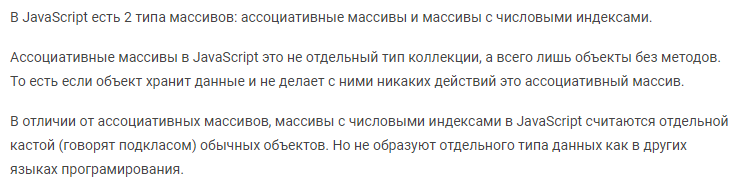
**Array- of (создание массива путем меппинга (присвоения индекса) отдельным элементам)**

**Array- from (создание массива путем добавления каждого символа элемента, как нового элемента массива)**

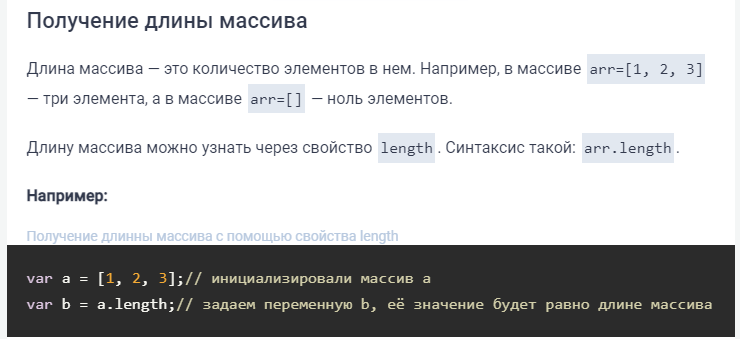
Индексы



виды массивов



длина массива



**как что делал**

**многомерный массив**

Многомерный массив в JavaScript (также называемый массив массивов) это массив, который содержит один или более массивов. В отличие от обычного массива, для доступа к элементу многомерного массива необходимо использовать два индекса.

Пример:

const multiDimensionalArray = [

[1, 2, 3],

[4, 5, 6],

[7, 8, 9]

];

// Для доступа к элементу 5 необходимо использовать индексы 1 и 1

console.log(multiDimensionalArray[1][1]); // 5

**фор ин, фор оф**

For in - это стандартный цикл для перебора свойств объектов. Наиболее часто используется для перебора ключей в объекте.

Пример:

const obj = {

a: 1,

b: 2,

c: 3

}

for (let key in obj) {

console.log(key, obj[key]);

}

// выведет 'a 1', 'b 2', 'c 3'

For of - это итератор, который позволяет перебирать элементы итерируемых объектов (таких как массивы, строки, последовательности и многое другое).

Пример:

const array = [1, 2, 3];

for (let value of array) {

console.log(value);

}

// выведет '1', '2', '3'

**методы перебора массива**

1. forEach()

Этот метод позволяет проходить по каждому элементу массива и выполнять над ним действие. Выглядит он так:

array.forEach(function(item, index, array) {

// твой код

});

В функцию передаются три аргумента: текущий элемент массива, его индекс и сам массив.

2. map()

Этот метод принимает в качестве аргумента функцию, которая будет вызвана для каждого элемента массива. Возвращает массив, состоящий из результатов вызовов функции. Выглядит он так:

const newArray = array.map(function(item, index, array) {

// твой код

});

3. filter()

Этот метод принимает в качестве аргумента функцию, которая будет вызвана для каждого элемента массива. Возвращает массив, состоящий из элементов, для которых функция вернула true. Выглядит он так:

const newArray = array.filter(function(item, index, array) {

// твой код

});

4. reduce()

Этот метод принимает в качестве аргументов функцию и начальное значение. Функция будет вызвана для каждого элемента массива, а в неё будут передаваться аккумулятор и текущий элемент массива. Возвращает значение аккумулятора. Выглядит он так:

const result = array.reduce(function(accumulator, item, index, array) {

// твой код

}, initialValue);

5. find()

Этот метод принимает в качестве аргумента функцию, которая будет вызвана для каждого элемента массива. Возвращает первое значение, для которого функция вернула true. Выглядит он так:

const result = array.find(function(item, index, array) {

// твой код

});

6. every()

Этот метод принимает в качестве аргумента функцию, которая будет вызвана для каждого элемента массива. Возвращает true, если для всех элементов массива функция вернула true. Выглядит он так:

const result = array.every(function(item, index, array) {

// твой код

});

7. some()

Этот метод принимает в качестве аргумента функцию, которая будет вызвана для каждого элемента массива. Возвращает true, если хотя бы для одного элемента массива функция вернула true. Выглядит он так:

const result = array.some(function(item, index, array) {

// твой код

});