



# JavaScript

**ΜΑΘΗΜΑ 7**  
**ΕΙΣΑΓΩΓΗ ΣΤΙΣ**  
**ΣΥΝΑΡΤΗΣΕΙΣ**

## ΠΕΡΙΕΧΟΜΕΝΑ:

1. **Συναρτήσεις**
  1. Παράμετροι - Επιστρεφόμενη Τιμή
  2. Spread κατά την κλήση - αναδρομή
  3. Η JS είναι (μόνο) by value
  4. Debugging με τον Google Chrome
2. **Συναρτησιακές Εκφράσεις**
  1. Άλλες Χρήσεις των Συναρτησιακών Εκφράσεων
3. **Arrow Functions**

Γεώργιος Κ.

Σμαραγδένιος Χορηγός Μαθήματος

Εμμανουήλ Α.

Χρυσός Χορηγός Μαθήματος

- Μία **Συνάρτηση (Function)** είναι:
  - Ένα τμήμα κώδικα με ένα όνομα το οποίο έχει σχεδιαστεί για να κάνει μια συγκεκριμένη δουλειά

### Ορισμός συνάρτησης:

- Ακολουθούμε το πρότυπο:

```
function functionName(param1, param2, ...) {
    // do something
    return value;
}
```

- **function**: ορίζει ότι θα ακολουθήσει ορισμός συνάρτησης
- **functionName**: **όνομα** το οποίο ορίζουμε εμείς
- **param1, param2,...**: Είναι οι παράμετροι της συνάρτησης [0..n]. Εντίθενται σε παρενθέσεις.
- Ακολουθεί το **σώμα** με τις εντολές της συνάρτησης. Εντίθεται σε άγκιστρα.
- Στο σημείο που επιθυμούμε (συνήθως στο τέλος) ορίζουμε την επιστρεφόμενη τιμή με τη **return**
- Ενώ η **κλήση** της συνάρτησης, γίνεται ως:

```
myFunction(arg1, arg2,...)
```

### Παράδειγμα 1:

#### function definition

Μία συνάρτηση χωρίς παράμετρους και επιστρεφόμενη τιμή.

```
function hello() {
    console.log("Hello World!");
}
hello();
hello();
```

### Στη Μνήμη:

- Η συνάρτηση είναι και αυτή ένα αντικείμενο που σχετίζεται με ένα όνομα.
  - Έτσι π.χ. στο παράδειγμα 1, σκεφτόμαστε ότι με τη δήλωση κατασκευάζεται ένα αντικείμενο - συνάρτηση και το όνομα hello δένεται πάνω σε αυτό το αντικείμενο.
  - Θα σκεφτόμαστε ότι ένα αντικείμενο-συνάρτηση, έχει ένα όνομα, τις παραμέτρους του και το σώμα του κώδικά.
- Φαίνεται περίεργο, αλλά είναι πολύ σημαντικό!
  - Η JS είναι (για πολλούς) κυρίως συναρτησιακή γλώσσα.
  - Και λέμε ότι μία τέτοια απλή **συνάρτηση είναι αντικείμενο πρώτης τάξεως (first-class object)**.
  - [Θα δούμε στη συνέχεια των μαθημάτων ότι υπάρχουν και 2ης τάξεως (συνάρτηση που παίρνει όρισμα συνάρτηση) κ.ο.κ.]

### Παράδειγμα 2: function name

```
function hello() {
    console.log("Hello World!");
}
let myFunc = hello;
myFunc();
console.log(typeof myFunc, typeof hello, myFunc.name);
```

### Παράμετροι και Ορίσματα

- Η συνάρτηση μπορεί να έχει 0 ή παραπάνω παραμέτρους. Κατά την κλήση, μπορούμε να διοχετεύσουμε λιγότερα ή περισσότερα ορίσματα από τις παραμέτρους, χωρίς να προκαλείται συντακτικό λάθος.
- **rest operator (...name)** (τρεις τελείες ακολουθούμενες από ένα όνομα):
  - Μπαίνει το πολύ μία φορά, ως τελευταία παράμετρος και «μαζεύει» τα υπόλοιπα ορίσματα σε έναν πίνακα.
- **Default τιμές παραμέτρων (param=value):**
  - Τίθεται η τιμή value στην param, αν δεν έχει καθοριστεί τιμή, ή έχει διοχετευτεί ως όρισμα: undefined.
  - Προσοχή: Η default τιμή κάθε παραμέτρου, εφόσον δεν καθοριστεί διαφορετικά, είναι undefined.
- **Η έμμεση παράμετρος arguments:**
  - Είναι ένα αντικείμενο που περιέχει όλα τα ορίσματα.
  - Είναι κάτι-σαν-πίνακας (υποστηρίζει τις arguments.length, arguments[i], spread operator)

### Επιστρεφόμενη τιμή

- Αν δεν κάνουμε return, τότε επιστρέφεται αυτόματα undefined από την κλήση της συνάρτησης.
- Αν κάνουμε return **επιστρέφουμε ακριβώς μία τιμή.**

### Παράδειγμα 3: parameters\_rest

```
function mult(initial, ...rest) {
  let result = initial;
  for (let item of rest)
    result *= item;
  return result;
}
console.log(mult(1,2,3,4,5));
```

### Παράδειγμα 4: parameters\_default

```
function myFunc(a,b="default",c) {
  console.log(a,b,c);
}
myFunc();
myFunc(1);
myFunc(1,2);
myFunc(1,2,3);
myFunc(1,undefined,3);
```

### Παράδειγμα 5: parameters\_arguments

```
function myFunc() {
  for (let item of arguments)
    console.log(item);
}
myFunc(1,2,3)
```

### Παράδειγμα 6: parameters\_arguments

```
function myFunc() {}
console.log(myFunc());
```

### Άσκηση 1:

- Κατασκευάστε μία συνάρτηση με όνομα average η οποία δέχεται αυθαίρετο πλήθος αριθμών και υπολογίζει και επιστρέφει το μέσο όρο τους.
- Αν υπάρχουν ορίσματα που δεν είναι αριθμοί, τότε αυτά να απορρίπτονται κατά τον υπολογισμό.

- Υπενθύμιση από Μάθημα 6.1:
  - Ο τελεστής spread απλώνει τα στοιχεία ενός πίνακα μέσα σε έναν άλλο πίνακα, π.χ.:

```
let array = [1, 2, 3, 4, 5];  
let array2 = [0, 0, ...array, 1, 1];
```

- Ο ίδιος τελεστής μπορεί να χρησιμοποιηθεί για να απλώσει τα στοιχεία ενός πίνακα ως ορίσματα κατά την κλήση μιας συνάρτησης (βλ. παράδειγμα)

#### Παράδειγμα 7: argument\_spread\_operator

```
let array = [1,2,3];  
  
function myFunc() {  
  let sum=0;  
  for (let item of arguments) {  
    sum += item;  
  }  
  return sum;  
}  
  
console.log(myFunc(2, ...array, 4));
```

#### Αναδρομή:

- Η JavaScript υποστηρίζει αναδρομή
- (Δεν είναι τόσο συχνό, αλλά βλέπουμε δύο παραδείγματα)

#### Παράδειγμα 8: factorial

```
function factorial(n) {  
  if (n===0)  
    return 1;  
  else  
    return n*factorial(n-1);  
}  
  
for (let i=0; i<10; i++)  
  console.log(`factorial(${i})=${factorial(i)}`)
```

#### Παράδειγμα 9: fibonacci

```
function fibonacci(n) {  
  if (n===0 || n===1)  
    return 1;  
  else  
    return fibonacci(n-1)+fibonacci(n-2);  
}  
  
for (let i=0; i<10; i++)  
  console.log(`fibonacci(${i})=${fibonacci(i)}`)
```

#### Η JS είναι μόνο by value!

- Για primitives - παραμέτρους σε συνάρτηση:
  - Οποιαδήποτε αλλαγή της τιμής μιας παραμέτρου μέσα στη συνάρτηση, δεν διατηρείται μετά το πέρας της κλήσης.




#### Σημείωση:

- Έρχεται σε αντίθεση με άλλες γλώσσες προγραμματισμού (π.χ. C/C++) οι οποίες έχουν και άλλους τρόπους διοχέτευσης ορισμάτων (by reference/by pointer) που διοχετεύουν αυτούσια τη μεταβλητή ως όρισμα.

#### Παράδειγμα 10: by value primitive

```
function myFunction(x) {
  console.log(`inside myFunction: ${x}`);
  x=2;
  console.log(`inside myFunction: ${x}`);
}

let a = 1;
console.log(`outside myFunction: ${a}`);
myFunction(a);
console.log(`outside myFunction: ${a}`);
```



 top ▼
 
 Filter

outside myFunction: 1
inside myFunction: 1
inside myFunction: 2
outside myFunction: 1




- Για αντικείμενα - παραμέτρους σε συνάρτηση (όπως π.χ. οι πίνακες)
  - Αν αλλάξουμε το ίδιο το αντικείμενο (π.χ. ορίζοντας ένα νέο αντικείμενο στο ίδιο όνομα) οι αλλαγές δεν διατηρούνται μετά το πέρας της κλήσης.
  - Αν αλλοιώσουμε τις ιδιότητες του αντικειμένου (π.χ. να προσθέσουμε τιμές σε έναν πίνακα), τότε αυτές οι αλλαγές διατηρούνται μετά το πέρας της κλήσης.

#### Παράδειγμα 11: by value object

```
function alterProperty(ar) {
  ar.push(2);
}

function alterTable(ar) {
  ar = [1,2,3,4,5];
}

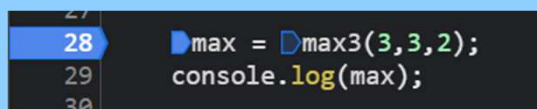
let array = [1];
alterProperty(array);
console.log(`Altering a property: ${array}`);
alterTable(array);
console.log(`Altering the table: ${array}`);
```



 top ▼
 
 Filter

Altering a property: 1,2
Altering the table: 1,2

**Debugging με τον Google Chrome**

- Επιλογή “Sources” στα Developer Tools (F12)
- Βάζουμε ένα breakpoint (σημείο παύσης) στα σημεία του προγράμματος που θέλουμε να ελέγξουμε (κλικ στον αριθμό γραμμής):



- Πατάμε Refresh για να ξανατρέξει ο κώδικας.
- Η εκτέλεση του script θα διακοπεί στο σημείο του breakpoint.
- Δεξιά πάνω έχουμε τις επιλογές:



- που αντίστοιχα είναι:
  - Εκτέλεση επόμενης εντολής, χωρίς να μπει σε κώδικα συνάρτησης
  - Εκτέλεση επόμενης εντολής, μπαίνοντας σε κώδικα συνάρτησης.
  - Εκτέλεση επόμενης εντολής, έξω από την τρέχουσα συνάρτηση.
- Παρατηρήστε (βλ. και βίντεο):
  - Την καρτέλα «Watch» στην οποία προσθέτουμε υπολογιζόμενη έκφραση
  - Την καρτέλα «Scope» με τις τιμές των ονομάτων στην τρέχουσα εμβέλεια.

**Άσκηση 2:**

- Με τη βοήθεια του debugger του google chrome, εντοπίστε το λάθος στον ακόλουθο κώδικα:

```
function max3(a,b,c) {  
  if (a>b && a>c)  
    return a;  
  else if (b>a && b>c)  
    return b;  
  else  
    return c;  
}  
  
let max = max3(1,2,3);  
console.log(max);  
  
max = max3(3,2,1);  
console.log(max);  
  
max = max3(3,3,2);  
console.log(max);
```

### Ανώνυμες Συναρτήσεις (Anonymous Functions)

- Είναι συνάρτησεις χωρίς όνομα.
- Σύνταξη (παράμετροι και σώμα: ίδιοι κανόνες όπως πριν)

```
function(parameters) {
  ...
}
```

### Παράδειγμα 12: function expression

```
let add = function(x, y) {
  return x+y;
}
console.log(add(1, 2));
let func = add;
console.log(add.name, func.name);
```

### Παρατήρηση:

- Μία ανώνυμη συνάρτηση, δεν ευσταθεί ως εντολή. Ωστόσο, μπορεί να είναι μέρος σε πιο σύνθετη εντολή:
  - Με παρενθέσεις (δημιουργεί αντικείμενο)
  - Σε συναρτησιακή έκφραση (ορισμός συνάρτησης)
  - Ως όρισμα σε άλλη συνάρτηση (callback)
  - Όστε να εκτελεστεί αμέσως (IIFE)

### Διαφορές Ορισμού Συνάρτησης από Συναρτησιακή Έκφραση:

- Hoisting:
  - Σε συναρτήσεις που ορίζονται με δήλωση, γίνεται hoisting.
  - Σε συναρτήσεις που ορίζονται με συναρτησιακή έκφραση, δεν γίνεται hoisting
- Μέλος name του αντικειμένου - συνάρτηση
  - Με δήλωση - συναρτησιακή έκφραση έχει το όνομα που δίνουμε στη «μεταβλητή»
  - Σε άλλες χρήσεις (όπως π.χ. στο IIFE), το όνομα της είναι η κενή συμβολοσειρά.

### Παράδειγμα:

```
> function(){}
✖ Uncaught SyntaxError:
> (function(){}())
< 1
```

### Συναρτησιακές Εκφράσεις (Function Expressions)

- Είναι η χρήση ανώνυμων συναρτήσεων σε πιο περίπλοκες εκφράσεις (που είναι συντακτικά έγκυρες), π.χ. η ακόλουθη έκφραση, ορίζει μία συνάρτηση και τη σχετίζει με ένα όνομα.

```
let name = function(parameters) {
  ...
}
```

### Παράδειγμα 13: function hoisting

```
declared();
function declared() {
  console.log("declared");
}

expressed();
var expressed = function() {
  console.log("expressed");
}
```



### Συναρτησιακή Έκφραση: Όρισμα σε συνάρτηση

- (Πολύ συχνή χρήση) αναφέρεται και ως callback.
- Μία συνάρτηση που παίρνει ως όρισμα μια άλλη συνάρτηση

### Παράδειγμα 14: function expression as argument

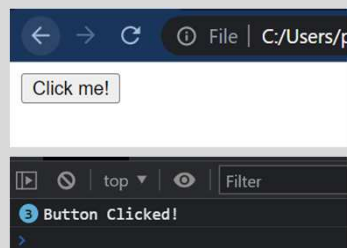
```
function calculator(left, right, operand) {
  return operand(left, right);
}

let add = calculator(2, 3, function(a,b){return a+b});
let sub = calculator(2, 3, function(a,b){return a-b});
console.log(add, sub);
```

- Στο μέλλον θα δούμε δύο καθιερωμένες χρήσεις:
  - Για να κάνουμε register κώδικα που θα τρέξει όταν π.χ. θα γίνει κλικ σε ένα κουμπι
  - Στην εκτέλεση ασύγχρονου κώδικα

### Παράδειγμα 15: function expression as callback

```
<button id="btn">Click me!</button>
<script>
  document.querySelector("#btn").addEventListener(
    "click",
    function() {
      console.log("Button Clicked!");
    }
  );
</script>
```



### Συναρτησιακή Έκφραση: IIFE

- IIFE = Immediately Invoked Function Expression
- Μία ανώνυμη συνάρτηση που ενεργοποιείται αμέσως

### Παράδειγμα 16: iife

```
(function(){
  // something useful
})();
```

- Το πλεονέκτημα του IIFE έχει να κάνει με την εμβέλεια:
  - Θα δούμε σε αναλυτικό επόμενο μάθημα, ότι η συνάρτηση ορίζει τη δική της εμβέλεια: Οι τοπικές μεταβλητές σε συνάρτηση (ακόμη και αν έχουν δηλωθεί με var) δεν μεταφέρονται στην αρχή του κώδικα (είναι τοπικές στη συνάρτηση).
  - Παλιότερα (προ ES6), αυτό ήταν ένα σημαντικό θέμα: Μέσω του IIFE μπορούσε ο κώδικας να χωριστεί σε μέρη, καθένα με το δικό του «χώρο ονομάτων»
  - Από την ES6 και μετά, ο χωρισμός σε μέρη, το κάθε ένα με τη δική του εμβέλεια, γίνεται με τα modules.
- Συνεπώς ήταν πολύ συχνό παλιότερα (προ ES6), αλλά σήμερα η χρήση modules είναι καθιερωμένη και προτιμώτερη.



**Arrow Functions (ή Lambda Functions)**

- Νέος τρόπος (ES6) για να ορίσουμε μια συνάρτηση
- Σύνταξη (παράμετροι και σώμα: ίδιοι κανόνες όπως πριν)

```
(parameters) => {  
  ...  
}
```

- Είναι μια συντακτική σύντμηση της συναρτησιακής έκφρασης με ίδια συμπεριφορά (διαφορές μόνο σε προχωρημένα θέματα όπως το this (επόμενο μάθημα))

**Παράδειγμα 17: arrow functions**

```
let add = (param1, param2) => {  
  return param1+param2;  
};  
console.log(add(1,2));
```

**Άσκηση 3:**

- Μετατρέψτε το παράδειγμα «function\_expression\_as\_argument», ώστε να χρησιμοποιεί arrow function, αντί για συναρτησιακές εκφράσεις.

**Συντακτική Παραλλαγή για τις παραμέτρους:**

- Αν έχει ακριβώς μία παράμετρο, οι παρενθέσεις μπορούν να παραληφθούν:

```
param => {...}
```

- (Σε κάθε άλλη περίπτωση οι παρενθέσεις είναι υποχρεωτικές)

**Συντακτικές Παραλλαγές για το Σώμα:**

- Αν το σώμα είναι απλά μια επιστρεφόμενη έκφραση/τιμή, τότε δεν απαιτείται ούτε η return, ούτε τα άγκιστρα, π.χ.:

```
x => x+x
```

- (Ειδικά όμως για την περίπτωση που η επιστρεφόμενη τιμή είναι αντικείμενο, το παραπάνω απαιτεί παρενθέσεις)

```
x => ({name: value})
```

**Προσοχή:**

- Για αποφυγή ιδιαίτερων συντακτικών λαθών, οι παράμετροι και το βέλος, πρέπει να είναι στην ίδια γραμμή.

**Παράδειγμα 18: arrow function as callback**

```
document.querySelector("#btn").addEventListener(  
  "click",  
  ()=>console.log("Button Clicked!")  
);
```