Arrays

Массивы
Методы массивов

TeachMeSkills

Урок

Введение

Для хранения упорядоченных коллекций существует особая структура данных, которая называется массив, `Array`.

Массив - это лаконичный способ хранения списка элементов под одним именем.

Массив – это особый подвид объектов

Введение

```
let array    =       ['Bob', 'John', 'Donald', 'Stacy', 'Jack']
```

Создание массива

```
let array = new Array();
let arr = [];
```

Особенности работы с массивом

```
let arr = [0, 'cat', null, undefined, []]
```

В массиве могут хранится
Любые типы данных

```
arr[3]
```

Доступ к отдельным элементам в массиве, можем получить используя квадратные скобки

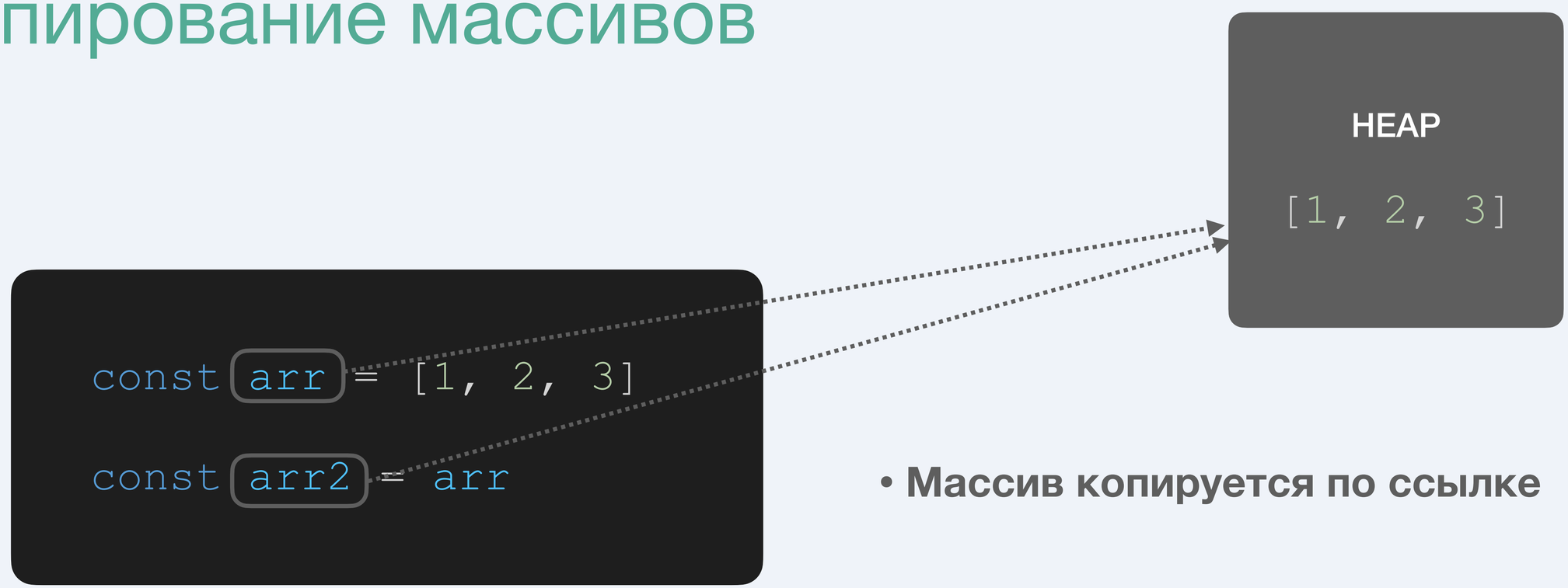
```
arr[2] = 99
```

Можно изменять отдельные элементы массива

```
arr.length
```

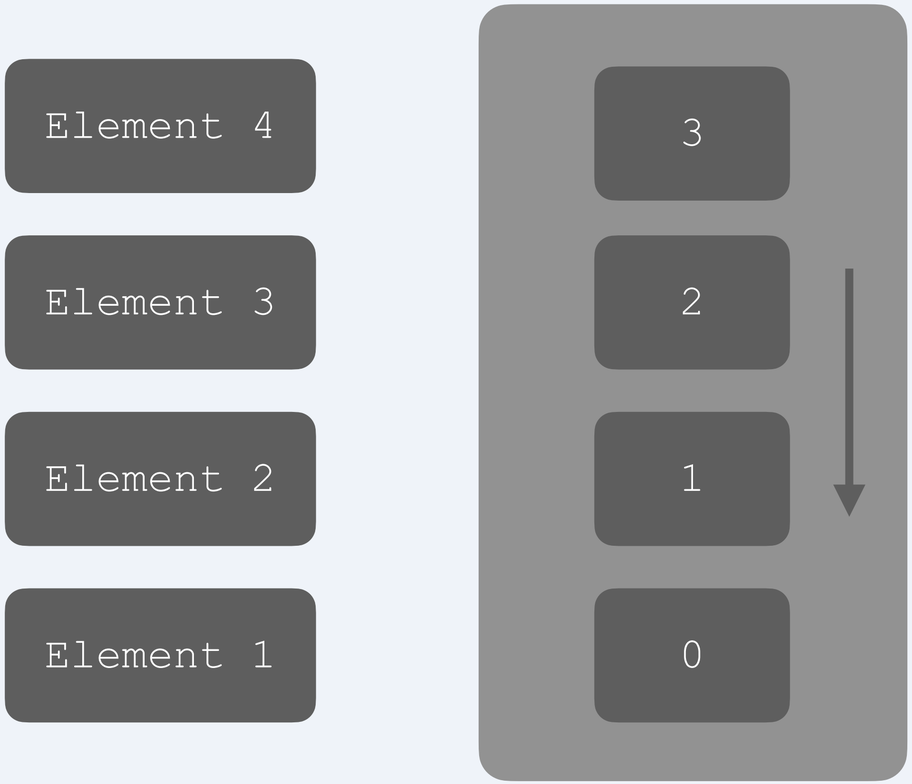
Вы можете найти длину массива (количество элементов в нём) используя свойство `length`

Копирование массивов



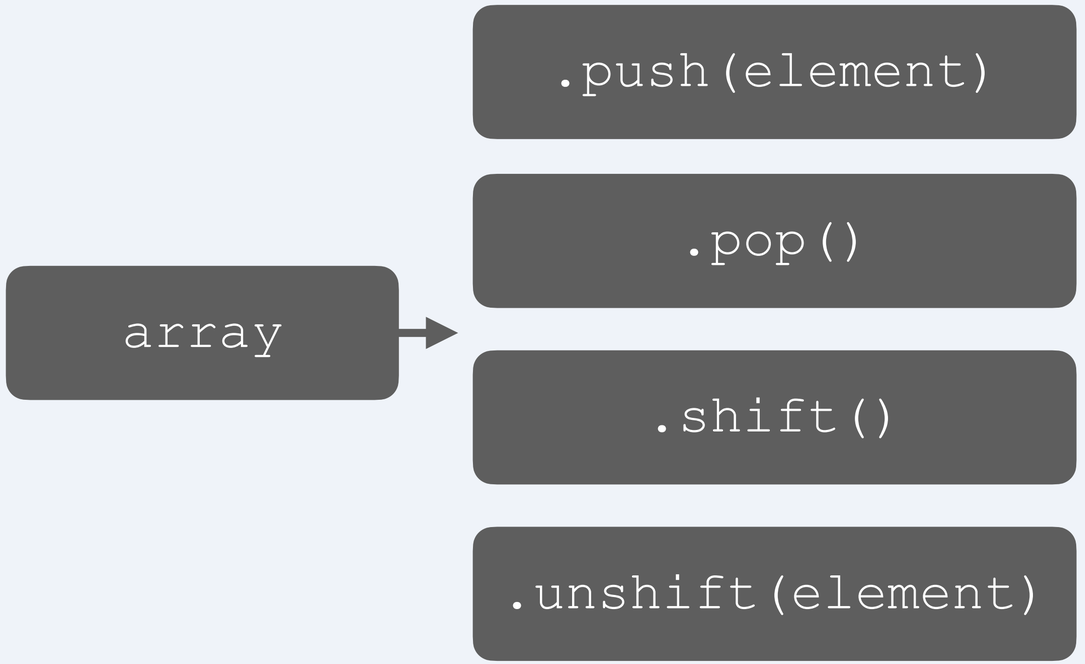
- **Массив копируется по ссылке**

Очередь



- **Отсчет элементов массива начинается с 0**

Добавление и удаление элементов массива



Добавляет элемент в конец массива

Удаляет последний элемент из массива и возвращает его

Удаляет из массива первый элемент и возвращает его

Добавляет элемент в начало массива

Перебор элементов массива

Цикл for

Цикл for of

```
const numbers = [1, 2, 3]

for(let i = 0; i < numbers.length; i++){
    console.log(numbers[i]);
}

for(let number of numbers){
    console.log(number);
}
```

Часто используемые методы массивов

				.reverse()
			.pop()	.map()
.concat()	.filter()	.forEach()	.push()	.slice()
.every()	.find()	.indexOf()	.reduce()	.some()
.findIndex()	.includes()	.join()	.splice()	.sort()

Выполняет указанную функцию один раз для каждого элемента в массиве.

`.forEach()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

array.forEach(function(number){
    console.log(number);
})
```

Возвращает первый индекс, по которому данный элемент может быть найден в массиве или -1, если такого индекса нет.

`.indexOf()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.indexOf(8) // 6
```

Удаление существующих элементов и/или добавление новых.

`.splice()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.splice(1, 4)  // [1, 1, 2, 3]
```

Возвращает новый массив, содержащий копию части исходного массива.

`.slice()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.slice(1, 3)  // [1, 1]
```

Создаёт новый массив с результатом вызова указанной функции для каждого элемента массива.

`.map()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.map(function(number){
    return number ** 2
}) // [0, 1, 1, 4, 9, 25, 64]
```

Возвращает **значение** первого найденного в массиве элемента, которое удовлетворяет условию переданному в callback функции. В противном случае возвращается undefined.

`.find()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.find(function(number){
    return number === 8
}) // 8
```

Создаёт новый массив со всеми элементами, прошедшими проверку, задаваемую в передаваемой функции.

`.filter()`

```
const array = [0, 1, 1, 2, 3, 5, 8]

let result = array.filter(function(number){
    return number > 3
}) // [5, 8]
```

Часто используемые методы массивов

`.concat()` → **Метод `concat()` возвращает новый массив, состоящий из массива, на котором он был вызван, соединённого с другими массивами и/или значениями, переданными в качестве аргументов.**

`.every()` → **Метод `every()` проверяет, удовлетворяют ли все элементы массива условию, заданному в передаваемой функции.**

`.findIndex()` → **Метод `findIndex()` возвращает индекс в массиве, если элемент удовлетворяет условию проверяющей функции. В противном случае возвращается -1.**

Часто используемые методы массивов

`.includes()` → **Метод `includes()` определяет, содержит ли массив определённый элемент, возвращая в зависимости от этого `true` или `false`.**

`.join()` → **Метод `join()` объединяет все элементы массива в строку.**

`.reduce()` → **Метод `reduce()` применяет функцию reducer к каждому элементу массива (слева-направо), возвращая одно результирующее значение.**

Часто используемые методы массивов

`.some()` → **Метод `some()` проверяет, удовлетворяет ли какой-либо элемент массива условию, заданному в передаваемой функции.**

`.sort()` → **Метод `sort()` *на месте* сортирует элементы массива и возвращает отсортированный массив. Порядок сортировки по умолчанию соответствует порядку кодовых точек Unicode.**

`.reverse()` → **Метод `reverse()` на месте обращает порядок следования элементов массива. Первый элемент массива становится последним, а последний — первым.**

Деструктуризация

- **Деструктуризация просто подразумевает разбивку сложной структуры на простые части.**
- **В JavaScript, таковая сложная структура обычно является объектом или массивом.**
- **Используя синтаксис деструктуризации, вы можете выделить маленькие фрагменты из массивов или объектов. Такой синтаксис может быть использован для объявления переменных или их назначения.**

Деструктурируем массив

```
const rgb = [255, 200, 0];

const [red, green, blue] = rgb;

console.log(red);   // 255
console.log(green); // 200
console.log(blue);  // 0
```

Дефолтные значения и пропуск значений

```
const rgb = [255, 200];

const [ , green, blue = 155] = rgb;

console.log(green); // 200
console.log(blue);  // 155
```

Для самостоятельного чтения на learn.javascript.ru

Типы данных → **Главы 5.4 - 5.5**