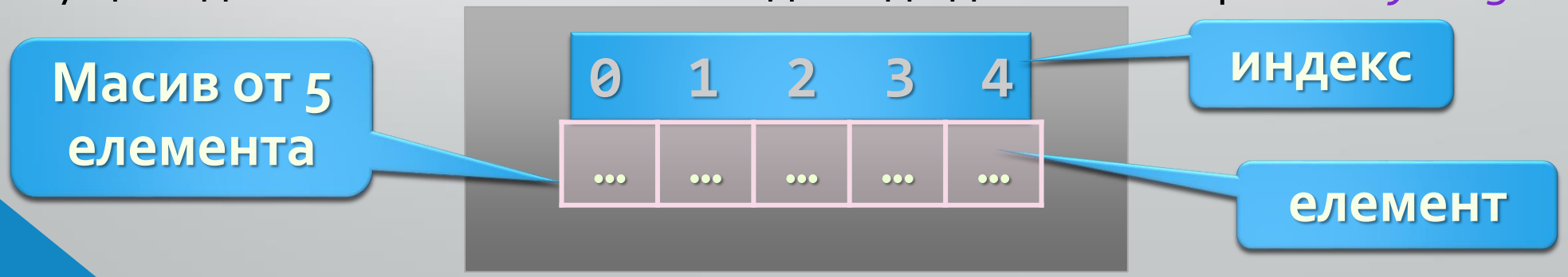
IT-Academy.BG

Масиви в JavaScript

Димитър Митев

Масиви (arrays)

- Масивите представляват поредица от елементи
- В предварително заделена част от паметта, съществува масива
- Поредността на елементите е фиксирина, т.е. ясно е кой елемент къде се намира
- Масивите в JS нямат фиксирана дължина
- Текущата дължина на масива може да бъде достъпена чрез *Array.length*



Деклариране на масив

- В JS няма строга типизация и масивите могат да съдържат различни по вид елементи
- Масив по може се декларира чрез
  - Конструктора *new Array(elements)* -> *var numbers = new Array(1, 2, 3, 4, 5);*
  - Конструктора *new Array(initialLength)* -> *var numbers = new Array(5);*
  - Литерал [ ] -> *var numbers = [ ];* -> **препоръчителен метод**

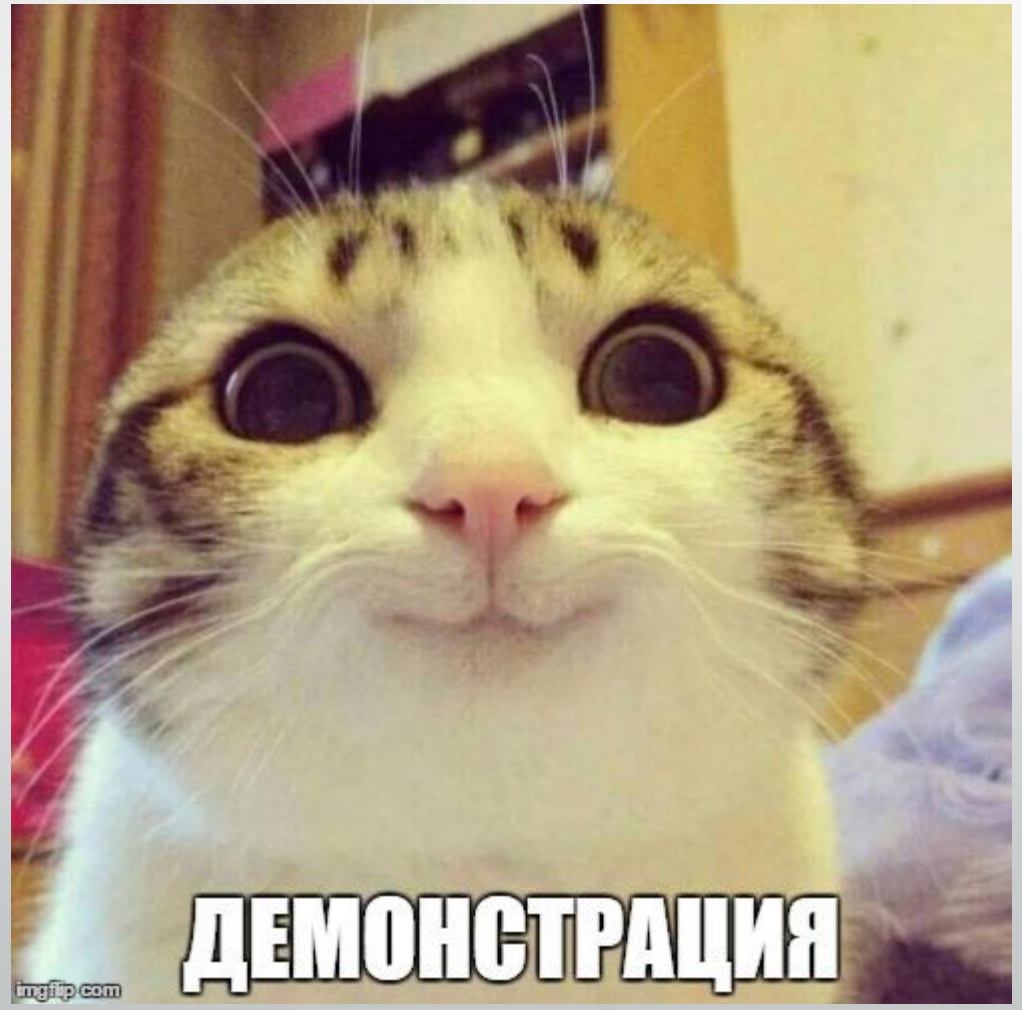
Деклариране на масив(1)

```
// Array holding integers
var numbers = [1, 2, 3, 4, 5]
;
// Array holding strings
var weekDays = ['Monday', 'Tuesday', 'Wednesday',
  'Thursday', 'Friday', 'Saturday', 'Sunday,
];

// Array of different types
var mixedArr = [1, new Date(), 'hello'];

// Array of arrays (matrix)
var matrix = [
  ['0,0', '0,1', '0,2'],
  ['1,0', '1,1', '1,2'],
  ['2,0', '2,1', '2,2']
];
```

Достъпване елементите на масива

- Елементите на масива могат да бъдат достъпвани посредством техния индекс чрез оператора *[ ]*
  - Диапазона на индекса е от 0 /нула/ до дължината на масива -1 (*0 …. length – 1*)
  - 1вият елемент има индекс 0 /нула/
  - Последният елемент има индекс *length-1*
- Елементите на масива могат да бъдат достъпвани и променяни чрез оператора *[ ]*

Създаване и достъп до елементите на масив



Обработка и обходане на масиви

- Тъй като масивите представляват поредица от елементи е много удобно за тяхната обработка и обхождане да се използват масиви /поради повтаряемия характкер на действията/
- Най-често се използва *for* – цикъл
- Може да се използват и други цикли

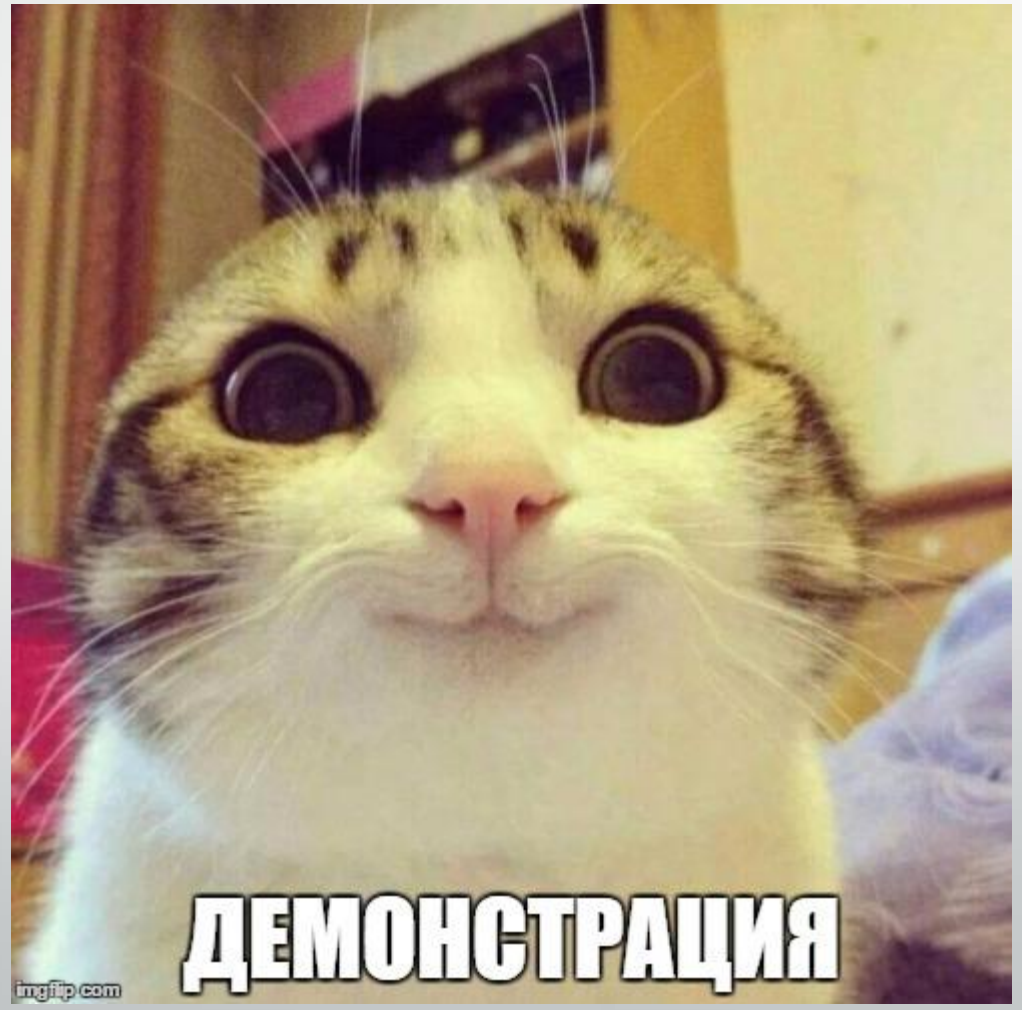
```
for (var i = 0; i < weekDays.length; i++) {
    console.log(weekDays[i]);
}
```

Обхождане на масиви. Алтернативи

- Освен чрез *for* цикъл масиви могат да се обхождат и чрез другите цикли
- Когато индексите на масива не са ясни е подходящо да се използва *for-in* цикъл
  - Не е гарантиран реда на елементите в масива
  - Елементите се достъпват 1 по 1

```
var weekDays = ['Monday', 'Tuesday', 'Wednesday',
    'Thursday', 'Friday', 'Saturday', 'Sunday'
];
for (var day in weekDays) {
    console.log(day + ' --> ' + weekDays[day]);
}
```

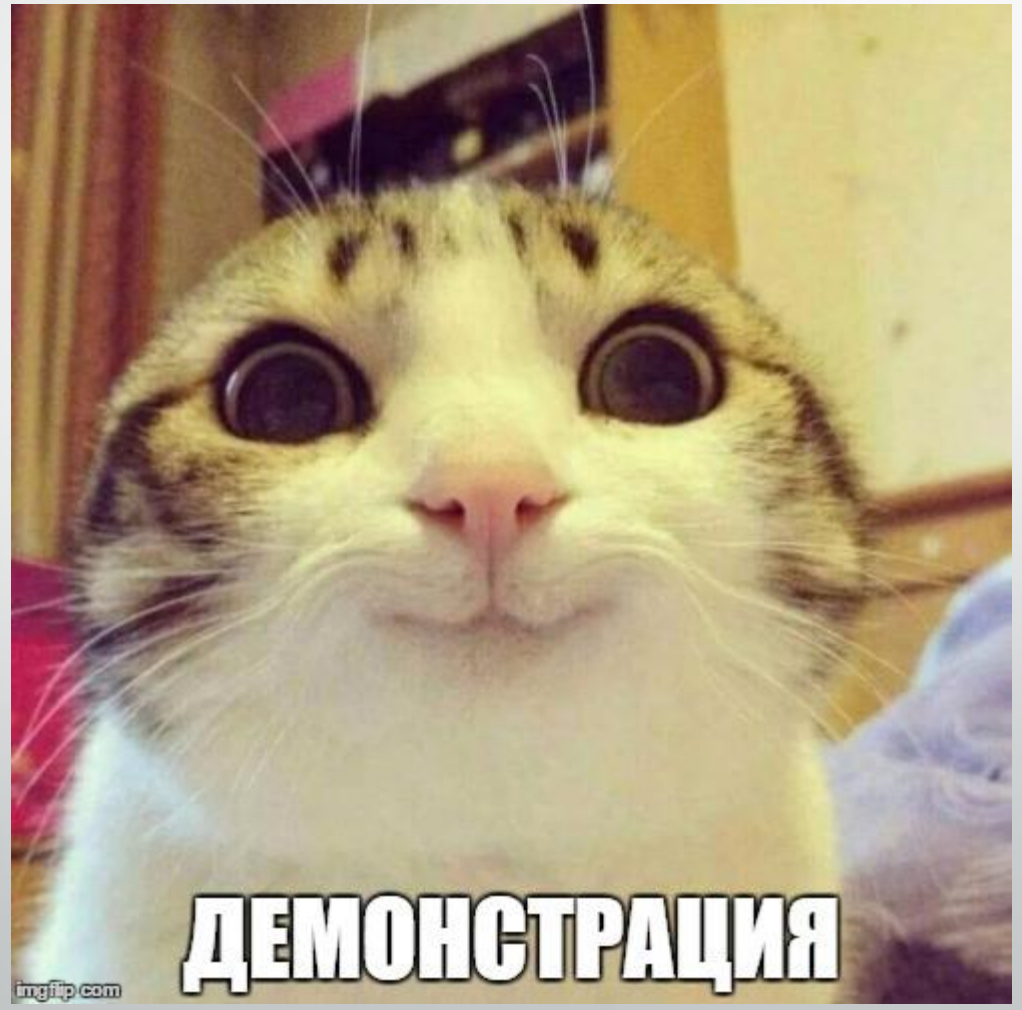
Обхождане на масив



Динамични масиви

- Всички масиви в JS са динамични по дизайн
  - Големината им може да бъде променяна по време на изпълнение
  - Нови елементи могат да бъдат вкарвани в масива
  - Елементи могат да бъдат премахвани
- Методи за манипулация на масиви
  - *[ ].push(element)* -> добавя нов елемент на края на масива, т.е. след последния
  - *[ ].pop()* -> премахва посление елемент от масива, като самата функция връща елемента като резултат
  - *[ ].unshift(element)* -> добавя нов елемент в началото на масива
  - *[ ].shift()* -> премахва 1вия елемент от масива

Динамични масиви



Вградени функции върху масиви

- *[ ].reverse()* -> връща нов масив, който съдържа елементите на масива, върху който е извикана фунцкията, в обратен ред
- *[ ].join(separator)* -> връща символен низ от елементите на масива, разделени чрез подадения сепаратор
- *arr1.concat(arr2)* -> долепя елементите на *arr2* към *arr1* и връща нов масив като оставя *arr1* и *arr2* непроменени
- *[ ].slice(from[, to])* -> връща част от масива като „*плитко копие*“ (*shallow copy*) на масива; може да се използва за клониране на масиви

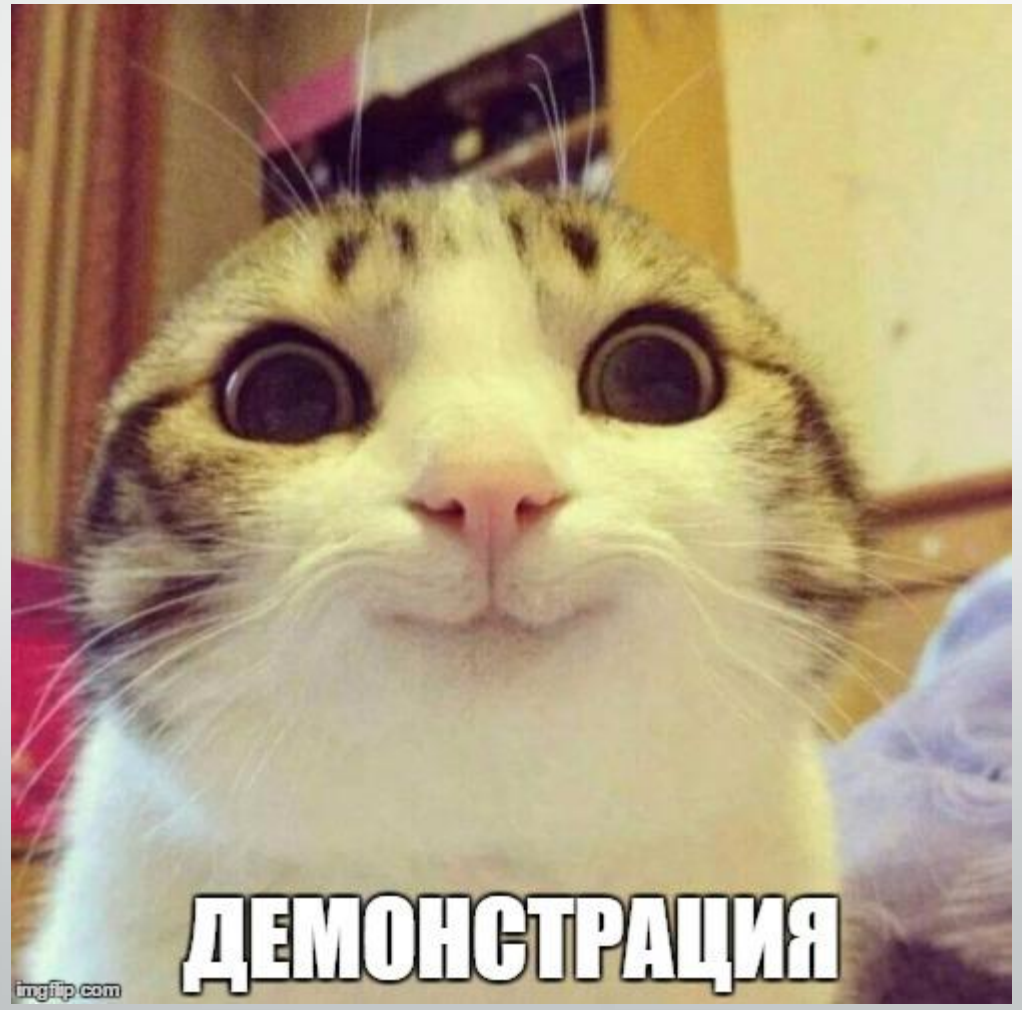
Вградени функции върху масиви(1)

- *[ ].splice(start, deleteCount[, item1[, item2[, ...]]])* -> добавя, премахва нови елементи на дадена позиция -> противоречи на привилата за качествен код
  - *start* -> индекса, от който да започне промяната на масива; ако е по-голям от дължината на масива, то промяната ще започне от края на масива; ако е отрицателен, то промяната ще започне от индекса който е разликата между дължината и дадения индекс (напр. Масив с дължина 5 и start = -2 -> промяна от индекс 1)
  - *deleteCount* -> броя на премахнатите елементи
  - *Item1,item2...* -> елементи, които да бъдат добавени

Вградени функции върху масиви(2)

- *[ ].indexOf(searchedElement [, fromIndex])* -> връща индекса на първия съвпадащ елемент на дадения масив или -1 ако не е намерено съвпадение
- *[ ].lastIndexOf(searchedElement [, fromIndex])* -> връща индекса на последния съвпадащ елемент на дадения масив или -1 ако не е намерено съвпадение
- За сравнение се използва стриктен подход (екв. на ===)

Вградени функции върху масиви



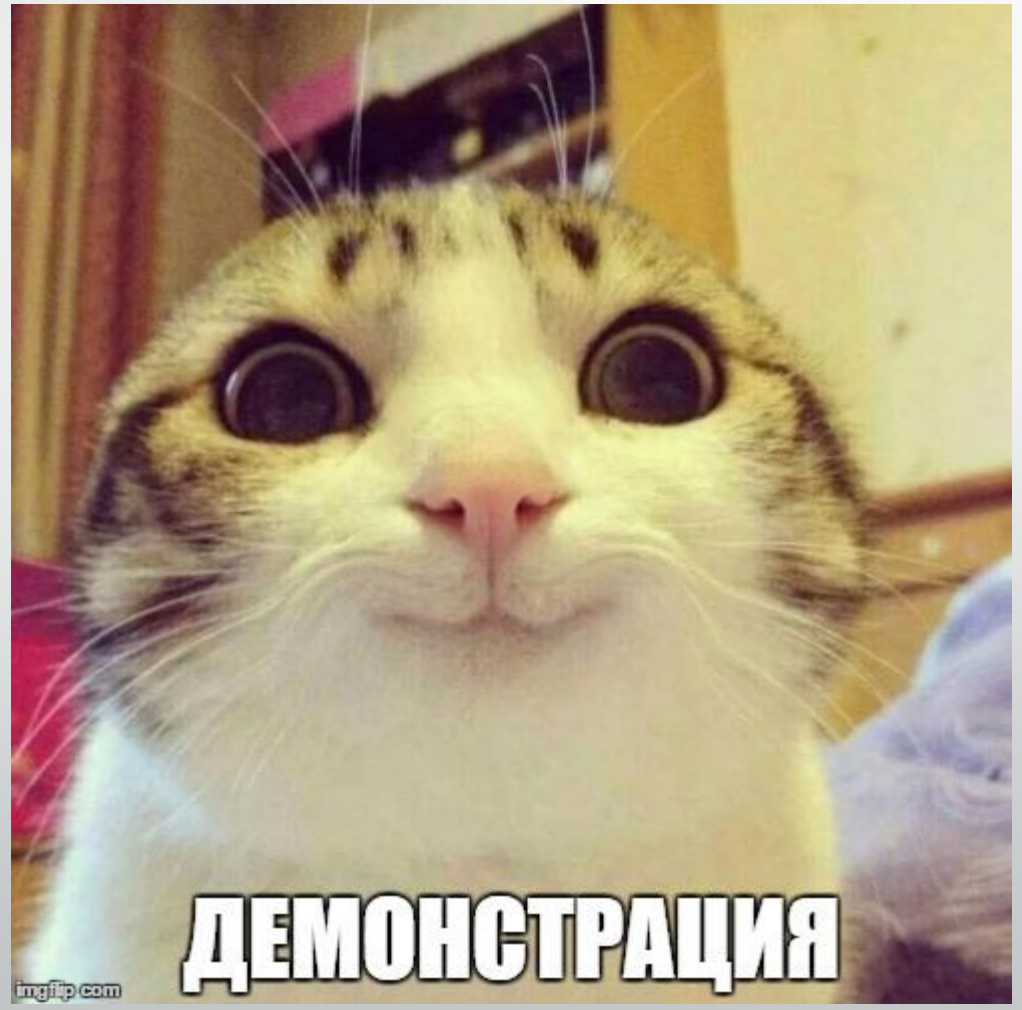
Проверка за масив

- *typeof([1,2,3])* -> *"object"* , защото в JS всичко е обект
- Проверката дали дадена променлива е от тип масив се прави чрез статичния метод
  - *Array.isArray([1,2,3])* -> *true*
  - Метода се поддържа от всички модерни браузъри

Асоциативен масив

- Асоциативният масив е масив с именувани индекси, т.е индексите не са поредни числа, ами зададени от потребителя имена
- В действителност асоциативния масив не е от тип масив, ами е от тип обект
  - Повечето от предефинираните методи за масиви не работят върху него
- Обхожда се чрез for-in цикъл

Асоциативни масиви




НЯКОЙ ИМА ЛИ
ВЪПРОСИ?

IT-Academy.BG

Домашна работа

1. Напишете JS код, който създава масив от 20 числа и го инициализира като всяко число има стойност съответния индекс, умножен по 3
2. Напишете JS код, който по зададен масив да намира най-дългата поредица от еднакви числа. Напр. 2, 1, 1, 2, 3, 3, **2, 2, 2**, 1 -> 2, 2, 2
3. Напишете JS код, който намира кое е най-често срещаното число в даден масив. Напр. 4, 1, 1, **4**, 2, 3, **4**, **4**, 1, 2, **4**, 9, 3 -> **4**
4. Напишете JS код, който да сортира даден масив. *Има и хитър начин на решение… Помислете и потърсете ☺
5. Напишете JS код, който да сортира масив по „метода на мехурчето“ (bubble sort).
6. Напишете JS код, който намира индекса на даден елемент в сортиран масив като се използва двоично търсене (binary search)