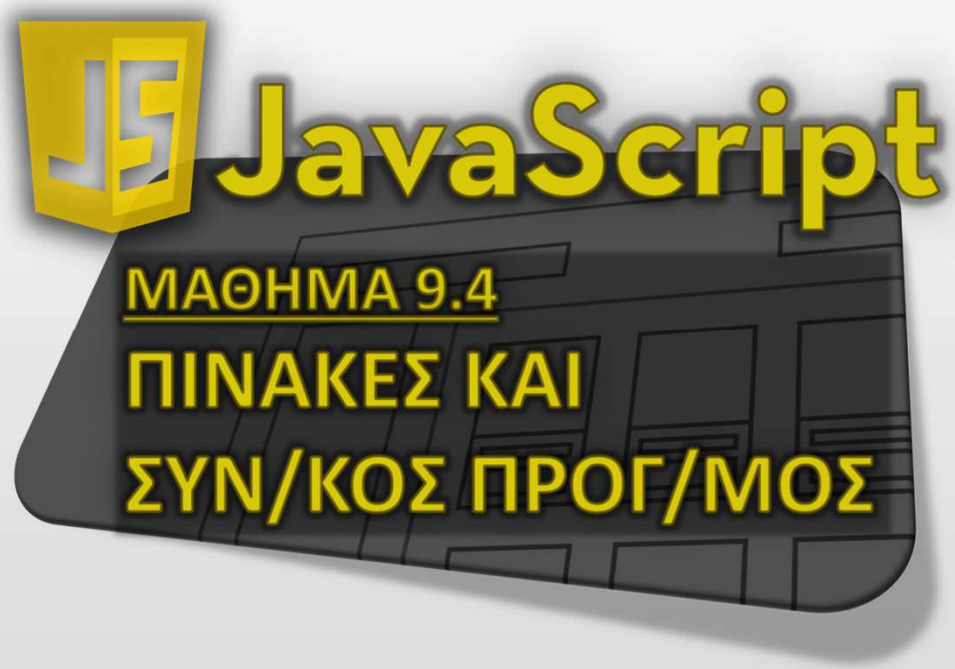



ΠΕΡΙΕΧΟΜΕΝΑ:

# ΜΑΘΗΜΑ 9.4: Πίνακες και Συν/κός Προγ/μός — 1. Συναρτησιακός Προγραμματισμός

JavaScript psounis

## Συναρτησιακός Προγραμματισμός

- Ο **συναρτησιακός προγραμματισμός (functional programming)** βασίζεται στο μοντέλο του λ-λογισμού (Church, 1930s) και είχε εκφραστεί αρχικά από γλώσσες όπως οι Lisp και η Haskell
- Οι δημοφιλείς γλώσσες (Python, Java, JS κ.λπ.) ενσωμάτωσαν ιδέες από τον συναρτησιακό προγραμματισμό στο συντακτικό τους. Βλέπουμε εδώ κάποιες ιδέες που ενσωματώθηκαν στην JS.

## Ιδέα 1:

- Οι συναρτήσεις δεν πρέπει να «αλλάζουν» την κατάσταση ενός αντικειμένου (λέμε ότι **δεν έχουν «παρενέργειες» (side effects)**)
- Π.χ. να προτιμούμε να επιστρέφουμε νέα εκδοχή του αντικειμένου αντί να τροποποιούμε την υφιστάμενη.

## Παράδειγμα 1: increment_array_functional

```
function incrementArrayNonFunctional(array, value) {
  for (let i=0; i<array.length; i++)
    array[i] += value;
}
function incrementFunctional(array, value) {
  let newArray = [...array];
  for (let i=0; i<array.length; i++)
    newArray[i] += value;
  return newArray;
}
```

## Σημείωση:

- Τα immutable αντικείμενα (π.χ. strings) είναι χαρακτηριστικό παράδειγμα κλάσεων, που οι μέθοδοί τους δεν έχουν παρενέργειες.

## Ιδέα 2:

- Κάθε συνάρτηση πρέπει να είναι «αυτόνομη», να μην εξαρτάται από άλλα αντικείμενα (π.χ. να παίρνει ορίσματα αντί εξαρτήσεων από καθολικές μεταβλητές).

## Παράδειγμα 2: prefix_functional

```
const prefix = "Sir";
function prefixNameNonFunctional(name) {
  return prefix + name;
}
function prefixNameFunctional(name, prefix) {
  return prefix + name;
}
```

## Ιδέα 3:

- Σύνθετες ενέργειες μπορούν να γίνουν με αλυσίδωση ενεργειών (function call chaining) ή με συναρτήσεις ανώτερης ταξης (higher order functions)

## Παράδειγμα 3: higher_order_function

```
function f(x) { // first order function
  return x*x;
}
function g(func, array) { // higher order function
  let newArray = [...array];
  for (let i=0; i<newArray.length; i++)
    newArray[i] = f(newArray[i]);
  return newArray;
}
let array = [1,2,3];
console.log(g(f,array));
```

# ΜΑΘΗΜΑ 9.4: Πίνακες και Συν/κός Προγ/μός — 1.1. Η μέθοδος map()

JavaScript psounis

- Η **μέθοδος map() του Array.prototype:**

```
map(function)
```

  - Εφαρμόζει τη συνάρτηση function, διαδοχικά, πάνω στα στοιχεία ενός πίνακα
  - Επιστρέφει νεό πίνακα με τα αποτελέσματα της εφαρμογής της συνάρτησης

**Παράδειγμα 4: map**

```
function square(x) {
   return x*x;
}

let array = [1,2,3,4,5];
let squares = array.map(square);
console.log(squares);
```

```
▼Array(5) i
   0: 1
   1: 4
   2: 9
   3: 16
   4: 25
   length: 5
▶ [[Prototype]]: Array(0)
```

**Παρατήρηση:**

- Συχνά χρησιμοποιούνται arrow functions ως όρισμα στη map (που οδηγεί σε ιδιαίτερα συμπαγή απεικόνιση σύνθετων ενεργειών.

**Παράδειγμα 5: map_arrow**

```
console.log([1,2,3,4,5].map(x=>x*x));
```

**Άσκηση 1:**

- Θέωρήστε έναν πίνακα από άτομα σαν τον ακόλουθο (βλ. exercise01_initial.html):

```
let persons = [
  {name: "John Snow", occupation: "King in the North", house: "Stark"},
  {name: "Daenerys Targaryen",
   occupation: "Queen of the Seven Kingdoms", house: "Targaryen"},
  {name: "Brandon Stark",
   occupation: "Three-eyed Raven", house: "Stark"}
]
```

- Χρησιμοποιήστε την map επί αυτού του πίνακα, ώστε να πάρετε νέους πίνακες:
  - Έναν πίνακα που περιέχει μόνο τις συμβολοσειρές - ονόματα των ατόμων:

```
▼Array(3) i
   0: "John Snow"
   1: "Daenerys Targaryen"
   2: "Brandon Stark"
```

  - Έναν πίνακα που περιέχει αντικείμενα που περιέχουν μόνο το όνομα και τον οίκο του ατόμου:

```
▼(3) [{…}, {…}, {…}] i
  ▶0: {name: 'John Snow', house: 'Stark'}
  ▶1: {name: 'Daenerys Targaryen', house: 'Targaryen'}
  ▶2: {name: 'Brandon Stark', house: 'Stark'}
```

**Παρατήρηση:**

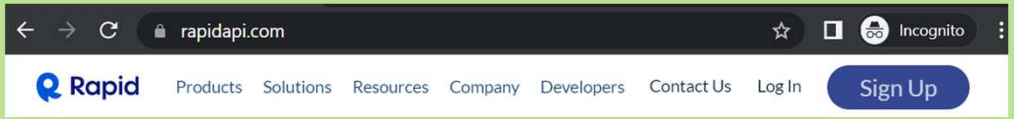
- Η ακόλουθη διαδικασία είναι αυξημένης δυσκολίας (βλ. υποχρεωτικά και βίντεο)

**Δεδομένα από REST API:**

- REST API: «Χοντρικά»: ένα URL προκαλεί ενέργειες στον Server
- Π.χ. μπορούμε να «χτυπήσουμε» ένα URL του IMDB και να μας επιστρέφεται μία λίστα από ταινίες σε μορφή JSON.

**Προετοιμασία σύνδεσης με REST API IMDB:**

- Κάνετε εγγραφή στη σελίδα rapidapi.com (sign up)




- Αφού ολοκληρώσετε την εγγραφή, αναζητήστε το API του IMDB και κάνετε εγγραφή σε αυτό::


- Στον κώδικα (δεξιά) επιλέξτε JavaScript και έπειτα «fetch»:


- Αντιγράψτε τον κώδικα και παρατηρήστε ότι αποτελείται από μία δήλωση και μία εντολή. Η δήλωση είναι:

```
const options = {
  method: 'GET',
  headers: {
    'X-RapidAPI-Key': '34af04aad3msheeae8e9eb10bf26p1b6986jsn3410ef566fbd',
    'X-RapidAPI-Host': 'imdb8.p.rapidapi.com'
  }
};
```

  - Εδώ δηλώνονται απαραίτητες ρυθμίσεις (προσωπικοι «κωδικοί») για τη σύνδεση στο imdb μέσω του rapidapi (δεν το πειράζουμε)

- Τροποποιήστε τον κώδικα της εντολής ως ακολούθως:

```
let query = "game of ";
fetch('https://imdb8.p.rapidapi.com/auto-complete?q=' + query, options)
  .then(response => response.json())
  .then(response => {
    console.log(response);
  })
  .catch(err => console.error(err));
```

  - Εδώ δίνουμε εντολή να μας «φέρουν» (fetch) τα δεδομένα σε ένα ερώτημα (τίτλοι οι όποίοι περιέχουν το «game of»)
  - Στο τελευταίο “then” γράφουμε κώδικα διαχείρισης του αποτελέσματος του ερωτήματος.

- (βλ. βίντεο και imdb_api_example.html)

**Άσκηση 2:**

- Χρησιμοποιήστε το Rapid API για να αναζητήσετε όλους τους τίτλους που περιέχουν το “Matrix”
- Εκτυπώστε έναν πίνακα με τα αποτελέσματα, που να περιέχει μόνο τους τίτλους τους ταινιών:

```
(8) ['The Matrix', 'The Matrix Resurrections', 'The Matrix Reloaded', 'The Matrix Revolutions', 'Matrix', 'Matrix', 'A Glitch in the Matrix', "Making 'The Matrix'"]
  0: "The Matrix"
  1: "The Matrix Resurrections"
  2: "The Matrix Reloaded"
  3: "The Matrix Revolutions"
  4: "Matrix"
  5: "Matrix"
  6: "A Glitch in the Matrix"
  7: "Making 'The Matrix'"
  length: 8
```

**Άσκηση 3:**

- Χρησιμοποιήστε το Rapid API για να αναζητήσετε όλους τους τίτλους που περιέχουν το “Matrix”
- Εκτυπώστε έναν πίνακα με τα αποτελέσματα, που να περιέχει για κάθε τίτλο: τον τίτλο, το έτος και τους ηθοποιούς, π.χ.:

```
Array(8)
  0: {title: 'The Matrix', year: 1999, actors: 'Keanu Reeves, Laurence Fishburne'}
  1: {title: 'The Matrix Resurrections', year: 2021, actors: 'Keanu Reeves, Carrie-Anne Moss'}
  2: {title: 'The Matrix Reloaded', year: 2003, actors: 'Keanu Reeves, Laurence Fishburne'}
  3: {title: 'The Matrix Revolutions', year: 2003, actors: 'Keanu Reeves, Laurence Fishburne'}
  4: {title: 'Matrix', year: 1993, actors: 'Nick Mancuso, Phillip Jarrett'}
  5: {title: 'Matrix', year: 2020, actors: 'Chris Harvey'}
  6: {title: 'A Glitch in the Matrix', year: 2021, actors: 'Nick Bostrom, Joshua Cooke'}
  7: {title: "Making 'The Matrix'", year: 1999, actors: 'Laurence Fishburne, Joe Pantoliano'}
  length: 8
```

- Η μέθοδος **filter() του Array.prototype:**

```
filter(function)
```

  - Εφαρμόζει τη συνάρτηση function, διαδοχικά, πάνω στα στοιχεία του πίνακα
    - Η συνάρτηση πρέπει να παίρνει ένα όρισμα και να επιστρέφει true/false
  - Επιστρέφει πίνακα με τα στοιχεία που «πέρασαν» τον έλεγχο (προσοχή είναι shallow copy)

**Παράδειγμα 6: filter**

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

let oddNumbers = array.filter(x => x%2===1);

console.log(oddNumbers);
```

**Σύνηθες στο Συναρτησιακό Προγραμματισμό:**

- Να γίνεται κλήση συνάρτησης επί του αποτελέσματος κλήσης συνάρτησης (αλυσίδωση ενεργειών).

**Παράδειγμα 7: consecutive_calls**

```
let oddSquares = array.map(x => x*x)
                      .filter(x => x%2===1);
```

**Άσκηση 4:**

Χρησιμοποιώντας το Rapid API, επιστρέψτε τους τίτλους των ταινιών που περιέχουν το Matrix και παίζει ο Keanu Reeves.

**Παράδειγμα 8: filter_shallow**

```
let persons = [
  {name: "John Snow", occupation: "King in the North", house: "Stark"},
  {name: "Daenerys Targaryen",
   occupation: "Queen of the Seven Kingdoms",  house: "Targaryen"},
  {name: "Brandon Stark", occupation: "Three-eyed Raven",
   house: "Stark"}
]

let theStarks = persons.filter(person=>person.house==="Stark");

persons[0].house = "Targaryen";
console.log(persons, theStarks)
```

**Άσκηση - βλ. και filter_deep:**

Σκεφθείτε πως θα πάρετε deep copy στο παράδειγμα 8.

- Η μέθοδος **reduce() του Array.prototype:**

```
reduce(function[, initialValue])
```

  - function: Συνάρτηση δύο ορισμάτων
  - Θα επιστρέψει μία ΤΙΜΗ που θα την υπολογίσει ως εξής:
    - Αρχικά ΤΙΜΗ = initialValue
    - Επαναληπτικά ΤΙΜΗ = function(ΤΙΜΗ, array[i]), i=0,..,length-1
  - Χωρίς initialValue:
    - Αρχικά ΤΙΜΗ = array[0]

**Παράδειγμα 9: reduce**

```
let array = [1,2,3,4,5];
let mult = array.reduce((x,y) => x*y);
let sum = array.reduce((x,y) => x+y, 10);
console.log(sum, mult);
```

**Παράδειγμα 10: reduce2**

```
let noDuplicates = [1,2,1,3,2,4,7,5].reduce((acc, elem)=>{
   if (!(elem in acc))
      acc.push(elem);
   return acc;
},[]);

console.log(noDuplicates);
```

**Παράδειγμα 11: reduce3**

```
let grades = [
   {name: "tom", grade: 5},
   {name: "bob", grade: 6},
   {name: "pam", grade: 6},
   {name: "jim", grade: 5},
]
let groupsByGrade = grades.reduce((groups, person)=>{
   if (person.grade in groups)
      groups[person.grade].push(person.name);
   else groups[person.grade] = [person.name]
   return groups;
}, []);

console.log(groupsByGrade);
```

**Άσκηση 5:**

Ομαδοποιήστε τις ταινίες που περιέχουν το τίτλο Matrix με βάση το είδος τους (πεδιο «q»), όπως στην ακόλουθη εκτύπωση:

```
▼ {feature: Array(6), TV series: Array(1), TV movie: Array(1)}
  ▶ TV movie: ["Making 'The Matrix'"]
  ▶ TV series: ['Matrix']
  ▶ feature: (6) ['The Matrix', 'The Matrix Resurrections', 'The Matrix Rel
  ▶ [[Prototype]]: Object
```

## Παραλλαγές της map και της filter:

- Η συνάρτηση που τίθεται ως όρισμα, μπορεί να έχει μία από τις παρακάτω παραλλαγές όσον αφορά τα ορίσματά της:

| Ορίσματα Συνάρτησης | Επεξήγηση |
|---|---|
| (elem, ind)=>... | ind: Index του τρέχοντος στοιχείου |
| (elem, ind, arr)=>... | arr: Ο πίνακας επί του οποίου έγινε η κλήση |

- ενώ προσφέρεται και παραλλαγή map(function, thisArg), όπου thisArg είναι το "this" κατά την κλήση.

## Παράδειγμα 12: map_alternatives

```
let result = [1,2,3,4,5].map((element, index, array) => ({
      prev: array[index-1],
      index: element,
      next: array[index+1]
   }));

console.log(result);
```

## Παραλλαγές της reduce:

- Η συνάρτηση που τίθεται ως όρισμα στη reduce, επιδέχεται αντίστοιχες παραλλαγές:

| Ορίσματα Συνάρτησης | Επεξήγηση |
|---|---|
| (acc, elem, ind)=>... | ind: Index του τρέχοντος στοιχείου |
| (acc, elem, ind, arr)=>... | arr: Ο πίνακας επί του οποίου έγινε η κλήση |

## Παράδειγμα 13: reduce_alternatives

```
let array = [1,2,3,4,5];
let sum = array.reduce((accumulator, element, index, array) => {
    console.log(array, index, element, accumulator);
    return accumulator + element;
 }, 10);
console.log(sum);
```

```
▸ (5) [1, 2, 3, 4, 5] 0 1 10
▸ (5) [1, 2, 3, 4, 5] 1 2 11
▸ (5) [1, 2, 3, 4, 5] 2 3 13
▸ (5) [1, 2, 3, 4, 5] 3 4 16
▸ (5) [1, 2, 3, 4, 5] 4 5 20
25
```

## Άσκηση 6:

Σε κάθε τίτλο του IMDB API διατηρείται το πεδίο «rank» που είναι η κατάταξη του τίτλου με βάση τη βαθμολογία των θεατών (π.χ. No. 1 είναι ο «Νονός», No.2 είναι το "Shawshank Redemption κ.οκ.)

- Επιστρέψτε τον τίτλο που περιέχει το Matrix και είναι ο καλύτερος (δηλαδή έχει το μικρότερο rank)

## Άλλες Συναρτησιακές Μέθοδοι Πινάκων:

- Οι παρακάτω μέθοδοι πινάκων εμπνέονται από τον συναρτησιακό προγραμματισμό:

| Συνάρτηση | Επεξήγηση |
|---|---|
| every(func) | Η func είναι αληθής για όλα τα στοιχείο |
| some(func) | Η func είναι αληθής για κάποιο στοιχείο |

- func είναι συνάρτηση που επιστρέφει boolean, με παραλλαγές στα ορίσματα, αντίστοιχες με τη map:
  - func(element[, index[, array]])
- ενώ προσφέρεται και παραλλαγή π.χ. every(function, thisArg), όπου thisArg είναι το “this” κατά την κλήση.

### Παράδειγμα 14: every_some

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

console.log(array.every(x=>x%2==0));
console.log(array.some(x=>x%2==0));
console.log(!array.some(x=>x%2==0)); // none
console.log(!array.every(x=>x%2==0)); // some not
```

## Η συνάρτηση forEach:

- Είναι η γενικής χρήσης, συνάρτηση επανάληψης επί των στοιχείων πίνακα:

| Συνάρτηση | Επεξήγηση |
|---|---|
| forEach(func) | Η func εφαρμόζεται σε κάθε στοιχείο |

- Η συνάρτηση δεν έχει επιστρεφόμενη τιμή (κάνει ενέργειες επί του αντικειμένου
- Υπάρχουν παραλλαγές στα ορίσματα, αντίστοιχες με τις every/some.

### Παράδειγμα 15: forEach

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

array.forEach(x=>console.log(x));

let copy = [];
array.forEach(x=>copy.push(x));
console.log(copy);
```

## Η συνάρτηση flatMap:

- Αντίστοιχη της flat() αλλά επί των αποτελεσμάτων της map:

### Παράδειγμα 16: flatMap

```
console.log([1,[1,2]].flat());
console.log([1,[1,2]].flatMap(x=>x));
```