SOFT CLUB

JAVA SCRIPT

LECTURE 3

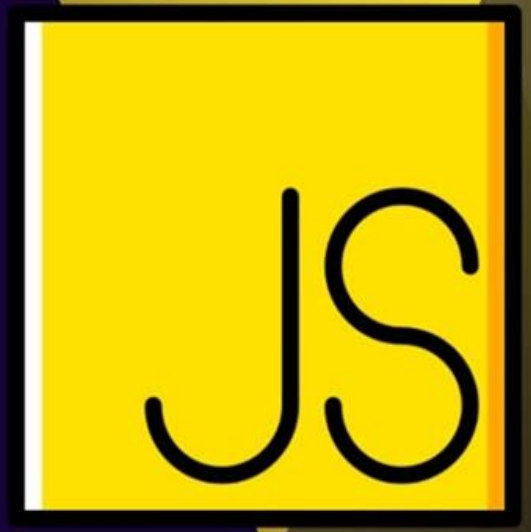
TABLE OF CONTENTS

SOFT CLUB

What is a method in java script?

A method is a block of code which only runs when it is called. You can pass data, known as parameters, into a method. Methods are used to perform certain actions, and they are also known as functions.

STRING IN JAVA SCRIPT



CREATE STRING IN JAVA SCRIPT

SOFT CLUB

"Double Quotes"

```
"Hello"
```

'Single Quotes'

```
'Hello'
```

`Backticks`

```
`Hello ${hi}`
```

JAVA SCRIPT STRING METHODS

➢ charAt()	➢ slice()
➢ concat()	
➢ trim()	➢ substring()
➢ includes()	➢ split()
➢ indexOf()	
➢ replace()	➢ toString()
➢ replaceAll()	➢ toLowerCase()
➢ repeat()	➢ toUpperCase()

JavaScript String method charAt()

The charAt() method returns the character at a specified index (position) in a string.
The index of the first character is 0, the second 1, ...
The index of the last character is string length - 1 .

Get the first character in a string:

```
// 1
let text = "HELLO WORLD";
let letter = text.charAt(0);
console.log(letter);
```

```
node /tmp/1d1LKTWZwx.js
H
```

Get the second character in a string:

```
// 2
let text = "HELLO WORLD";
let letter = text.charAt(1);
console.log(letter);
```

```
node /tmp/1d1LKTWZwx.js
E
```

Get the last character in a string:

```
let text = "HELLO WORLD";
let letter = text.charAt(text.length-1);
console.log(letter);
```

```
node /tmp/Z0DgPxhxm5.js
D
```

JavaScript String method at()

The **at()** method takes an integer value and returns a new String.
This method allows for positive and negative integers. Negative integers count back from the last string character.

```
const sentence = 'The quick brown fox jumps over the lazy dog';

console.log(sentence.at(-1)) // "g"
console.log(sentence.at(4))  // "q"
```

JavaScript String method concat()

The concat() method joins two or more strings.
The concat() method does not change the existing strings.
The concat() method returns a new string.

Join two strings:

```
const text1 = 'hello';
const text2 = 'world';
const result1 = text1.concat(' ', text2);
console.log(result1);
```

```
node /tmp/FNBL7MN5w2.js
hello world
```

Join three strings:

```
// 2
let text1 = "Hello";
let text2 = "world!";
let text3 = "Have a nice day!";
let result = text1.concat(" ", text2, " ", text3);
console.log(result);
```

```
node /tmp/FNBL7MN5w2.js
Hello world! Have a nice day!
```

JavaScript String method replace()

The **replace()** method searches a string for a value or a regular expression.
The **replace()** method returns a new string with the value(s) replaced.
The **replace()** method does not change the original string.

Replace Microsoft:

```
// 1
let text = "Visit Microsoft!";
let result = text.replace("Microsoft", "Soft club");
console.log(result);
```

```
node /tmp/FNBL7MN5w2.js
Visit Soft club!
```

JavaScript String method replaceAll()

The **replaceAll()** method returns a new string with all matches of a pattern replaced by a replacement.

```
const p = 'The quick brown dog fox jumps over the lazy dog.';
console.log(p.replaceAll('dog', 'monkey'));
//"The quick brown monkey fox jumps over the lazy monkey."
```

JavaScript String method split()

The **split() method** splits a string into an array of substrings. The split() method returns the new array. The split() method does not change the original string. If (" ") is used as separator, the string is split between words.

Examples:

```
1  // converting the string to an array
2  const text = 'hello';
3  const result = text.split();
4  console.log(result);
5  //2
6  let text2 = "How are you, doing today?";
7  const myArray = text2.split(" ");
8  console.log(myArray);
9  //3
10 let text3 = "How are you, doing today?";
11 const myArray2 = text3.split(" ", 3);
12 console.log(myArray2)
```

```
node /tmp/TGHxjCVANO.js
[ 'hello' ]

[ 'How', 'are', 'you,', 'doing', 'today?' ]

[ 'How', 'are', 'you,' ]
```

JavaScript String method substring(*start,end*)

The substring() method extracts characters, between two indices (positions), from a string, and returns the substring.
The substring() method extracts characters from start to end (exclusive).
The substring() method does not change the original string.
If start is greater than end, arguments are swapped: (4, 1) = (1, 4).
Start or end values less than 0, are treated as 0.

Examples:

```
1  //  Extract a substring from text:
2  let text = "Hello world!";
3  let result = text.substring(2);
4  console.log(result)
5  //  If "start" is less than 0, it will start from index 0:
6  let text2 = "Hello world!";
7  let result2 = text2.substring(-3);
8  console.log(result2)
9  //  Only the last:
10 let text3 = "Hello world!";
11 let result3 = text3.substring(text.length - 1);
12 console.log(result3)
```

```
node /tmp/q12McE42mX.js
llo world!

Hello world!

!
```

JavaScript String method slice(*start, end*)

The **slice()** method **returns a shallow copy of a portion of an array into a new array object selected from start to end** (end not included) where start and end represent the index of items in that array.

Examples:

```javascript
// Slice the first 5 positions:
let text = "Hello world!";
let result = text.slice(0, 5);
console.log(result)
// From position 3 to the end:
let text2 = "Hello world!";
let result2 = text2.slice(3);
console.log(result2)
// The whole string:
let text3 = "Hello world!";
let result3 = text3.slice(0);
console.log(result3)
```

```
node /tmp/q12McE42mX.js
Hello

lo world!

Hello world!
```

JavaScript String method toLowerCase()

The toLowerCase() method converts a string to lowercase letters.
The toLowerCase() method does not change the original string.

Example:

```
// Convert to lowercase:
let text = "Hello World!";
let result = text.toLowerCase();
console.log(result)
```

```
node /tmp/q12McE42mX.js
hello world!
```

JavaScript String method toUpperCase()

The toUpperCase() method converts a string to uppercase letters, using current locale.
The toUpperCase() method does not change the original string.

Example:

```
// Convert to uppercase:
let text = "Hello World!";
let result = text.toLocaleUpperCase();
console.log(result)
```

```
node /tmp/q12McE42mX.js
HELLO WORLD!
```

JavaScript String method trim()

Method **trim()** removes whitespace from both sides of a string.
The **trim()** method does not change the original string.

Example:

```
1  // Remove spaces with trim():
2  let text = "       Hello World!        ";
3  let result = text.trim();
4  console.log(result)
```

```
node /tmp/q12McE42mX.js
HELLO WORLD!
```

JavaScript String method includes()

The includes() method returns true if a string contains a specified string.
Otherwise it returns false.
The includes() method is case sensitive.

Examples:

```
ludes "world".
let text = "Hello world, welcome to the universe.";
let result = text.includes("world");
console.log(result)
// Start at position 6:
let text2 = "Hello world, welcome to the universe.12";
let result2 = text2.includes("world", 6);
console.log(result2)
```

```
node /tmp/q12McE42mX.js
true
true
```

JavaScript String method toString()

The toString() method returns a string representing the object.

By default toString() takes no parameters.

C: > Users > adnan > OneDrive > Desktop > JS numtostr.js > ...

```
var n = 99;
console.log(typeof(n));
var st = n.toString();
console.log(typeof(st));
```

Code

PROBLEMS OUTPUT DEBUG CONSOLE ... Filter (e.g. text, !exclude)

```
C:\Program Files\nodejs\node.exe .\numtostr.js
number
string
```

Output

JavaScript String method indexOf()

The `indexOf()` method returns the position of the first occurrence of a value in a string.
The `indexOf()` method returns -1 if the value is not found.
The `indexOf()` method is case sensitive.

```
1  const message = "JavaScript is not Java";
2  const index = message.indexOf("is");
3  console.log('index: ' + index);  // index: 2
```

```
node /tmp/Ql80ZbdYct.js
index: 11
```

JavaScript String method repeat()

The repeat() method creates a new string by repeating the given string a specified number of times and returns it.

```
const holiday = "Happy holiday!";
const result = holiday.repeat(3);
console.log(result);
```

```
node /tmp/3SJlvCrvwM.js
Happy holiday!Happy holiday!Happy holiday!
```

JavaScript Number methods

JavaScript Number methods Math.round(),ceil(),floor()

The Math.floor() function rounds down a number to the next smallest integer.

```
let number = 38.8;
let roundedNumber = Math.floor(number);
console.log(roundedNumber);
```

```
node /tmp/Jww9kTELqr.js
38
```

The Math.round() function returns the number rounded to the nearest integer.

```
let number = 3.87;
let roundedNumber = Math.round(number);
console.log(roundedNumber);
```

```
node /tmp/mH3w7DA7Rw.js
4
```

The ceil() method rounds a decimal number up to the next largest integer and returns it.

```
let number = Math.ceil(4.3);
console.log(number);
```

```
node /tmp/yVro3yT8qw.js
5
```

JavaScript Number methods Math.max() and Math.min()

The max() method finds the maximum value among the specified values and returns it.

```
let numbers = Math.max(12, 4, 5, 9, 0, -3);
console.log(numbers);
```

```
node /tmp/Y7L1E2Mj22.js
12
```

The min() method finds the minimum value among the specified values and returns it.

```
let numbers = Math.min(12, 4, 5, 9, 0, -3);
console.log(numbers);
```

```
node /tmp/dNVmPNFZKB.js
-3
```

JavaScript Number methods Math.pow() and Math.sqrt()

The pow() method computes the power of a number by raising the second argument to the power of the first argument.

```
let power = Math.pow(5, 2);
console.log(power);
```

```
node /tmp/fnDIIV7ssk.js
25
```

The sqrt() method computes the square root of a specified number and returns it

```
let number = Math.sqrt(4);
console.log(number);
```

```
node /tmp/6ENxHqGWMY.js
2
```

JavaScript String method Math.abs() and Math.random()

The abs() method finds the absolute value of the specified number (without any sign) and returns it.

```
let number= Math.abs(-2);
console.log(number);
```

```
node /tmp/Mjdgn7d8Cr.js
2
```

The Math.random() function returns a floating-point, pseudo-random number between **0** (inclusive) and **1** (exclusive).

```
let randomNumber = Math.random()*10
console.log(randomNumber)
```

```
node /tmp/vLBwWfhdK3.js
4.051126874036138
```

JavaScript Number method isNaN()

The isNaN() function checks if a value is **NaN (Not-a-Number)** or not.

```
let number = NaN;
let number2 = 1;
let result = isNaN(number);
let result2 = isNaN(number2);
console.log("Is number a NaN ? ->", result);
console.log("Is number a NaN ? ->", result2);
```

```
node /tmp/TZgV5YfWrI.js
Is number a NaN ? -> true
Is number a NaN ? -> false
```

SOFT CLUB

What is **Array** in **JavaScript** ?

ARRAY IN JAVASCRIPT

An **array** is an object that holds values (of any type) not particularly in named properties/keys, but rather in numerically indexed position
In JavaScript, an array is an ordered list of values. Each value is called an element specified by an index. ... First, an array can hold values of mixed types.
An array is a special variable, which can hold more than one value:

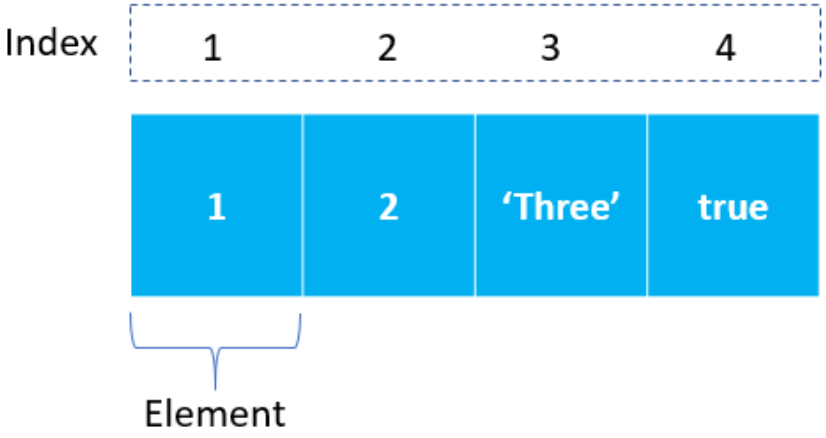
Creating an Array

1. Using an array literal

```
const array_name = [item1, item2, ...];
```

2. Using the new keyword

```
const array2 = new Array("eat", "sleep");
```



CHANGE ELEMENTS IN ARRAY

SOFT CLUB

You can also add elements or change the elements by accessing the index value

Suppose, an array has two elements. If you try to add an element at index 3 (fourth element), the third element will be undefined. For example,

```
// Example: 1
let array=["go","sleep","eat",1,true,{}]
array[3]=2
console.log(array)  // [ 'go', 'sleep', 'eat', 2, true, {} ]


// Example: 2
let array2=["go","sleep"]
array2[3]="eat"
console.log(array2)  // [ 'go', 'sleep', <1 empty item>, 'eat' ]
```

ARRAY METHODS

SOFT CLUB

pop()	forEach()	join()
shift()	map()	includes()
push()	find()	indexOf()
unshift()	filter()	splice()
concat()	reduce()	toString()
slice()	toSorted()	toReversed()

ARRAY METHOD PUSH

SOFT CLUB

The push() method adds one or more elements to the end of an array and returns the new length of the array.
The element(s) to add to the end of the array.

Syntax: push(element0, element1, /* … ,*/ elementN)

```
const animals = ['pigs', 'goats', 'sheep'];

const count = animals.push('cows',"dogs");
console.log(count);
// Expected output: 5
console.log(animals);
// Expected output: Array ["pigs", "goats", "sheep", "cows"]

animals.push('chickens', 'cats', 'dogs');
console.log(animals);
// Expected output: Array ["pigs", "goats", "sheep", "cows", "chickens", "cats", "dogs"]
```

ARRAY METHOD POP

The pop() method removes the last element from an array and returns that element. This method changes the length of the array.

Syntax: pop()

```
const plants = ['broccoli', 'cauliflower', 'cabbage', 'kale', 'tomato'];

console.log(plants.pop());
// Expected output: "tomato"

console.log(plants);
// Expected output: Array ["broccoli", "cauliflower", "cabbage", "kale"]

plants.pop();

console.log(plants);
// Expected output: Array ["broccoli", "cauliflower", "cabbage"]
```

ARRAY METHOD UNSHIFT

The unshift() method adds one or more elements to the beginning of an array and returns the new length of the array.

Syntax: unshift(element0, element1, /* … ,*/ elementN)

```
const array1 = [1, 2, 3];

console.log(array1.unshift(4, 5));
// Expected output: 5

console.log(array1);
// Expected output: Array [4, 5, 1, 2, 3]
```

ARRAY METHOD SHIFT

The pop() method removes the last element from an array and returns that element. This method changes the length of the array.

Syntax: shift()

```
const array1 = [1, 2, 3];

const firstElement = array1.shift();

console.log(array1);
// Expected output: Array [2, 3]

console.log(firstElement);
// Expected output: 1
```

ARRAY METHOD TOSTRING

The toString() method returns a string representing the specified array and its elements.
A string representing the elements of the array.

Syntax: toString()

```
const array1 = [1, 2, 'a', '1a'];

console.log(array1.toString());
// Expected output: "1,2,a,1a"
```

JAVASCRIPT ARRAY METHODS

SOFT CLUB

indexOf()

```
1  // Find the first index of "Apple":
2  const fruits = ["Banana", "Orange", "Apple", "Mango" , "Apple"];
3  let index = fruits.indexOf("Apple");
4  console.log(index)
5  //  Start at index 3:
6  const fruits2 = ["Banana", "Orange", "Apple", "Mango", "Apple"];
7  let index2 = fruits2.indexOf("Apple", 3 );
8  console.log(index2)
9  // Find the first index of "Apple", starting from the last element:
10 const fruits3 = ["Banana", "Orange", "Apple", "Mango", "Apple"];
11 let index3 = fruits3.indexOf("Apple", -1);
12 console.log(index3)
```

```
node /tmp/qhX0w49VLY.js
2

4

4
```

includes()

```
1  // includes() returns true if an array contains a specified element:
2  const fruits = ["Banana", "Orange", "Apple", "Mango"];
3  console.log(fruits.includes("Mango"));
4  // Check if fruit[] contains "Banana", starting the search from position 3:
5  const fruits2 = ["Banana", "Orange", "Apple", "Mango"];
6  console.log(fruits2.includes("Banana", 2));
```

```
node /tmp/qhX0w49VLY.js
true

false
```

JAVASCRIPT ARRAY METHODS

SOFT CLUB

slice()

```
const fruits = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const citrus = fruits.slice(1, 3);
console.log(citrus);

const fruits2 = ["Banana", "Orange", "Lemon", "Apple", "Mango"];
const myBest = fruits2.slice(-3, -1);
console.log(myBest);
```

```
node /tmp/qhX0w49VLY.js
[ 'Orange', 'Lemon' ]
[ 'Lemon', 'Apple' ]
```

concat()

```
const arr1 = ["Cecilie", "Lone"];
const arr2 = ["Emil", "Tobias", "Linus"];
const arr3 = ["Robin"];
const children = arr1.concat(arr2,arr3);
console.log(children);
```

```
node /tmp/qhX0w49VLY.js
[ 'Cecilie', 'Lone', 'Emil', 'Tobias', 'Linus', 'Robin' ]
```

JAVASCRIPT ARRAY METHODS

SOFT CLUB

splice()

```javascript
const fruits = ["Banana", "Orange", "Apple", "Mango"];
// At position 2, add 2 elements:
fruits.splice(2, 0 , "Lemon", "Kiwi");
console.log(fruits)


const fruits2 = ["Banana", "Orange", "Apple", "Mango", "Kiwi"];
// At position 2, remove 2 items:
fruits2.splice(2, 2);
console.log(fruits2)

var fruits3 = ["Banana", "Orange", "Apple", "Mango"];
// At position 2, add 2 elements, remove 1:
fruits3.splice(2, 1, "Lemon", "Kiwi");
console.log(fruits3)
```

```
node /tmp/qhX0w49VLY.js
[ 'Banana', 'Orange', 'Lemon', 'Kiwi', 'Apple', 'Mango' ]

[ 'Banana', 'Orange', 'Kiwi' ]

[ 'Banana', 'Orange', 'Lemon', 'Kiwi', 'Mango' ]
```

Syntax

```
array.splice(index, howmany, item1, ....., itemX)
```

JAVASCRIPT ARRAY METHODS CALLBACKS

JAVASCRIPT ARRAY METHOD MAP

SOFT CLUB

map()

```javascript
// Return a new array with the square root of all element values:
const numbers = [4, 9, 16, 25];
const newArr = numbers.map(Math.sqrt)
console.log(newArr)
// Multiply all the values in an array with 10:
const numbers2 = [65, 44, 12, 4];
const newArr2 = numbers2.map(myFunction)

function myFunction(num) {
  return num * 10;
}
console.log(newArr2)
```

```
node /tmp/qhX0w49VLY.js
[ 2, 3, 4, 5 ]
[ 650, 440, 120, 40 ]
```

JAVASCRIPT ARRAY METHOD FOREACH()

SOFT CLUB

forEach()

```javascript
const words = ['hello', 'bird', 'table', 'football', 'pipe', 'code'];
const capWords = words.forEach(capitalize);

function capitalize(word, index, arr) {
  arr[index] = word[0].toUpperCase() + word.substring(1);
}
console.log(words);
```

```
node /tmp/qhX0w49VLY.js
[ 'Hello', 'Bird', 'Table', 'Football', 'Pipe', 'Code' ]
```

Syntax

```
Array.forEach(callback(item, index, arr), thisValue)
```

JAVASCRIPT ARRAY METHOD FIND()

SOFT CLUB

find()

```
1  // Find the value of the first element with a value over 18:
2  const ages = [3, 10, 19, 20];
3  function checkAge(age) {
4    return age > 18;
5  }
6  console.log(ages.find(checkAge));
```

```
node /tmp/qhX0w49VLY.js
19
```

Syntax

```
array.find(function(currentValue, index, arr),thisValue)
```

METHOD REDUCE

The **reduce()** method executes a user-supplied "reducer" callback function on each element of the array, in order, passing in the return value from the calculation on the preceding element. The final result of running the reducer across all elements of the array is a single value.

```
const array1 = [1, 2, 3, 4];

// 0 + 1 + 2 + 3 + 4
const initialValue = 0;
const sumWithInitial = array1.reduce(
  (accumulator, currentValue) => accumulator + currentValue,
  initialValue
);

console.log(sumWithInitial);
// Expected output: 10
```

METHOD FILTER

The **filter()** method creates a shallow copy of a portion of a given array, filtered down to just the elements from the given array that pass the test implemented by the provided function.

```
const words = [1,2,4,50,5,6,7,8,88];
const result = words.filter(n => n>5);
console.log(result);
// Expected output: Array [ 50, 6, 7, 8, 88 ]
```

METHOD TOSORTED

The **toSorted()** method of Array instances is the copying version of the sort() method. It returns a new array with the elements sorted in ascending order.

```
const values = [1, 100, 21, 2];
const sortedValues = values.toSorted((a,b)=>a-b);
console.log(sortedValues); // [1, 2, 21, 100]
console.log(values); // [1, 100, 21, 2]
```

METHOD DESTRUCTURING

The destructuring **assignment** syntax is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables.

```
let a, b, rest;
[a, b] = [10, 20];

console.log(a);
// Expected output: 10

console.log(b);
// Expected output: 20

[a, b, ...rest] = [10, 20, 30, 40, 50];

console.log(rest);
// Expected output: Array [30, 40, 50]
```

METHOD SPREAD

The **spread (...)** syntax allows an iterable, such as an array or string, to be expanded in places where zero or more arguments (for function calls) or elements (for array literals) are expected. In an object literal, the spread syntax enumerates the properties of an object and adds the key-value pairs to the object being created.

```
function sum(x, y, z) {
  return x + y + z;
}

const numbers = [1, 2, 3];

console.log(sum(...numbers));
// Expected output: 6
```

METHOD REST

The **rest parameter** syntax allows a function to accept an indefinite number of arguments as an array.

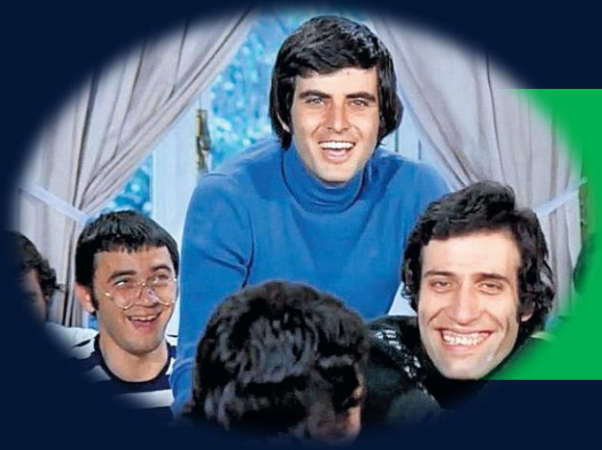
```
function sum(...theArgs) {
   return theArgs
}

console.log(sum(1, 2, 3));
// Expected output: [1, 2, 3]
```

SOFT CLUB

Thanks!

Be happy and Smile



THE END
LECTURE 3