JS: Arrays

AIT
Hi-tech team

НАШИ ПРАВИЛА

Включенная камера

Вопросы по поднятой руке

Не перебиваем друг друга

Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору

Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

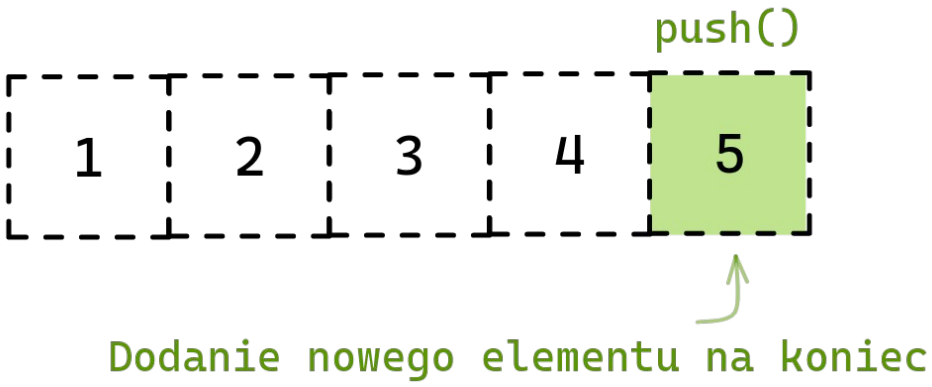
ЦЕЛЬ

Изучить особенности и синтаксис работы с массивами в js, методы push, pull, shift, unshift, виды циклов

ПЛАН ЗАНЯТИЯ

- Массивы: доступ к элементам, длинна, добавление, удаление
- Циклы: for, while, do while

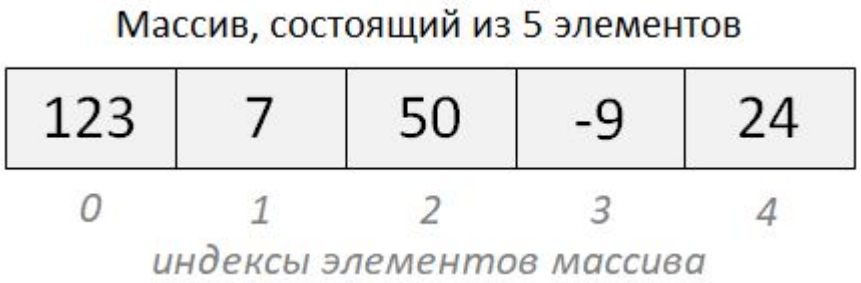
Массивы



Довольно часто мы понимаем, что нам необходима упорядоченная коллекция данных, в которой присутствуют 1-й, 2-й, 3-й элементы и т.д.

Например, она понадобится нам для хранения списка чего-либо: пользователей, товаров, элементов HTML и т.д.

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



Объявление

Синтаксис для создания пустого массива:

```
2  let arr = [];
```

Практически всегда используется второй вариант синтаксиса. В скобках мы можем указать начальные значения элементов:

```
1  let fruits = ["Яблоко", "Апельсин", "Слива"];
```

Объявление

Элементы массива нумеруются, начиная с нуля.

Мы можем получить элемент, указав его номер в квадратных скобках:

```
1  let fruits = ["Яблоко", "Апельсин", "Слива"];
2
3  alert( fruits[0] ); // Яблоко
4  alert( fruits[1] ); // Апельсин
5  alert( fruits[2] ); // Слива
```

Мы можем заменить элемент:

```
1  fruits[2] = 'Груша'; // теперь ["Яблоко", "Апельсин", "Груша"]
```

…Или добавить новый к существующему массиву:

```
1  fruits[3] = 'Лимон'; // теперь ["Яблоко", "Апельсин", "Груша", "Лимон"]
```

Длина массива

Общее число элементов массива содержится в его свойстве length:

```
1  let fruits = ["Яблоко", "Апельсин", "Слива"];
2
3  alert( fruits.length ); // 3
```

Методы pop/push, shift/unshift

Очередь – один из самых распространённых вариантов применения массива. В области компьютерных наук так называется упорядоченная коллекция элементов, поддерживающая два вида операций:

- push добавляет элемент в конец.
- shift удаляет элемент в начале, сдвигая очередь, так что второй элемент становится первым.

На практике необходимость в этом возникает очень часто. Например, очередь сообщений, которые надо показать на экране.

Методы pop/push, shift/unshift

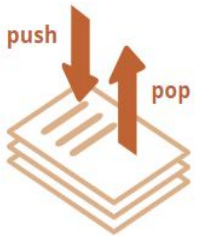
Существует и другой вариант применения для массивов – структура данных, называемая стек.

Она поддерживает два вида операций:

- push добавляет элемент в конец.
- pop удаляет последний элемент.

Таким образом, новые элементы всегда добавляются или удаляются из «конца».

Примером стека обычно служит колода карт: новые карты кладутся наверх и берутся тоже сверху:



Методы, работающие с концом массива:

pop

Удаляет последний элемент из массива и возвращает его:

```
1  let fruits = ["Яблоко", "Апельсин", "Груша"];
2
3  alert( fruits.pop() ); // удаляем "Груша" и выводим его
4
5  alert( fruits ); // Яблоко, Апельсин
```

Методы, работающие с концом массива:

push

Добавляет элемент в конец массива:

```
let fruits = ["Яблоко", "Апельсин"];

fruits.push("Груша");

alert( fruits ); // Яблоко, Апельсин, Груша
```

Методы, работающие с началом массива:

shift

Удаляет из массива первый элемент и возвращает его:

```
1  let fruits = ["Яблоко", "Апельсин", "Груша"];
2
3  alert( fruits.shift() ); // удаляем Яблоко и выводим его
4
5  alert( fruits ); // Апельсин, Груша
```

Методы, работающие с началом массива:

unshift

Добавляет элемент в начало массива:

```
1  let fruits = ["Апельсин", "Груша"];
2
3  fruits.unshift('Яблоко');
4
5  alert( fruits ); // Яблоко, Апельсин, Груша
```

Методы push и unshift могут добавлять сразу несколько элементов:

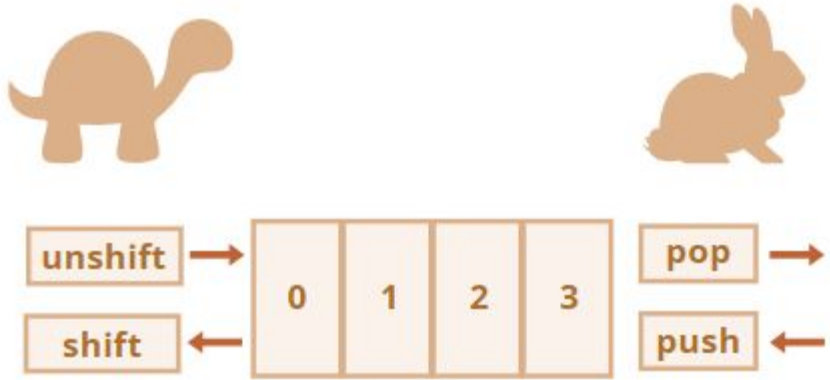
```
let fruits = ["Яблоко"];

fruits.push("Апельсин", "Груша");
fruits.unshift("Ананас", "Лимон");

// ["Ананас", "Лимон", "Яблоко", "Апельсин", "Груша"]
alert( fruits );
```

Эффективность

Методы `push/pop` выполняются быстро, а методы `shift/unshift` – медленно.



Циклы

Цикл “for”

Более сложный, но при этом самый распространённый цикл — цикл for.

```
1  for (начало; условие; шаг) {
2    // ... тело цикла ...
3  }
```

Цикл “for”

Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет alert(i) для i от 0 до (но не включая) 3:

```
1 for (let i = 0; i < 3; i++) { // выведет 0, затем 1, затем 2
2   alert(i);
3 }
```

Перебор элементов массива - for

Одним из самых старых способов перебора элементов массива является цикл for по цифровым индексам:

```
1  let arr = ["Яблоко", "Апельсин", "Груша"];
2
3  for (let i = 0; i < arr.length; i++) {
4    alert( arr[i] );
5  }
```

Цикл while

Цикл while имеет следующий синтаксис:

```
while (condition) {
  // код
  // также называемый "телом цикла"
}
```

Цикл while

Код из тела цикла выполняется, пока условие condition истинно.

Например, цикл ниже выводит i, пока i < 3:

```
let i = 0;
while (i < 3) { // выводит 0, затем 1, затем 2
  alert( i );
  i++;
}
```

Цикл while

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

```
let i = 3;
while (i) { // когда i будет равно 0, условие станет ложным
  alert( i );
  i--;
}
```

Цикл “do…while”

Проверку условия можно разместить под телом цикла, используя специальный синтаксис do..while:

```
do {
  // тело цикла
} while (condition);
```

Цикл “do…while”

Цикл сначала выполнит тело, а затем проверит условие condition, и пока его значение равно true, он будет выполняться снова и снова.

```
let i = 0;
do {
  alert( i );
  i++;
} while (i < 3);
```

Такая форма синтаксиса оправдана, если вы хотите, чтобы тело цикла выполнилось **хотя бы один раз**

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team