







Frontender[1.0] JavaScript - Arrays - методы-циклы, forEach, map, filter, reduce, some, every, find, slice, splice

 YouTube	https://youtu.be/iEZg-fOohDg
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	23



Задачи к этому уроку тут https://boosty.to/dmitry_ko

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <script src="script.js"></script>
  <title>Arrays - forEach, filter, map, reduce</title>
</head>
<body>
</body>
</html>
```

```
// https://learn.javascript.ru/array-methods
/*
Arrays - Methods (Методы)
forEach, indexOf, lastIndexOf, find, findIndex,
some, every, slice, splice, toSpliced,
filter, map, reduce
*/

const log = console.log;

const skills = ["html", "css", "scss", "js", "git", "ts", "react", "css"];

// ----- 💣Перебор массива💣 -----
```

```

// -- 💡 forEach (перебирает массив как for, for of) --

// skills.forEach((value) => log(value));
// skills.forEach((value, index) => log(value, index));
// skills.forEach((value, index, sourceArr) => log(value, index, sourceArr));

const logValues = (value) => log(value);

function logValuesFn(value) {
  log(value);
};

// skills.forEach(logValues);
// skills.forEach(logValuesFn);

// ----- 💣 Поиск в массиве 💣 -----

// -- 💡 indexOf, lastIndexOf (возвращают индекс запрашиваемого элемента) --

log(skills.indexOf("js"));
log(skills.indexOf("js", 2));
log(skills.indexOf("js", 4)); // -1 если не найдено

log(skills.indexOf("css")); // отдает индекс первого найденного элемента
log(skills.lastIndexOf("css")); // отдает индекс последнего найденного элемента

log(skills.indexOf("git")); // если элемент один, то индексы совпадают
log(skills.lastIndexOf("git")); // если элемент один, то индексы совпадают

// -- 💡 some, every (возвращают true/false если по условию элемент/элементы найдены/не найдены)

const isJs = skills.some((value) => value === "js");
log("isJs", isJs);

const phones = [
  {id: 1, title: "samsungA50"},
  {id: 2, title: "iphone10"},
  {id: 3, title: "nokia3310"},
  {id: 4, title: "xiaomi"},
];

const everyHasTitles = phones.every((phone) => "title" in phone);
const someHasTitles = phones.some((phone) => "title" in phone);

log("allHasTitles", everyHasTitles);
log("someHasTitles", someHasTitles);

// -- 💡 find (находит и возвращает первый найденный элемент по условию), findIndex --

const nokia = phones.find((phone) => phone.title === "nokia3310");

log(nokia);

const nokiaIndex = phones.findIndex((phone) => phone.title === "nokia3310");

```

```

log(nokiaIndex);

// -- 💡 filter -- похож на find, только возвращает все элементы подходящие условию.

const filteredSkills = skills.filter((skill) => skill.includes("c"));
log(filteredSkills);

const evenNumbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].filter((num) => num % 2 === 0);

log(evenNumbers) // только четные

const clients = [
  {id: 1, level: 3, name: "Lucy", status: "online"},
  {id: 2, level: 1, name: "Rick", status: "offline"},
  {id: 3, level: 3, name: "Jack", status: "online"},
  {id: 4, level: 2, name: "Helen", status: "online"},
  {id: 5, level: 1, name: "Alice", status: "offline"},
  {id: 6, level: 1, name: "Derek", status: "offline"},
  {id: 7, level: 3, name: "Megan", status: "online"},
];

const clientsHighLevel = clients.filter(client => client.level === 3);

log(clientsHighLevel);

// ----- 💣Добавление и удаление элементов💣 -----

/*
💡 slice - возвращает новый массив, в который копирует
все элементы с индекса start до end
*/
const newPhones = phones.slice(0, 2);
log(newPhones);

// delete phones[nokiaIndex];
// log(phones);

/*
💡 splice - изменяет исходный массив, умеет добавлять, удалять и заменять элементы
*/
// log(phones.splice(nokiaIndex, 1));
// log(phones);
// log(phones.splice(1, 0, {id: 5, title: "iphone13"}));
// log(phones);

/*
💡 toSpliced, тот же splice, но только делает копию массива, а не изменяет исходный массив
*/
// const phones2025 = phones.toSpliced();
const phones2025 = phones.toSpliced(2, 1, {id: 5, title: "iphone16"});

log(phones2025);

// ----- 💣Преобразование элементов💣 -----

const str = "hello";
log(str.split("").reverse().join(""));

```

```
/*
💡 map
один из наиболее полезных и часто используемых.

map вызывает функцию для каждого элемента массива и возвращает массив результатов выполнения
этой функции.
*/
```

```
const clientsNames = clients.map(client => client.name);
```

```
const clientNamesAndStatuses = clients.map(client => {
  return {
    name: client.name,
    status: client.status
  }
}).map(client => {
  if (client.status === "online") {
    client.status = "online 🟢"
  } else {
    client.status = "offline 🔴"
  }
  return client;
}).filter(c => c.status.startsWith("on"));
```

```
log(clients);
log(clientsNames);
log(clientNamesAndStatuses);
```

```
/*
💡 reduce
используется для вычисления единого значения на основе всего массива
```

```
arr.reduce((accumulator, item, index, array) => {
  // ...
}, [initial]);
```

Аргументы:

accumulator – результат предыдущего вызова этой функции, равен initial при первом вызове (если передан initial),
item – очередной элемент массива,
index – его позиция,
array – сам массив.

```
*/
```

```
const numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
```

```
const sumAllNumbers = numbers.reduce((accumulator, currentValue) => {
  return accumulator + currentValue;
}, 0);
```

```
log(sumAllNumbers);
```

```
const books = [
  {id: 1, title: "Гарри Поттер", price: 59, category: "fantasy" },
  {id: 2, title: "Чистый код", price: 109, category: "science" },
  {id: 3, title: "Темная Башня", price: 149, category: "fantasy" },
  {id: 4, title: "Грокаем алгоритмы", price: 173, category: "science" },
  {id: 5, title: "Многопоточный JavaScript", price: 79, category: "science" },
```

```
];

const sumOfBooks = books.reduce((sum, book) => {
  return sum + book.price;
}, 0);

log(sumOfBooks);

let categoriesCount = 0;

const categoryMap = books.reduce((res, book) => {
  if (book.category in res) {
    res[book.category]++;
  } else {
    res[book.category] = 1;
    categoriesCount++;
  }

  // book.category in res ? res[book.category]++ : res[book.category] = 1;

  return res;
}, {});

log(categoryMap);

log(categoriesCount);
// log(Object.keys(categoryMap).length);
```