Προγραμματισμός και Συστήματα στον Παγκόσμιο Ιστό

Θέματα JavaScript Frameworks, Angular

Δρ. Δημήτριος Κουτσομητρόπουλος
Ιωάννης Γαροφαλάκης, καθηγητής

Περιεχόμενα

Σήμερα
- Εκφράσεις συναρτήσεων, ανώνυμες συναρτήσεις, lambdas
- Συναρτησιακός προγραμματισμός στη JavaScript
  - Currying, Closures
- `this`
- Frameworks, Angular

Την επόμενη φορά
- Promises
- Fetch API
- Asynchronous JavaScript and HTTP Requests (*AJAX*)
- REST APIs
- CORS

Εκφράσεις συναρτήσεων

Εκφράσεις συναρτήσεων (Function expressions)

```
// defines a function using a function expression
var sub = function subtotal(price,quantity)
{
     return price * quantity;
};
// invokes the function
var result = sub(10,2);
```

Συνήθως στις εκφράσεις το όνομα της συνάρτησης παραλείπεται

Συναρτήσεις ως αντικείμενα

Οι συναρτήσεις στη JavaScript είναι αντικείμενα.

- Μπορούν να αποθηκευτούν σε μεταβλητές
- **Μπορούν να περαστούν ως παράμετροι**
- Έχουν ιδιότητες, όπως τα άλλα αντικείμενα
- Μπορούν να οριστούν ανώνυμα, χωρίς αναγνωριστικό.
  - Ορισμένες φορές καλούνται **εκφράσεις λάμδα** (lambda expressions)

First-class functions

- Μία συνάρτηση είναι αντικείμενο τύπου Function, που έχει ιδιότητες, όπως:
  - name
  - toString()
  - call()
  - Ο τελεστής () απλά καλεί την call()

```
const greeting = function() {
 console.log('hello, world');
}
console.log(greeting.name);
console.log(greeting.toString());
greeting.call();
greeting();
```


Ανώνυμες εκφράσεις συναρτήσεων

```
// defines a function using an anonymous function expression
var calculateSubtotal = function (price,quantity)
{
     return price * quantity;
};
// invokes the function
var result = calculateSubtotal(10,2);
```

Εκφράσεις συναρτήσεων και hoisting

Function declaration is **hoisted** to the beginning of its scope

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    function calculateTax(subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    }
}
```

Variable declaration is hoisted to the beginning of its scope

BUT

Variable assignment is **not** hoisted

```
function calculateTotal(price,quantity) {
    var subtotal = price * quantity;
    return subtotal + calculateTax(subtotal);

    var calculateTax = function (subtotal) {
        var taxRate = 0.05;
        var tax = subtotal * taxRate;
        return tax;
    };
}
```

THUS

The value of the `calculateTax` variable here is undefined

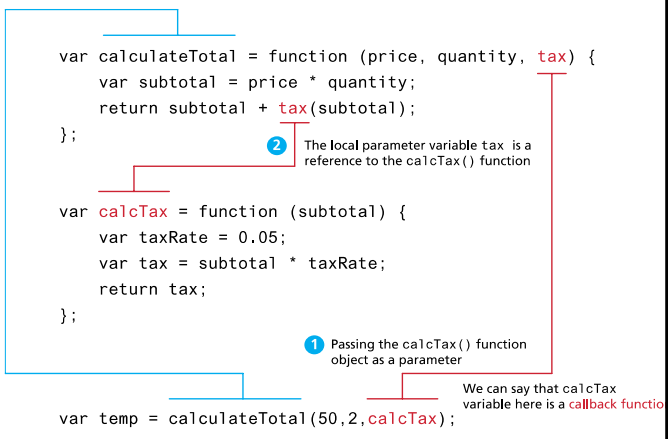
Συναρτήσεις και αντικείμενα

```
var order = {
    salesDate : "May 5, 2017",
    product : {
        type: "laptop",
        price: 500.00,
        output: function () {
            return this.type + ' $' + this.price;
        }
    },
    customer : {
        name: "Sue Smith",
        address: "123 Somewhere St",
        output: function () {
            return this.name + ', ' + this.address;
        }
    },
    output: function () {
            return 'Date' + this.salesDate;
    }
};
```

Callbacks

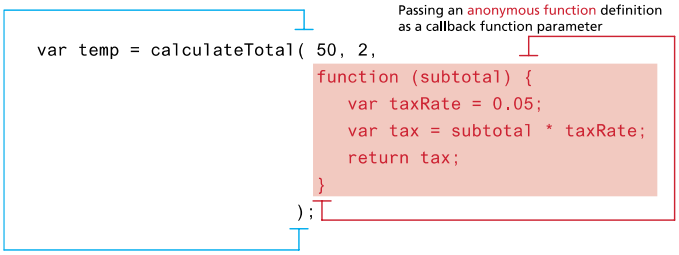
Callback: Μια συνάρτηση που περνιέται ως παράμετρος σε μια άλλη συνάρτηση, συνήθως ως απόκριση σε κάτι.

Οι παράμετροι της callback τροφοδοτούνται από την εξωτερική συνάρτηση



Callbacks

Για callback μπορούν επίσης να χρησιμοποιηθούν *ανώνυμες εκφράσεις συναρτήσεων*



JavaScript `this`

JavaScript `this`

Η λέξη κλειδί `this` στη JavaScript **ανατίθεται δυναμικά**

- μπορεί να σημαίνει διαφορετικά πράγματα, αναλόγως του πώς χρησιμοποιείται
- Όταν είναι έξω από κάποια συνάρτηση, αναφέρεται στο **global αντικείμενο window**
- Όταν είναι εντός μιας συνάρτησης:
- Αν δεν έχει τεθεί πριν, αναφέρεται στο global object
- Αν η συνάρτηση είναι ***μέθοδος αντικειμένου***, **αναφέρεται στο αντικείμενο που την καλεί**
- Αν η συνάρτηση είναι ***event handler***, **αναφέρεται στο στοιχείο DOM που πυροδότησε το event**

JavaScript ‘this’ – για Objects

Πως λειτουργεί η JavaScript εδώ?

- Δίνει την τιμή 42 στο ‘the_answer’
- Δημιουργεί μια μέθοδο ‘ask_question’

Τί γίνεται όταν καλείται η ‘ask_question’?

- Δημιουργείται ένα πλαίσιο εκτέλεσης για το deep_thought (context)
- Θέτει το **this** στο object ‘deep_thought’

```
<script type="text/javascript">
  var deep_thought = {
   the_answer: 42,
   ask_question: function () {
    return this.the_answer;
   }
  };

  var the_meaning =
deep_thought.ask_question();
  console.log(the_meaning);
 </script>
```

JavaScript ‘this’ – για Functions

Τί γίνεται όταν καλείται η ‘test_this’?

Τι θα εμφανίσει το i?

- Δεν έχει δημιουργηθεί κάποιο context ούτε έχουμε object.
- Επομένως η JavaScript ανεβαίνει προς τα πάνω την αλυσίδα μέχρι και το root object (= window)

```
<script type="text/javascript">
  function test_this() {
      return this;
    }
  var i = test_this();
  console.log(i);
</script>
```

JavaScript 'this' – για Events

Τι περιέχει το this?

- Το DOM στοιχείο που στην περίπτωση μας είναι το '<button>' στοιχείο.

```
const button = document.querySelector('#abutton');
button.addEventListener('click', click_handler);

function click_handler() {
  alert(this);
}
```

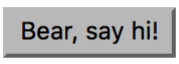
object HTMLButtonElement

Callback και this

```
const bear = {
  characterName: 'Ice Bear',
  hobbies: ['knitting', 'cooking', 'dancing'],
  greeting: function() {
     console.log(this.characterName + ' says hello');
  }
}
bear.greeting();

const button = document.querySelector('button');
button.addEventListener('click', bear.greeting);
```

```
<button>Bear, say hi!</button>
```




```
Ice Bear says hello
undefined says hello
```

Τι θα τυπώσει ο κώδικας;

Όταν καλείται σε μέθοδο αντικειμένου, το this δείχνει στο αντικείμενο που κάλεσε τη μέθοδο

Όταν καλείται σε έναν event handler, το this δείχνει στο DOM element που συνδέεται με το event (το κουμπί, <button>)

Functional JavaScript

Παράδειγμα: `findIndex`

***array*.findIndex(*callback*, *thisArg*)**:
Επιστρέφει τον δείκτη ενός στοιχείου
Η παράμετρος ***callback*** είναι μια συνάρτηση με τις ακόλουθες παραμέτρους: <- τροφοδοτούνται από την findIndex

- **element:** Το τρέχον στοιχείο του οποίου γίνεται επεξεργασία
- **index:** Ο δείκτης του τρέχοντος στοιχείου (optional)
- **array**: ο πίνακας που κάλεσε την `findIndex` (optional)

Οι τιμές παρέχονται από την **findIndex**

Η ***callback*** χρησιμοποιείται από την findIndex για κάθε στοιχείο του πίνακα, και επιστρέφει true αν βρεθεί, διαφορετικά false.
Η ***thisArg*** είναι η τιμή του `this` που θα χρησιμοποιηθεί στην ***callback*** (optional)

findIndex

```
const flavors =
    ['vanilla', 'chocolate', 'strawberry', 'green tea'];
function isStrawberry(element) {  //Callback: Αναζητά μια συγκεκριμένη γεύση
  return element === 'strawberry';
}

const indexOfStrawberry = flavors.findIndex(isStrawberry);
```

Η `findIndex` επιστρέφει 2, επειδή το πρώτο στοιχείο που ικανοποιεί την συνάρτηση ελέγχου βρέθηκε στη θέση 2.

Arrow function

Μπορούμε να ορίσουμε την συνάρτηση ελέγχου απευθείας μέσα στην findIndex:

```
const index = flavors.findIndex(
  function(element) { return element === 'strawberry'; });
```

Μπορούμε να χρησιμοποιήσουμε τον **συμβολισμό βέλους** (arrow function, ES6):

```
const index = flavors.findIndex(
  (element) => { return element === 'strawberry'; });
```

Μπορούμε να χρησιμοποιήσουμε τη συνοπτική εκδοχή του συμβολισμού βέλους:

- Όταν υπάρχει **μόνο μία παράμετρος** παραλείπονται η παρενθέσεις
- Όταν η συνάρτηση **μόνο έχει return** παραλείπονται τα άγκιστρα, return

```
const index = flavors.findIndex(
  element => element === 'strawberry');
```

Σύγκριση

Ποιο κομψός και καθαρός κώδικας:

```
const index = flavors.findIndex(
   element => element === 'strawberry');
```

vs

```
for (var i = 0; i < flavors.length; i++) {
  if (flavors[i] === 'strawberry') {
    break;
  }
}
const index = i;
```


Closures, Currying

Έστω ότι θέλουμε μια γενική συνάρτηση ελέγχου, για οποιαδήποτε γεύση:

```
function isFlavor(flavor, element) {
  return element === flavor; }
```

Δεν μπορούμε να τη χρησιμοποιήσουμε, γιατί η callback δέχεται συγκεκριμένες παραμέτρους (element,…)

Πώς θα περάσουμε το flavor ως παράμετρο;

Closure

Ουσιαστικά, έχουμε φτιάξει μια νέα συνάρτηση isFlavor, που δέχεται τις σωστές παραμέτρους (element), αλλά μπορούμε να τις περάσουμε και την τιμή της επιθυμητής επιπλέον παραμέτρου (flavor)

Θα δημιουργήσουμε μια συνάρτηση που δέχεται το flavor ως παράμετρο και επιστρέφει μια συνάρτηση ελέγχου για αυτήν την παράμετρο

```
const flavors =
    ['vanilla', 'chocolate', 'strawberry', 'green tea'];

function createFlavorTest(flavor) {
  function isFlavor(element) {
    return element === flavor;
  }
  return isFlavor;
}
const isStrawberry = createFlavorTest('strawberry');
const indexOfFlavor = flavors.findIndex(isStrawberry);
```

Μια συνάρτηση που ορίζεται μέσα σε μια άλλη ονομάζεται κλειστότητα **(closure).**
Ένα closure μπορεί να επιστρέφεται ως αντικείμενο από την εξωτερική συνάρτηση.

Currying

```
function isFlavor(flavor, element) {
  return element === flavor;
}
```

```
function createFlavorTest(flavor) {
  function isFlavor(element) {
    return element === flavor;
  }
  return isFlavor;
}
```

```
flavors.findIndex(isFlavor);
```

Currying: Η ανάλυση μια συνάρτησης με πολλές παραμέτρους, εφαρμόζοντας μία κάθε φορά σε μια ακολουθία παραγόμενων συναρτήσεων.

Προς τιμήν του **Haskell Brooks Curry** που μελέτησε μαθηματικά το θέμα: «*Οι συναρτήσεις πολλών μεταβλητών μπορούν να αναλυθούν σε ακολουθίες συναρτήσεων μίας μόνο μεταβλητής*»

JavaScript Frameworks
Angular

JavaScript Frameworks

Αποτελούν μια «εφαρμογή» γραμμένη σε JavaScript

Επιτρέπουν τη συγγραφή κώδικα με νέο συντακτικό και σχεδιαστικές συμβάσεις
- Αποκρύπτουν την πολυπλοκότητα
- Επιταχύνουν τον προγραμματισμό

Κάνουν "transpile" τον κώδικα σε απλή JavaScript

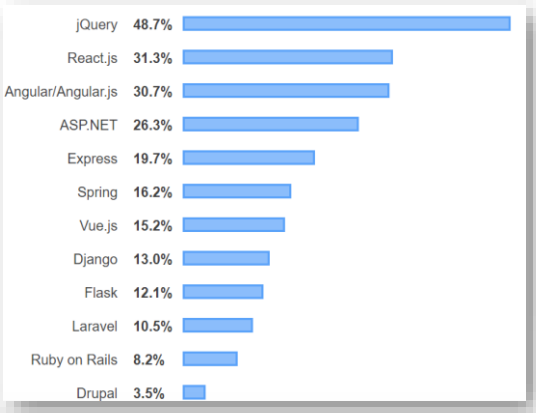
Περιλαμβάνουν βιβλιοθήκες
- Παρέχουν το δικό τους API και μεθόδους

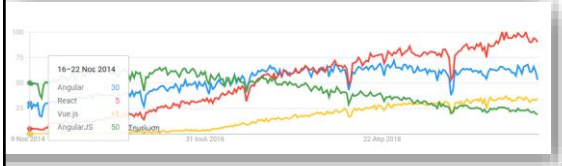
MVC JavaScript Frameworks:
- AngularJS
- ReactJS
- VueJS

Παλιότερα frameworks:
- jQuery (JavaScript βιβλιοθήκη)

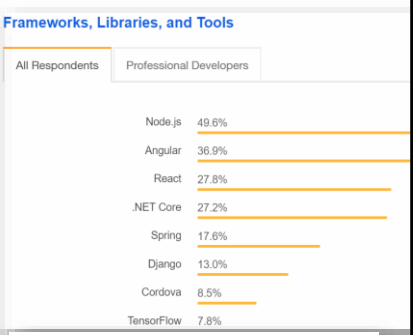
Javascript Frameworks

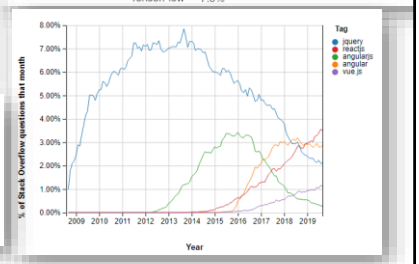
StackOverflow, 2019





StackOverflow, 2018





Google trends

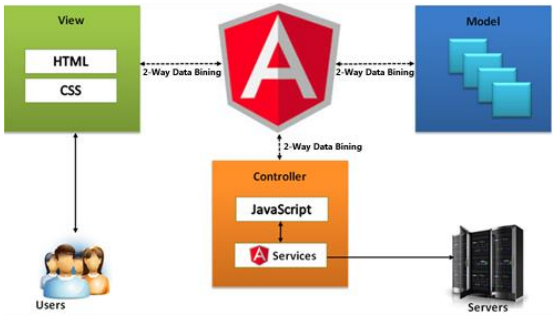
JavaScript Frameworks

- Το **Angular** βοηθά τον προγραμματιστή να αναπτύξει πλήρως μια web εφαρμογή, ακολουθώντας μια καλώς ορισμένη και αξιόλογη δόμηση. Χρησιμοποιεί το *πρότυπο* (pattern) **MVC (Model-View-Controller)**. Υποστηρίζεται από τη Google κ.α.

- Το **React** είναι μια βιβλιοθήκη και framework που αναπτύσσεται από το Facebook. Το React δεν είναι πλήρες MVC framework. Αντίθετα εστιάζει κυρίως στο View.

- Το **Vue** ξεκίνησε ως ένα απλοποιημένο υποσύνολο της 1ης έκδοσης του Angular (AngularJS), κατάλληλο για single-page εφαρμογές. Έχει πολλές ομοιότητες με το React στο UI, αλλά αποτελεί και MVC framework.

Angular MVC



- Τα ***μοντέλα (models)*** είναι συνήθη JavaScript αντικείμενα. Αντιπροσωπεύουν το μέρος της εφαρμογής που διαχειρίζεται την ανάκτηση και την αποθήκευση **δεδομένων**.
- Η ***όψη (view)*** είναι αυτό που βλέπει ο χρήστης στην οθόνη (HTML&CSS)
- Τα ***components*** (***controllers*** στο AngularJS) καθορίζουν το πώς γίνεται η αλληλεπίδραση ανάμεσα στα models και το view. Δέχονται είσοδο και δίνουν εντολές στο model και στο view.
- ***2-way data binding*** σημαίνει ότι υπάρχει αμφίδρομη επικοινωνία ανάμεσα στο μοντέλο και στην όψη. Τυχόν αλλαγές δεδομένων στο μοντέλο αντανακλώνται απευθείας στην όψη και τυχόν αλλαγές στην όψη (είσοδος χρήστη) φαίνονται αυτόματα στο μοντέλο.
- Το React π.χ. υποστηρίζει *1-way binding* (model → view)

Angular

- Βασικό δομικό στοιχείο: **component**
  - Ένα angular project περιλαμβάνει ένα ή περισσότερα components
- Ο προγραμματισμός γίνεται σε ***TypeScript***
  - Υπερσύνολο της JavaScript με *ισχυρούς τύπους* και decorators (*@Decorator*)
- Ένα **component** περιλαμβάνει 3 αρχεία:
  - Αρχείο .html (view): Καθορίζει τον τρόπο εμφάνισης του component
  - Αρχείο .css (view)
  - Αρχείο .ts (controller)
- Το model αντιστοιχεί σε ένα **service**
  - Περιλαμβάνει .ts αρχεία
  - Αναλαμβάνει τη διαχείριση των δεδομένων
  - Όμως σε απλές περιπτώσεις ένα component μπορεί να τα προσπελάζει απευθείας
- Τα components επικοινωνούν με το view παρεμβάλλοντας στοιχεία και tags στον html κώδικα (π.χ. `{{title}}`)
- Ο επιμερισμός και η δρομολόγηση της λειτουργικότητας της εφαρμογής ανάμεσα σε διαφορετικές «σελίδες» (URLs) καθορίζεται από τα **router modules** (controller)
- Παρέχονται πολλές αυτοματοποιήσεις μέσω του Angular CLI
  - Dev web server με live reload
  - Έτοιμο κέλυφος (*app* component) και σκελετός της εφαρμογής (έτοιμα πρότυπα για components, html, css)
  - Αυτόματη εγκατάσταση dependencies

Παράδειγμα app component

Tour of Heroes tutorial

- src/app/app.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'Tour of Heroes';
}
```

`Import Symbol:` import a module than has been exported with `export`

Decorator (*TS specific*) that marks a class as an Angular component and provides configuration metadata

The CSS selector for the component

The HTML rendering (*template*) of the component

So it can be imported elsewhere

- src/app/app.component.html

```
<h1>{{title}}</h1>
```

Angular interpolation (*Angular specific*)

stackblitz

Δημιουργία Hero component

Μοντελοποίηση των δεδομένων της εφαρμογής

- src/app/heroes/heroes.component.ts

```
1. import { Component, OnInit } from '@angular/core';
2. import { Hero } from '../hero';
3.
4. @Component({
5.   selector: 'app-heroes',
6.   templateUrl: './heroes.component.html',
7.   styleUrls: ['./heroes.component.css']
8. })
9. export class HeroesComponent implements OnInit {
10.   hero: Hero = {
11.     id: 1,
12.     name: 'Windstorm'
13.   };
14.
15.   constructor() { }
16.
17.   ngOnInit() {
18.   }
19.
20. }
```

Μπορεί να αρχικοποιεί πεδία της κλάσης

Καλείται *μετά τον constructor* και εκτελεί κώδικα αρχικοποίησης. Χρειάζεται το *interface* OnInit (*TS specific*)

- src/app/heroes.ts

```
export class Hero {
  id: number;
  name: string;
}
```

Ισχυροί τύποι (*TS specific*)

- src/app/heroes.component.html

```
<h2>{{hero.name | uppercase}} Details</h2>
<div><span>id: </span>{{hero.id}}</div>
<div>
  <label>name:
    <input [(ngModel)]="hero.name" placeholder="name">
  </label>
</div>
```

Angular 2-way data binding (*Angular specific*): data can flow *in both directions*: from the hero.name property to the textbox, and from the textbox back to the hero.name

- src/app/app.component.html

```
<h1>{{title}}</h1>
<app-heroes></app-heroes>
```

Selector για το Hero component

stackblitz

Εμφάνιση λίστας Heroes

- src/app/heroes/heroes.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Hero } from '../hero';

@Component({…
})
export class HeroesComponent implements OnInit {

  selectedHero: Hero;
  heroes: Hero[] = [{
    id: 1,
    name: 'Windstorm'
  }, {
    id: 2,
    name: 'Superman'
  }];

  onSelect(hero: Hero): void {
    this.selectedHero = hero;
  }

  constructor() { }

  ngOnInit() {
  }}
```

- src/app/heroes.component.html

```html
<div *ngFor="let h of heroes" (click)=onSelect(h)
>
  <h2>{{h.name | uppercase}} Details</h2>
  <div>id:{{h.id}}</div>
</div>
<div *ngIf="selectedHero">
  name:
  <input [(ngModel)]="selectedHero.name"
placeholder="name">
</div>
```

stackblitz

Πηγές: Connolly, Hoar, "Fundamentals of Web Development, 2nd ed", V. Kirst, Stanford CS-193X course, https://angular.io/tutorial, G. Stern, "How to create Angular Hello World", medium.com