S T R I N G S

(1). typeof Operator (checking for a type) – number, string, object (като резултат в конзолата)…

- **typeof p;**
- **typeof(p)**

(2). число в стринг?

променлива.toString() ИЛИ String(променлива)

let outputAsString = String(**output**);

let outputAsString = output.toString().includes('9');

(3). .replace(), .includes()

let output = str.replace('_', char);

let outputAsString = output.toString().includes('9');

(includes задава въпрос и получава: true ИЛИ false като резултат в конзолата.)

(4). String.fromCharCode() - value in decimal; ASCII

(5). .toFixed(n) превръща числото в string!

=> let numAsString = String(num); // '123'

=> let numAsNumber = Number(num); // 123

(6). parseInt – Преобразува **string в число**.

Изрязва само цялата част на този string. **Int** означава цяло число

// '3.24' или '3.24ygkwr' -> 3

(7). parseFloat – преобразува **string в число**. Не се влияе от символите. Взима **числото след дробната част** до където има цифри.

// '3.24' или '3.24ygkwr' -> 3.24

(8). Number(нещо си, както в Basics) – може да преработи само ако стринга е изцяло число ,с цифри написан.

return ключова дума във функция, прекъсва функцията. return връща резултат и по-надолу не се изпълнява никакъв код.
След return трябва всичко да е на същия ред (а не на нов ред), защото след return не се чете.

- **Strings have a length property.**

-> word**.length** (брой; дължина тотал)
-> word**.length – 1** (индекс)
-> word**[word.length – 1]** (буквата на последния индекс)

- **Strings Concatenated using the "+" operator**
- **Strings are immutable.**
- **Strings Accessible by index.**

```
let str = "Hello, JS";
let ch = str[2];                // Expected output: l
ch = str.charAt(2);             // Expected output: l
                                // Both declarations are the same
```

- **Strings can also be iterated using For-Of Loop (and For classic Loop).**
- **Strings can be declared with three types of quotes:**

```
let str = "Hello";      let str = 'Hello';  let str = `Hello`;
```

Manipulating Strings

Concatenating

- **Use the "+" or the "+=" operators**
- **Use the concat() method**

```
let greet = "Hello, ";
let name = "John";
let result = greet.concat(name);
console.log(result);                // Expected output: "Hello, John"
```

Searching for Substrings

- **.includes("text") -> expected output true/false (търси текст, TRUE or FALSE)**
- **Starts With/Ends with**

Use **.startsWith()** to determine whether a string begins with the characters of a specified substring // Expected output: **true / false**

Use **.endsWith()** to determine whether a string ends with the characters of a specified substring // Expected output: **true / false**

- **.indexOf(substr) -> case sensitive (от ляво на дясно). Дава СТАРТОВИЯ индекс.**

```
let str = "I am JavaScript developer";
console.log(str.indexOf("Java"));          // Expected output: 5
console.log(str.indexOf("java"));          // Expected output: -1 (няма го)
```

може и .indexOf('text', positionIndex) -> от кой символ нататък да търси

```
function countStringOccurrences(text, word) {
    let counter = 0;
    let index = 0;
    let found = text.indexOf(word);

    while (found != -1) {
        counter++;
        index = found + 1;
        found = text.indexOf(word, index);
    }

    console.log(counter)
}

countStringOccurrences(
    'This is a word and it also is a sentence',
```

• **.lastIndexOf(substr) -> case sensitive (от дясно на ляво). Дава СТАРТОВИЯ индекс.**

```
let str = "Intro to programming";
let last = str.lastIndexOf("o");
console.log(last);                    // Expected output: 11
```

substring започва винаги с по-малката от двете стойности (независимо кое е старт и кое енд индекс), а **slice** ги кара както са записани и върви само надясно.

Extracting Substrings: взимаме ПАРЧЕ от стринга

• **.substring(startIndex, endIndex (exclusive)) -> endIndex е екслузив**

```
let str = "I am JavaScript developer";
let sub = str.substring(5, 10);
console.log(sub);                     // Expected output: JavaS
```

Може и само startIndex нататък (let sub = str.substring(**5**))
Както и друг трик: **.substring(start, start + count) -> endIndex** e start + count (изкл.)

ИЗВОД: .slice е по-полезен от .substring, защото може с отрицателни символи да взима назад от края на думата

```
1  function substring(str, startIndex, count) {
2
3      let endIndex = startIndex + count;
4      str = str.substring(startIndex, endIndex);
5      console.log(str);
6  }
7  substring('ASentence', 1, 8);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE
"C:\Program Files\nodejs\node.exe" ".\10-Text Proces
Debugger attached.
Sentence
```

- **.slice() получава (startIndex, endIndex (exclusive))**

.slice печата и с – (отрицателни бройки -> вземи ми последните 3 символа)

Колко символа да вземем: .slice(start, start + count) -> endIndex е start + count (изкл.)
(start + count дават количеството символи които искаме да вземем от стартовия напред еди колко си)

- **.replace** заменя само веднъж.

.replace(search, replacement)

let text = "Hello, john@softuni.bg, you have been using john@softuni.bg in your registration.";
let replacedText = text.replace(".bg", ".com");
console.log(replacedText);
*// Hello, john@softuni**.com**, you have been using john@softuni**.bg** in your registration.*

ПРИМЕР:

```
let myString = "Hello, john@softuni.bg, you have been using john@softuni.bg";

let found = myString.indexOf('.bg');
while (found != -1) {
    myString = myString.replace('.bg', '.com');
    found = myString.indexOf('.bg');
}
```

.replaceAll заменя всички. (нов метод е, може да не работи навсякъде) **или ползваме алтернативен подход с**
regex: / /g (заменя навсякъде)
ПРИМЕР: let replacedText = text.replace(/.bg/g, ".com");

(Others) - Превръщане от МАЛКИ в ГОЛЕМИ букви и обратното:

- **.toUpperCase, .toLowerCase**

- **.toLocaleUpperCase, .toLocaleLowerCase (хваща и КИРИЛИЦА символи и букви)**

Splitting and Repeating Strings

• **.split(‘separator‘)** **-> връща МАСИВ от стрингове.** Може да се сплитне и по думи и символи, не само по символи.

• **.repeat(count) - Creates a new string repeated count times**
‘banana’.repeat(4) **-> bananabananabananabanana**

Trimming Strings

• Use **trim()** method to remove **whitespaces** (spaces, tabs, no-break space, etc.) from both ends of a string

• Use **trimStart()** or **trimEnd()** to remove **whitespaces** only at the beginning or at the end

Padding at the Start and End

• Use **padStart()** to add to the current string another substring at the start until a length is reached

• Use **padEnd()** to add to the current string another substring at the end until a length is reached

```
let text = "010";
console.log(text.padStart(8, '0'));          // Expected output: 00000010
```

/ ДОПЪЛНИ го с НУЛИ ОТПРЕД докато ЦЯЛАТА ДЪЛЖИНА СТАНЕ с 8 символа общо /

```
1   let hour = 12;     // programming basics часове - минути - секунди добави символи отпред/отзад вместо If проверки
2   let min = '2';
3
4   let minFrom0to9 = min.padStart(2, "0");     // countTotalWord, extraSymbolsAdd
5   console.log(hour + ":" + minFrom0to9);
```

```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE                    Filter (e.g. text, !exclude)
"C:\Program Files\nodejs\node.exe" ".\10-Text Processing\demo-06-padStart-Clock example.js"
Debugger attached.
12:02
```

```
let sentence = "He passed away";
console.log(sentence.padEnd(20, '.'));          // Expected output: He passed away.....
```

RE-ASIGGIGN (символ за присвояване =) мога да преправя STRING, (който string всъщност е immutable):

let **text** = ‘’Lubomir”;

text = ‘’Lubomir” + “ “ + “Jiponov” =>

// **Lubomir Jiponov**

// Създава се нова референция в паметта на компютъра в Stack за променливата text, а старата референция се изтрива.