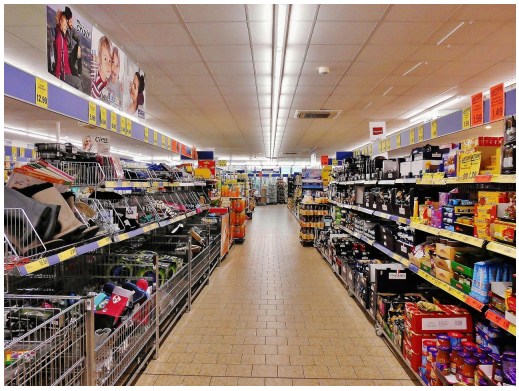
Массивы в JavaScript — введение в JavaScript

Давайте представим себе книжную полку на которой стоит множество книг. Или камеру хранения на вокзале со множеством ячеек. У каждой ячейки есть номер и в каждой из них может находиться чемодан.

В JavaScript тоже есть похожие "книжные полки" или "камеры хранения". Называются они массивами и в массиве мы можем хранить множество переменных. Мы можем хранить на одной полке только книги, можем вазы, можем там хранить стринги, числа или даже всё вместе.

Как в магазине на полке. У нас конфеты стоят рядом с чаем и печеньем.



И это прям вот реальный пример из программирования. Мы можем поместить все товары магазина в один массив. Нам никто не запрещает этого сделать.

Массив определение в JavaScript

Массив — это пронумерованная структура данных, в которой хранится конечное число элементов.

Один элемент - одна ячейка.

Давайте просто создадим уже массив и посмотрим код.

Создание массива

```
// создание пустого массива;
let arr1 = [];
// второй способ создания пустого массива.
let arr2 = new Array();
// Создание массива с элементами, числами
let arr3 = [0, 1, 1, 2, 3, 5, 8, 13];
let arr4 = new Array(21, 34, 55);
// Создание массива с элементами, булевые элементы
let arr5 = [true, false, true, false];
//создание смешанного массива
let arr6 = [true, 1, "Школа", 42, false, "JavaScript"];
//создание массива со строками
let arr7 = ["дом", "улица", "фонарь", "аптека"];
```

Мы можем объявлять массив на нескольких строчках. Иногда так лучше видно код.

```
let germanCities = [
    "Берлин",
    "Дрезден",
    "Кёльн",
    "Франкфурт",
    "Мюнхен",
];
```

Прошу обратить внимание на последнюю запятую, она никому не мешает и не запрещена. В некоторых случаях добавления новых элементов она полезна. С последним массивом **germanCities** мы и "поиграем".

```
console.log(germanCities[0]);
```

Мы можем обратиться к каждому элементу массива обратившись к массиву по имени и указав в квадратных скобках его порядковый номер. Нумерация элементов начинается с ноля. Мы можем обратиться и к последнему элементу, потому что JavaScript даёт нам возможность всегда знать длину массиву.

```
let arraySize = germanCities.length;
console.log(germanCities[arraySize - 1]);
```

Если в массиве 5 элементов, то длина массива 5. Если при этом элементы пронумерованы с нуля, то последний элемент стоит на четвёртом месте. **4 = 5 - 1;**

Конечно же мы можем обратиться к последнему элементу не прибегая к промежуточной переменной.

```
console.log(germanCities[germanCities.length - 1]);
```

Мы можем вывести на экран весь массив целиком. **console.log()** умеет работать и с массивами.

```
console.log(germanCities)
```

Замена элемента массива, update

Если нам надо заменить элемент на новый, то мы можем это сделать просто обратившись к порядковому номеру.

```
germanCities[4] = "NewCity";
```

Предыдущий элемент будет безвозвратно утрачен. У нас пока нет шестого элемента и пятое место в массиве свободно. Давайте попробуем на его место записать новый город.

Добавление нового элемента в массив, create

```
germanCities[5] = "BigCity";
```

Наш массив увеличился на один элемент. Если вы укажите номер порядкового элемента превышающий последний уже имеющийся, то вы создадите в промежутки пустые элементы, в которых ничего нет.

```
germanCities[7] = "TestCity";
console.log(germanCities)
```

Одна из причин почему не рекомендуется использовать этот метод в том, что при ошибке мы можем безвозвратно потерять элемент или создать пустой. На самом деле неправильное обращение по индексу очень распространённая ошибка начинающих программистов. По этому давайте сегодня рассмотрим безопасные и рекомендованные методы работы с массивами в JavaScript.

Для безопасного добавления элемента в массив есть специальный метод **push()**

push

```
let belarusCities = ["Минск", "Могилёв", "Мозырь"]
belarusCities.push("Мосты");
console.log(belarusCities); // [ 'Минск', 'Могилёв', 'Мозырь', 'Мосты' ]
```

Метод push() добавляет новый элемент в конец массив.

Если мы хотим добавить элемент в начало массива, то мы можем воспользоваться методом **unshift()**

unshift

```
let belarusCities = ["Минск", "Могилёв", "Мозырь"]
belarusCities.unshift("Мосты");
console.log(belarusCities); // [ 'Мосты', 'Минск', 'Могилёв', 'Мозырь' ]
```

Это намного более дорогая операция. Потому что под капотом мы обработали все элементы массива. На больших объёмах данных разница в "добавить просто в конце массива" и "поменять порядок всех элементов" может быть очень существенная.

Удаление элементов из массива очень важная операция. Представьте себе, что ваш клиент ходит по вашему виртуальному магазину и добавляет товары в корзинку. По сути мы именно это мы и можем сделать с помощью метода **push** или ** unshift**.

Удаление элемента из массива, delete

Если мы смогли добавить, то можем и удалить элементы из массива. Для этого у нас тоже есть два метода: pop и shift.

pop

Всё достаточно предсказуемо. Нам ничего не надо указывать. Просто вызвать метод.

```
let belarusCities = ["Минск", "Могилёв", "Мозырь"]
belarusCities.pop(); // удаляется просто последний элемент
console.log(belarusCities); // [ 'Минск', 'Могилёв' ]
```

shift

Здесь всё один в один, только удаляется первый элемент.

```
let belarusCities = ["Минск", "Могилёв", "Мозырь"]
belarusCities.pop(); // удаляется ПЕРВЫЙ элемент
console.log(belarusCities); // [ 'Могилёв', 'Мозырь' ]
```

length и создание пустого массива фиксированной длины

У массива всегда есть длина. В любой момент времени мы знаем сколько там элементов. Или можем узнать:

```
let belarusCities = ["Минск", "Могилёв", "Мозырь"]
console.log(belarusCities.length); // 3
```

У пустого массива тоже есть длина:

```
let simpleArr = [];
console.log(simpleArr.length) //0
```

И есть не совсем понятный код:

```
let magicArr = new Array(10);
console.log(magicArr.length) //10
```

Длина этого магического массива 10. Если мы передаём в круглые скобки одно единственное число, то JS понимает это как длину массива. И создаёт пустые элементы.

```
console.log(magicArr2); // [ <10 empty items> ]
console.log(magicArr2[0]); // undefined
```

Теперь мы можем заполнять этот массив элементами.
Это свойство массива иногда используется в работе.

Примеры использования

```
let myFirstArray = [1965, 1975, 1985, 1990, 1997, 2003];

function calcAge(birthYear) {
    let now = 2021;
```

```
    return now - birthYear;
}

console.log(calcAge(myFirstArray[3]));
console.log(calcAge(myFirstArray[myFirstArray.length - 1]));
```

Полезные материалы

1. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
2. https://habr.com/ru/company/plarium/blog/483958/
3. https://habr.com/ru/company/ruvds/blog/430380/
4. https://habr.com/ru/company/plarium/blog/446902/
5. https://habr.com/ru/company/ruvds/blog/358306/
6. https://habr.com/ru/company/ruvds/blog/413169/
7. https://habr.com/ru/company/ruvds/blog/422091/
8. https://learn.javascript.ru/array