








Frontender[1.0] JavaScript - Массивы - методы, rest, spread, деструктуризация, for of

 YouTube	https://youtu.be/CcmYGLNVqCA
 Telegram	https://t.me/Dmitry_Kolotilshikov
 Github	https://github.com/DmitryKolotilshikov/
 Boosty	https://boosty.to/dmitry_ko
# Номер урока	22

 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 - methods, spread, rest</title>
</head>
<body>
</body>
</html>
```

```
// Arrays (Массивы) - for of, методы, деструктуризация, spread, rest

const styles = `background: lightsalmon;`
const log = console.log;

const cars = ["audi", "ford", "mercedes", "mazda", "tesla"];

const users = [
  { id: 1, name: "Alex", age: 35, position: "manager" },
  { id: 2, name: "Kate", age: 22, position: "qa" },
  { id: 3, name: "Nikita", age: 29, position: "developer" },
];

// for of (цикл по массиву)
```

```

for (const car of cars) {
    log(`Car -> ${car}`);
}
for (const user of users) {
    log(`User -> ${user.name} ${user.age} ${user.position}`);
}
// ----- Методы -----

const numbers1 = [4, 33, 2, 1];

log(numbers1.sort((a, b) => a - b)); // sort - изменяет исходный массив
log(numbers1.sort((a, b) => b - a));
log(numbers1);
log("Reverse ", numbers1.reverse()); // reverse - изменяет исходный массив
log("Reverse ", numbers1);

const numbers2 = [4, 33, 2, 1];

const newNumbers2 = numbers2.toSorted((a, b) => a - b); // toSorted - делает копию массива
log(newNumbers2);
log(numbers2);
log("toReversed ", numbers2.toReversed()); // toReversed - делает копию массива
log("toReversed ", numbers2);

// -----

const newCars = ["nissan", "volvo"];

const allCars = cars.concat(newCars); // concat - приклеивает к исходному массиву другие массивы
log(allCars);

log(allCars.toString()); // toString - просто делает строку из массива
log(allCars.join()); // join - делает строку с различными разделителями
log(allCars.join(''));
log(allCars.join(','));
log(allCars.join('-'));

// -----

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

log(nestedArrays);
log(nestedArrays[2][0]); // 3
log(nestedArrays[3][1]); // 6
log(nestedArrays[4][1][0]); // 8

log(nestedArrays.flat()); // flat - делает плоским массив
log(nestedArrays.flat(2));
log(nestedArrays.flat(Infinity));

// -----

log(cars.includes("audi")); // проверяет есть ли переданный аргумент в массиве, возвращает true или false
log(cars.includes("lada"));

```

```

log(cars.at(1)); // at - возвращает элемент массива по указанному индексу
log(cars.at(-1));

// ----- static methods (статические методы объекта/класса Array) -----

const num = 123;

log(Array.isArray(cars)); // Array.isArray - проверяет является ли переданный аргумент массивом, возвращает true или false
log(Array.isArray(num));

const arrayOf = Array.of(1, 2, "hello"); // Array.of - создаем массив из переданных аргументов
log(arrayOf);

const arrayFrom = Array.from("12345"); // Array.from - создаем массив из итерируемых массивоподобных элементов (строка, SET, MAP, Массив)
log(arrayFrom);

// ----- деструктуризация -----

/*
Деструктурирующее присваивание – это специальный синтаксис, который
позволяет нам «распаковать» массивы или объекты в несколько переменных,
так как иногда они более удобны.
Деструктурирующее присваивание не изменяет исходный массив, а только копирует нужные значения
*/

const names = ["Alex", "Mike", "Angelina"];
const names2 = ["Kate"];

// const nameAlex = names[0];
// const nameMike = names[1];

const [nameAlex, nameMike] = names;
const [nameKate, nameOlga = "Olga"] = names2;

log(names);
log(nameAlex);
log(nameMike);

log(nameKate);
log(nameOlga);

/*
Кортеж – это упрощенная структура данных, которая имеет определенное число и последовательность значений.
*/

// const [counter, setCounter] = useState(0); // Пример из React

// ----- spread (оператор расширения) -----

const allNames = [...names, "Nik", ...names2];
log(allNames);

const strHello = "Hello";

```

```

log(...strHello]);

const allNumbers = [1, 55, 9, 33, 124, 765, 0, 3, 4];

log(Math.max(...allNumbers));

// ----- rest (остаточные параметры) -----

const sum = (...numbers) => {
  let sum = 0;

  for (const value of numbers) {
    sum += value;
  }

  return sum;
}

log(sum(1, 2, 3, 4, 5));

const [firstName, ...restNames] = names;

log(firstName, restNames);

// Остаточные параметры должны располагаться в конце !

// const fn = (arg1, ...rest, arg2) => {
//   // Error
// }

const getTitleAndDescription = (name, ...description) => {
  log(`%c${name}`, styles);

  for (const value of description) {
    log(`${value} 💣`);
  }
}

getTitleAndDescription("JavaScript", "- Язык программирования", "- На нем можно кодить фронт
и бэк, моб приложения, игры, десктопные программы", "- гибкий и очень популярный");

```