O B J E C T S

• Съвкупност **KEY + VALUE = променлива**

• **Празно свойство го правим null или undefined:**

```
let person = {
    name: 'Peter',
    age: undefined        ИЛИ        age: null
};
```

• **Чрез .chaining (property calling) начин достъпваме стойности навътре в обекта.**

person.grades.Math[0]

• Проверка дали съществува дадено **VALUE** – чрез If

If (person.name != undefined)

При масив е: If (arr[0] != undefined)

• Ключовете са еквивалент на индексите при обикновения For-of цикъл на масивите.

VALUE TYPES: Boolean, String, Number, null, undefined, Symbol, BigInt.
REFERENCE TYPES: Arrays, Functions, Objects (collectively Objects).

копиране на обект (плитко копие)

let newArr = inputArr.slice(); ~ копиране на масив.

let **newObj** = **Object.assign({}, inputObj)**; ~ **копие на обекта с Object.assign() method**

копие на обекта с json (дълбоко копие)

DEEP COPY METHOD FOR OBJECTS:

let copy = **JSON.parse(JSON.stringify(**obj**));**

let x = **JSON.parse(JSON.stringify(obj))** ,за да разруши релация. По-добре е да се ползва тая хитрина, защото прави цялостно копие. Докато spread и assign само частично.

- **let text = JSON.stringify(obj);** ~ Object into JSON string method
- **let obj = JSON.parse(text);** ~ JSON string into Object method

.hasOwnProperty() проверява ключовете в ОБЕКТА (**дали съществува** даден key в Object).

delete person.age; -> изтриване на property (напълно, не остава следа). KEY & VALUE DELETE.

object.keys(obj) -> **създава масив с ключовете като стринг.**

object.values(obj) -> **създава масив със стойностите като стринг.**

object.entries(obj) -> **създава масив (матрица) със двойките KVP като стринг.**

.length **Object.keys(obj).length == 0** **(дали е ПРАЗЕН ОБЕКТА)**

VALUE или стойност **се изтрива**: като set-нем ключа на **undefined** **или** **null.** ПРИМЕРИ: myAssocArr[myKey] = undefined; myAssocArr.beta = null;

- **For-IN LOOP за ОБЕКТИ цикъл. (for object inside iteration). We can use for-in loop to iterate through the keys.**

- **For-OF LOOP за МАСИВИ цикъл. Връзката между Object и Array я правим с Object.entries(Obj) || Obejct.keys(Obj).** To iterate over the object properties **by key (from ARRAY).**

Iterate destructured entries: при масив с For-Of Loop.

• Обектите **НЕ СЕ СОРТИРАТ**. Трябва първо да превърнем **обектите -> в масиви** и да ползваме sorting, filtering, mapping за arrays.

let entriesArr **= Object.entries(Obj);**
The every entry is turned into an array of [[key, value], [key, value]]
(To sort by key, use the first element of each entry)

Set() уникални стойности пази, обект е. С forEach към нов масив всеки el. Storing Unique Elements. Store unique values of any type, whether **primitive** values or **object** references. Set objects are collections of values.

```
let set = new Set([1, 2, 2, 4, 5]);
// Set(4) { 1, 2, 4, 5 }
set.add(7)); // Add value
console.log(set.has(1));
// Expected output: true
```

set.add()
set.has()

Essential Set() Methods

Method	**Description**
new Set()	Creates a new Set
add()	Adds a new element to the Set
has()	Returns true if a value exists in the Set
delete()	Removes an element from a Set
forEach()	Invokes a callback for each element in the Set
values()	Returns an iterator with all the values in a Set

Property	**Description**
size	Returns the number of elements in a Set

ПРИМЕРИ: Set()

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

09-Associative Arrays > JS 2.exe-06.companyUsers-01.js > companyUsers

```
function companyUsers(inputArr) {

    let result = {};

    for (let el of inputArr) {
        let token = el.split(' -> ');
        //console.log(token);
        let name = token[0];
        let id = token[1];
        if (!result.hasOwnProperty(name)) {
            result[name] = new Set();
            result[name].add(id)
        }
        result[name].add(id)
    }
    console.log(result);

    let sorted = Object
        .keys(result)
        .sort((a, b) => a.localeCompare(b))

    for (let el of sorted) {
        console.log(`${el}`);
        for (let subEl of result[el]) {
            console.log(`-- ${subEl}`);
        }
    }
    // 2-ри начин
    // let entries = Object.entries(result).sort((a, b) => a[0].localeCompare(b[0]));
    // for (let [company, ids] of entries) {
    //     console.log(company);
    //     for (let id of ids) {
    //         console.log(`-- ${id}`);
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
{
  SoftUni: Set(2) { 'AA12345', 'BB12345' },
  Microsoft: Set(1) { 'CC12345' },
  HP: Set(1) { 'BB12345' }
}
HP
-- BB12345
Microsoft
-- CC12345
SoftUni
-- AA12345
-- BB12345
```

```js
function wordOccurrences(input) {
    let words = {};

    for (let el of input) {
        if (!words.hasOwnProperty(el)) {
            words[el] = 0
        }
        words[el]++;
    }

    let sorted = Object
        .entries(words)
        .sort((a, b) => b[1] - a[1])
        .forEach(([key, value]) => console.log(`${key} -> ${value} times`));
}
wordOccurrences(["Here", "is", "the", "first", "sentence", "Here", "is", "another", "sentence", "And", "finally", "the", "third", "sentence"]);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
"C:\Program Files\nodejs\node.exe" ".\09-Associative Arrays\1.lab-06.wordOccurrences-MYCODE-2.js"
Debugger attached.
sentence -> 3 times
Here -> 2 times
is -> 2 times
the -> 2 times
first -> 1 times
another -> 1 times
And -> 1 times
finally -> 1 times
third -> 1 times
```

- let [...] -> по-къс синтаксис, деструктуриране

- + и Number() правят едно и също нещо -> parse към число

calling property keys:

demo-00.js

08-Objects and Classes > demo-00.js > solve

```js
function solve() {
    let person = {
        name: 'Peter',
        age: 20,
    };

    console.log(person.age);            //1 .chainingProperty calling

    let propName = 'age';
    console.log(person[propName]);      //2 from [variable]

    console.log(person['a' + 'ge']);    //3 concatenation

    console.log(person['age']);         //4 ['key']
}
solve();
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
"C:\Program Files\nodejs\node.exe" ".\08-Objects and Classes\demo-00.js"
4 20
```

For-Of Loop < - > forEach

forEach: В метода НЕ работи break И return И continue.
forEach няма как да го спреш.
.forEach() има ли алтернатива на break и continue: С if

- **word.toLocaleLowerCase() -> превръща стринг в малки букви**

например: PhP -> php

Difference Between isnan() and Number.isnan()

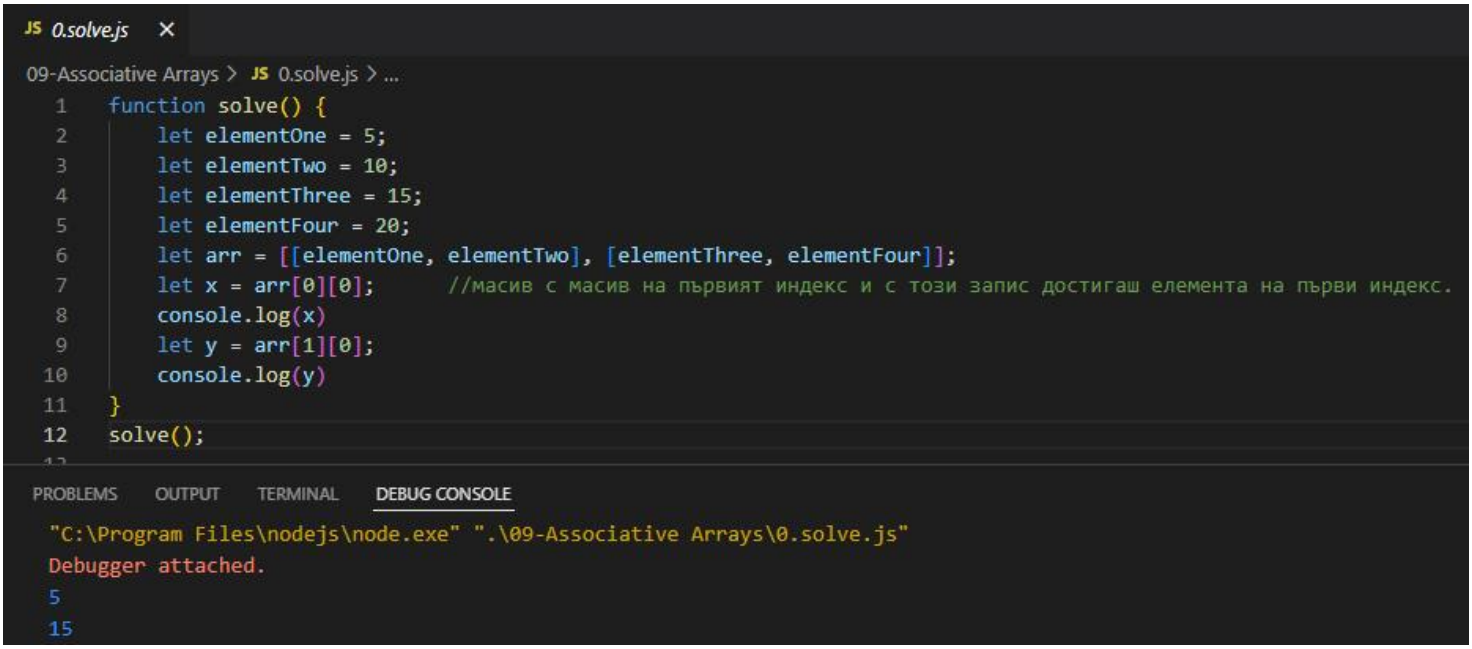
isNaN() method returns true if a value is **Not-a-Number**.

Number.isNaN() returns true if a number is **Not-a-Number**.

In other words: isNaN() converts the value to a number before testing it.

variable[0][0]

variable е масив с масив на първият индекс и с този запис достигаш елемента на първи индекс.
[[elementOne,elementTwo], [elementOne,elementTwo]]



forEach е алтернатива на For-Of.

.map() е алтернатива на forEach с разликата че създава нов масив.

Разлики ?

-forEach изпълнява някаква функция върху всеки един елемент от масив

-map() също изпълнява функция върху всеки един елемент от масива, но променя елементите му и ги запазва в нов масив

Най-често в практиката се ползва forEach, ако става на въпрос за обхождане на масив. Set(): Сета пази уникални стойности. И е по-близо до масива от колкото до обекта.