A R R A Y S

0 до **length-1.**

arr**[**0**]** **=** 4

let arr = **[]**

Shift+Alt+A прави блок комент **/* */**

How do you comment multiple lines?
Press Ctrl + /
- Select all the lines that you would like to be commented.
- Press Ctrl + / Two slashes "//" will be added to the front of each line, causing them to be recognized as a comment.

console.table(arr) : показва индекси и стойности от масив.

ЗАНУЛЯВАНЕ НА МАСИВА (всички ел. да станат НУЛА)->ПРИМЕР:

catalog[name].ratings**.length** = **0**;

For-of Loop:

for (let el **of** collection**)**

let arr = [10, 20, 30, 40, 50, 60, 70]; -> *примерен масив*
arr.length дава брой елементи в масива (число)

arr.concat() --- Accessor Methods ---

arr.shift() – remove from the beginning (-) --- Mutator Method ---
изрязва първия елемент от масива и го връща в променлива или където искаме (например: накрая на масива с .push).

arr.pop() – remove from the end (-) --- Mutator Method ---

arr.unshift(50**) – add to the beginning** (+) --- Mutator Method ---
ПРИМЕР: arr**.unshift(**arr**.pop())**;

arr.push(50**) – add to the end** (+) --- Mutator Method ---
слага нов ел. накрая на масива с дадена стойност

arr.includes(50) – look for value (boolean: true or false, as result). Дали дадена стойност се намира вътре в масива**.** --- Accessor Methods ---
Може и с индекс от който да търси стойност (от индекс X натам проверка (съществува ли)) -> arr.includes(50, 2)

arr.indexOf(50) – find index of value (from 0 to length-1 as result)
--- Accessor Methods ---
Output is **-1** if element is **not** present; if **-1** index as result: no such character;

arr.split(“) един стринг на масив от елементи.

arr.toString() превръща масива в стринг, като всички стойности ги изрежда със запетая и ще върне string.

arr.join(‘,’) превръща масива в стринг, методът join прави същото като toString само че избира какъв да е разделителя (докато toString е все едно join ама с разделител запетая). --- Accessor Methods ---

arr.reverse() завърта наобратно стойностите в масива. (Работи с масив, със СТРИНГ НЕ може!). --- Mutator Method ---

arr.fill(): Fills all the elements of an array from a **start index** to an **end index** with a **static value.** --- Mutator Method ---
let arr = [10, 20, 30, 40];
arr.fill(0, 2, 4) -> // [10, 20, 0, 0]

arr.slice(2,4) -> COPY Array Elements --- Accessor Methods ---
ПРИМЕР: let sortedArr = inputArr**.sort**((a, b) => a - b)**.slice**((inputArr.length / 2))

- **The slice() function creates a new array from part of another**
- **Gets a range of elements from selected start to end (exclusive) OR only start (and takes all to the end of an array)**
- **Note that the original array will not be modified**
- **има свойството с отрицателно число в скобите (-3) да взима отзад-напред елементите**
- **.slice() без параметри в скобите прави истинско копие на масива**

arr.splice(2, 4, "twenty", "twenty-five") --- Mutator Method ---
ПРИМЕР: let result = inputArr**.sort**((a, b) => a - b)**.splice**(0, 2)

Splice(): **insert, replace, delete**. (index, replace 0 / del 1,2.., +add..)
И връща при **delete** масив с елемент/и **!**

• The .splice() adds/removes items to/from an array, and returns the removed item(s) as an array

• This function changes the original array

• Gets a range of elements from selected start, delete-count (deletes these elements from the original array) AND can add additional elements to the original array.

• отрицателни стойности важат за индекс (както например ги ползваме в другия метод slice) ,НО ВАЖАТ СТОЙНОСТИ за COUNT and DELETE:

- **При плюс число .splice(1) -> означава от индекс 1 нататък до края DELETE**
- **При минус число .splice(-3) -> означа последните 3 елемента DELETE**

• можем да слагаме променливи в скобите

• при .splice бройката параметри които изрязваме НЕ е нужно да отговаря на бройката параметри, които ще вкараме

Note: **Removing elements** with splice() receives two parameters:

- **Start Index**
- **Count of elements you want to remove**

Note: **Inserting elements** with splice() receives three parameters:

- **Start Index**
- **Count of elements to remove – if none enter 0**
- **Elements to insert at that position**

--- Iteration Methods ---:

arr.method(**value, index, arr**) => ……

- **.forEach()** == for-of loop

myArr.forEach((**value, index, arr**) => console.log(value, index, arr));
forEach НЕ ВРЪЩА РЕЗУЛТАТ (връща undefined)

- **.some()**: дали **ПОНЕ ЕДИН ел.** отговаря на сравняващата функция (true/false)

- **.every()**: **дали ВСИЧКИ ел.** отговарят на сравняващата функция (true/false)

- **.find()**: като some(), но дали **поне 1 елемент** да отговаря на сравняващата функция и **ВРЪЩА СТОЙНОСТТА като РЕЗУЛТАТ**.
(**връща 1 СТОЙНОСТ** или **undefined** ако няма)

- **.filter()**: които елементи отговарят на сравняващата функция ги **ВРЪЩА** като **СТОЙНОСТ като РЕЗУЛТАТ**.

- **.map() creates a new array by applying a function to every element (Transform Elements)**

- **създава нов масив и прилага функция на всеки елемент**

ПРИМЕРИ: .map(x => x.length); .map(Number); .map(x => x+1);

.filter() creates a new array from elements matching predicate.
Predicate is a function returning a Boolean value (true or false)
(Filter Elements)

- **създава нов масив и прилага функция на всеки елемент чрез Булев израз, който ако върне true на елементи ги вкарва в новия масив.**

Filter е метод на масив, който връща нов масив. Във филтър подаваш функция с условие. И елементите които преминат условието се записват в новия масив.

ПРИМЕРИ: .filter(x => x.length > 3); .filter(x => x > 0);
.filter(el => el !== Number(item[1]));

let newArr = arr.filter(value => value > 18) ИЛИ
let newArr = arr.filter(function(value) {
return value > 18;
});

let arr=[1,2,3,4,5,6,7,8,9] let filterArr = arr.filter(x=>x>5)
//Expected output: [6,7,8,9]

.sort() function sorts the items of an array:
default, alphabetic (without regard for the case used) or numeric.
- arr.sort()
- arr.sort((a,b) = > a.localeCompare(b))
- arr.sort((a,b) = > a – b) up OR arr.sort((a,b) = > b – a) down

Depending on the provided compare function, sorting can be **alphabetic or numeric**,
and either **ascending** (up) **or descending** (down).

By default, the sort() function sorts the values as strings in alphabetical and ascending order.

If you want to **sort numbers** or other values, you need to provide the correct compare function!

• .sort() **По дефолт** ги сортира спрямо **ASCII** таблицата (тоест по Азбучен ред като взима ПЪРВО ГОЛЕМИ БУКВИ, после МАЛКИ букви).

• .sort() The localeCompare() method is used to compare any two characters **without regard for the case used**. It's a string method so it can't be used directly on an array.
Pass localeCompare() as the comparison function:
.sort((a, b) => a.localeCompare(b)); (от A до Z)
.sort((a, b) => b.localeCompare(a)); (от Z до A)

• .sort() за ЧИСЛА от малките към големите и обратния вариант:

.sort(a, b) => a - b == ascending

.sort(a, b) => b - a == descending

ПРИМЕРИ:

const arr = ['aa','aaa','b','bb','c','ccc'];
const sortedArr = arr.sort((a,b) => a.length - b.length || a.localeCompare(b));
console.log(sortedArr);
резултат: ['b', 'c', 'aa', 'bb', 'aaa', 'ccc']
Първо се подреждат по дължина в нарастващ ред и след това по азбучен ред

```
arr.sort((a, b) => {
    return a.length - b.length || a.localeCompare(b)
});

arr.forEach(x => console.log(x));
```

--- Reducing Arrays ---:

reduce(): The reduce() method executes a reducer function on each element of the array, resulting in a single output value.
Reduce method приема в скобите 2 параметъра – reducer func & initial first value.
Reducer function приема 4 параметъра.

- **The reducer function takes four arguments:**

- Accumulator
- Current Value
- Current Index (Optional)
- Source Array (Optional)

- **Your reducer function's returned value is assigned to the accumulator**

- **Accumulator's value - the final, single resulting value**

ПРИМЕР: let sum = arr.reduce**((a, v)** => **a + v, 0)**

```
02-Arrays and Nested Arrays > JS demo-10-reduce2.js > ...
1   const myArr = [10, 20, 30, 40];
2
3   const result = myArr.reduce((a, v) => a + v, 0)
4
5   console.log(result);

PROBLEMS   OUTPUT   TERMINAL   DEBUG CONSOLE
"C:\Program Files\nodejs\node.exe" ".\02-Arrays and Ne
Debugger attached.
100
```

```
    console.log(num.toString().split('').every((el, i, arr) => (el == arr[0])));
    console.log(num.toString().split('').map(Number).reduce((a, v) => a + v, 0));
}

sameNumbers(2222222);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
"C:\Program Files\nodejs\node.exe" ".\01-Syntax, Functions and Statements\2.exercise-03
Debugger attached.
true
14
```

```
function largestNumber(...params) {    // 100/100

    console.log(`The largest number is ${Math.max(...params)}.`);
}

largestNumber(5, -3, 16);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
"C:\Program Files\nodejs\node.exe" ".\01-Syntax, Functions and Statemer
Debugger attached.
The largest number is 16.
```

```
function sumFirstAndLast(input) {    // 100/100

    let first = [...input].shift();     // let first = input[0]
    let second = [...input].pop();      // let second = input[input.length-1]

    let result = Number(first) + Number(second);
    return result;
}

sumFirstAndLast(['20', '30', '40']);        // 60
```

Set()

```
let set = new Set(arr);

if (!result.hasOwnProperty(name)) {
        result[name] = new Set();
        result[name].add(id)
    }
    result[name].add(id)
```

Каква е разликата между РЕСТ и СПРЕД оператор?

rest operator (прави масив от променливи)
spread operator (прави променливи от масив) -> ПЛИТКО КОПИЕ НА МАСИВ
Функция ПРЕДИКАТ – функция която връща TRUE или FALSE!

const се използва , когато знаеш че стойността в тази променлива никога не искаш да се променя. Ако се опиташ да присвоиш нова стойност ще ти гръмне с грешка.