JS Syntax Fundamentals

22 February 2023 08:29

JavaScript is a high-level programming language

* One of the core technologies of the World Wide Web
* Enables interactive web pages and applications
* Can be executed on the server and on the client
* C-like syntax
* Multi-paradigm
* Dynamic typing

What is Node.js?

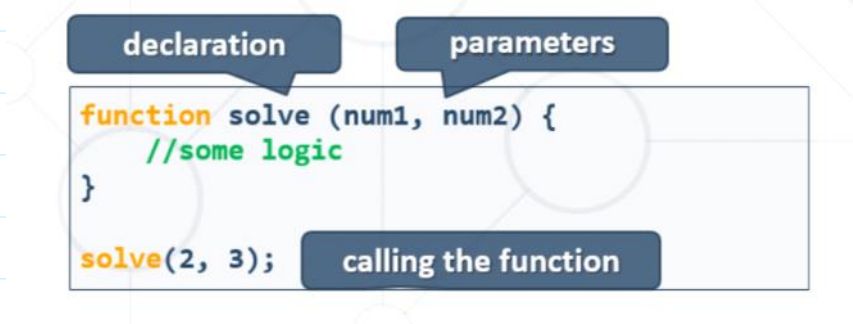
- Server-side JavaScript runtime

JavaScript Syntax

Defining and Initializing variables:

```
let a = 5;
let b = 10;
```

```
if (b > a) {
  console.log(b);
}
```



- Text can be composed easier using interpolated strings:

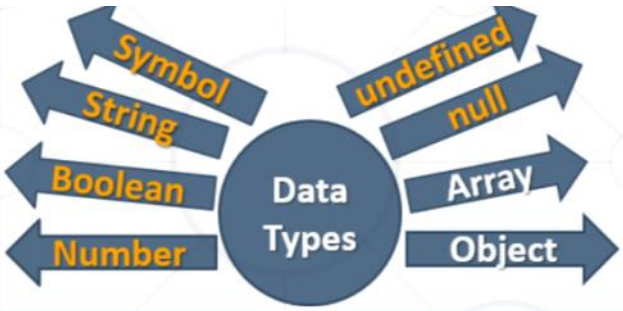
```
console.log(`The name is: ${name}, grade: ${grade}`);
```

- To format a number, use the toFixed() method (converts to string):

Number of decimal places

```
grade.toFixed(2);     //The name is: Petar, grade: 3.56
```

14




```
let number = 10;                                    // Number
let person = {name: 'George', age: 25};             // Object
let array = [1, 2, 3];                              // Array
let isTrue = true;                                  // Boolean
let name = 'George';                                // String
let empty = null;                                   // null
let unknown = undefined;                            // undefined
```

Недекларирана променлива.

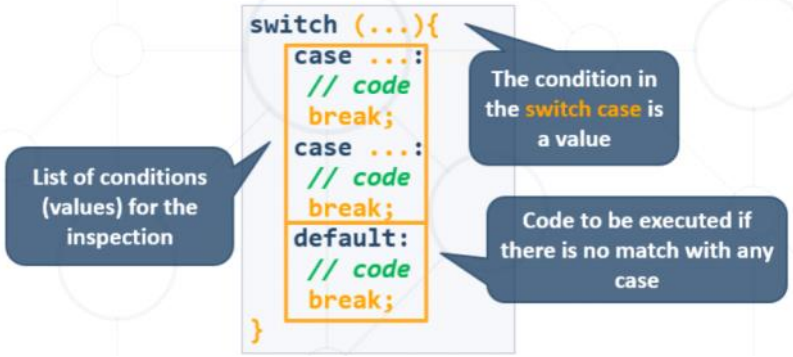
```
// Camel case
let myFirstName = '';
// Pascal case
SolveThisProblem
```

Comparison Operators

```
console.log(1 == '1');  // true
console.log(1 === '1'); // false
console.log(3 != '3');  // false
console.log(3 !== '3'); // true
console.log(5 < 5.5);   // true
console.log(5 <= 4);    // false
console.log(2 > 1.5);   // true
console.log(2 >= 2);    // true
console.log((5 > 7) ? 4 : 10); // 10
```

Ternary operator

- Works as a series of **if / else if / else if...**



- The **typeof** operator returns a string indicating the type of an operand

```
const val = 5;
console.log(typeof val);    // number
```

What is an Array?

- Arrays are **list-like objects**

- Arrays are a **reference type**, the variable points to an address in memory
- Neither the **length** of a JavaScript array **nor** the **types** of its elements are **fixed**
- An array's **length can be changed** at any time
- Data can be stored at non-contiguous locations in the array

- Array elements are accessed using their **index**

```
let cars = ['BMW', 'Audi', 'Opel'];
let firstCar = cars[0];   // BMW
let lastCar = cars[cars.length - 1];  // Opel
```

- Accessing indexes that do not exist in the array returns **undefined**

```
console.log(cars[3]);   // undefined
console.log(cars[-1]);  // undefined
```

Destructuring Syntax

Software University

- Expression that **unpacks values** from **arrays** or **objects**, into distinct **variables**

```
let numbers = [10, 20, 30, 40, 50];
let [a, b, ...elems] = numbers;

console.log(a) // 10
console.log(b) // 20
console.log(elems) // [30, 40, 50]
```

Rest operator

For-of Loop

Software University

- Iterates through all **elements** in a collection
- Cannot access the current index

```
for (let el of collection) {
    // Process the value here
}
```

Methods

Pop ->

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

Shift ->

The **shift()** method **removes** the **first element** from an array and **returns** that **removed element**

Unshift - >

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

Splice - >

- Changes the contents of an array by **removing** or **replacing** existing **elements** and / or **adding new** elements

```
let nums = [1, 3, 4, 5, 6];
nums.splice(1, 0, 2); // inserts at index 1
console.log(nums); // [ 1, 2, 3, 4, 5, 6 ]
nums.splice(4, 1, 19); // replaces 1 element at index 4
```

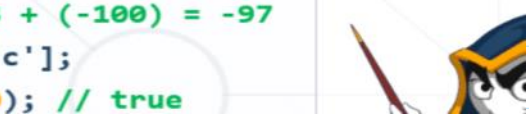
Reverse ->

- Reverses the array
  - The first array element becomes the last, and the last array element becomes the first

Includes ->

- Determines whether an array contains a certain element, returning true or false as appropriate

```
// array length is 3
// fromIndex is -100
// computed index is 3 + (-100) = -97
let arr = ['a', 'b', 'c'];
arr.includes('a', -100); // true
```



Предикат - функция, която се изпълнява върху друга функция.

indexOf ->

- The indexOf() method returns the first index at which a given element can be found in the array
  - Output is -1 if element is not present

ForEach ->

- The forEach() method executes a provided function once for each array element
- Converting a for loop to forEach

```
const items = ['item1', 'item2', 'item3'];
const copy = [];

// For Loop
for (let i = 0; i < items.length; i++) {
  copy.push(items[i]);
}

// ForEach
items.forEach(item => { copy.push(item); });
```

Map

- Creates a new array with the results of calling a provided function on every element in the calling array

```
let numbers = [1, 4, 9];
let roots = numbers.map(function(num, i, arr) {
  return Math.sqrt(num)
});
// roots is now [1, 2, 3]
// numbers is still [1, 4, 9]
```

Filter

- Creates a new array with filtered elements only
- Calls a provided callback function once for each element in an array
- Does not mutate the array on which it is called

```
let fruits = ['apple', 'banana', 'grapes', 'mango', 'orange'];
 // Filter array items based on search criteria (query)
function filterItems(arr, query) {
  return arr.filter(function(el) {
      return el.toLowerCase().indexOf(query.toLowerCase()) !== -1;
  });
};
console.log(filterItems(fruits, 'ap')); // ['apple', 'grapes']
```

```
// Mutator Methods (that change the array)
// splice, sort

// Accessor Methods (don't change the array)
// slice, includes, indexOf

// Iteration Methods
// forEach, map, filter (receive a callback)
```

Manipulating String

Use the "+" or the "+=" operators

Use the concat() method

- indexOf(substr)
- lastIndexOf(substr)

- substring(startIndex, endIndex)

- replace(search, replacement)

- repeat(count) - Creates a new string repeated count times

```
let n = 3;
for(let i = 1; i <= n; i++) {
  console.log('*'.repeat(i));
}
```

```
// *
// **
// ***
```

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

```
let text = "   Annoying spaces       ";
console.log(text.trim()); // Expected output: "Annoying spaces"
```

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

```
let text = "   Annoying spaces       ";
text = text.trimStart(); text = text.trimEnd();
console.log(text); // Expected output: "Annoying spaces"
```