Существует 2 варианта синтаксиса создания объекта:

1. Через фигурные скобки

let options = {

свойство1: true;

свойство2: false

}

1. С помощью new Object

let options = new Object () - но таким синтаксисом никто не пользуется

Создадим объект для примера:

Если мы хотим получить свойства объекта, то это можно сделать через точку или квадратные скобки. Но, если мы хотим задать новое свойство, то либо нужно дописать новое свойство прямо в объекте, либо задать точно такую же пару после объекта.

В объект можно поместить любой тип данных, соответственно в один объект можно поместить другой, что часто используется на практике.

//delete - для удаления свойства объекта

delete options.bool;

//для перебора свойств объекта:

for (let key in options) {

    console.log('Свойство ' + key + ' имеет значение' + options[key]);

}

В объект можно записывать функции, тогда они становятся методами объекта.

console.log(arr.length); //Для массивов свойство length работает необычным образом, выводя значение равное последнему индексу массива +1.

let first = [1,"second",3,4,5];

first[99] = 99;

console.log(first.length);

first.forEach(function(item, i, arr){

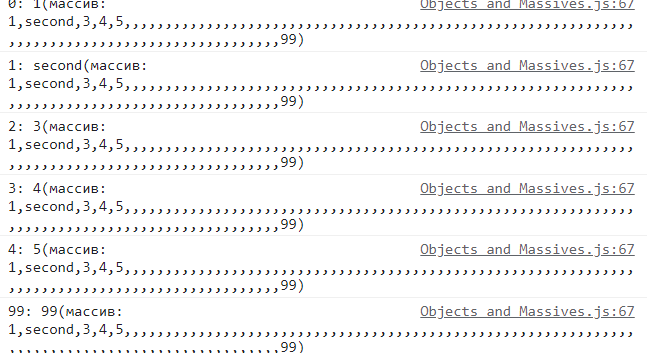
    console.log (i + ": " + item + "(массив: " + arr + ")");

})

Как параметр в данную функцию передаются item – каждый элемент нашего массива, i – index, т.е номер по порядку и arr – массив

Поэтому в данном случае хоть и в самом массиве 6 чисел в консоли будет выведено число 100

Если мы выведем в данном случае массив через цикл for each, то в консоли получим следующее:



Это происходит потому, что элементы в массиве должны быть по порядку, если элемент пропущен, то он заменяется пустым местом.

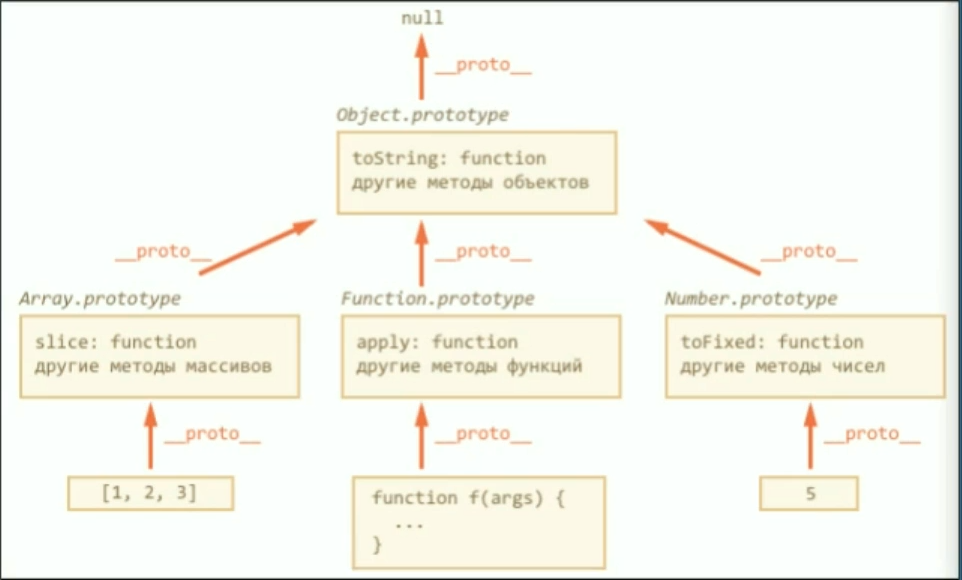
Методы для трансформации массивов через callback-функцию: filter, map, reduce – но они используются редко. Чаще используется – split, join, sort.

Пример:

Псевдомассив – объект, структура которого совпадает со структурой массива, он также хранит элементы в индексах, но у него отсутствуют методы массивов.

**ООП**

ООП – наука как делать правильную архитектуру. В JS массивы и функции – объекты. Поэтому главным является объект. На рисунке видно, что объект является прототипом для всех остальных сущностей, родителем. Поэтому у него они берут свои свойства и методы и если необходимо переопределяют и подстраивают под себя (прототипно наследуются от объекта)



Пример: