JS: Functions, 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)

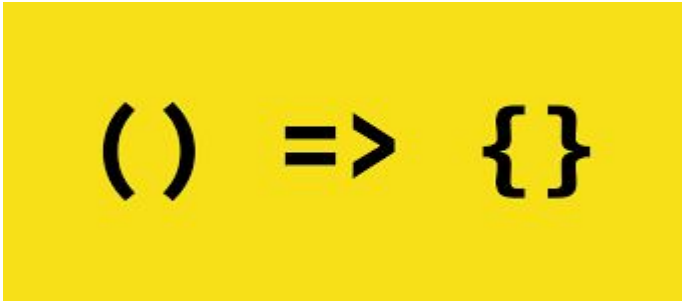
ЦЕЛЬ

Повторить пройденный материал

ПЛАН ЗАНЯТИЯ

- Опрос
- Повторение пройденного материала

Callback function



Функция в качестве параметра

Функция может передаваться в качестве аргумента при вызове другой функции (callback function). Например, функция, которая может выполнить произвольную операцию между двумя числами.

```
function operateOnNumbers(a, b, operation) {
  return operation(a, b);
}

// Функция сложения
function add(x, y) {
  return x + y;
}

// Функция вычитания
function subtract(x, y) {
  return x - y;
}

const sumResult = operateOnNumbers(5, 3, add);
console.log(sumResult);

const differenceResult = operateOnNumbers(8, 3, subtract);
console.log(differenceResult);
```

Передача анонимной функции в качестве параметра

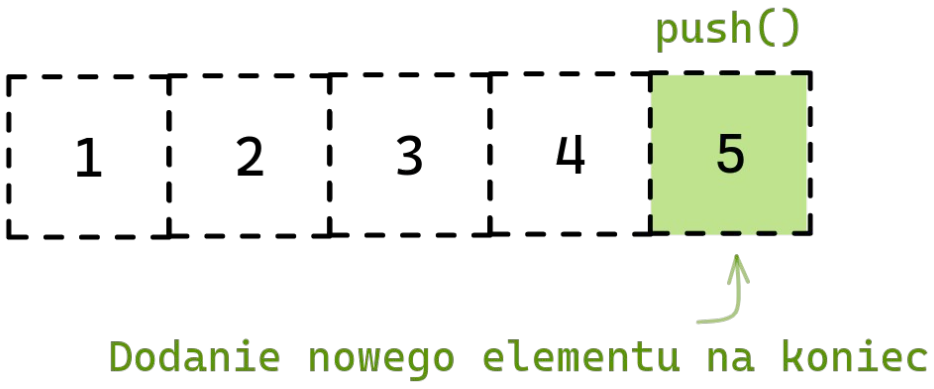
Анонимные функции в JavaScript - это функции, которые не имеют имени и обычно определяются прямо в месте, где они используются.

```
// Функция, принимающая функцию в качестве параметра
function performOperation(x, y, operation) {
  return operation(x, y);
}

// Используем функцию performOperation с анонимной функцией-выражением
const result = performOperation(8, 3, function(a, b) {
  return a - b;
});

console.log(result); // Вывод: 5
```

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

Существует два варианта синтаксиса для создания пустого массива:

```
1  let arr = new Array();
2  let arr = [];
```

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

```
1  let fruits = ["Яблоко", "Апельсин", "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
```

Содержание массива

В массиве могут храниться элементы любого типа.

```
let mixedArray = [42, "Hello", true, [1, 2, 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

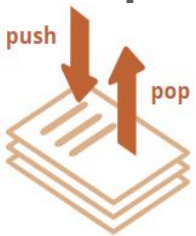
Существует и другой вариант применения для массивов – структура данных, называемая стек.

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:

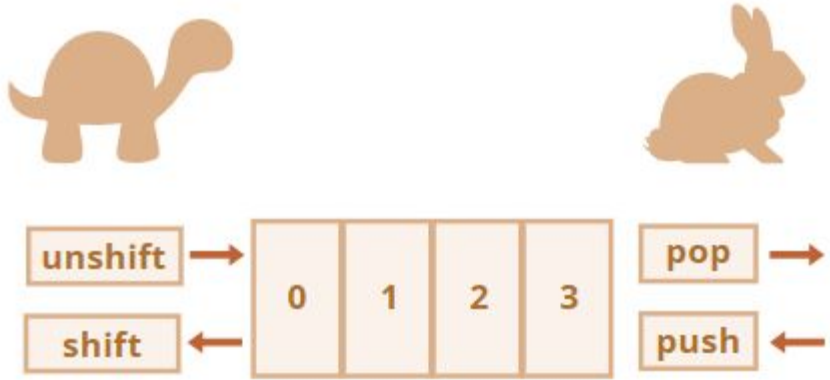
```
let fruits = ["Яблоко"];

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.



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team