Массивы

Мікалай Янкойць

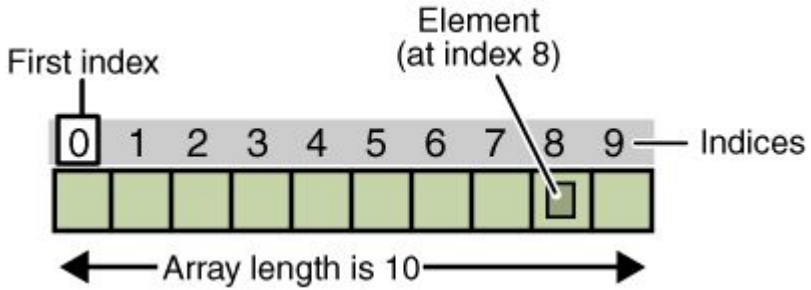
Адукар

Повестка дня

1. Что такое массив и как его создать
2. Базовые свойства и методы массивов
3. Простой поиск и сортировка

Что такое массивы

Массив – структура данных, которая позволяет хранить упорядоченный **набор** значений. Предназначен для работы со списками и коллекциями. Отдельные значения называются **элементами** массива, их количество – **длиной** массива.
Доступ к элементам можно получить по их номеру в наборе – **индексу**.



В JavaScript* индексы массивов начинаются с нуля.

*как и во многих других языках

Как создать массив в JS

Способ 1 – квадратные скобки (самый частый)

```
let arr = []; // пустой массив
let primeNums = [2, 3, 5, 7, 11, 13]; // массив из 6 элементов
let drinks = ["Кофе", "Чай", "Кола", "Вино"]; // массив длиной 4
```

Способ 2 – new Array() (есть свои преимущества)

```
let emptyArr = new Array(); // пустой массив
let fibNums = new Array(0, 1, 1, 2, 3, 5); // массив из 6 элементов
let audit2019 = new Array(365);
    // массив длиной 365, но все его элементы – undefined
```

Элементы массива

Доступ к элементам массива: имяМассива[индекс_элемента]

```
1  let cities = ["Брест", "Витебск", "Гродно", "Борисов", "Минск"];
2  console.log(cities[0]); // в консоли – "Брест"
3  console.log(cities[4]); // в консоли – "Минск"
```

Точно так же можно менять значения элементов (и создавать

```
5  cities[3] = "Гомель"; // заменили "Борисов"
6  cities[5] = "Могилёв"; // добавили элемент, длина массива теперь 6
```

В одном массиве могут храниться элементы разных типов:

```
8  let strangeArr = ["строка", 42, false];
```

Длина массива

Длину массива можно узнать с помощью его свойства length:

```
1  let summer = ["June", "July", "August"];
2  console.log(summer.length); // 3
```

Это свойство изменяется динамически:

```
4  let seasons = ["Winter", "Spring", "Summer"];
5  console.log(seasons.length); // 3
6  seasons[seasons.length] = "Fall"; // да, в [] может быть и выражение
7  console.log(seasons.length); // 4
```

Свойство length можно менять*:

```
9  seasons.length = 3; // последний элемент исчез. навсегда!
```

*но лучше этого не делать

Методы массивов: pop/push

Массивы часто используют как **стек** – упорядоченную коллекцию, элементы которой берутся и добавляются **с конца** (LIFO: last in, first out).

Для такого подхода у массивов есть **методы** push и pop.

pop() удаляет последний элемент массива (и возвращает его):

```
1  let names = ["Bill", "Bob", "Joe"];
2  console.log(names.pop()); // "Joe"; этот элемент удалён
3  console.log(names); // ["Bill", "Bob"]
```

push() добавляет элемент в конец массива:

```
4  names.push("Jack"); // добавили
5  console.log(names); // ["Bill", "Bob", "Jack"]
```

Методы массивов: shift/unshift

shift() удаляет первый элемент массива (и возвращает его):

```
let names = ["Jenny", "Zoe", "Mary"];
console.log(names.shift()); // в консоли – "Jenny"
    // элемент "Jenny" удалён, остальные – сдвинуты вперёд
console.log(names); // ["Zoe", "Mary"]
console.log(names[0]); // Zoe
```

unshift() добавляет элемент в начало массива:

```
names.unshift("Anne"); // добавили элемент в начало, сдвинули остальные
console.log(names); // ["Anne", "Zoe", "Mary"]
```

Ещё одно применение массивов – **очередь**: коллекция, обработка которой идёт с начала, а новые элементы добавляются в конец (FIFO: first in, first out). Очередь можно получить, комбинируя методы push и shift.

Методы массивов: очередь и стек

С помощью push и unshift в массив можно добавлять сразу несколько новых элементов:

```
1  let letters = ["c", "d", "e"];
2  letters.unshift("a", "b");
3  console.log(letters); // ["a", "b", "c", "d", "e"]
4  letters.push("f", "g");
5  console.log(letters); // ["a", "b", "c", "d", "e", "f", "g"]
```

Методы pop/push работают **значительно быстрее**, чем shift/unshift. Это связано с тем, что при работе с очередью после добавления/удаления нужно сдвинуть все элементы массива. При работе со стеком всё остаётся на своих местах.

Перебор элементов массива

Перебрать все элементы массива можно с помощью цикла:

```
1  let powersOfTwo = [1, 2, 4, 8, 16, 32];
2  for (let i = 0; i < powersOfTwo.length; i++)
3      console.log(powersOfTwo[i]); // выведет все элементы
```

Точно так же внутри цикла можно заполнять массив значениями:

```
5  let powersOfThree = [];
6  for (let i = 0; i < 5; i++)
7      powersOfThree[i] = 3 ** i; // добавляем в массив 5 эл-в
8  console.log(powersOfThree); // [1, 3, 9, 27, 81]
```

Многомерные массивы

Элементами массива могут быть другие массивы. Массив из массивов называют многомерным.

Доступ к элементам работает точно так же, как с одномерными массивами.

```
let twoDimensions = [
    [1, 2, 3],
    [2, 3, 1],
    [3, 1, 2]
];

console.log(twoDimensions[0][2]); // 3
console.log(twoDimensions[1][0]); // 2
```

Практика

1. Создайте массив и в цикле заполните его чётными числами от 2 до 20.
2. Преобразуйте массив из задачи 1 так, чтобы его элементы стали равны своему индексу, умноженному на 5.
3. Преобразуйте массив из задачи 2 так, чтобы его элементы расположились в обратном порядке.
4. Получите от пользователя три числа, создайте из них массив. Используя циклы, найдите наибольшее из чисел и разделите на него каждый элемент массива.

Методы массивов: join

Метод join «склеивает» все элементы массива в одну строку, используя свой единственный аргумент в качестве разделителя.

```
let asianCities = ["Tokyo", "Seoul", "Shanghai", "Karachi"];
let citiesStr = asianCities.join(", ");
console.log(citiesStr); // "Tokyo, Seoul, Shanghai, Karachi"
let citiesLine = asianCities.join("--");
console.log(citiesLine); // "Tokyo--Seoul--Shanghai--Karachi"
```

Если нужно, тип значений перед склейкой неявно приводится к String.

```
let strangeArr = ["строка", 42, false];
console.log(strangeArr.join(" ")); // "строка 42 false"
```

Создание массива из строки: split

split – метод, обратный join: он собирает массив из строки.

Обязательный первый аргумент указывает, по каким символам разбивать строку на части.

```
let europeCities = "Istanbul, Moscow, London, Berlin";
let citiesArr = europeCities.split(", ");
console.log(citiesArr); // ["Istanbul", "Moscow", "London", "Berlin"]
let vaderSay = "luke.i.am.your.father";
console.log(vaderSay.split("."));
    // ["luke", "i", "am", "your", "father"]
```

Если указать split второй аргумент, он ограничит длину итогового массива.

```
let nums = "1-2-3-4-5-6";
console.log(nums.split("-", 3)); // ["1", "2", "3"]
```

Методы массивов: slice

Метод slice(start, end) копирует часть массива от индекса start (включительно) до индекса end (не включая) и возвращает её в **новом** массиве. Исходный массив не меняется.

```
let letters = ["a", "b", "c", "d", "e", "f"];
let part = letters.slice(1, 4);
console.log(part); // ["b", "c", "d"]
console.log(letters); // ["a", "b", "c", "d", "e", "f"]
```

- если не указать end, массив копируется от start до конца
- если start < 0, элементы отсчитываются с конца массива

```
console.log(letters.slice(3)); // ["d", "e", "f"]
console.log(letters.slice(-4, 4)); // ["c", "d"]
    //то же, что slice(letters.length-4, 4)
```

Методы массивов: splice

Метод splice(start, deleteCount, elem1, ..., elemN):

- удаляет deleteCount элементов из массива
- начиная с индекса start
- вставляет на их место элементы elem1, ..., elemN
- возвращает массив из удалённых элементов

```
1  let digits = [1, 2, 3, 17, 25, 6];
2  let newDigits = digits.splice(3, 2, 4, 5);
3      // начиная с digits[3], удаляем 2 элемента
4      // и вставляем вместо них 4 и 5
5  console.log(digits); // [1, 2, 3, 4, 5, 6]
6  console.log(newDigits); // [17, 25]
```

Методы массивов: fill

Метод fill(value, start, end) заменяет значения всех элементов массива от start (включительно) до end (не включая) на значение value.

```
let languages = ["html", "css", "java", "c#", "js"];
languages.fill("js", 2, 4);
console.log(languages); // ["html", "css", "js", "js", "js"]
```

- если не указать end, заменяются элементы от start до конца
- если не указать ни end, ни start, значение value получат все элементы

```
let zerosAndOnes = [0, 0, 1, 0, 1, 0];
zerosAndOnes.fill(1, 3);
console.log(zerosAndOnes); // [0, 0, 1, 1, 1, 1]
let someArray = [4, 6, 1, 9, 2];
someArray.fill(3);
console.log(someArray); // [3, 3, 3, 3, 3]
```

Методы массивов: concat и reverse

Метод arr.concat(elem1, …, elemN) создаёт и возвращает новый массив, в который копируются все элементы из arr и все аргументы elem1 – elemN.

```
let domains = ["com", "org", "net"];
let moreDomains = domains.concat("biz", "info");
console.log(moreDomains); // ["com", "org", "net", "biz", "info"];
```

Если один из аргументов concat – массив, в результат включается не он сам, а все его элементы!

```
console.log(moreDomains.concat(["io", "by"]));
    // ["com", "org", "net", "biz", "info", "io", "by"];
```

Метод reverse возвращает массив с элементами в обратном порядке.

```
console.log([1,2,3,4,5].reverse()); // [5,4,3,2,1]
```

Методы массивов: indexOf / lastIndexOf

Метод arr.indexOf(search, start) ищет в массиве **первый** элемент search и возвращает его индекс или -1, если такой элемент не найден. Поиск начинается с индекса start (или с начала массива, если start не указан).

Элементы сравниваются с искомым через оператор ===.

```
let letters = ["a", "e", "a", "o", "u", "o"];
console.log(letters.indexOf("o")); // 3
console.log(letters.indexOf("y")); // -1
console.log(letters.indexOf("a", 2)); // 2
console.log(letters.indexOf("a", 3)); // -1
```

Метод lastIndexOf действует так же, но перебирает массив задом наперёд.

```
console.log(letters.lastIndexOf("o")); // 5
console.log(letters.lastIndexOf("o", 2)); // -1, "o" с индексами 3 и 5
```

Методы массивов: sort

Метод sort сортирует массив по возрастанию. При сортировке все элементы массива неявно приводятся к типу String.

```
1  let words = ["xyz", "abc", "sum", "lol"];
2  words.sort();
3  console.log(words); // ["abc", "lol", "sum", "xyz"]
4  let nums = [6, 12, 23, 2];
5  nums.sort();
6  console.log(nums); // [12, 2, 23, 6] – из-за приведения типов!
```

Внеклассное чтение

https://learn.javascript.ru/array

https://learn.javascript.ru/array-methods

Практика

1. Создайте массив из чисел от 1 до 35. Вырежьте из него первые 10 элементов, а затем добавьте их в конец массива. Разверните в обратном порядке элементы с 11 по 20. Удалите элементы с 21 по 25, на их место вставьте пять первых степеней двойки. Элементы с 26 по 30 замените на единицы. Элементы с 31 по 35 склейте в одну строку, разделяя пробелами, и замените на итоговую строку.
2. Напишите функцию, которая удаляет из массива повторяющиеся элементы и возвращает обновлённый массив.
3. Напишите функцию, удаляющую из массива все элементы, которые при приведении к типу Boolean дают false.