Методы массивов

Методы для работы с массивами

Array.forEach(cb) - метод, который позволяет применить функцию callback последовательно к каждому
элементу массива. Мутирует исходный массив

Params:

cb* (currentValue* ,index ,array)

thisArg

Вовзращает undefined.

```
const names = ['Anna', 'Ihor', 'Kate', 'Vlad'];

names.forEach((name, index, arr) => {
  console.log(`${index + 1}.My name is ${name}. Next name is ${arr[index+1]}`)
} , this);

const colors = ['blue', 'green', 'white'];

function iterate(item) {
  console.log(item);
}

colors.forEach(iterate);

function iterate2(item, index, array) {
  console.log(item);
  if (index === array.length - 1) {
    console.log('The last iteration!');
  }
}
colors.forEach(iterate2);
```

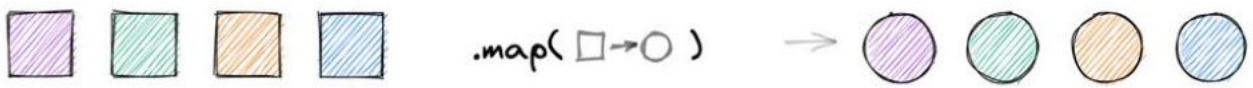
Array.map(cb) - создает новый массив, такой же длинны как и исходный. Этот метод НЕ мутирует исходный
массив. Элементами нового массива становятся результаты
функции callback.

Params:

cb* (currentValue* ,index ,array)

thisArg

Возвращает новый массив.




```
const numbers = [2, 4, 8, 16, 32];

const squaresReturn = numbers.map((number) => {
    return Math.pow(number, 2)
});
const squares = numbers.map((number) => Math.pow(number, 2))


const str = 'abcd'

const strChanger = str =>
  str
  .split('')
  .map((a , index) => a.repeat(index + 1))
  .map(a => a.slice(0 , 1).toUpperCase() + a.slice(1))
  .join('-')
```

Array.filter(cb) - этот метод позволяет удалить из
исходного массива значения, которые не подходят по
условиям функции callback. Этот метод НЕ мутирует исходный массив.

Params:

cb* (currentValue* ,index ,array)

thisArg

Возвращает новый массив.


```
const words = ['spray', 'limit', 'exuberant', 'destruction'];

const result = words.filter(word => word.length > 6);

[7, 'ate', '', false, 9].filter(Boolean)

// Функция возвращает новый массив, который наполнен теми значениями,
// которые не передавались как второй и последующие аргументы функции
const without = (arr , ...rest) => arr.filter(a => !rest.includes(a))
```

Array.sort(cb) - этот метод используется для сортировки массива. Этот метод МУТИРУЕТ исходный
массив

cb - функция, которая принимает два аргумента:
- currentValue – текущее значение.
- prevValue – предидущее значение.

Возвращает отсортированный массив.

```
const fruit = ['арбузы', 'бананы', 'Вишня']
fruit.sort()

const scores = [1, 2, 10, 21]
scores.sort()

const things = ['слово', 'Слово', '1 Слово', '2 Слова']
things.sort()

const phrase = 'Лупайте сю скалу! Нехай ні жар, ні холод не спинить вас';
const sortedWords = phrase.split(' ').sort((a,b) => a.length - b.length);
//сю ні ні не вас жар, Нехай холод скалу! Лупайте спинить
```

Array.find(cb) - этот метод позволяет найти элемент
по конкретному признаку. Обычно используется для поиска объектов внутри массива.

Params:

cb* (currentValue* ,index ,array)

thisArg

Возвращает значение элемента массива либо undefined.

Похожие методы: Array.findIndex() , Array.indexOf()


```
const arr = [1 , 2 , 3 , 4]

typeof arr.find(a => '' + a === '1')
//number

const orders = [
  { id: 1, product: "Лыжи", price: 100},
  { id: 2, product: "Шапка", price: 200},
  { id: 3, product: "Лыжные очки", price: 300},
  { id: 4, product: "Теплая куртка", price: 400},
  { id: 5, product: "Номер в буковель", price: 500}
];

const hat = orders.find((order) => order.product.toLowerCase() === "шапка");
```

Array.every(cb) - этот метод проверяет соответствует ли каждый элемент массива определенному условию

Params:

cb* (currentValue* ,index ,array)

thisArg

Возвращает true или false.


```
const orders = [
  { id: 1, product: "Лыжи", price: 100},
  { id: 2, product: "Шапка", price: 200},
  { id: 3, product: "Лыжные очки", price: 300},
  { id: 4, product: "Теплая куртка", price: 400},
  { id: 5, product: "Номер в буковель", price: 500}
];

const isBigger200 = orders.every((order) => order.price > 200)

const isBigger50 = orders.every((order) => order.price > 50)
```

Array.some(cb) - этот метод проверяет

Соответствует ли каждый элемент массива определенному условию

Params:

- cb* (currentValue* ,index ,array)
- thisArg

Возвращает true или false.


```
const orders = [
  { id: 1, product: "Лыжи", price: 100},
  { id: 2, product: "Шапка", price: 200},
  { id: 3, product: "Лыжные очки", price: 300},
  { id: 4, product: "Теплая куртка", price: 400},
  { id: 5, product: "Номер в буковель", price: 500}
];

const isBigger200 = orders.some((order) => order.price > 200);

const isBigger50 = orders.some((order) => order.price < 50);
```

Array.reduce(cb) - этот метод создан для того,
чтобы аккумулировать значение (то есть приводить массив к одному значению).

Params:

cb* (accumulator*, currentValue*, index, array)

initialValue


```
const orders = [
  { id: 1, product: "Лыжи", price: 100},
  { id: 2, product: "Шапка", price: 200},
  { id: 3, product: "Лыжные очки", price: 300},
  { id: 4, product: "Теплая куртка", price: 400},
  { id: 5, product: "Номер в буковель", price: 500}
];

var total = [0, 1, 2, 3].reduce((a, b) => a + b);

var initialValue = 0;
var sum = [{x: 1}, {x:2}, {x:3}].reduce(
    (accumulator, currentValue) => accumulator + currentValue.x,
    initialValue
);

var flattened = [[0, 1], [2, 3], [4, 5]].reduce((a, b) => a.concat(b))

const totalOrdersSum = orders.reduce((total, order) => total + order.price, 0)
```

Q&A