PHASE 1

WEEK 1

DAY 2

ELBRUS/BOOTCAMP

План

1. Массив
2. Перебор массива
3. Методы массива
4. Копирование массива
5. Советы по парному программированию

Массив

Упорядоченный набор значений

Массивы

```
const emptyArray = [];

// массив с однотипными данными
const padawans = ['Revan', 'Bastila Shan', 'Jolee Bindo',
'Juhani'];

// массив с данными разного типа
const revan = ['Revan', 99.9, true];

// доступ к значению по индексу
revan[1];
```

Перебор массива

Цикл for

```
// исходный массив
const letters = ['a', 'b', 'c'];

for (let index = 0; index < letters.length; index += 1) {
 // перебор всех элементов массива
}
```

Цикл for...of

```
// исходный массив
const letters = ['a', 'b', 'c'];

for (const letter of letters) {
 // перебор всех элементов массива
 // letter — значение текущего элемента
}
```

Цикл for...in

Не рекомендуется использовать!

```
// исходный массив
const letters = ['a', 'b', 'c'];

for (const index in letters) {
 // перебор элементов массива, которым присвоены значения
 // index — индекс текущего элемента
}
```

Методы массивов

push

Добавляет новый элемент **в конец** массива.

```
// исходный массив
const letters = ['a', 'b', 'c'];

// добавить элемент в конец массива
letters.push('d');

console.log(letters); // ['a', 'b', 'c', 'd']
```

pop

Извлекает **последний элемент** массива и возвращает его.

```
// исходный массив
const letters = ['a', 'b', 'c', 'd'];

// извлечь последний элемент массива
const lastLetter = letters.pop();

console.log(lastLetter); // 'd'
console.log(letters); // ['a', 'b', 'c']
```

unshift

Добавляет новый элемент **в начало** массива.

```
// исходный массив
const letters = ['x', 'y', 'z'];

// добавить элемент в начало массива
letters.unshift('w');

console.log(letters); // ['w', 'x', 'y', 'z']
```

shift

Извлекает **первый элемент** массива и возвращает его.

```
// исходный массив
const letters = ['w', 'x', 'y', 'z'];

// извлечь первый элемент массива
const firstElement = letters.shift();

console.log(firstElement); // 'w'
console.log(letters); // ['x', 'y', 'z']
```

split

Разбивает строку на массив подстрок.

```
// исходная строка
const nameData = 'Igor;Anna;Vasya';

// создать массив на основе строки
const names = nameData.split(';'); // ['Igor', 'Anna', 'Vasya']
```

join

“Склеивает” строку из массива подстрок.

```
// исходный массив
const names = ['Igor', 'Anna', 'Vasya'];

// создать строку на основе массива
const formattedNames = names.join(', '); // 'Igor, Anna, Vasya'
```

slice

Возвращает новый массив — срез исходного массива.

```
// исходный массив
const letters = ['a', 'b', 'c', 'd', 'e'];

// вернуть новый массив с 1 индекса
const allFromFirst = letters.slice(1); // ['b', 'c', 'd', 'e']

// вернуть новый массив со 2 индекса (включительно) по 4-й (не включая его)
const secondAndThird = letters.slice(2, 4); // ['c', 'd']
```

splice

Позволяет извлекать элементы массива, а также добавлять новые элементы.

Возвращает массив извлечённых элементов.

```
const letters = ['a', 'b', 'c', 'd', 'e'];

// извлечь 1 элемент, начиная с индекса 2
letters.splice(2, 1); // ['c'] — элемент с индексом 2 был удалён

console.log(letters); // ['a', 'b', 'd', 'e']
```

splice

Позволяет извлекать элементы массива, а также добавлять новые элементы.

Возвращает массив извлечённых элементов.

```
const letters = ['a', 'b', 'c', 'd', 'e'];

// добавить новый элемент на место элемента с индексом 2
letters.splice(2, 0, 'подставное лицо'); // [] — элементы не извлекались

console.log(letters); // ['a', 'b', 'подставное лицо', 'c', 'd', 'e']
```

concat

Создаёт новый массив, “склеивая” несколько исходных.

```
// исходные массивы
const maleNames = ['Василий', 'Игорь'];
const femaleNames = ['Анна', 'Марина'];

const names = ['Максим', 'Наталья'].concat(femaleNames, maleNames);
console.log(names); // ['Максим', 'Наталья', 'Анна', 'Марина', 'Василий', 'Игорь']
```

Альтернатива concat

Можно добиться предыдущего результата через **spread**-оператор:

```
const names = [
 'Максим',
 'Наталья',
 ...femaleNames, // “выложить” все элементы из массива femaleNames
 ...maleNames,   // “выложить” все элементы из массива maleNames
];
```

reverse

Переставляет элементы исходного массива в обратном порядке.

```
// исходный массив
const names = ['Максим', 'Наталья'];

names.reverse();
console.log(names); // ['Наталья', 'Максим']
```

Callback-функция

Функция, переданная параметром в другую функцию

forEach

Метод массива. Вызывает указанную callback-функцию для каждого элемента.

```
// исходный массив
const letters = ['a', 'b', 'c'];

// callback-функция
const doSomething = (letter) => { console.log(letter); };

// вызвать функцию для каждого элемента в массиве
letters.forEach(doSomething);
```

Всегда возвращает undefined — что метод, что callback-функция.

every

Проверяет, удовлетворяют ли **все элементы** массива заданному условию.

```
function isMoreThanTen(element) {
 return element > 10;
}

[12, 5, 8, 130, 44].every(isMoreThanTen);   // false
[12, 54, 18, 130, 44].every(isMoreThanTen); // true
```

some

Проверяет, удовлетворяет ли **хотя бы один** элемент массива заданному условию.

```
function isMoreThanTen(element) {
 return element > 10;
}

[12, 5, 8, 130, 44].some(isMoreThanTen);    // true
```

filter

Возвращает новый массив со всеми элементами, удовлетворяющими заданному условию.

```
const words = ["lemon", "papaya", "apple", "cherry"];

function getLongWord(word) {
 return word.length > 5;
}

words.filter(getLongWord); // ["papaya", "cherry"];
```

find / findIndex

Возвращает первый элемент/индекс элемента, удовлетворяющий заданному условию.

```
const words = ["lemon", "papaya", "apple", "cherry"];

function getLongWord(word) {
 return word.length === 6;
}

words.find(getLongWord); // "papaya"
words.findIndex(getLongWord); // 1
```

indexOf / lastIndexOf

.indexOf(...) находит и возвращает индекс первого элемента, совпадающего с аргументом.
.lastIndexOf(...) делает то же самое, только возвращает индекс последнего элемента.

```
// исходный массив
const words = ["lemon", "papaya", "apple", "apple", "apple", "cherry", "papaya"];

words.indexOf("apple"); // 2
words.lastIndexOf("apple"); // 4
```

map

Создаёт новый массив с результатом вызова функции

```
// исходный массив
const words = ["lemon", "papaya", "apple", "cherry"];

// копия массива words + изменения после выполнения map()
const newWords = words.map((item, index, arr) => {
 return `Текущее слово: ${item}, в нём ${item.length} букв из массива ${arr}`;
});
```

sort

Сортирует элементы массива на месте, возвращает отсортированный массив. По умолчанию сортировка лексикографическая — по тексту, а не по числам.

```
const numbers = [1, 5, 11, 28, 3, 30, 9, 4];
numbers.sort();
// [1, 11, 28, 3, 30, 4, 5, 9]

numbers.sort((a, b) => {
 return a - b;
});
// [1, 3, 4, 5, 9, 11, 28, 30]
```

reduce / reduceRight

reduce() применяет функцию к каждому значению массива (слева-направо), сводя его к одному значению (аккумулятору)
reduceRight() делает то же, что и reduce() для элементов массива справа-налево

```
const arr = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
const sum = arr.reduce((sum, current) => {
 if (current % 2 === 0) {
   return sum + current;
 }
 return sum;
}, 0); // 30
```

Копирование массива

Поверхностное копирование

```
// исходный массив, который будем копировать
const arr = ["a", "b", "c"];
```

Нельзя:

```
const arrCopy = arr;
```

Можно:

```
const arrCopy1 = […arr];
const arrCopy2 = arr.slice();
const arrCopy3 = arr.map(el => el);
```

Глубокое копирование

```
// исходный массив, который будем копировать
const arr = ["a", "b", "c"];

// глубокое копирование массива
const deepCopy = JSON.parse(JSON.stringify(arr));
```

Парное программирование, инструкция

1. Работать в 1 форке репозитория. Напарника нужно добавить в Collaborators репозитория.
2. Роли меняются по таймеру каждые 30 минут
3. Использовать 1 компьютер (второй нужно выключить, иначе можно уйти в соло-групповую работу) и монитор
4. Договариваться о времени обеда. В это время не работать по одиночке, а ждать напарника.
5. Менять роли Драйвера и Навигатора
6. В конце парного программирования дать обратную связь что стоит улучшить в следующий раз.

Парное программирование, преимущества

- ✔ Обмен опытом
- ✔ Знания о системе
- ✔ Коллективное владение кодом
- ✔ Наставничество
- ✔ Больше общения
- ✔ Стандарты кодирования
- ✔ Улучшение дисциплины
- ✔ Сопряжение потока
- ✔ Меньше прерываний

Парное программирование, анти-паттерны

- ✖ Наблюдай за Мастером
- ✖ Диктатор
- ✖ Сходи за кофе
- ✖ Молчаливые партнеры
- ✖ Разделение задач за одним столом
- ✖ Неудобно сидеть
- ✖ Партнер занят своим делом
- ✖ Свои настройки окружения

Обратная связь, как её дать?

1. Сильная сторона
2. Пожелание
3. Результат