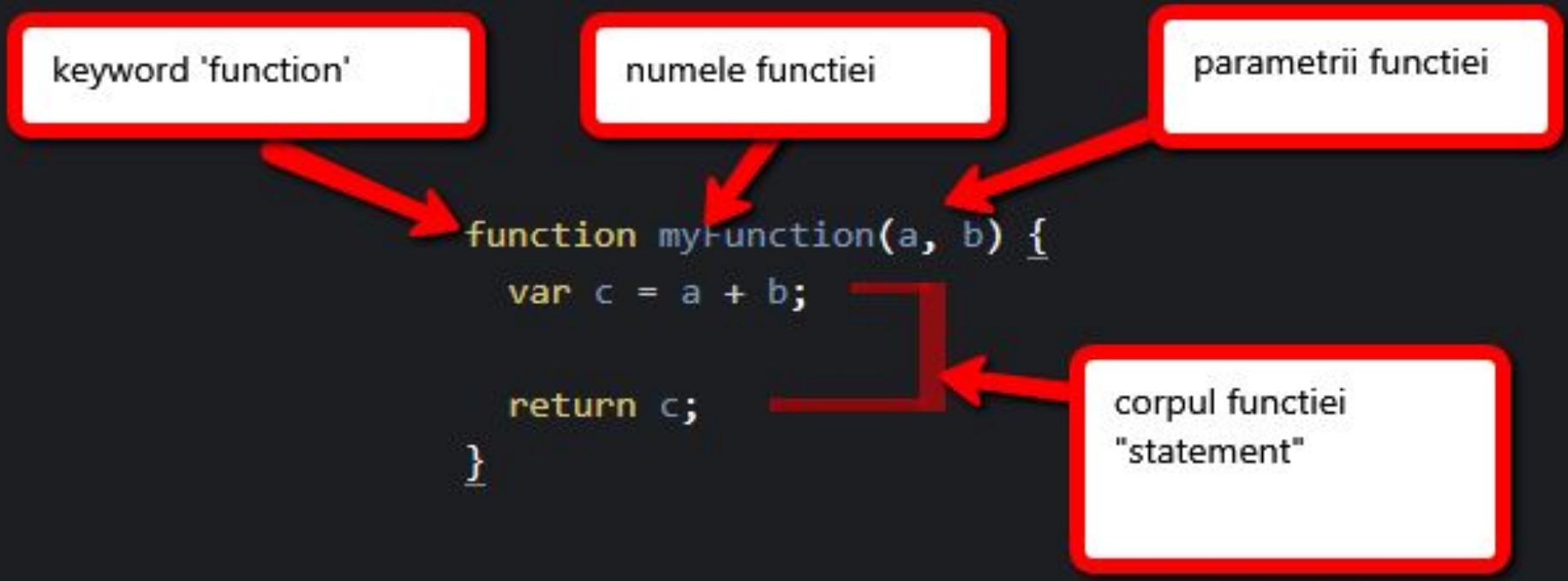
wantsome
The friendly IT Academy

Saptamana 5

Partea 2

Programare Front-End

1. JavaScript - Functions - General concept



JavaScript - Functions

- Functiile in JS se declara cu ajutorul *keyoword*-ului **function**
- Functiile incapsuleaza blocuri de instructiuni reutilizabile care indeplinesc unul / mai multe *task*-uri
- Avem doua tipuri de sintaxe specifice declararii unor functii: **function declaration** si **function expression**

Function declaration

```
function myFunction(a, b) {
  return a * b;
}
```

Function expression (sau ***functii anonime*** - nu au nume)

```
var x = function (a, b) {return a * b};
var z = x(4, 3);
```

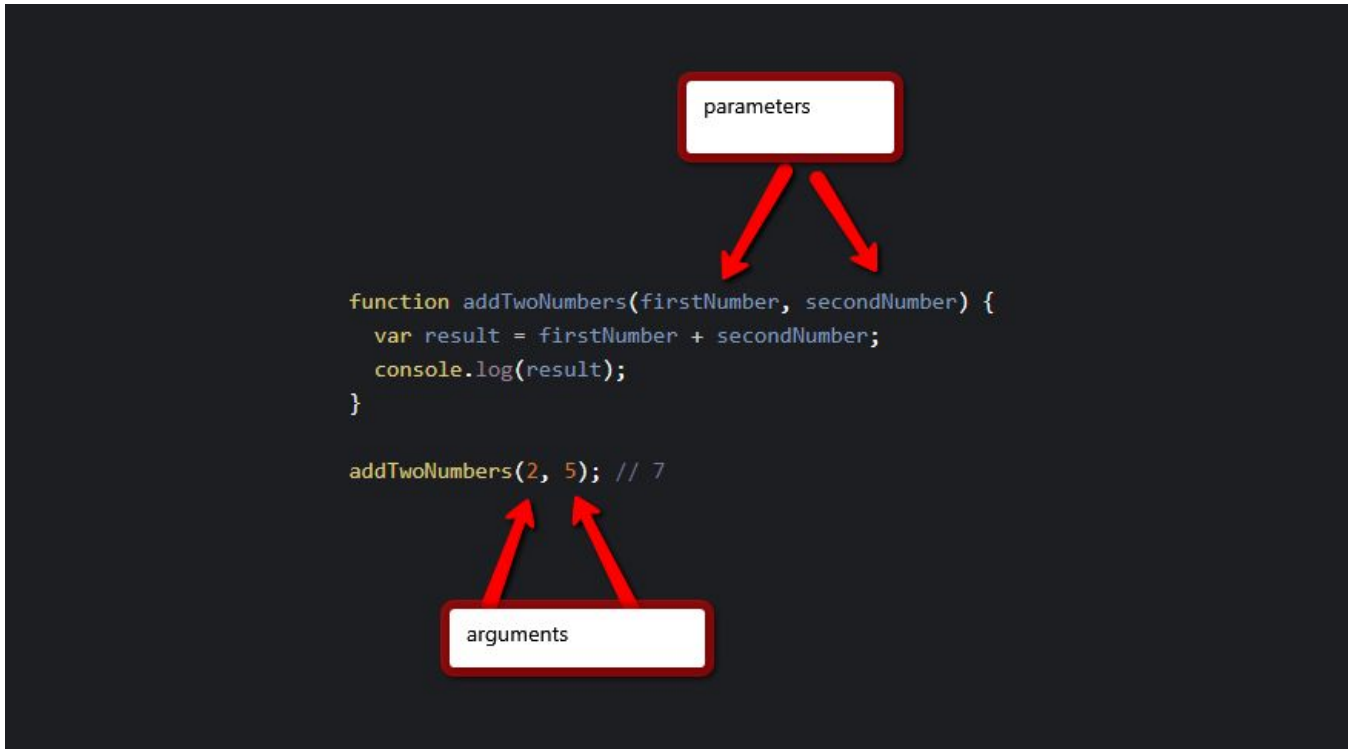
JavaScript - Functions

Function

parameters

and

Arguments



JavaScript - Functions

- Codul din interiorul une functii se va **executa** ori de cate ori se **invoca/chema (invoke function/call function)**
- Functia **NU** se executa atunci cand este **declarata** !

```
function addTwoNumbers(firstNumber, secondNumber) {
  var result = firstNumber + secondNumber;
  console.log(result);
}
// n-o sa vedem nimic in consola pentru ca functia nu este apelata.
```

JavaScript - Functions

- Orice functie in javascript trebuie sa **RETURNEZE** “ceva”; Daca nu specificam ce sa **return**eze va **return**a ***undefined***

```
function addTwoNumbers() {
}
console.log(addTwoNumbers());// undefined
```

```
function addTwoNumbers() {
  return true;
}
console.log(addTwoNumbers());// true
```

Atentie: instructiunea **return** opreste executia codului scris in continuare !

```
function isFun() {
  return 'yes';
  return 'no';
}
console.log(isFun());// 'yes'
```

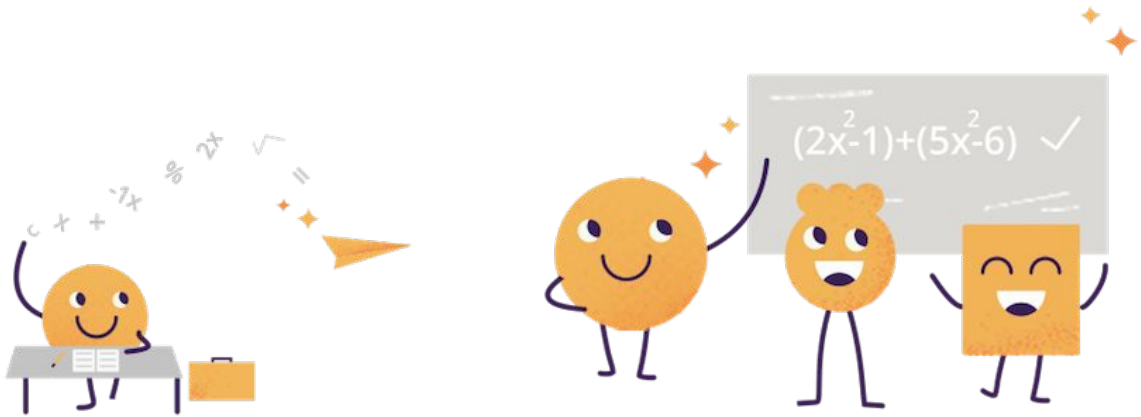
JavaScript - Functions

- Folosind **return** putem asigna valoarea returnata de catre o functie unei variabile

```
var multiply = function (a) {
  return a * 10;
}
var result = multiply(5);
console.log(result); //50
```

PRACTICE: **Functions**

http://bit.do/exJSFunctions



JavaScript - Strings

var myString = 'wewantJavaScript';

String Length(ex: console.log(myString.length) // 16) - verificam lungimea string-ului.

Special Characters(ex: **var** **aboutUs** = "Vrem sa "stim" JavaScript" // Uncaught SyntaxError: Invalid or unexpected token)

- Va fi interpretat Vrem sa Javascript dar ***stim*** e ceva ce nu se asteapta sa fie intre stringuri.
- Sa putem scrie ceva cu ghilimele vom folosi **backslash escape character.**

\'	'	Single quote
\"	"	Double quote
\\	\	Backslash

Ex : **var** **aboutUs** = "Vrem sa \"stim\" JavaScript" - va printa - Vrem sa stim Javascript

JavaScript - String Methods and Properties

indexOf() - Gasim indexul de unde incepe ceea ce cautam:

```
var string = "Ne place JavaScript"
var stringPosition = string.indexOf("place");
console.log(stringPosition);// 3
```

```
var string = "Ne place JavaScript"
var stringPosition = string.indexOf("plc");
console.log(stringPosition);// -1
```

lastIndexOf() - Gasim indexul la ultima aparitie a 'stringului' pe care trebuie sa-l gasim,

```
var string = "Ne place JavaScript si ne place CSS"
var stringPosition = string.lastIndexOf("place");
console.log(stringPosition);// 26
```

string.indexOf(searchValue, fromIndex) - ex string.indexOf('place', 10); - avem al doilea parametru unde ii zicem metodei de unde sa caute ce dorim sa gasim.

JavaScript - String Methods and Properties

search() - Ca idee principala se aseamana cu **indexOf().**

```
var string = "Hello World! Hello Wantsome!";
var pattern = /l+/g;
var result = str.match(pattern);
console.log(result) // ll,l,ll
```

- Diferentele dintre cele doua sunt ca **indexOf()** accepta al doilea parametru (fromIndex) iar **search()** poate face cautare dupa regexp.

REGEXP - O expresie care descrie un pattern de caractere.referinta regexp

JavaScript - String Methods and Properties

Metode de extragere ale string-urilor.

```
slice(start, end)
substring(start, end)
substr(start, length)
```

slice()

```
var string = "Mazda, Tesla, Toyota";
var result = string.slice(7, 12);
var result2 = string.slice(5, 11);
console.log(result) // Tesla
console.log(result2) // , Tesl
```

```
var string = "Mazda, Tesla, Toyota";
var result = string.slice(-13, -8);
console.log(result) // Tesla
```

- Daca parametrii sunt negativi se va incepe de la finalul stringului.
- Daca omitem ultimul parametru, va incepe de la primul numar si va face slice pana la finalul stringului.

JavaScript - String Methods and Properties

Metode de extragere ale string-urilor.

```
slice(start, end)
substring(start, end)
substr(start, length)
```

substring()

- Se comporta ca si slice() doar ca nu accepta index(indecsi?) negativi
- Mai multe diferente : link

substr()

- Se comporta ca si slice() doar ca la doilea parmetru este lungimea stringului

```
var string = "Mazda, Tesla, Toyota";
var result = string.substr(7, 4);
console.log(result) // Tesl
```

JavaScript - String Methods and Properties

replace() - Dupa cum ne asteptam, inlocuim un string cu alt string. (Metoda este caseSensitive)

```
var string = "Mazda, Tesla, Toyota";
var result = string.replace('Toyota', 'Hyundai');
console.log(result) // "Mazda, Tesla, Hyundai"
```

Convertirea stringurilor cu litere mari sau litere mici:

- **toLowerCase();**
- **toUpperCase();**

```
var string = "We love waNtSome";
var resultWithLower = string.toLowerCase();
var resultWithUpper = string.toUpperCase();
console.log(resultWithLower) // "we love wantsome"
console.log(resultWithUpper) // "WE LOVE WANTSOME"
```

JavaScript - String Methods and Properties

concat() - 'Concateneaza' / Uneste 2 sau mai multe stringuri.

```
var love = "We love";
var wantsome = "Wantsome!"

var result = love.concat(wantsome);
var result2 = love.concat(" ", wantsome);

console.log(result) // ""We loveWantsome!
console.log(result2) // ""We love Wantsome!
```

trim() - 'scoate' / remove la spatiile goale de la **inceput** si **sfarsit** dintr-un string.

```
var wantsome = "        We love Wantsome        ";
var result = wantsome.trim();
console.log(result) // "We love Wantsome"
```

JavaScript - String Methods and Properties

Metode de extras carterele unui string:

- `charAt(position)`
- `charCodeAt(position)`
- Property access [ ]

- **charAt()** - ne returneaza caracterul de la indexul(pozitia) specificat

```
var str = "Wantsome";
var result = str.charAt(0);
console.log(result) // "W"
```

- **charCodeAt()** - ne returneaza unicodul caracterului de la indexul(pozitia) specificat

```
var str = "Wantsome";
var result = str.charCodeAt(0);
console.log(result) // "87"
```

- **Property access** - ne returneaza stringul de la indexul specificat

```
var str = "Wantsome";
var result = str[0];
console.log(result) // "W"
```

JavaScript - String Methods and Properties

split() - Aceasta metoda transforma un string in array de acel string.

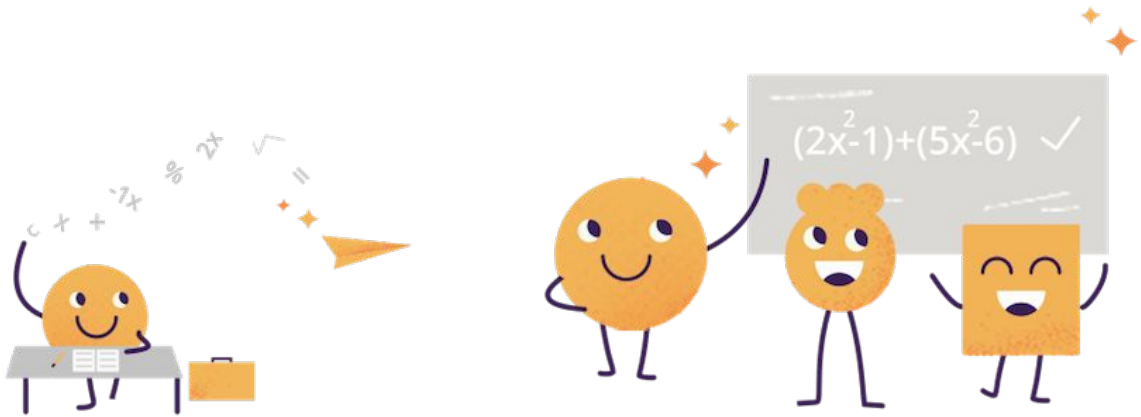
```
var str = "Wantsome";
var result = str.split();
console.log(result) // ["Wantsome"]
console.log(typeof(result)) // ??
```

```
var str = "Wantsome";
var result = str.split('');
console.log(result) // ["W", "a", "n", "t", "s", "o", "m", "e"]
```

```
var str = "Mazda,Tesla,Hyundai";
var result = str.split(',');
console.log(result) // ["Mazda", "Tesla", "Hyundai"]
```

PRACTICE: **String methods and properties**

http://bit.do/exJS-strings



JavaScript - Numbers

var x = 5;

var y = 5.25

```
var x = 123e5;    // 12300000
var y = 123e-5;   // 0.00123
```

Javascript Numbers Methods

isFinite() -

```
Number.isFinite(123) //true
Number.isFinite(-1.23) //true
Number.isFinite(0) //true
Number.isFinite('123') //false
Number.isFinite('Hello') //false
Number.isFinite(Infinity) //false
Number.isFinite(0 / 0) //false
```

isInteger()-

```
Number.isInteger(123) //true
Number.isInteger(-123) //true
Number.isInteger(0.5) //false
Number.isInteger('123') //false
Number.isInteger(false) //false
Number.isInteger(Infinity) //false
Number.isInteger(0 / 0) //false
```

JavaScript - Numbers

Javascript Numbers Methods

toFixed() - Converteste numarul in **string** si pastreaza numarul de decimale dupa punct cate am specificat.

```
var myNumber = 5.234353
var result = myNumber.toFixed(2);
console.log(result) // 5.23
console.log(typeof(result)) // string
```

toPrecision() - converteste numarul in **string** si se pastreaza length-ul dat.

```
var myNumber = 5.234353
var result = myNumber.toPrecision(2);
console.log(result) // 5.2
console.log(typeof(result)) // string
```

toString() - converteste un numar in string var num = 5 // num.toString() o sa fie num = “5”

Alte metode - **isNan(), isSafeInteger(), toExponential(x), valueOf().**