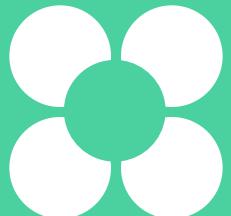


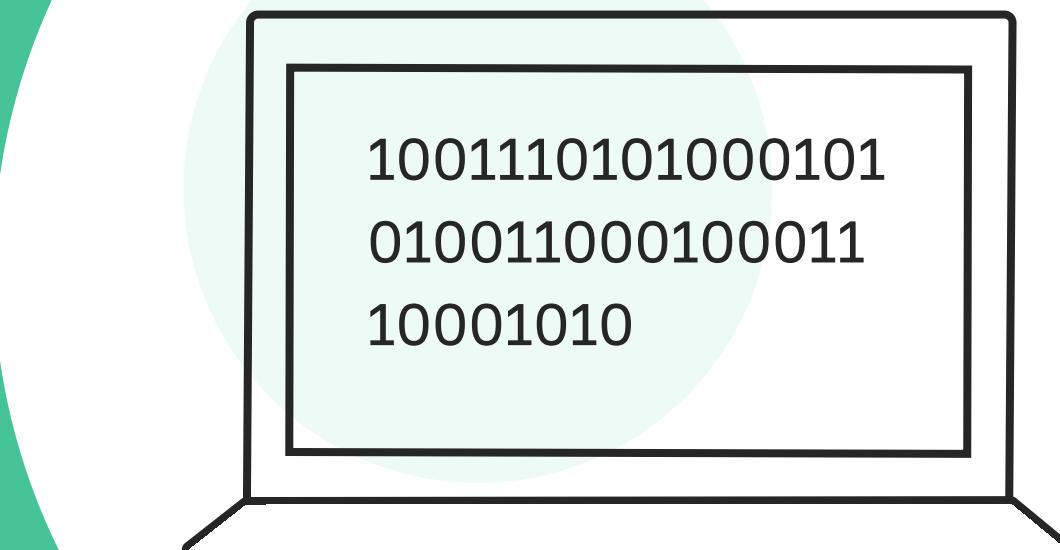
# Массивы и базовые методы работы с ними



# План занятия

- 1 Что такое массивы
- 2 Основы основ
- 3 Методы работы с массивами

# Что такое массивы



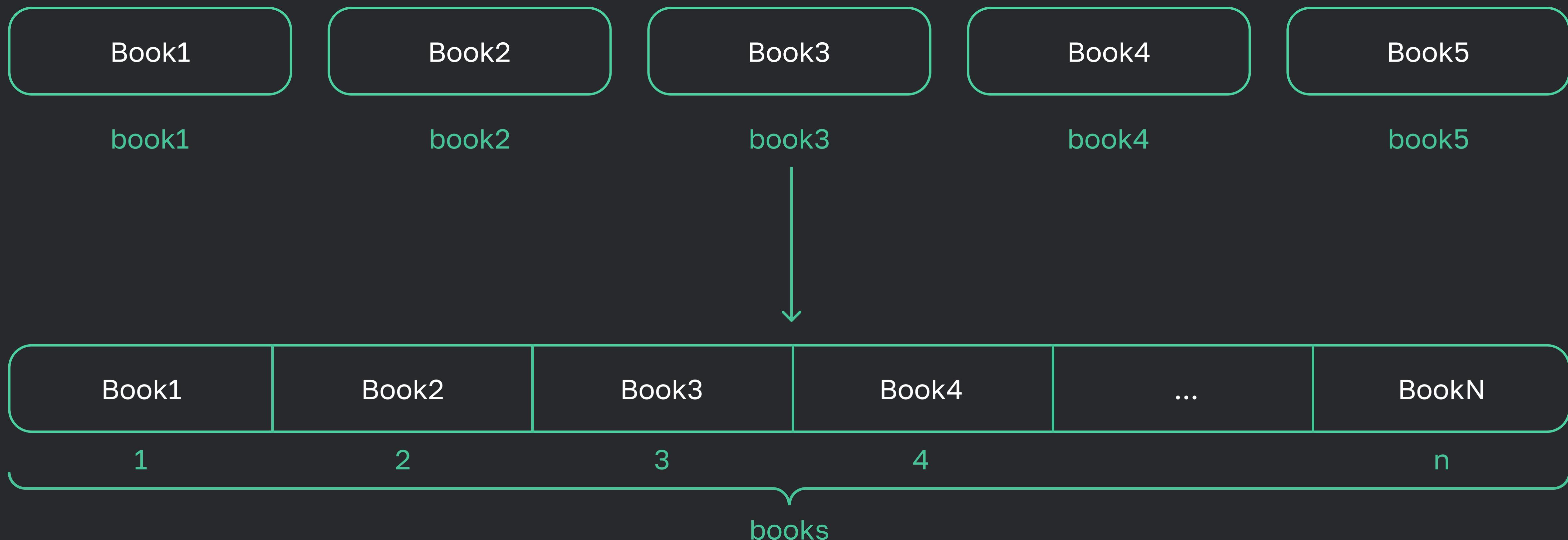
# Зачем нам массивы

В приложениях мы имеем очень много данных.

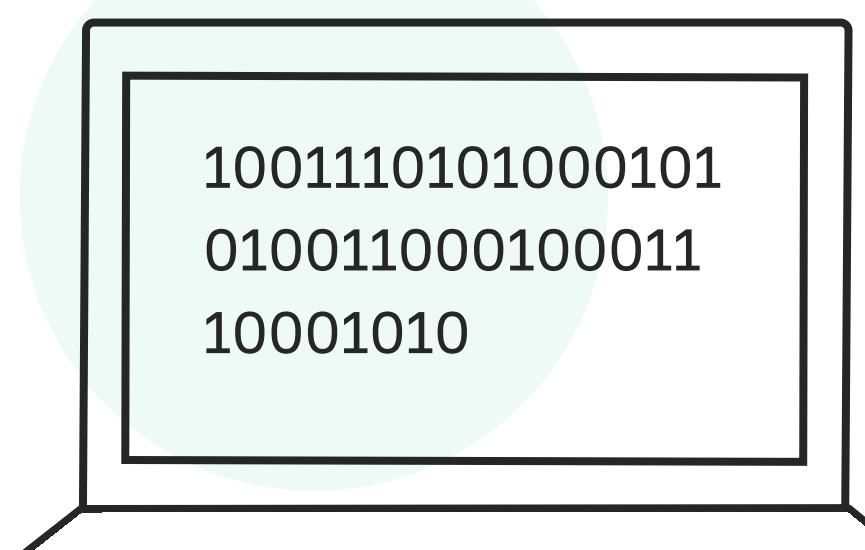
Предположим, что у нас в магазине есть 100 разных книг. Один из вариантов – хранить название каждой книги в своей переменной

# Зачем нам массивы

Однако гораздо проще объединить названия книг в общую переменную. В этом нам и помогут массивы



# Основы основ



# Индекс

У каждой единицы данных в массиве есть свой порядковый номер – индекс. **Нумерация индексов начинается с нуля:**

```
1 // Это может поначалу 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, но не смешивать

В JS тип данных может быть любым. 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] = 'ещё один элемент';
3
4 console.log(array);
```

# Массив

С чего начать:

- Массив объявляется с помощью квадратных скобок []
- При объявлении массива можно оставить его пустым и добавить данные позже, а можно сразу инициализировать его значениями ([1, 2, 4] , ['Маша', 'Паша'] )

```
1 let arr = [1, 2, 3];
2 let arr2 = [];
3 arr2[0] = 'Маша';
4
5 console.log(arr, arr2);
```

# Чтение элемента

Чтобы прочитать 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'
```

# Изменение элемента

Можно 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
```

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

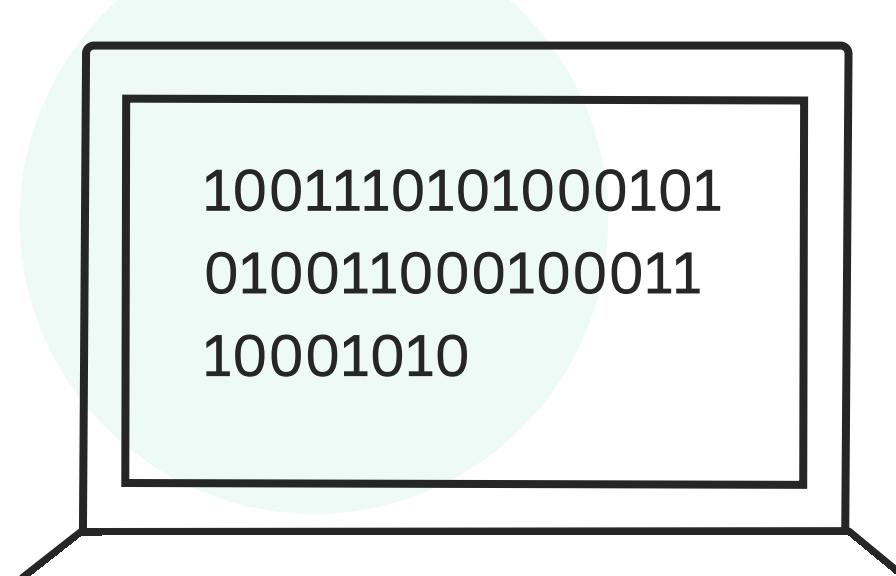
# Доступ к элементам вложенного массива

Для доступа к элементу вложенного массива необходимы два индекса:

- Первый, чтобы добраться до вложенного массива
- Второй, чтобы получить из него нужный элемент

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

# Методы работы с массивами



# Про свойства и методы

Почти все значения в JavaScript содержат вспомогательные функции и значения, доступные через точку.

Такие функции называют методами, а значения – свойствами.

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

# Ещё один помощник

Иногда мы знаем элемент, но не знаем его индекс. Найти индекс можно с помощью метода `indexOf`

# Ещё один помощник

Иногда мы знаем элемент, но не знаем его индекс. Найти индекс можно с помощью метода `indexOf`.

Нюансы:

- Если одинаковых элементов в массиве несколько, найдётся только первый
- Если элемент не найден, `indexOf` вернёт число `-1`

```
1 let arr = [1, 2, 3];
2
3 console.log(arr.indexOf(2));
4 // Получим 1 – не забываем, что нумерация начинается с нуля
5 console.log(arr.indexOf(99));
6 // Напечатает -1, потому что числа 99 в массиве нет
```

# 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']; 1
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

# Методы – наши верные солдаты

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

Каждый метод-солдат делает что-то с данными и докладывает результат своей работы, то есть возвращает какое-то значение. Его можно сохранить в отдельную переменную и использовать далее в программе, а можно проигнорировать.

Работу методов и функций мы разберём подробнее в следующих лекциях, но сейчас важно понять, что возвращаемый результат и модификации массива – это две разные вещи, происходящие «под капотом»

# push

Добавляет элементы в конец массива, возвращает текущую длину массива:

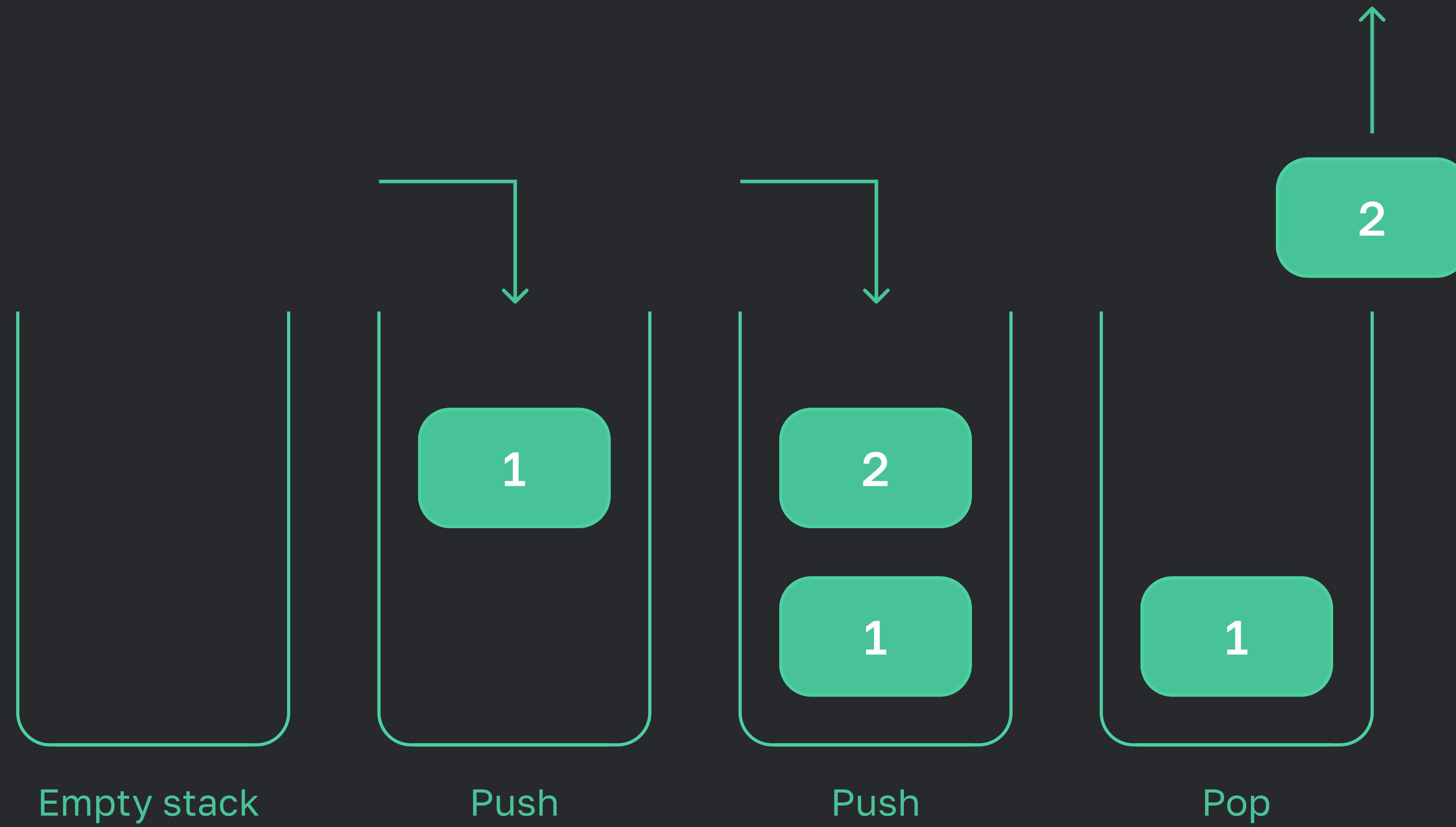
```
1 let arr = [];
2
3 arr.push(1, 2, 3); // 3
4 arr.push(11); // 4
5 console.log(arr); // [1, 2, 3, 11]
6 console.log(arr.length); // 4
```

# 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 немножко 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 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

## join

Объединяет элементы массива в строку с помощью переданного в качестве параметра разделителя, по умолчанию этим разделителем является запятая

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

# Строки, или туда и обратно

## join

Объединяет элементы массива в строку с помощью переданного в качестве параметра разделителя, по умолчанию этим разделителем является запятая

## split

Тут всё то же самое, только в обратную сторону. Берём строку, говорим, что хотим разделить её, например, 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

- slice
- splice
- concat

# slice

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

```
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` – универсальный метод работы с массивами. Эта функция может удалять элементы, вставлять элементы, заменять элементы по очереди и одновременно:

```
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 // в результате получим 'Почему надо учить JavaScript'
5 console.log(merged.join(' '));
```

# Чему мы научились

- Узнали, что такое массивы и каковы их особенности в JS
- Узнали, как оперировать сразу целым набором данных, тем или другим элементом в массиве
- Стали на шаг ближе к JS-ниндзя

# Как и где можно поиграть с массивами

- <https://www.codewars.com/>, 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/Learn/JavaScript/Objects/Basics>  
– по традиции документация на MDN
- <https://learn.javascript.ru/array> – learn.javascript.ru,  
опять же по традиции

# Что посмотреть

- <https://www.youtube.com/watch?v=3OjuRfR8RNq> – 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!

