

# МАССИВЫ И БАЗОВЫЕ МЕТОДЫ РАБОТЫ С НИМИ




**АЛЕНА БАТИЦКАЯ**



**АЛЕНА БАТИЦКАЯ**

 [ABatickaya](#)

 [@alenabat](#)



# ПЛАН ЗАНЯТИЯ

1. [Повторение – мать учения](#)
2. [Что же такое массивы?](#)
3. [Основы основ](#)
4. [Методы работы с массивами](#)
5. [Где попрактиковаться?](#)



# ВСПОМНИМ ПРОШЛЫЙ МАТЕРИАЛ

# ВСПОМНИМ ПРОШЛЫЙ МАТЕРИАЛ

Что мы получим в результате исполнения такого кода?

```
1 let duration = 1;
2 let myString = '';
3
4 myString += 'внимание!'.toUpperCase();
5 myString += ' Проверяем усвоенный ';
6 myString += duration;
7 myString += ' неделю назад материал! Вых!'.toLowerCase().substr(0, 22);
8
9 console.log(myString);
```

## ЧТО МЫ УВИДИМ В РЕЗУЛЬТАТЕ?

Мы увидим в консоли сообщение:

```
'ВНИМАНИЕ! проверяем усвоенный 1 неделю назад материал!'
```

Мы инициализировали переменную `myString` пустой строкой, а далее добавляли по-разному отформатированные строки в нее. Затем с помощью `substr` вырезали часть строки и вывели результат в консоль.

---

# ЧТО ЖЕ ТАКОЕ МАССИВЫ?



# ЗАЧЕМ НАМ МАССИВЫ?

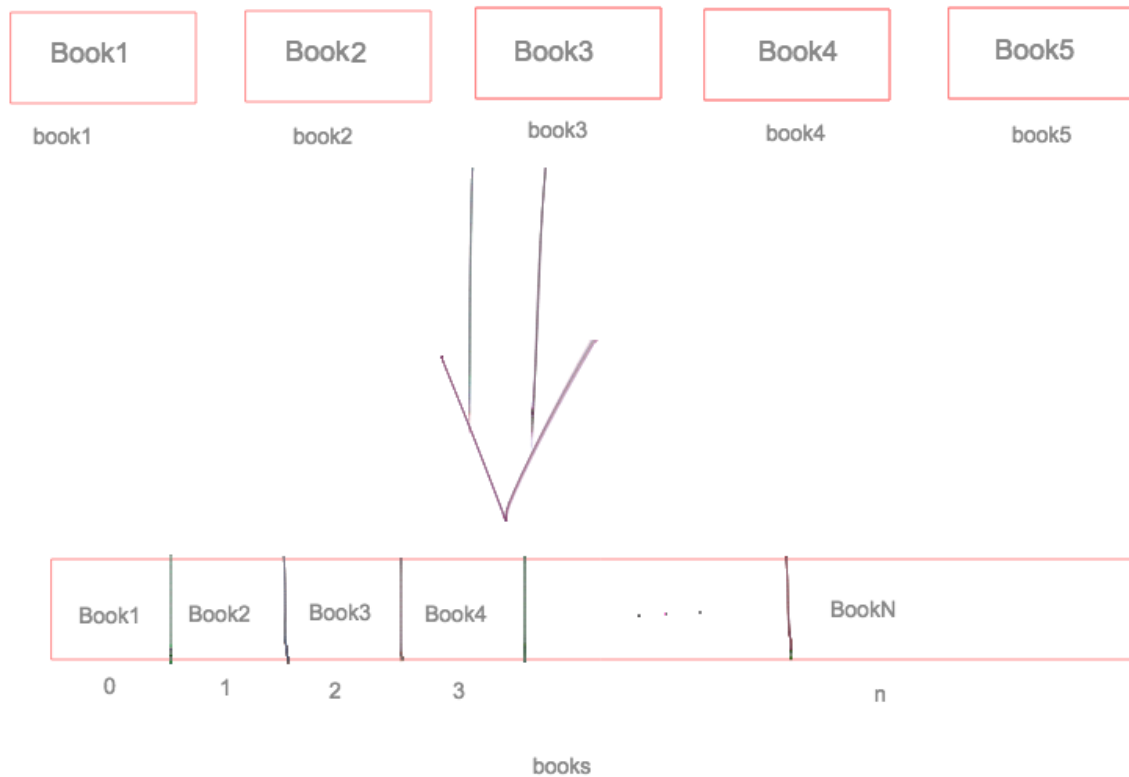
Как правило, в приложениях мы имеем много (а иногда даже очень много) данных.

Предположим, что у нас в магазине есть 100 разных книг. Один из вариантов — хранить имя каждой в своей переменной.



# ЗАЧЕМ НАМ МАССИВЫ?

Но гораздо проще объединить их всех в какую-то общую переменную. В этом нам и помогут массивы:





# ОСНОВЫ ОСНОВ

# НУМЕРАЦИЯ ЭЛЕМЕНТОВ

У каждой «единицы» данных в массиве есть свой порядковый номер. Он называется индексом. *Нумерация индексов начинается с нуля.*

```
1 // это может по началу быть непривычным, но в данном примере
2 // "первый" будет иметь индекс 0, "второй" – индекс 1, а "третий" – 2
3
4 let arr = ["первый", "второй", "третий"];
```

# ВЗБОЛТАТЬ, А НЕ СМЕШИВАТЬ

Во многих языках программирования данные в массивах должны быть одного типа и размер часто задается заранее. В JS все намного веселее — тип данных может быть любой и в любой момент мы можем добавить сколько угодно данных в существующий массив.

```
1 let array = [1, 2, 3, 'четыре', true, false];  
2 array[6] = 'еще один элемент';  
3  
4 console.log(array);
```

# В НАЧАЛЕ ~~БЫЛО СЛОВО~~ БЫЛ МАССИВ

Что нам стоит знать для начала:

- массив объявляется с помощью специального литерала — квадратных скобок (`[]`);
- при объявлении вы можете оставить его пустым и добавить данные позже, а можете и сразу инициализировать какими-то значениями (`[1, 2, 4]`, `['Маша', 'Паша']`).

```
1 let arr = [1, 2, 3];
2 let arr2 = [];
3 arr2[0] = 'Маша'; // про этот трюк мы еще поговорим
4
5 console.log(arr, arr2);
```

# ЧТЕНИЕ

Чтобы прочесть элемент из массива, нужно указать его индекс в квадратных скобках после имени массива:

```
1 | let arr = ['первый', 'второй', 'третий'];  
2 |  
3 | console.log(arr[0], arr[1], arr[2]); // 'первый', 'второй', 'третий'
```

Это как раз часть того самого «трюка», о котором мы только что говорили — `arr2[0]`.

## ...И ЗАПИСЬ

А вот и вторая часть «трюка». Вы можете не только читать, но и изменять любой элемент массива по его индексу:

```
1  let arr = [1, 2, 3];
2  arr[0] = 33;
3
4  console.log(arr); // [33, 2, 3]
5  console.log(arr[1]); // 2
6  // а вот и новый трюк — но можно догадаться, что
7  // таким образом мы получаем длину массива
8  console.log(arr.length); // 3
```

## МОЖНО ЛИ ТАК?

```
1 let array = ["привет", "Здравствуйте", "Добрый вечер"];
2 let arrayTwo = [1, 2, 3];
3
4 array[2] = arrayTwo;
5
6 console.log(array); // ???
```



# КОНЕЧНО МОЖНО!

Мы получили массив, одним из элементов которого является другой массив.

```
1 let array = ["привет", "Здравствуйтесь", "Добрый вечер"];
2 let arrayTwo = [1, 2, 3];
3
4 array[2] = arrayTwo;
5
6 console.log(array); // ["привет", "Здравствуйтесь", [1, 2, 3]]
```

## ДОСТУП К ЭЛЕМЕНТАМ ВЛОЖЕННОГО МАССИВА

Для доступа к элементу вложенного массива нам потребуется два индекса: первый, чтобы добраться до вложенного массива и второй, чтобы получить из него элемент:

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



# МЕТОДЫ РАБОТЫ С МАССИВАМИ

# ПРО СВОЙСТВА И МЕТОДЫ

Все значения в JavaScript, за исключением `null` и `undefined`, содержат набор вспомогательных функций и значений, доступных «через точку».

Такие функции называют «методами», а значения – «свойствами».

Свойства говорят о текущем состоянии массива, а методы выполняют наши команды.

Именно поэтому мы смогли так легко узнать длину нашего массива (`arr.length`). Если помните, у строк тоже есть такой метод-помощник.

## ЕЩЕ ОДИН ПОМОЩНИК

Иногда мы знаем элемент, но не знаем его индекс. Найти индекс можно с помощью метода `indexOf`. Важный нюанс: если одинаковых элементов в массиве несколько, то по умолчанию найдется только первый.

```
1 let arr = [1, 2, 3];  
2  
3 // получим 1 — не забываем, что нумерация начинается с нуля  
4 console.log(arr.indexOf(2));
```



# ИЗМЕНЯЕМ, ДОБАВЛЯЕМ, УБИРАЕМ

Рассмотрим основные методы, которые помогают нам манипулировать массивами, не обращаясь непосредственно к индексам.

Работать с методами удобно, когда нам нужно часто и много изменять изначальный массив.

# ИЗМЕНЯЕМ, ДОБАВЛЯЕМ, УБИРАЕМ

```
1  let books = ['Книга 1', 'Книга 2', 'Книга 3'];
2
3  /** допустим нам нужно добавить 100 книг в наш массив */
4
5  // с помощью конструкции "books[books.length] = any;"
6  // мы добавляем элемент в конец массива
7  books[books.length] = 'Книга 1';
8  books[books.length] = 'Книга 2';
9  books[books.length] = 'Книга 3';
10 // ...
11 books[99] = 'Книга 100';
```

Задачу мы решили, но слишком уж сложно, не находите?

# МЕТОДЫ – НАШИ ВЕРНЫЕ СОЛДАТЫ

Представьте, что массив — это генерал вашей армии, а обилие всех его методов — это подвластное ему войско. При этом каждый метод — это солдат со своими уникальными способностями.

Так вот. Каждый метод-солдат делает что-то с данными и докладывает результат своей работы (если говорить скучным техническим языком — возвращает какое-то значение). Его можно сохранить в отдельную переменную и переиспользовать далее в программе, а можно проигнорировать.

Как работают методы/функции мы еще разберем более подробно в следующих лекциях, но сейчас важно понять, что возвращаемый результат и модификации массива — это две разные вещи, происходящие «под капотом».



# PUSH

Добавляет элемент(ы) в конец массива, возвращает текущую длину массива:

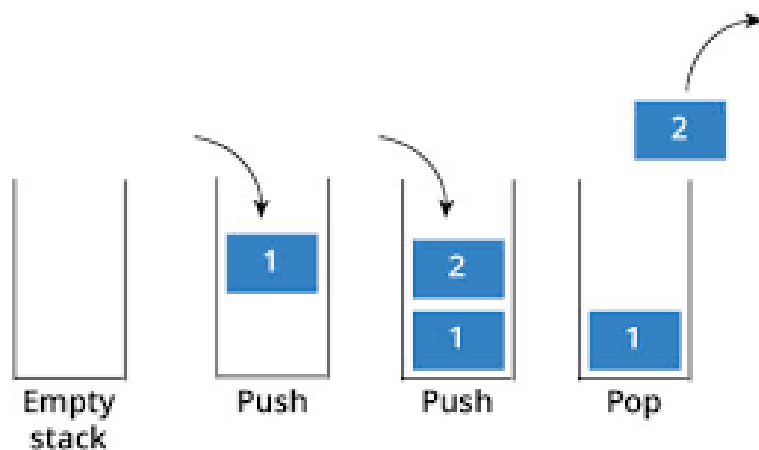
```
1  let arr = [];  
2  
3  arr.push(1, 2, 3); // 3  
4  arr.push(11); // 4  
5  console.log(arr); // [1, 2, 3, 11]  
6  console.log(arr.length); // 4
```

# POP

Удаляет элемент с конца массива, возвращает этот самый элемент:

```
1  let arr = [1, 2, 3, 11];  
2  
3  arr.pop(); // 11  
4  console.log(arr); // [1, 2, 3]  
5  console.log(arr.length); // 3
```

# И СНОВА НЕМНОГО ВИЗУАЛИЗАЦИИ



# SHIFT

Удаляет элемент из начала массива, возвращает этот самый элемент:

```
1  let arr = [1, 2, 3];  
2  
3  arr.shift(); // 1  
4  console.log(arr); // [2, 3]  
5  console.log(arr.length); // 2
```

# UNSHIFT

Добавляет элемент(ы) в начало массива, возвращает текущую длину массива:

```
1  let arr = [2, 3];  
2  
3  arr.unshift(9, 16, 25); // 5  
4  arr.unshift(36); // 6  
5  console.log(arr); // [36, 9, 16, 25, 2, 3]  
6  console.log(arr.length); // 6
```

# СТРОКИ, ИЛИ ТУДА И ОБРАТНО

Часто возникает необходимость превратить строку в массив или наоборот — склеить массив в строку. Для этого у нас есть помощники, которыми легко пользоваться:

— `join`

объединяет элементы массива в строку с помощью переданного в качестве параметра разделителя. По умолчанию этим разделителем является запятая;

— `split`

тут все то же самое — только в обратную сторону. Берем строку, говорим, что мы хотим разделить ее, скажем по пробелам — получаем массив слов.

```
1 | let arr = ['Мы', 'хотим', 'быть', 'одним', 'предложением'];  
2 |  
3 | arr.join(' '); // "Мы хотим быть одним предложением"  
4 | 'банан, клубника, молоко'.split(','); // ["банан", "клубника", "молоко"]
```

# ОБРЕЗАЕМ, ОБЪЕДИНЯЕМ, КОПИРУЕМ

Часто нам бывает нужно сделать копию данных или же получить какую-то часть исходного массива, не вызывая `pop/shift` много-много раз. В этом нам помогут следующие методы:

- `slice`;
- `splice`;
- `concat`.

Давайте посмотрим на примерах как это работает.

# SLICE

Метод `slice(begin, end)` копирует участок массива от `begin` до `end`, не включая `end`. Исходный массив при этом не меняется. Если не указать `end`, то копируем до конца массива.

```
1 let arr = ['Почему', 'не', 'надо', 'учить', 'JavaScript'];
2 let arr2 = arr.slice(0, 1); // только 1-ый элемент
3 let arr3 = arr.slice(2); // все элементы, начиная со второго
4
5 console.log(arr2, arr3); // ['Почему'], ['надо', 'учить', 'JavaScript']
```



# SPLICE

Метод `splice` – это универсальный раскладной нож для работы с массивами. Умеет все: удалять элементы, вставлять элементы, заменять элементы – по очереди и одновременно.

Пока достаточно освоиться с удалением.

```
1 let arr = ['Я', 'изучил', 'изучаю', 'JavaScript'];
2
3 arr.splice(1, 1); // начиная с позиции 1, удалить 1 элемент
4
5 alert( arr ); // осталось ['Я', 'изучаю', 'JavaScript']
```

# CONCAT

Метод `arr.concat(value1, value2, ... valueN)` создаёт новый массив, в который копируются элементы из `arr`, а также `value1`, `value2`, ... `valueN`.

```
1 let arr = ['Почему'];
2 let merged = arr.concat(['надо', 'учить', 'JavaScript']);
3
4 // помните метод join из лекции про строки?
5 // в результате получим 'Почему надо учить JavaScript'
6 console.log(merged.join(' '));
```

---

## ЧЕМУ МЫ НАУЧИЛИСЬ?

1. Узнали что же такое массивы и их особенности в JS;
2. Узнали как оперировать сразу целым набором данных или тем или другим элементом в массиве;
3. Стали на шаг ближе к JS ниндзя.

---

## РАДУЕМСЯ!



# ДОМАШНЕЕ ЗАДАНИЕ

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задаем в группе Facebook!
- Задачи можно сдавать по частям.
- Зачет по домашней работе проставляется после того, как приняты все **3 задачи**.

# ГДЕ И КАК МОЖНО ПОИГРАТЬСЯ С МАССИВАМИ

- [Codewars](#), programming challenges;
- [Массивы с числовыми индексами](#), `learn.javascript.ru`;
- [Массивы: методы](#), `learn.javascript.ru`;
- [Массивы: перебирающие методы](#), `learn.javascript.ru`.



# ЧТО ПОЧИТАТЬ

- [https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global\\_Objects/Array](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Array)  
- по традиции — документация на MDN;
- <https://learn.javascript.ru/array> — `learn.javascript.ru`, опять же по традиции.



## А ПОСМОТРЕТЬ?

- <https://www.youtube.com/watch?v=3OjuRfR8RNg> — отличный обзор того, что мы сегодня обсудили (и даже больше) от Sorax;
- <https://www.youtube.com/watch?v=orAS-MBh5f4> — хороший обзор на английском.



Спасибо за внимание! Время задавать вопросы

**АЛЕНА БАТИЦКАЯ**

