


**ΕΛΛΗΝΙΚΟ ΜΕΣΟΓΕΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ**

**ΣΧΟΛΗ ΕΠΙΣΤΗΜΩΝ ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΟΙΚΟΝΟΜΙΑΣ**

**ΤΜΗΜΑ ΔΙΟΙΚΗΤΙΚΗΣ ΕΠΙΣΤΗΜΗΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ**

---

# ΑΝΑΠΤΥΞΗ ΔΙΑΔΙΚΤΥΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΜΕ ΧΡΗΣΗ ANGULAR

---

# ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

---

Εισηγητής: Γεωργία Βοτζάκη, Α.Μ: 1571

Επιβλέπων: Ιωάννης Καπανταϊδάκης

**©**

**2024**




**HELLENIC MEDITERRANEAN UNIVERSITY**

**SCHOOL OF MANAGEMENT AND ECONOMICS SCIENCE**

**DEPARTMENT OF MANAGMENENT SCIENCE AND TECHNOLOGY**

---

# WEB APPLICATION DEVELOPMENT USING ANGULAR

---

# DIPLOMA THESIS

---

Student : Georgia Votzaki, A.M: 1571

Supervisor : Ioannis Kapantaidakis

©
**2024**

***Υπεύθυνη Δήλωση*** *: Βεβαιώνω ότι είμαι συγγραφέας αυτής της πτυχιακής εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην πτυχιακή εργασία. Επίσης έχω αναφέρει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επίσης βεβαιώνω ότι αυτή η πτυχιακή εργασία προετοιμάστηκε από εμένα προσωπικά ειδικά για τις απαιτήσεις του προγράμματος σπουδών του Τμήματος Διοικητικής Επιστήμης και Τεχνολογίας του ΕΛ.ΜΕ.ΠΑ.*

# ΠΕΡΙΛΗΨΗ

Η Πληροφορική και συνολικά οι Τεχνολογίες Πληροφορικής και Επικοινωνιών έχουν αλλάξει τον τρόπο ζωής των ανθρώπων. Ιδιαίτερα τα τελευταία χρόνια με την αύξηση της χρήσης των κινητών συσκευών έχουν αυξηθεί σημαντικά οι διαδικτυακοί τόποι που παρέχουν πληροφορίες και υπηρεσίες. Το περιβάλλον εργασίας των διαδικτυακών τόπων καθημερινά βελτιώνεται και γίνεται πιο φιλικό. Σε αυτό συμβάλλουν τα JavaScript framework και ιδιαίτερα η Angular που βελτιώνει σημαντικά την εμπειρία του χρήστη κατά την επίσκεψη σε μια ιστοσελίδα. Στην πτυχιακή μας θα μελετήσουμε σε βάθος την Angular και θα την χρησιμοποιήσουμε δημιουργώντας μια διαδικτυακή εφαρμογή.

**Λέξεις Κλειδιά** : JavaScript framework, Angular.

# ABSTRACT

Information and Communication Technologies have changed people's way of life, especially in recent years, with the increase in the use of mobile devices, the number of websites that provide information and services has increased significantly. The user interface of websites is improving and becoming friendlier every day. JavaScript frameworks and especially Angular contribute to this, which significantly improves the user experience when visiting a website. In our thesis we will study Angular in depth and use it by creating a web application.

**Key Words**: JavaScript framework, Angular.

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

**ΛΙΣΤΑ ΠΙΝΑΚΩΝ** ..... 5
**ΛΙΣΤΑ ΣΧΕΔΙΑΓΡΑΜΜΑΤΩΝ** ..... 5
**ΚΕΦΑΛΑΙΟ 1** ..... 6
**ΕΙΣΑΓΩΓΗ** ..... 6
**ΚΕΦΑΛΑΙΟ 2** ..... 8
**Τεχνολογίες Διαδικτύου** ..... 8
2.1 HTML ..... 8
2.2 JavaScript Frameworks ..... 11
2.2.1 Angular ..... 21
2.2.2 React ..... 24
2.2.3 Ext JS ..... 27
2.3.4 GSAP ..... 29
2.3 TypeScript ..... 30
**ΚΕΦΑΛΑΙΟ 3** ..... 35
**Αρχιτεκτονική της Angular** ..... 35
3.1 Εισαγωγή ..... 35
3.2 Ιστορική αναδρομή ..... 36
3.3 Αρχιτεκτονική ..... 39
3.3.1 Module ..... 40
3.3.2 Component ..... 45
3.3.3 Template ..... 46
3.3.4 Directive ..... 50
3.3.5 Service ..... 57
3.3.6 Dependency injection ..... 58
3.3.7 Router ..... 62
3.3.8 State management ..... 64
3.3.9 HTTP Client ..... 71
3.4 Δημιουργία μιας Angular εφαρμογής ..... 72
3.5 Δομή μιας Angular εφαρμογής ..... 74
**ΚΕΦΑΛΑΙΟ 4** ..... 78
**Σχεδιασμός και Ανάπτυξη εφαρμογής** ..... 78
4.1 Εισαγωγή ..... 78
4.2 Καταγραφή απαιτήσεων ..... 78
4.3 Δομή της εφαρμογής ..... 78
**ΚΕΦΑΛΑΙΟ 5** ..... 81
**Λειτουργία εφαρμογής** ..... 81
5.1 Εισαγωγή ..... 81
5.2 Αναλυτική παρουσίαση του τρόπου λειτουργίας ..... 81
**ΚΕΦΑΛΑΙΟ 6** ..... 85
**Συμπεράσματα** ..... 85
**ΒΙΒΛΙΟΓΡΑΦΙΑ** ..... 86
Α. ΞΕΝΟΓΛΩΣΣΗ ..... 86
**ΠΑΡΑΡΤΗΜΑ Α** ..... 88
Index.html ..... 88
app.component.ts ..... 89

book-info.ts ..... 89
bookaction.service.ts ..... 89
routes.ts ..... 95
details.component.ts ..... 95
book-data.component.ts ..... 96
book-data.component.html ..... 97
book.component.ts ..... 97
book.component.html ..... 98

# ΛΙΣΤΑ ΠΙΝΑΚΩΝ

Πίνακας 1: Κυριότερες βιβλιοθήκες JavaScript (GeeksForGeeks, 2024a) ..... 12
Πίνακας 2: Κυριότερα JavaScript framework (GeeksForGeeks, 2024a) ..... 15
Πίνακας 3: Μερίδιο αγοράς των 10 JavaScript frameworks (Wappalyzer, 2024) ..... 20
Πίνακας 4: Παραδείγματα σύνταξης για κάθε κατηγορία σύνδεσης δεδομένων (Angular, 2024) ..... 49
Πίνακας 5: Τύποι Directive (Angular, 2024) ..... 50
Πίνακας 6: Οι πιο κοινές οδηγίες Attribute directive (Angular, 2024) ..... 51
Πίνακας 7: Πιο συχνές structural directives (Angular, 2024) ..... 54

# ΛΙΣΤΑ ΣΧΕΔΙΑΓΡΑΜΜΑΤΩΝ

Εικόνα 1: Ανατομία ενός HTML element (mdn, 2024) ..... 8
Εικόνα 2: Χαρακτηριστικά σε στοιχεία (mdn, 2024) ..... 9
Εικόνα 3: Απεικόνιση DOM ενός Table (Robie, n.a.) ..... 10
Εικόνα 4: DOM (Deshpande, 2024) ..... 11
Εικόνα 5: JavaScript framework τάσεις και σύγκλιση (Hempenius & Osmani, 2024) ..... 17
Εικόνα 6: Παράγοντες που πρέπει να ληφθούν υπόψη κατά την επιλογή ενός JavaScript framework (Sencha, 2024) ..... 19
Εικόνα 7: Angular ..... 22
Εικόνα 8: Χαρακτηριστικά της Angular (Sharma, 2023) ..... 24
Εικόνα 9: React ..... 25
Εικόνα 10: Browser DOM vs Virtual DOM (GeeksforGeeks, 2024b) ..... 25
Εικόνα 11: React component (GeeksForGeeks, 2024b) ..... 27
Εικόνα 12: Ext JS (Sencha,2024) ..... 28
Εικόνα 13: GSAP (GSAP, 2024) ..... 29
Εικόνα 14: Η Typescript σε σχέση με την έκδοση ES6 της JavaScript (Javatpoint, 2024) ..... 31
Εικόνα 15: Διαδικασία μεταγλώττισης TypeScript (Javatpoint, 2024) ..... 31
Εικόνα 16: Χαρακτηριστικά της TypeScript(Krishnakumar, 2023) ..... 34
Εικόνα 17: Χαρακτηριστικά της Angular (ZobiWebSolutions, 2020) ..... 35
Εικόνα 18: Χαρακτηριστικά Angular 2 (Sakthi, 2024) ..... 37
Εικόνα 19: Οι εκδόσεις της Angular (Sakthi, 2024) ..... 39
Εικόνα 20: Αρχιτεκτονική της Angular (NgDevelop, 2024) ..... 39
Εικόνα 21: Συσχέτιση Agular εφαρμογής με NgModule (Ganatan, 2024) ..... 41
Εικόνα 22: Αλληλεπίδραση Component με Module ..... 42
Εικόνα 23: Δομή του app.module.ts (Ganatan, 2024) ..... 43
Εικόνα 24: Angular Service (ScholarHat, 2024) ..... 57
Εικόνα 25: Dependency injection στην Angular (Kiran, 2018) ..... 59
Εικόνα 26: Αποτέλεσμα ng serve ..... 73
Εικόνα 27: Αποτέλεσμα της πρώτης εφαρμογής ..... 74
Εικόνα 28: Δέντρο ένθετων στοιχείων (AngularStart, 2024) ..... 76
Εικόνα 29: Αρχική σελίδα εφαρμογής ..... 81
Εικόνα 30: Εμφάνιση πληροφοριών βιβλίου ..... 82
Εικόνα 31: Εισαγωγή ονόματος προς αναζήτηση ..... 83
Εικόνα 32: Αποτέλεσμα αναζήτησης με βάση τον τίτλο ..... 83

# ΚΕΦΑΛΑΙΟ 1

# ΕΙΣΑΓΩΓΗ

Αντικείμενο της πτυχιακής εργασίας ήταν η ανάπτυξη μιας διαδικτυακής εφαρμογής με χρήση της Angular. Ο σκοπός της συγκεκριμένης πτυχιακής εργασίας ήταν η κατανόηση του τρόπου χρήσης των JavaScript frameworks στους διαδικτυακούς τόπους και το πώς αυτά μπορούν να δράσουν βελτιώνοντας την εμπειρία του χρήστη. Επίσης, ήταν η σε βάθος μελέτη της Angular, η κατανόηση της δομής και του τρόπου χρήσης έτσι ώστε να μπορέσουμε να αναπτύξουμε μια διαδικτυακή εφαρμογή με αυτή, για να επιδείξουμε τη βασική λειτουργία της.

Για το σκοπό αυτό μελετήσαμε βιβλιογραφικές πηγές τόσο για τα JavaScript framework όσο και για την Angular, επιλέγοντας διαδικτυακούς τόπους με ενημέρωση την τρέχουσα χρονιά έτσι ώστε να έχουμε πρόσβαση στις τελευταίες πληροφορίες και δεδομένα.

Στο κεφάλαιο 2 παρουσιάζουμε τις πιο σημαντικές τεχνολογίες του διαδικτύου, ξεκινώντας από την HTML για την οποία κάνουμε μια μικρή εισαγωγή. Στη συνέχεια παρουσιάζουμε το τί είναι τα JavaScript framework, ποια είναι τα πλεονεκτήματά τους και που μπορούν να χρησιμοποιηθούν στους διαδικτυακούς τόπους. Στην ενότητα αυτή αναφερόμαστε και σε ορισμένα JavaScript framework (επιλέγοντας ορισμένα από το μεγάλο σύνολο των JavaScript framework) παρουσιάζοντας τα πλεονεκτήματα και μειονεκτήματά τους. Κλείνουμε το κεφάλαιο αναφερόμενοι την TypeScript μια αντικειμενοστραφή γλώσσα προγραμματισμού ανοιχτού κώδικα. Η TypeScript είναι ένα υπερσύνολο της JavaScript, προσφέροντας τυποποίηση και χρησιμοποιείται από πολλά JavaScript framework όπως είναι η Angular.

Στο κεφάλαιο 3, εμβαθύνουμε στην Angular. Ξεκινάμε με μια σύντομη ιστορική αναδρομή και στη συνέχεια παρουσιάζουμε την αρχιτεκτονική της. Θα δούμε τα μέρη από τα οποία αποτελείται και πώς μπορεί ο προγραμματιστής να τα χρησιμοποιήσει και να τα συνδυάσει έτσι ώστε το αποτέλεσμά του να είναι ένας πιο φιλικός διαδικτυακός τόπος. Κατά την παρουσίαση αυτή, για να βελτιώσουμε την κατανόηση χρησιμοποιούμε αρκετά παραδείγματα. Ολοκληρώνοντας τη μεγάλη ενότητα της αρχιτεκτονικής της Angular συνεχίζουμε με τον τρόπο δημιουργίας μιας εφαρμογής. Παρουσιάζουμε τα βήματα που πρέπει να ακολουθήσει ένας προγραμματιστής για να δημιουργήσει στον υπολογιστή του μια νέα εφαρμογή Angular. Τέλος περιγράφουμε την τυπική δομή μιας εφαρμογής Angular.

Στο κεφάλαιο 4 θα αναφερθούμε στο σχεδιασμό και την ανάπτυξη της διαδικτυακής εφαρμογής. Θα παρουσιάσουμε την εφαρμογή που θα υλοποιήσουμε και θα καταγράψουμε τις απαιτήσεις χρήσης της. Τέλος θα αναφερθούμε στη δομή της εφαρμογής και στα πιο σημαντικά μέρη από τα οποία αποτελείται.

Στο κεφάλαιο 5 θα παρουσιάσουμε τον τρόπο χρήσης της εφαρμογής που αναπτύξαμε. Η παρουσίαση θα γίνει με αναλυτικά βήματα που μπορεί να εκτελέσει ο χρήστης και τα αποτελέσματα που θα δει στην οθόνη του.

Τέλος στο κεφάλαιο 6 θα δούμε τα συμπεράσματα που αποκομίσαμε από την ολοκλήρωση της πτυχιακής εργασίας.

# ΚΕΦΑΛΑΙΟ 2

# Τεχνολογίες Διαδικτύου

## 2.1 HTML

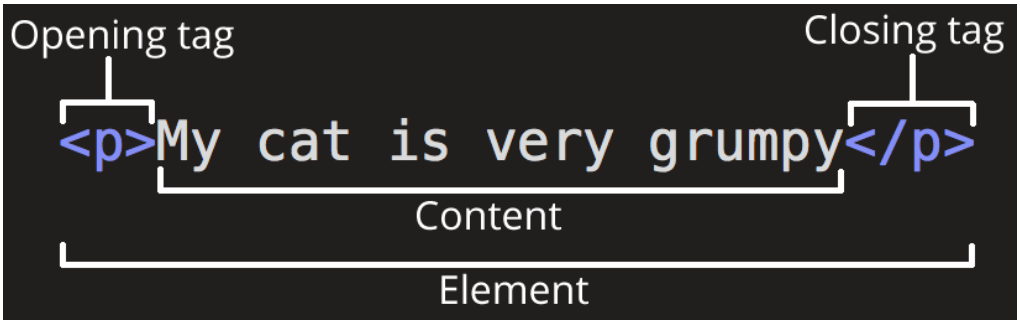
Η HTML είναι μια γλώσσα σήμανσης που καθορίζει τη δομή του περιεχομένου που θέλουμε να εμφανίσουμε. Η HTML αποτελείται από μια σειρά στοιχείων, τα οποία χρησιμοποιούμε για να περικλείσουμε διάφορα μέρη του περιεχομένου έτσι ώστε να εμφανίζεται με έναν συγκεκριμένο τρόπο ή να προσφέρει στους χρήστες συγκεκριμένους τρόπους ενέργειας. Οι ετικέτες που περικλείουν μπορούν να κάνουν μια λέξη ή μια εικόνα να λειτουργήσει ως υπερσύνδεση, να τροποποιήσουν τον τρόπο εμφάνισης μιας λέξης π.χ. πλάγια γράμματα, να τροποποιήσουν το μέγεθος μιας γραμματοσειράς και ούτω καθεξής. Έστω ότι θέλαμε να εμφανιστεί το κείμενο

My cat is very grumpy

Εάν θέλαμε η γραμμή να είναι μόνη της, θα μπορούσαμε να καθορίσουμε ότι είναι μια παράγραφος, περικλείοντάς την σε ετικέτες παραγράφου:

My cat is very grumpy

Ας αναλύσουμε λίγο περισσότερο την παραπάνω γραμμή μπορούμε να διακρίνουμε τέσσερα στοιχεία:




**Εικόνα 1: Ανατομία ενός HTML element (mdn, 2024)**

Η ετικέτα ανοίγματος: Αποτελείται από το όνομα του στοιχείου (στην περίπτωση αυτή, p), τυλιγμένο σε γωνιακές αγκύλες ανοίγματος και κλεισίματος. Αυτό δηλώνει πού αρχίζει ή αρχίζει να τίθεται σε ισχύ το στοιχείο, στην περίπτωσή μας ότι αρχίζει η παράγραφος.

- Η ετικέτα κλεισίματος: Είναι ίδια με την ετικέτα ανοίγματος, με τη διαφορά ότι περιλαμβάνει μια κάθετο προς τα εμπρός πριν από το όνομα του στοιχείου. Αυτό δηλώνει πού τελειώνει το στοιχείο, στην περίπτωσή μας πού τελειώνει η παράγραφος.
- Το περιεχόμενο: Αυτό είναι το περιεχόμενο του στοιχείου, το οποίο στην περίπτωσή μας είναι απλώς κείμενο.
- Το στοιχείο: Η ετικέτα ανοίγματος, η ετικέτα κλεισίματος και το περιεχόμενο μαζί αποτελούν το στοιχείο.

Τα στοιχεία μπορούν επίσης να έχουν χαρακτηριστικά που μοιάζουν με τα ακόλουθα:




**Εικόνα 2: Χαρακτηριστικά σε στοιχεία (mdn, 2024)**

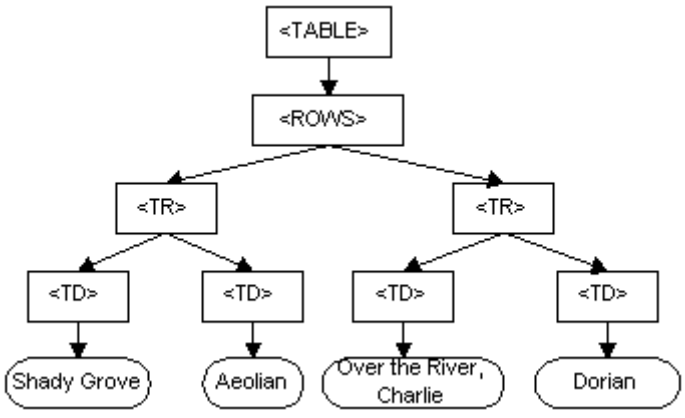
Τα χαρακτηριστικά περιέχουν επιπλέον πληροφορίες σχετικά με το στοιχείο που δεν θέλουμε να εμφανίζεται στο πραγματικό περιεχόμενο. Στην περίπτωσή μας class είναι το όνομα του χαρακτηριστικού και editor-note είναι η τιμή του (mdn, 2024).

Στο σημείο αυτό θα σταματήσουμε την ανάλυση της HTML και θα ασχοληθούμε με το Document Object Model. Το Document Object Model (DOM) είναι ένα API προγραμματισμού για έγγραφα HTML και XML. Καθορίζει τη λογική δομή των εγγράφων και τον τρόπο πρόσβασης και χειρισμού τους. Με το Document Object Model, οι προγραμματιστές μπορούν να δημιουργήσουν έγγραφα, να περιηγηθούν στη δομή τους και να προσθέσουν, να τροποποιήσουν ή να διαγράψουν στοιχεία και περιεχόμενο. Οτιδήποτε βρίσκεται σε ένα έγγραφο HTML ή XML μπορεί να προσπελαστεί, να αλλάξει, να διαγραφεί ή να προστεθεί χρησιμοποιώντας το DOM.

Αν για παράδειγμα έχουμε το ακόλουθο table σε HTML:

```
<TABLE>
    <ROWS>
    <TR>
    <TD>Shady Grove</TD>
    <TD>Aeolian</TD>
    </TR>
    <TR>
    <TD>Over the River, Charlie</TD>
    <TD>Dorian</TD>
    </TR>
    </ROWS>
 </TABLE>
```

Τότε το DOM που απεικονίζει το συγκεκριμένο table θα είναι:



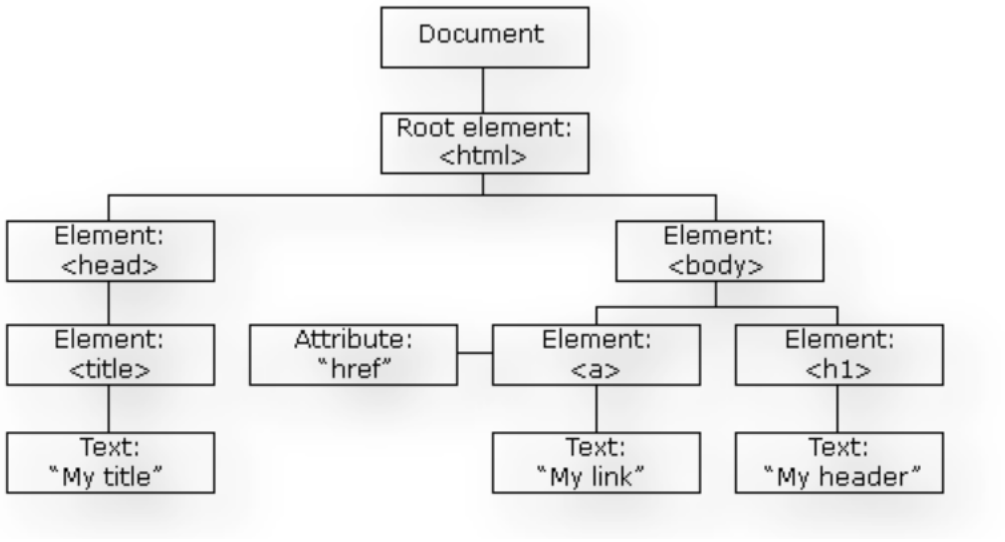

**Εικόνα 3: Απεικόνιση DOM ενός Table (Robie, n.a.)**

Το DOM χρησιμεύει ως ένας ανεξάρτητος από πλατφόρμα και ουδέτερος ως προς τη γλώσσα σύνδεσμος μεταξύ γλωσσών script όπως η JavaScript και του βασικού εγγράφου Ιστού, καθώς

απεικονίζει τη δομή του εγγράφου σε μια δενδρική δομή στην οποία κάθε κόμβος είναι ένα ανεξάρτητο, επιλέξιμο αντικείμενο.

Τα HTML mark-ups ορίζουν σχέσεις μεταξύ των διαφορετικών ετικετών. Με αυτόν τον τρόπο, τα στοιχεία του εγγράφου ιστού που προσδιορίζονται από ετικέτες είναι ανώτερα ή δευτερεύοντα, ανάλογα με τον ρόλο τους. Επιπλέον, ορισμένες ετικέτες μπορούν να περιέχονται σε άλλες. Για να μπορέσουν να εμφανιστούν επαρκώς αυτές οι ιεραρχίες στο DOM, χρησιμοποιείται συνήθως η δομή δέντρου που αναφέρθηκε παραπάνω, η οποία καθιστά δυνατή την ανάλογη διάταξη των αντικειμένων (Robie, n.a.).

Ακολουθεί το DOM ενός HTML εγγράφου:




**Εικόνα 4: DOM (Deshpande, 2024)**

## 2.2 JavaScript Frameworks

Έχοντας παρουσιάσει τις δυνατότητες της JavaScript, θα συνεχίσουμε στην ενότητα αυτή μελετώντας τα JavaScript Frameworks.

Πριν όμως ξεκινήσουμε θα πρέπει να ξεκαθαρίσουμε τη διαφορά μεταξύ JavaScript Framework και JavaScript Library (βιβλιοθήκη JavaScript). Μια βιβλιοθήκη JavaScript

αποτελείται από διάφορες συναρτήσεις, από προ-γραμμένα τμήματα κώδικα που μπορούν να κληθούν από την εφαρμογή ή τον ιστότοπο με σκοπό να εκτελεστούν συγκεκριμένες λειτουργίες. Με τον τρόπο αυτό παρέχουν στους προγραμματιστές προκαθορισμένες λειτουργίες για να τους βοηθήσουν στη διαδικασία ανάπτυξης ιστότοπου ή εφαρμογών.

Στον πίνακα που ακολουθεί μπορούμε να δούμε τις πιο γνωστές βιβλιοθήκες JavaScript.

**Πίνακας 1: Κυριότερες βιβλιοθήκες JavaScript (GeeksForGeeks, 2024a)**

| JavaScript Library | Πρώτη κυκλοφορία | Περιγραφή | Εφαρμογές |
|---|---|---|---|
| **jQuery** | 26 Αυγούστου 2006 | Μια βιβλιοθήκη JavaScript που είναι γρήγορη, μικρή και πλούσια σε δυνατότητες.<br>Βοηθά στον χειρισμό DOM και στο χειρισμό συμβάντων σε εφαρμογές web. | Παρέχει συμβατότητα μεταξύ προγραμμάτων περιήγησης και απλοποιεί τον κώδικα. |
| **ReactJS** | 29 Μαΐου 2013 | Μια βιβλιοθήκη JavaScript ανοιχτού κώδικα που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη με δηλωτικό και αποτελεσματικό τρόπο. | Δημιουργεί απλές προβολές για κάθε κατάσταση στην εφαρμογή και ενημερώνει αποτελεσματικά αποδίδοντας ακριβώς τη σωστή τιμή καθώς |

| JavaScript Library | Πρώτη κυκλοφορία | Περιγραφή | Εφαρμογές |
|---|---|---|---|
| | | Η βιβλιοθήκη front-end που βασίζεται σε στοιχεία είναι υπεύθυνη μόνο για το επίπεδο προβολής μιας αρχιτεκτονικής MVC. | αλλάζουν τα δεδομένα. |
| **Lodash** | 23 Απριλίου 2012 | Μια βιβλιοθήκη JavaScript που έχει μεγάλο αριθμό συναρτήσεων. | Η χρήση ενσωματωμένων συναρτήσεων της παρέχει τη δυνατότητα για δημιουργία αρθρωτών και γρήγορων εφαρμογών web. |
| **ElectronJS** | 15 Ιουλίου 2013 | Αναπτύχθηκε κυρίως από το GitHub.<br>Χρησιμοποιεί το NodeJs στον πυρήνα του για την εξυπηρέτηση των σελίδων | Υποστηρίζει την εκτέλεση σε διάφορες πλατφόρμες (cross-platform). |

Αντίθετα ένα JavaScript Framework, είναι κάτι πιο γενικό. Είναι ένα πλήρες σύνολο εργαλείων που βοηθά στη διαμόρφωση και οργάνωση του ιστότοπου ή μιας εφαρμογής. Μπορούμε να πούμε ότι παρέχει τα θεμέλια στους προγραμματιστές και με τον τρόπο αυτό ορίζει το σχεδιασμό μιας ιστοσελίδας ή μιας εφαρμογής. Προσφέρει μια βάση επαναχρησιμοποιήσιμου κώδικα, βιβλιοθηκών και εργαλείων που μειώνουν και απλοποιούν τη διαδικασία δημιουργίας διαδραστικών και δυναμικών διεπαφών χρήστη.

Τα JavaScript framework προσφέρουν συνήθως μια σειρά λειτουργιών/χαρακτηριστικών τα οποία έχουν σχεδιαστεί με σκοπό να βελτιώσουν την παραγωγικότητα των προγραμματιστών και την απόδοση της εφαρμογής. Αυτά τα χαρακτηριστικά μπορεί να περιλαμβάνουν:

- Αρχιτεκτονική βασισμένη σε στοιχεία: Τα JavaScript framework υιοθετούν συχνά μια αρχιτεκτονική που βασίζεται σε στοιχεία (Component-based Architecture), επιτρέποντας στους προγραμματιστές να αναλύουν τις εφαρμογές τους σε επαναχρησιμοποιήσιμα και αυτοτελή στοιχεία. Αυτά τα στοιχεία ενσωματώνουν τόσο τη δομή όσο και τη συμπεριφορά, προάγοντας αρθρωτό και επαναχρησιμοποιήσιμο κώδικα.
- Data binding: Τα framework παρέχουν μηχανισμούς δέσμευσης δεδομένων, διευκολύνοντας τον αυτόματο συγχρονισμό δεδομένων μεταξύ του μοντέλου (δεδομένων) της εφαρμογής και του επιπέδου προβολής (UI). Αυτή η ροή δεδομένων και προς τις δύο κατευθύνσεις, διασφαλίζει ότι οι αλλαγές στα υποκείμενα δεδομένα αντικατοπτρίζονται αμέσως στη διεπαφή χρήστη και αντίστροφα, χωρίς να είναι απαραίτητη η υλοποίηση από την πλευρά του προγραμματιστή πολύπλοκων λειτουργιών του DOM. Η σύνδεση δεδομένων απλοποιεί τη διαδικασία ανάπτυξης, μειώνει τον κώδικα boilerplate και ενισχύει την ανταπόκριση των εφαρμογών web.
- Δρομολόγηση: Οι δυνατότητες δρομολόγησης επιτρέπουν στους προγραμματιστές να ορίσουν τη δομή πλοήγησης των εφαρμογών τους και να αντιστοιχίσουν διευθύνσεις URL σε συγκεκριμένες προβολές ή στοιχεία. Αυτό επιτρέπει τη δημιουργία εφαρμογών μιας σελίδας (SPA) όπου η πλοήγηση πραγματοποιείται χωρίς επαναφόρτωση πλήρους σελίδας. Με τη διαχείριση της δρομολόγησης εντός του πλαισίου, οι προγραμματιστές μπορούν να δημιουργήσουν πιο ρευστές και διαδραστικές εμπειρίες χρήστη, να βελτιώσουν την απόδοση της εφαρμογής και να βελτιώσουν τη βελτιστοποίηση

μηχανών αναζήτησης (SEO) ενεργοποιώντας την απόδοση από την πλευρά του διακομιστή.

- Virtual DOM: Πολλά σύγχρονα framework αξιοποιούν την εικονική τεχνολογία DOM για τη βελτιστοποίηση της απόδοσης και τη βελτίωση της αποτελεσματικότητας των ενημερώσεων της διεπαφής χρήστη. Το εικονικό DOM λειτουργεί ως αναπαράσταση στη μνήμη του πραγματικού DOM, επιτρέποντας στα πλαίσια να εκτελούν λεπτομερείς ενημερώσεις και να ελαχιστοποιούν δαπανηρούς χειρισμούς DOM. Συγκρίνοντας το εικονικό DOM με το πραγματικό DOM, τα πλαίσια μπορούν να αναγνωρίσουν και να εφαρμόσουν μόνο τις απαραίτητες αλλαγές, μειώνοντας τον χρόνο απόδοσης και βελτιώνοντας την ανταπόκριση της εφαρμογής.

Στον πίνακα που ακολουθεί μπορούμε να δούμε τα πιο γνωστά JavaScript Framework.

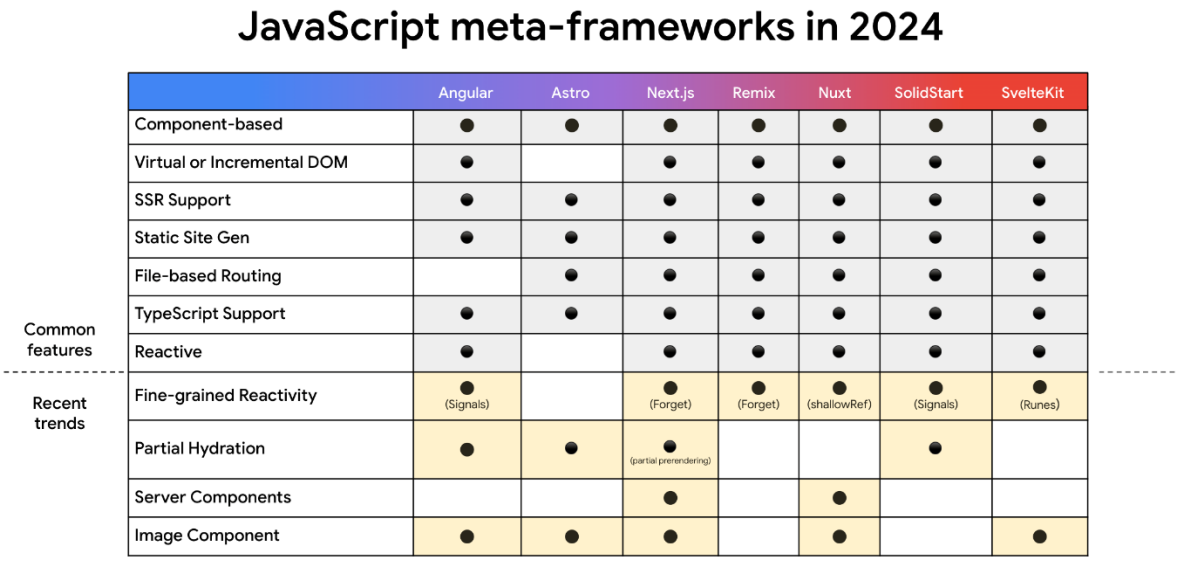
**Πίνακας 2: Κυριότερα JavaScript framework (GeeksForGeeks, 2024a)**

| JavaScript Framework | Πρώτη κυκλοφορία | Περιγραφή | Εφαρμογές |
|---|---|---|---|
| **Angular js** | 20 Οκτωβρίου 2010 | Ένα ισχυρό JavaScript framework που βοηθά στη δημιουργία δυναμικών και διαδραστικών εφαρμογών Ιστού. | Δημιουργεί εφαρμογή με βάση την αρχιτεκτονική MVC που οδηγεί σε καθαρότερο και πιο οργανωμένο κώδικα. |
| **Bootstrap** | 19 Αυγούστου 2011 | Ένα ευρέως χρησιμοποιούμενο πλαίσιο ανοιχτού κώδικα για την ανάπτυξη ιστού, που παρέχει μια συλλογή | Δημιουργεί ανταποκρίσιμους ιστότοπους και παράλληλα παρέχει συμβατότητα μεταξύ διαφορετικών προγραμμάτων περιήγησης. |

| JavaScript Framework | Πρώτη κυκλοφορία | Περιγραφή | Εφαρμογές |
|---|---|---|---|
| | | στοιχείων και εργαλείων HTML, CSS και JavaScript.<br>Δείνει τη δυνατότητα στους προγραμματιστές να δημιουργήσουν εύκολα ανταποκρίσιμους ιστότοπους που προορίζονται για κινητές συσκευές. | |
| **Vue.js** | Φεβρουάριος 2014 | Μπορεί να προσαρμόζεται σταδιακά και έχει μια βιβλιοθήκη που εστιάζει μόνο στο επίπεδο προβολής. | Βοηθά στη δημιουργία γρήγορων και δρομολογούμενων εφαρμογών web. |
| **Node.js** | 27 Μαΐου 2009 | Ένα περιβάλλον χρόνου εκτέλεσης ανοιχτού κώδικα και cross platform που βασίζεται στη μηχανή JavaScript V8 του Chrome για την εκτέλεση κώδικα JavaScript εκτός προγράμματος περιήγησης. | Παρέχει ένα περιβάλλον χρόνου εκτέλεσης με βάση τα events, non blocking I/O υποστηρίζοντας πολλές πλατφόρμες με σκοπό τη δημιουργία εξαιρετικά επεκτάσιμων εφαρμογών από την πλευρά του διακομιστή |

| JavaScript Framework | Πρώτη κυκλοφορία | Περιγραφή | Εφαρμογές |
|---|---|---|---|
| | | | χρησιμοποιώντας JavaScript. |

Πλέον, τα JavaScript framework έχοντας εισέλθει σε μια εποχή ωριμότητας, συγκλίνουν σε πολλά παρόμοια χαρακτηριστικά και αρχιτεκτονικές, μαθαίνοντας το ένα από το άλλο και υιοθετώντας βέλτιστες πρακτικές. Στην εικόνα που ακολουθεί μπορούμε να δούμε τα σημεία σύγκλισης που είναι αρχιτεκτονική που βασίζεται σε στοιχεία (component-based architecture), δρομολόγηση βάσει αρχείων και σύγχρονη υποστήριξη SSR. Αυτή η σύγκλιση καταδεικνύει την ωριμότητα και την εξέλιξη του οικοσυστήματος καθώς τα πλαίσια μαθαίνουν το ένα από το άλλο και υιοθετούν βέλτιστες πρακτικές.




**Εικόνα 5: JavaScript framework τάσεις και σύγκλιση (Hempenius & Osmani, 2024)**

Ταυτόχρονα, μια σειρά από πρόσφατες τάσεις (όπως στοιχεία διακομιστή και διαφορετικές προσεγγίσεις για fine grade reactivity) έχουν κάνει την εμφάνισή τους, οι οποίες υποστηρίζονται από ορισμένα framework (Hempenius & Osmani, 2024).

Ας δούμε τί προσφέρουν  τα JavaScript framework στους προγραμματιστές που τα χρησιμοποιούν:

- Μειωμένος χρόνος ανάπτυξης: οι προγραμματιστές αντί να γράφουν από το μηδέν τον κώδικά τους, μπορούν να αξιοποιήσουν τον προκατασκευασμένο κώδικα για κοινές λειτουργίες, εξοικονομώντας σημαντικό χρόνο και προσπάθεια.
- Βελτιωμένη ποιότητα κώδικα: τα framework συχνά επιβάλλουν συμβάσεις κωδικοποίησης και βέλτιστες πρακτικές, οδηγώντας σε καθαρότερο και πιο διατηρήσιμο κώδικα.
- Βελτιωμένη λειτουργικότητα: τα framework διαθέτουν συχνά ενσωματωμένες λειτουργίες όπως η αντιδραστικότητα, ο χειρισμός δεδομένων, στοιχεία διεπαφής χρήστη και δρομολόγηση, μειώνοντας έως εκμηδενίζοντας την ανάγκη συγγραφής από τον προγραμματιστή των λειτουργιών αυτών.
- Μεγαλύτερη κοινότητα και υποστήριξη: τα δημοφιλή framework έχουν μεγάλες κοινότητες προγραμματιστών, οι οποίες παρέχουν πρόσβαση σε εκτενείς πόρους, σεμινάρια και υποστήριξη σε περίπτωση προβλημάτων.

Ποιοι είναι οι παράγοντες που πρέπει να ελέγξει ένας προγραμματιστής για να επιλέξει να χρησιμοποιήσει ένα JavaScript framework; Η εικόνα που ακολουθεί τους συνοψίζει και στη συνέχεια θα τους παρουσιάσουμε αναλυτικότερα.




**Εικόνα 6: Παράγοντες που πρέπει να ληφθούν υπόψη κατά την επιλογή ενός JavaScript framework (Sencha, 2024)**

- Απαιτήσεις Έργου: Θα πρέπει να ληφθούν υπόψη οι απαιτήσεις του προς υλοποίηση έργου, όπως ενδεικτικά το μέγεθος, η πολυπλοκότητα, η ανάγκη για επεκτασιμότητα καθώς επίσης και οι απαιτήσεις απόδοσης της εφαρμογής Ιστού. Αν λάβουμε υπόωψη μας τα παραπάνω θα επιλέξουμε το καλύτερο πλαίσιο JavaScript για το συγκεκριμένο έργο. Για παράδειγμα, εάν το έργο που θέλουμε να αναπτύξουμε αφορά δημιουργία εφαρμογής ιστού μίας σελίδας, η Angular είναι μια καλή επιλογή με καλό χειρισμό δεδομένων και συμβατότητα μεταξύ πλατφορμών.
- Καμπύλη εκμάθησης και κοινότητα προγραμματιστών: Είναι πολύ σημαντικός ο χρόνος που απαιτείται για την εκμάθηση του framework. Αν για την εκμάθηση του πλαισίου είναι απαραίτητη η αφιέρωση μεγάλου χρονικού διαστήματος, αυτό σημαίνει ότι θα καθυστερήσει σημαντικά η ανάπτυξη της εφαρμογής ιστού. Για το λόγο αυτό, πρέπει να επιλεγεί ένα εύχρηστο και εύκολο framework. Παράλληλα με το χρόνο εκμάθησης θα πρέπει να ληφθούν υπόψη και οι διαθέσιμοι πόροι εκμάθησης. Τέλος θα

πρέπει να αξιολογηθεί και αν το framework υποστηρίζεται από κάποια προγραμματιστική κοινότητα, μέσω της οποίας θα μπορούσαμε να λάβουμε υποστήριξη σε περίπτωση προβλημάτων.

- Απόδοση και Βελτιστοποίηση: Το καλύτερο πλαίσιο JavaScript έχει σχεδιαστεί για να λειτουργεί αποτελεσματικά σε διαφορετικά είδη εφαρμογών ιστού και να χειρίζεται τα δεδομένα απρόσκοπτα. Για παράδειγμα, το Ext JS προσφέρει ένα πλέγμα JS υψηλής απόδοσης που μπορεί να επεξεργαστεί αποτελεσματικά εκατομμύρια εγγραφές χωρίς να επηρεάσει την απόδοση της εφαρμογής, βελτιώνοντας την εμπειρία του χρήστη.
- Ενοποίηση με άλλες βιβλιοθήκες: Ένα καλό framework Javascript συνοδεύεται από τα περισσότερα από τα απαραίτητα εργαλεία που χρειάζονται οι προγραμματιστές για να δημιουργήσουν πλούσιες σε χαρακτηριστικά, ανταποκρινόμενες και διαδραστικές διεπαφές χρήστη. Όμως, υπάρχει περίπτωση οι ανάγκες της εφαρμογής να απαιτούν πρόσθετες βιβλιοθήκες για επεξεργασία δεδομένων, επεξεργασία εμπλουτισμένου κειμένου, χειρισμό DOM και πολλά άλλα. Επομένως, θα πρέπει να αξιλογηθεί εάν το framework υποστηρίζει την ενοποίηση με άλλες βιβλιοθήκες.
- Μακροπρόθεσμη βιωσιμότητα: Η αξιολόγηση της μακροπρόθεσμης βιωσιμότητας του framework είναι ένας άλλος κρίσιμος παράγοντας που πρέπει να ληφθεί υπόψη έτσι ώστε να διασφαλιστεί ότι η εφαρμογή θα συνεχίσει να λειτουργεί αποτελεσματικά για μεγάλο χρονικό διάστημα. Για την αξιολόγηση της μακροπρόθεσμης βιωσιμότητας μπορούν να ληφθούν υπόψη παράγοντες όπως αν κυκλοφορούν νέες ή/και βελτιωμένες εκδόσεις, αν στις εκδόσεις αυτές περιλαμβάνονται νέες δυνατότητες και εάν το framework διαθέτει μια ενεργή κοινότητα προγραμματιστών (Sencha, 2024).

Στον πίνακα που ακολουθεί μπορούμε να δούμε το μερίδιο αγοράς που κατέχουν τα 10 πιο γνωστών JavaScript framework.

**Πίνακας 3: Μερίδιο αγοράς των 10 JavaScript frameworks (Wappalyzer, 2024)**

| Α/Α | Javascript framework | Πλήθος δικτυακών τόπων | Μερίδιο αγοράς |
|---|---|---|---|
| 1 | React | 2.215.000 | 32% |

| | | | |
|---|---|---|---|
| 2 | GSAP | 915.000 | 13% |
| 3 | Vue.js | 803.000 | 12% |
| 4 | styled-components | 630.000 | 9% |
| 5 | Emotion | 539.000 | 8% |
| 6 | Backbone.js | 451.000 | 7% |
| 7 | Redux | 437.000 | 6% |
| 8 | RequireJS | 357.000 | 5% |
| 9 | AngularJS | 291.000 | 4% |
| 10 | AMP | 280.000 | 4% |

### 2.2.1 Angular

Η Angular είναι ένα framework και πλατφόρμα ανάπτυξης, η οποία βασίζεται στην TypeScript. Χρησιμοποιείται για τη δημιουργία διαδικτυακών εφαρμογών μιας σελίδας (single-page web applications). Ως πλατφόρμα, η Angular περιλαμβάνει:

- Ένα framework βασισμένο σε στοιχεία για τη δημιουργία επεκτάσιμων εφαρμογών Ιστού
- Μια συλλογή από καλά ενσωματωμένες βιβλιοθήκες που καλύπτουν μια μεγάλη ποικιλία λειτουργιών, όπως δρομολόγηση, διαχείριση φορμών, επικοινωνία πελάτη-διακομιστή και άλλα

- Μια σουίτα εργαλείων προγραμματιστών η οποία θα προσφέρει στους προγραμματιστές τη δυνατότητα να αναπτύξουν, δημιουργήσουν, δοκιμάσουν και να ενημερώσουν τον κώδικά σας, εύκολα.
- Παρέχει μια ομάδα που υποστήριξης με πάνω από 1,7 εκατομμύρια προγραμματιστές, συγγραφείς βιβλιοθηκών και δημιουργούς περιεχομένου.

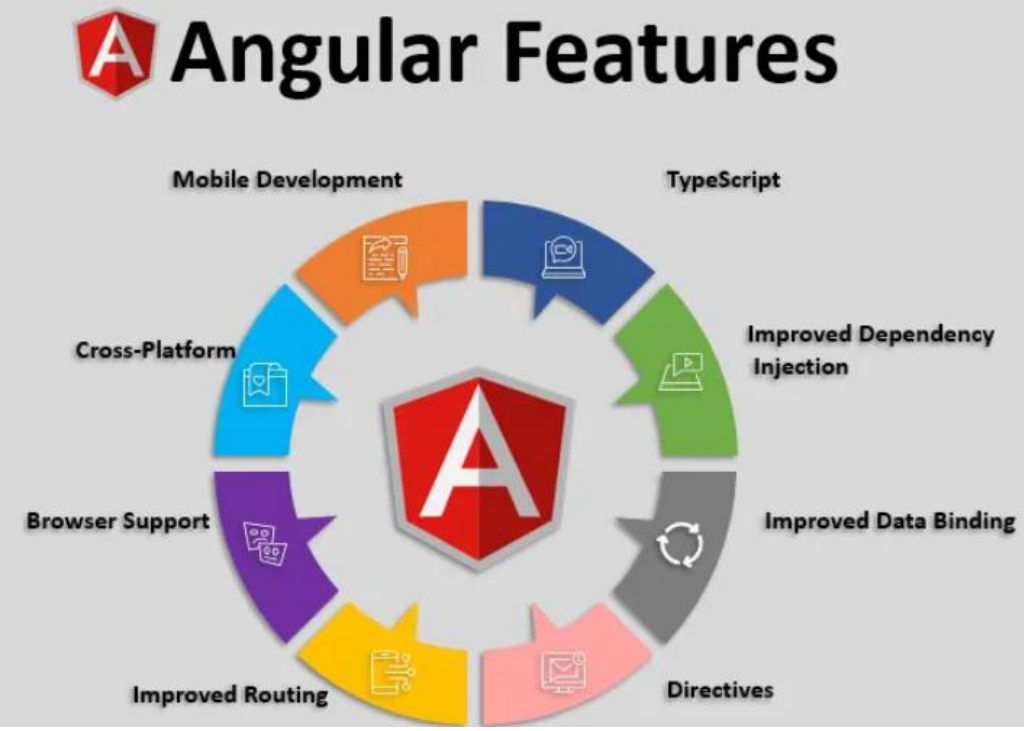


**Εικόνα 7: Angular**

Τα βασικά χαρακτηριστικά που προσφέρει στον προγραμματιστή η Angular είναι:

- Αρχιτεκτονική Model-View-Controller (MVC): Αυτό το μοτίβο σχεδίασης χωρίζει μια εφαρμογή σε τρία διασυνδεδεμένα στοιχεία - μοντέλο, προβολή και ελεγκτής - επιτρέποντας στους προγραμματιστές να διαχειρίζονται σύνθετα έργα πιο αποτελεσματικά.
- Αμφίδρομη σύνδεση δεδομένων (two-way data binding): Με το σύστημα αυτό, ενημερώνεται αυτόματα η προβολή όταν γίνονται αλλαγές στο μοντέλο και αντίστροφα. Η αμφίδρομη σύνδεση δεδομένων εξοικονομεί χρόνο από τους προγραμματιστές και ταυτόχρονα μειώνει την πολυπλοκότητα του έργου.
- Κινούμενα σχέδια υψηλής ποιότητας και δυνατότητες προσβασιμότητας: και τα δύο χαρακτηριστικά συμβάλουν στη σημαντική βελτίωση της εμπειρίας του χρήστη.
- Υποστηρίζει την επαναχρησιμοποίηση στοιχείων, η οποία ενισχύει την παραγωγικότητα των προγραμματιστών, επιτρέποντάς τους να επαναχρησιμοποιήσουν εύκολα τον κώδικα που έχουν γράψει, σε διαφορετικά μέρη μιας εφαρμογής.

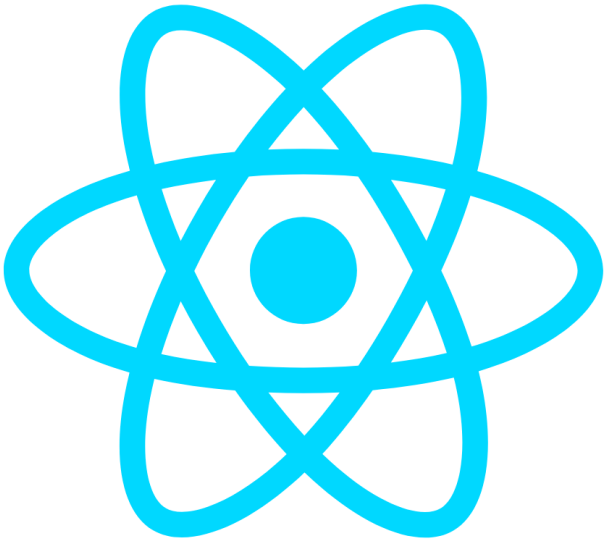
- Οι δυνατότητες δοκιμών της Angular είναι επίσης ισχυρές, με την υποστήριξη του Jasmine (ένα JavaScript framework για δοκιμές το οποίο βασίζεται στη συμπεριφορά). Με τον τρόπο αυτό θα διευκολυνθεί η διασφάλιση της βέλτιστης απόδοσης των εφαρμογών πριν από την ανάπτυξη.
- Χρήση της Typescript: Η TypeScript ορίζει ένα σύνολο τύπων σε JavaScript, το οποίο βοηθά τους χρήστες να γράψουν κώδικα JavaScript που είναι πιο κατανοητός. Όλος ο κώδικας TypeScript μεταγλωττίζεται με JavaScript και μπορεί να εκτελεστεί ομαλά σε οποιαδήποτε πλατφόρμα. Η TypeScript δεν είναι υποχρεωτική για την ανάπτυξη μιας εφαρμογής Angular. Ωστόσο, συνιστάται ιδιαίτερα καθώς προσφέρει καλύτερη συντακτική δομή και με τον τρόπο αυτό οδηγεί στη συγγραφή κώδικα που είναι ευκολότερος στην κατανόηση και τη διατήρηση.
- Υψηλή ταχύτητα και βέλτιστη απόδοση: Η Angular παρέχει πολλές δυνατότητες και βέλτιστες πρακτικές που μπορούν να βοηθήσουν στη διασφάλιση υψηλής ταχύτητας και βέλτιστης απόδοσης της εφαρμογής.
- Dependency Injection: ένα μοτίβο σχεδίασης λογισμικού που επιτρέπει στα αντικείμενα να εισάγονται με τις εξαρτήσεις τους κατά το χρόνο εκτέλεσης, αντί να χρειάζεται να τα ορίζουν στατικά κατά το χρόνο μεταγλώττισης. Το χαρακτηριστικό αυτό διευκολύνει τη σύνταξη επαναχρησιμοποιήσιμου κώδικα καθώς και την εκτέλεση ελέγχων - δοκιμών στην εφαρμογή.
- Angular CLI: Το Angular CLI είναι ένα εργαλείο γραμμής εντολών που διευκολύνει τη δημιουργία, τη διαχείριση και τη δημιουργία εφαρμογών Angular. Το CLI παρέχει μια σειρά από χρήσιμα χαρακτηριστικά, όπως τη δημιουργία στοιχείων, υπηρεσιών και μονάδων, την εκτέλεση δοκιμών καθώς και τη δημιουργία της τελικής έκδοσης της εφαρμογής για το παραγωγικό περιβάλλον. Με τη βοήθεια του οι προγραμματιστές μπορούν να διαχειρίζονται την εφαρμογή καθ' όλη τη διάρκεια του κύκλου ζωής της (Abaas, 2023).




**Εικόνα 8: Χαρακτηριστικά της Angular (Sharma, 2023)**

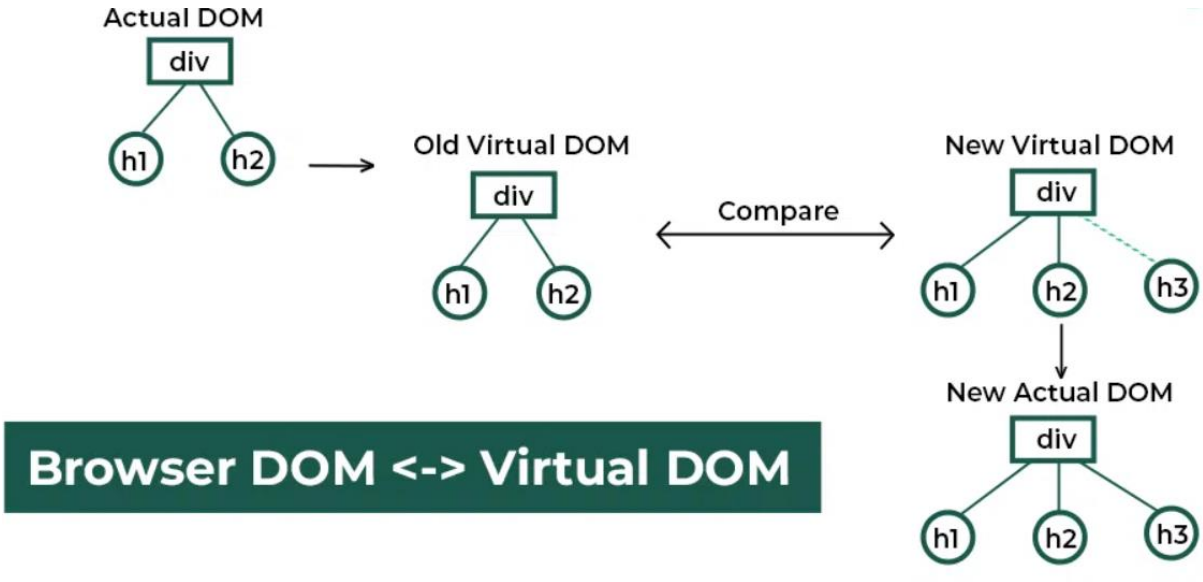
### 2.2.2 React

Η React είναι μια βιβλιοθήκη JavaScript για τη δημιουργία διεπαφών χρήστη (UI) στον Ιστό. Η React είναι μια δηλωτική, βασισμένη σε στοιχεία βιβλιοθήκη που επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα στοιχεία διεπαφής χρήστη και ακολουθεί την προσέγγιση Virtual DOM (Document Object Model), η οποία βελτιστοποιεί την απόδοση rendering ελαχιστοποιώντας τις ενημερώσεις DOM. Ένα από τα πιο σημαντικά πλεονεκτήματα της React είναι η ταχύτητα και ότι λειτουργεί καλά με άλλα εργαλεία και βιβλιοθήκες.



**Εικόνα 9: React**

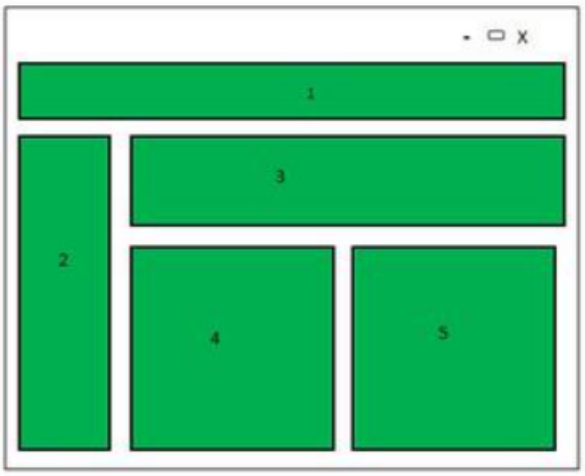
Η React επινοήθηκε από προγραμματιστές του Facebook που βρήκαν το παραδοσιακό DOM αργό. Με την εφαρμογή ενός εικονικού DOM, h React αντιμετώπισε αυτό το ζήτημα και κέρδισε γρήγορα δημοτικότητα.




**Εικόνα 10: Browser DOM vs Virtual DOM (GeeksforGeeks, 2024b)**

Ας δούμε ορισμένα από τα χαρακτηριστικά της React

- Component-Based Architecture: React παρέχει τη δυνατότητα διάσπασης της διεπαφής χρήστη σε μικρότερα, αυτόνομα στοιχεία. Κάθε στοιχείο μπορεί να έχει τη δική του κατάσταση.
- JSX (JavaScript Syntax Extension): Το JSX είναι μια επέκταση σύνταξης για JavaScript που επιτρέπει στους προγραμματιστές να γράφουν κώδικα τύπου HTML μέσα στα αρχεία JavaScript τους. Κάνει τα στοιχεία του React πιο ευανάγνωστα και εκφραστικά.
- Virtual DOM: Η React διατηρεί μια ελαφριά αναπαράσταση του πραγματικού DOM στη μνήμη. Όταν συμβαίνουν αλλαγές, το React ενημερώνει αποτελεσματικά μόνο τα απαραίτητα μέρη του DOM.
- One-way data binding: Τα δεδομένα στη React ρέουν μόνο προς μία κατεύθυνση, δηλαδή τα δεδομένα μεταφέρονται από πάνω προς τα κάτω, δηλαδή από τα μητρικά συστατικά στα θυγατρικά στοιχεία. Οι ιδιότητες στο θυγατρικό στοιχείο δεν μπορούν να επιστρέψουν τα δεδομένα στο γονικό στοιχείο, αλλά μπορούν να επικοινωνούν με τα γονικά στοιχεία για να τροποποιήσουν τις καταστάσεις σύμφωνα με τις παρεχόμενες εισόδους.
- Performance: Η React χρησιμοποιεί εικονικό DOM και ενημερώνει μόνο τα τροποποιημένα μέρη. Έτσι, αυτό κάνει το DOM να λειτουργεί πιο γρήγορα. Επιπλέον το DOM εκτελείται στη μνήμη με αποτέλεσμα να εκτελείται πιο γρήγορα.
- Components: Η React διαιρεί την ιστοσελίδα σε πολλαπλά στοιχεία καθώς βασίζεται σε στοιχεία. Κάθε στοιχείο είναι μέρος του σχεδίου διεπαφής χρήστη που έχει τη δική του λογική και σχεδιασμό όπως φαίνεται στην παρακάτω εικόνα. Έτσι, η λογική του στοιχείου που είναι γραμμένη σε JavaScript το καθιστά εύκολο και ταχύτερο και μπορεί να επαναχρησιμοποιηθεί.




**Εικόνα 11: React component (GeeksForGeeks, 2024b)**

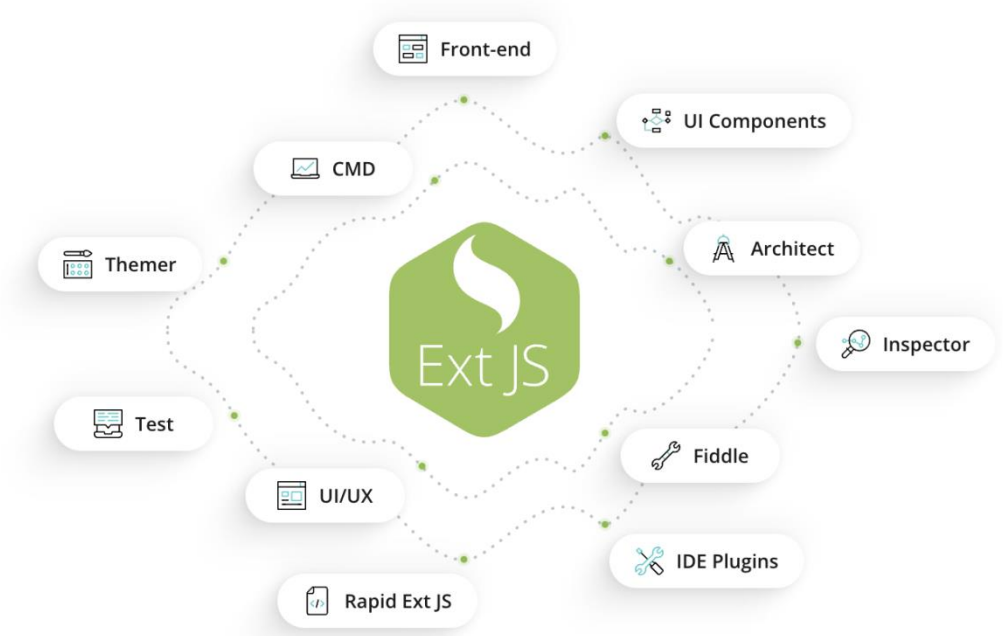
- Single-Page Applications: Η React συνιστάται για τη δημιουργία SPA, επιτρέποντας ομαλές ενημερώσεις περιεχομένου χωρίς επαναφόρτωση σελίδων. Η εστίασή του σε επαναχρησιμοποιήσιμα εξαρτήματα το καθιστά ιδανικό για εφαρμογές σε πραγματικό χρόνο (GeeksForGeeks, 2024b).

### 2.2.3 Ext JS

Το Ext JS είναι ένα ισχυρό front-end Javascript framework για ταχύτερη και αποτελεσματικότερη ανάπτυξη εφαρμογών ιστού και κινητών σε επαγγελματικό επίπεδο. Το Ext JS χρησιμοποιείται σε περισσότερες από 100.00 επιχειρήσεις παγκοσμίως, συμπεριλαμβανομένων των Apple, Canon, American Airlines και Cisco, για την ανάπτυξη λειτουργικών και ασφαλών εφαρμογών. Ας δούμε ορισμένα από τα βασικά χαρακτηριστικά του Ext JS:

- Το Ext JS προσφέρει περισσότερα από 140 προκατασκευασμένα στοιχεία διεπαφής χρήστη υψηλής απόδοσης. Όλα τα εξαρτήματα είναι εξαιρετικά προσαρμόσιμα και πλήρως ελεγμένα. Αυτά τα στοιχεία έχουν σχεδιαστεί για να συνεργάζονται απρόσκοπτα σε οποιαδήποτε εφαρμογή.
- Το JavaScript framework υποστηρίζει επίσης ένα ευρύ φάσμα γραφημάτων, όπως γραφημάτων περιοχών, πίτας, ράβδων και στηλών, καθώς και οπτικοποιήσεις D3, όπως

χάρτες δέντρων και χάρτες θερμότητας, τα οποία μπορούν να χρησιμοποιηθούν σε εφαρμογές ιστού.



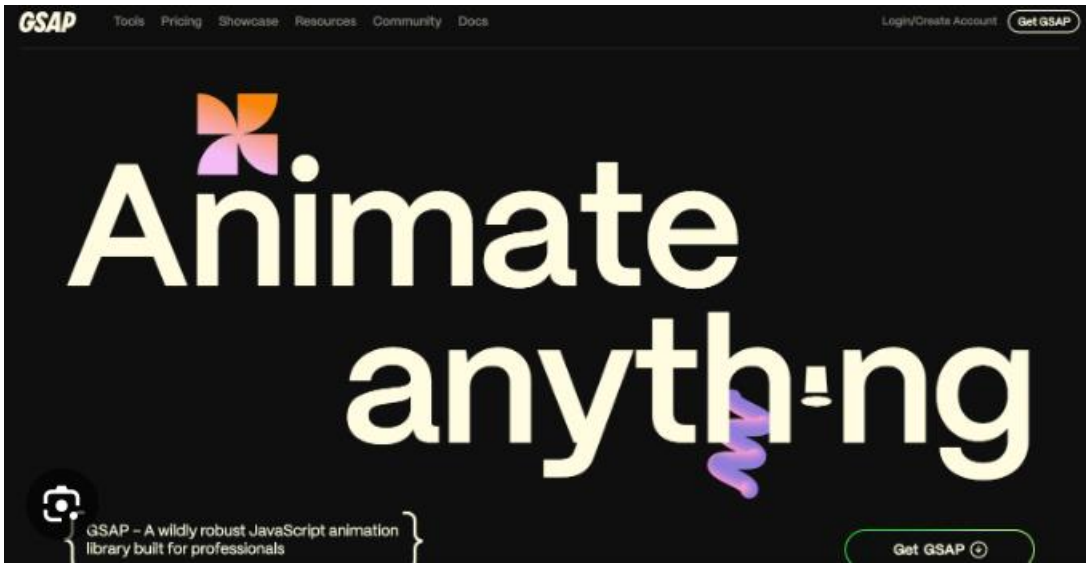

**Εικόνα 12: Ext JS (Sencha,2024)**

- Το Ext JS προσφέρει ένα από τα ταχύτερα και πιο αποτελεσματικά JS data grid. Το συγκεκριμένο data grid είναι προ-δοκιμασμένο σε μια σειρά από προγράμματα περιήγησης και πλατφόρμες και είναι σε θέση να χειρίζεται αποτελεσματικά εκατομμύρια εγγραφές χωρίς συμβιβασμούς στην απόδοση.
- Το συγκεκριμένο framework υποστηρίζει και τις δύο αρχιτεκτονικές MVC και MVVM (Model-View-ViewModel).
- Το Ext JS προσφέρει μια ενιαία βάση κώδικα για την ανάπτυξη εφαρμογών μεταξύ διαφορετικών πλατφορμών. Με τον τρόπο αυτό παρέχει στους προγραμματιστές τη δυνατότητα να προσφέρουν εξαιρετική εμπειρία χρήστη σε επιτραπέζιους υπολογιστές, smartphone και tablet.
- Το Ext JS συνοδεύεται από λεπτομερή τεκμηρίωση που παρέχει όλες τις πληροφορίες που χρειάζεται ο προγραμματιστής για να κατανοήσει τις δυνατότητες, τις δυνατότητες και τη λειτουργικότητά του.

Το μόνο αρνητικό του συγκεκριμένου framework είναι ότι είναι επί πληρωμή, αν και διατίθεται μια δωρεάν έκδοση (Sencha, 2024).

### 2.3.4 GSAP

Το GSAP (GreenSock Animation Platform) είναι ένα JavaScript framework με τη βοήθεια του οποίου παρέχεται animation σε οποιοδήποτε κώδικα JavaScript. Είναι γνωστό για τις υψηλές επιδόσεις του και το ευρύ φάσμα χαρακτηριστικών του, γεγονός που το καθιστά δημοφιλές μεταξύ των προγραμματιστών.




**Εικόνα 13: GSAP (GSAP, 2024)**

Ορισμένα από τα βασικά χαρακτηριστικά του είναι:

- Ομαλή απόδοση, ακόμη και με πολύπλοκα κινούμενα σχέδια
- Εκτεταμένο σύνολο εργαλείων, συμπεριλαμβανομένων των Scroll, SVG, Text και άλλα
- Υποστηρίζει διάφορους τύπους κινούμενων εικόνων: CSS, SVG, canvas, React, Vue, WebGL, χρώματα, συμβολοσειρές, διαδρομές κίνησης κ.λπ.
- Προηγμένη αλληλουχία και έλεγχος κινούμενων εικόνων

- Επιτάχυνση GPU και άλλες βελτιστοποιήσεις
- Προσαρμόσιμα χειριστήρια διευκόλυνσης και αναπαραγωγής

Στα πλεονεκτήματα του συγκεκριμένου framework μπορούμε να αναφέρουμε

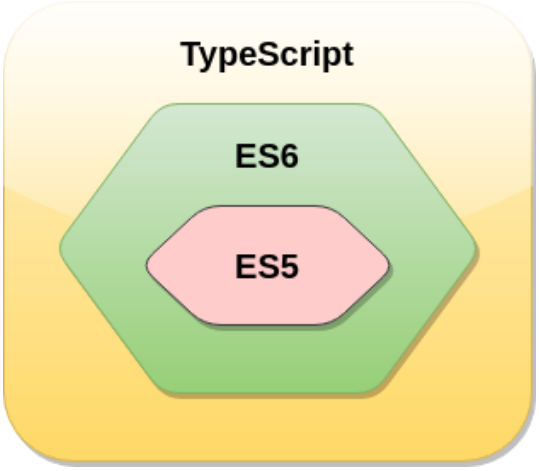
- Υψηλής απόδοσης
- Εκτεταμένο σύνολο εργαλείων
- Υποστηρίζει διάφορους τύπους κινούμενων εικόνων
- Προηγμένη αλληλουχία και έλεγχος
- Προσαρμόσιμα χειριστήρια διευκόλυνσης και αναπαραγωγής

Ταυτόχρονα αναγνωρίζονται και ορισμένα μειονεκτήματα τα οποία είναι:

- Απότομη καμπύλη εκμάθησης για πολύπλοκα κινούμενα σχέδια
- Μπορεί να απαιτεί επιπλέον ρύθμιση για πολύπλοκα κινούμενα σχέδια
- Περιορισμένη υποστήριξη για παλαιότερα προγράμματα περιήγησης
- Δεν είναι ιδανικό για απλά κινούμενα σχέδια
- Μπορεί να χρειάζονται πρόσθετες προσθήκες για συγκεκριμένες λειτουργίες (Kramer, 2024)

### 2.3 TypeScript

Η TypeScript είναι μια καθαρά αντικειμενοστραφή γλώσσα προγραμματισμού ανοιχτού κώδικα. Είναι ένα υπερσύνολο της JavaScript, προσφέροντας τυποποίηση και μεταγλωττίζεται σε απλή JavaScript. Η Typescript είναι σχεδιασμένη για ανάπτυξη εφαρμογών JavaScript μεγάλης κλίμακας, η οποία μπορεί να εκτελεστεί σε οποιοδήποτε πρόγραμμα περιήγησης, οποιοδήποτε κεντρικό υπολογιστή και οποιοδήποτε λειτουργικό σύστημα, αφού πρώτα όμως μεταγλωττιστεί. Η TypeScript είναι η έκδοση ES6 της JavaScript με ορισμένες πρόσθετες δυνατότητες.




**Εικόνα 14: Η Typescript σε σχέση με την έκδοση ES6 της JavaScript (Javatpoint, 2024)**

Όπως αναφέραμε και παραπάνω, η TypeScript δεν μπορεί να εκτελεστεί απευθείας στο πρόγραμμα περιήγησης. Χρειάζεται έναν μεταγλωττιστή για να μεταγλωττίσει το αρχείο και να το δημιουργήσει σε αρχείο JavaScript, το οποίο μπορεί να εκτελεστεί απευθείας στο πρόγραμμα περιήγησης. Το αρχείο προέλευσης TypeScript βρίσκεται στην επέκταση ".ts" (Javatpoint, 2024).




**Εικόνα 15: Διαδικασία μεταγλώττισης TypeScript (Javatpoint, 2024)**

Τα χαρακτηριστικά που προσφέρει η TypeScript είναι:

- Static Typing: Σε αντίθεση με την JavaScript, όπου οι τύποι καθορίζονται κατά το χρόνο εκτέλεσης, στην TypeScript μπορούμε να δηλώσουμε τον τύπο μιας μεταβλητής και ο μεταγλωττιστής TypeScript ελέγχει εάν τηρούμε τους τύπους που έχουμε δηλώσει.
- Type Annotations: Μπορούμε να καθορίσουμε ρητά τύπους χρησιμοποιώντας σχολιασμούς. Για παράδειγμα:

```
let myNumber: number = 42;

let myString: string = "Hello, TypeScript!";
```

- Interfaces: Η TypeScript επιτρέπει να οριστούν προσαρμοσμένοι τύποι δεδομένων χρησιμοποιώντας διεπαφές. Τα interface λειτουργούν σαν σχέδια για αντικείμενα:

```
interface Person {
 name: string;
 age: number;
 }
let john: Person = { name: "John", age: 25 };
```

- Classes and Inheritance: Η TypeScript υποστηρίζει αντικειμενοστραφή χαρακτηριστικά προγραμματισμού όπως κλάσεις και κληρονομικότητα.

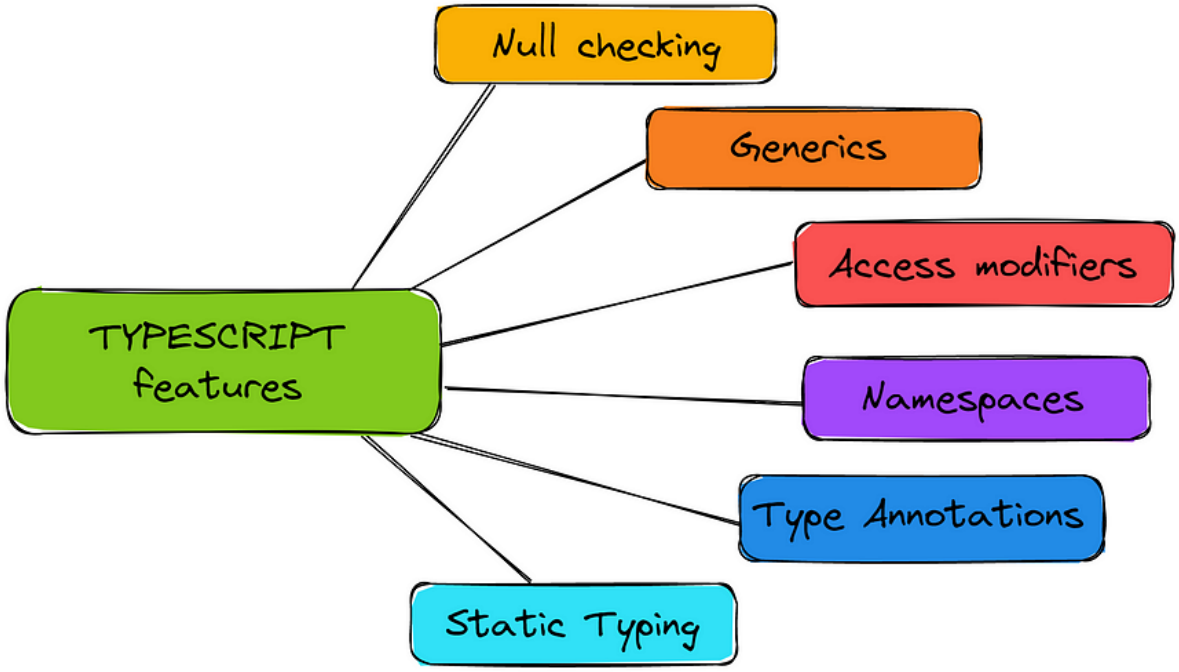
```
class Animal {
   name: string;
   constructor(name: string) {
    this.name = name;
   }
   makeSound() {
     console.log("Some generic sound");
   }
}
class Dog extends Animal {
```

```
  makeSound() {
    console.log("Woof, woof!");
  }
}
let myDog: Dog = new Dog("Buddy");
```

- Generics: Η TypeScript έχει μια ισχυρή δυνατότητα που ονομάζεται generics, η οποία επιτρέπει στους προγραμματιστές να γράφουν συναρτήσεις και κλάσεις που λειτουργούν με διάφορους τύπους δεδομένων.

```
function identity<T>(arg: T): T {
 return arg;
 }
let result: number = identity(42);
```

- Tooling Support: Η TypeScript έχει εξαιρετική υποστήριξη εργαλείων με δυνατότητες όπως αυτόματη συμπλήρωση, ανακατασκευή και έλεγχος σφαλμάτων απευθείας στο πρόγραμμα επεξεργασίας κώδικα.
- Συμβατότητα με JavaScript: Ο υπάρχων κώδικας JavaScript είναι έγκυρος κώδικας TypeScript (Krishnakumar, 2023).




**Εικόνα 16: Χαρακτηριστικά της TypeScript(Krishnakumar, 2023)**

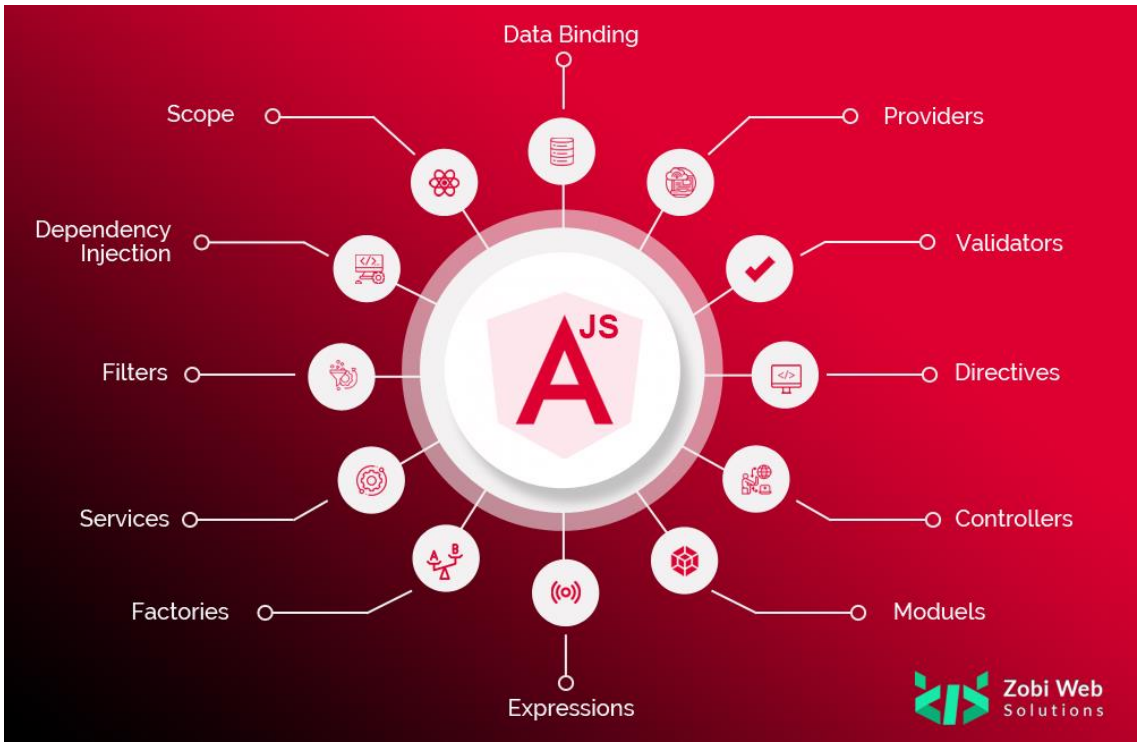
# ΚΕΦΑΛΑΙΟ 3

# Αρχιτεκτονική της Angular

## 3.1 Εισαγωγή

Στο προηγούμενο παρουσιάσαμε τα πιο γνωστά JavaScript framework και μέσα σε αυτά είδαμε και την Angular. Στο κεφάλαιο αυτό θα μελετήσουμε πιο αναλυτικά την Angular, ξεκινώντας από μια σύντομη ιστορική αναδρομή, θα παρουσιάσουμε την Αρχιτεκτονική της Angular και στη συνέχεια θα δούμε τη πώς γίνεται η δημιουργία μιας Angular εφαρμογής και ποια είναι η δομή της.

Η Angular έχει γίνει αρκετά γνωστή και καθημερινά αυξάνει το μερίδιο της στην χρήση σε web εφαρμογές επειδή παρέχει στους προγραμματιστές μια πληθώρα από χαρακτηριστικά όπως μπορούμε να δούμε στην εικόνα που ακολουθεί:




**Εικόνα 17: Χαρακτηριστικά της Angular (ZobiWebSolutions, 2020)**

### 3.2 Ιστορική αναδρομή

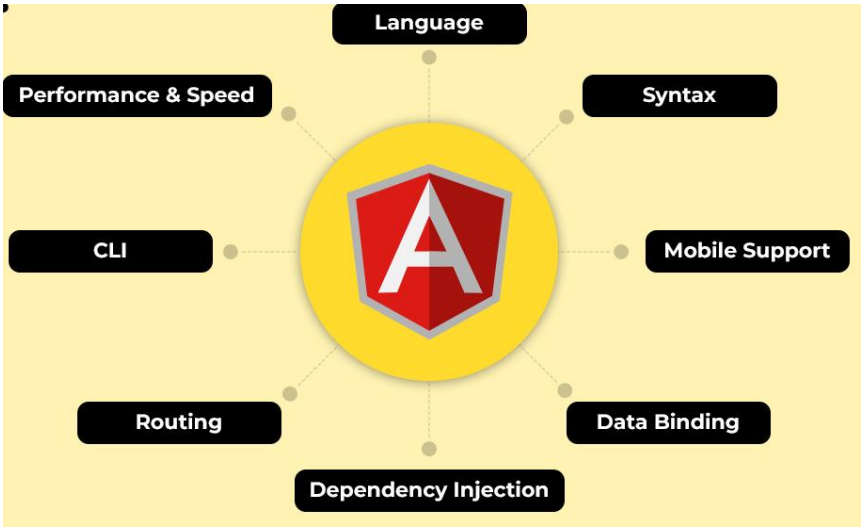
Πριν από την κυκλοφορία της Angular, ένας υπάλληλος της Google με το όνομα Miško Hevery, ανέπτυξε ένα έργο το οποίο είχε ως στόχο να διευκολύνει τη δημιουργία διαδικτυακών εφαρμογών για μερικά εσωτερικά έργα στα οποία εργαζόταν. Αυτό το δευτερεύον έργο έγινε αργότερα γνωστό ως AngularJS (Angular λόγω του < > σε HTML). Ο Misko και ο Adam Abrons (μηχανικός της Google) άρχισαν να δημιουργούν μερικές ακόμη εσωτερικές εφαρμογές με το AngularJS, το οποίο τελικά κυκλοφόρησε ως έργο ανοιχτού κώδικα το 2010.

Διαθέτοντας ένα σημαντικό σύνολο πλεονεκτημάτων, η AngularJS αύξησε γρήγορα τη δημοτικότητά της. Αλλά δυστυχώς για την AngularJS, οι τεχνολογίες αναπτύχθηκαν ακόμη πιο γρήγορα. Σε λίγο, η πλατφόρμα σταμάτησε να ανταποκρίνεται στις αυξανόμενες απαιτήσεις του διαδικτύου και έχασε τη θέση της από νεότερους ανταγωνιστές που συνέχιζαν να εμφανίζονται. Ακόμη και οι συνεχείς βελτιώσεις δεν μπορούσαν να βελτιώσουν τη θέση της AngularJS σε σχέση με τον ανταγωνισμό. Έτσι, το 2014, η Google αποφάσισε να την ξαναγράψει. Για αρκετά χρόνια, η AngularJS υπήρχε δίπλα-δίπλα με τον διάδοχό της που ονομάζεται απλά Angular. Αλλά το 2022, έφτασε στο τέλος της ζωής.

Τον Σεπτέμβριο του 2016, η Google κυκλοφόρησε την Angular 2. Η διαφορά μεταξύ της παλιάς AngularJS και της νέας έκδοσης ήταν τόσο ριζική που ήταν αδύνατη η μετάβαση από το ένα framework στο άλλο.. Οι βασικές αλλαγές είναι οι ακόλουθες:

- TypeScript αντί για JavaScript. Σε αντίθεση με τον προκάτοχό του που βασίζεται στο JS, το Angular χρησιμοποιεί TypeScript.
- Τα Component είναι τα πιο βασικά δομικά στοιχεία.
- Καλύτερη απόδοση. Οι εφαρμογές που έχουν γραφτεί χρησιμοποιώντας το framework Angular μπορούν να είναι πέντε φορές πιο γρήγορες από αυτές που χρησιμοποιούν το AngularJS.
- CLI. Η AngularJS δεν έχει τη δική της διεπαφή γραμμής εντολών (CLI). Η Angular 2 (και οι επόμενες εκδόσεις), από την άλλη πλευρά, διαθέτει ένα CLI που επιτρέπει τη γρήγορη δημιουργία Component, Service κ.λπ.
- Φιλικότητα προς το κινητό. Η AngularJS δεν σχεδιάστηκε με γνώμονα τα προγράμματα περιήγησης για κινητά. Κατά την ανάπτυξη της Angular, η Google έλαβε

υπόψη αυτό το κενό, οπότε το νέο πλαίσιο έλαβε την απαραίτητη υποστήριξη για προγράμματα περιήγησης για κινητές συσκευές.




**Εικόνα 18: Χαρακτηριστικά Angular 2 (Sakthi, 2024)**

Μετά την Angular 2, το framework έχει προσκολληθεί στη σημασιολογική έκδοση (SemVer) που σημαίνει ότι οι νέες εκδόσεις, εμφανίζονται κάθε έξι μήνες και διαθέτουν συμβατότητα προς τα πίσω. Παράλληλα παρέχουν υποστήριξη για 18 μήνες. Η Angular 4 διέθετε έναν βελτιωμένο μεταγλωττιστή που μείωσε το μέγεθος του παραγόμενου κώδικα στο μισό.

Με την Angular 7, το CLI βελτιώθηκε με προτροπές για την επεξήγηση των λειτουργιών και των σκοπών των στοιχείων, με αποτέλεσμα η χρήση του CLI έγινε πιο διαισθητική.

Η Angular 8 έφερε μια προεπισκόπηση του Ivy, ενός renderer νέας γενιάς που προορίζεται να αντικαταστήσει τον παλαιότερο μεταγλωττιστή— View Engine. Ο στόχος του Ivy ήταν να κάνει την εφαρμογή να τρέχει πιο γρήγορα και να μειώσει το μέγεθος του πακέτου. Μια πλήρης αλλαγή στο Ivy συνέβη στην Angular 9 όπου έγινε προεπιλεγμένος μεταγλωττιστής για όλες τις εφαρμογές. Μια άλλη ευρέως διαφημιζόμενη καινοτομία της ένατης έκδοσης είναι η αυτόματη ενσωμάτωση γραμματοσειρών κατά το στάδιο της μεταγλώττισης το Angular CLI πραγματοποιεί λήψη και ενσωματώνει γραμματοσειρές που χρησιμοποιούνται στο έργο.

Η Angular 10 επικεντρώθηκε στις βελτιώσεις ποιότητας και απόδοσης αντί σε νέες δυνατότητες. Στην Angular 11 παρουσιάστηκαν Component Test Harnesses για να βοηθήσουν στη δοκιμή στοιχείων.

Η έκδοση 12 επικεντρώθηκε σε στιλιστικές βελτιώσεις, δίνοντας στους προγραμματιστές περισσότερο έλεγχο του έργου και αφαιρώντας στοιχεία παλαιού τύπου. Αυτή η τάση επεκτάθηκε και στην επόμενη έκδοση. Με την Angular 13, έληξε η υποστήριξη για το View Engine (το πρώτο Angular renderer), τον Internet Explorer 11 και τις εκδόσεις Node.js πριν από την 12.20.0.

Με την Angular 14 εμφανίστηκαν οι Strictly Typed Forms, μια προσθήκη που ζητούσαν πολύ καιρό οι προγραμματιστές. Στις προηγούμενες εκδόσεις οι φόρμες δεν είχαν ορισμούς τύπων με αποτέλεσμα, το TypeScript δεν μπορούσε να εντοπίσει πολλά κοινά σφάλματα.

Η έκδοση 15 έφερε ένα νέο σύστημα API και εισήχθησαν τα σταθερά, αυτόνομα API που επιτρέπουν στους προγραμματιστές να δημιουργούν εφαρμογές χωρίς να βασίζονται σε μονάδες Ng. Αυτή η ενημέρωση όχι μόνο βελτιώνει τη συνολική εμπειρία και απόδοση, αλλά μειώνει επίσης την ανάγκη για κώδικα boilerplate.

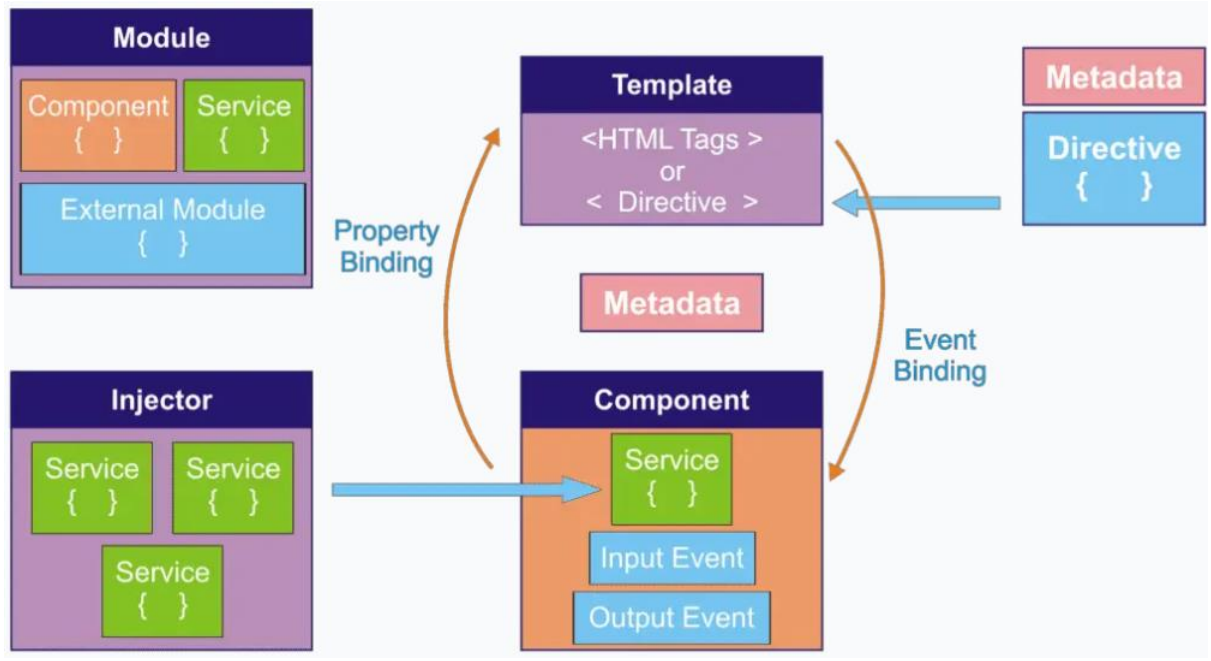
Η έκδοση 16 που παρουσιάστηκε το Μάιο του 2023, ενσωμάτωσε περισσότερες από 2500 απαιτήσεις των προγραμματιστών για νέες λειτουργίες. Τον Νοέμβριο του 2023 παρουσιάστηκε η έκδοση 17 που έφερε την ενσωματωμένη σύνταξη για τον έλεγχο της ροής. Με το εργαλείο αυτό οι προγραμματιστές έχουν στα χέρια τους μια πιο εκφραστική και αποτελεσματική σύνταξη προτύπων, εμπλουτίζοντας περαιτέρω τις δυνατότητες της Angular. Με την έκδοση 17 εμφανίστηκε και ο ενσωματωμένος έλεγχος ροής (@if, If @else If @else, @For) (Sakthi, 2024).




**Εικόνα 19: Οι εκδόσεις της Angular (Sakthi, 2024)**

### 3.3 Αρχιτεκτονική

Στην αρχιτεκτονική της Angular μπορούμε να διακρίνουμε αρκετά διαφορετικά μέρη όπως τα Component, Template, Directive, Service και Module.




**Εικόνα 20: Αρχιτεκτονική της Angular (NgDevelop, 2024)**

Αν θέλουμε να υλοποιήσουμε μια εφαρμογή στην Angular, θα πρέπει πρώτα απ' όλα να γνωρίζουμε την αρχιτεκτονική της Angular και στη συνέχεια αναλύσουμε την εφαρμογή περιγράφοντας τις δυνατότητες που θέλουμε να έχει. Με τον τρόπο αυτό θα μπορέσουμε να αντιστοιχήσουμε κάθε τμήμα της εφαρμογής με τα αντίστοιχα τμήματα της αρχιτεκτονικής της Angular και η εφαρμογή μας θα είναι πιο γρήγορη και πιο ομαλή στις μεταβάσεις της.

Για την ανάπτυξη οποιασδήποτε διαδικτυακής εφαρμογής, η Angular ακολουθεί τα μοτίβα σχεδίασης MVC (Model – View - Controller) και MVVM (Model – View -ViewModel), τα οποία διευκολύνουν μια δομημένη και οργανωμένη προσέγγιση στο σχεδιασμό της εφαρμογής, ενώ παράλληλα διευκολύνουν τη διαχείριση κώδικα, αυξάνουν την ευκολία στη συντήρησή του κ.λπ. Αυτοί οι τύποι μοτίβων σχεδίασης διατηρούν συνήθως μια σαφή διάκριση μεταξύ των δεδομένων (Model - Μοντέλο), της διεπαφής χρήστη (View - Προβολή) και της λογικής (Controller/ViewModel), το οποίο έχει σαν αποτέλεσμα να μας οδηγήσει σε πιο επεκτάσιμες και ελεγχόμενες εφαρμογές. Παράλληλα, η χρήση των συγκεκριμένων μοντέλων παρέχει στον προγραμματιστή τη δυνατότητα επαναχρησιμοποίησης κώδικα και διασφαλίζει μια ομαλότερη διαδικασία ανάπτυξης για μεγάλα και πολύπλοκα έργα.

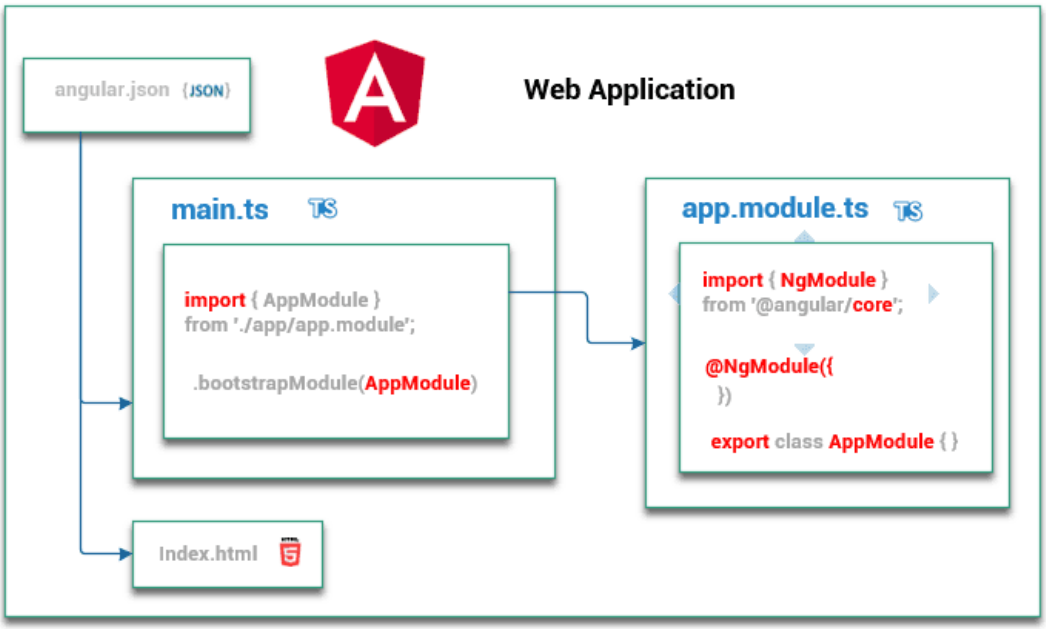
Στη συνέχεια θα προχωρήσουμε στην ανάλυση των διαφορετικών μερών της αρχιτεκτονικής και θα παρουσιάσουμε παράλληλα και τη δομή τους.

### 3.3.1 Module

Τα τελευταία χρόνια, οι διαδικτυακές εφαρμογές έχουν γίνει όλο και πιο περίπλοκες. Ακόμα και η χρήση της JavaScript, για την υλοποίηση των προγραμμάτων δε βελτίωσε ούτε μείωσε την πολυπλοκότητα. Έπρεπε να δημιουργηθεί ένας μηχανισμός που να επιτρέπει στα προγράμματα JavaScript να χωρίζονται σε διάφορα μέρη και στη συνέχεια να μπορούν να συνεργαστούν μεταξύ τους για την επίτευξη του σκοπού της εφαρμογής. Η λύση ήταν τα modules. Με τη χρήση των module επιτεύχθηκαν:

- Δόμηση της εφαρμογής σε πολλά αρχεία αντί για ένα
- Επαναχρησιμοποίηση κώδικα
- Ενθυλάκωση κώδικα και καλύτερος έλεγχος
- Καλύτερη διαχείριση των εξαρτήσεων.

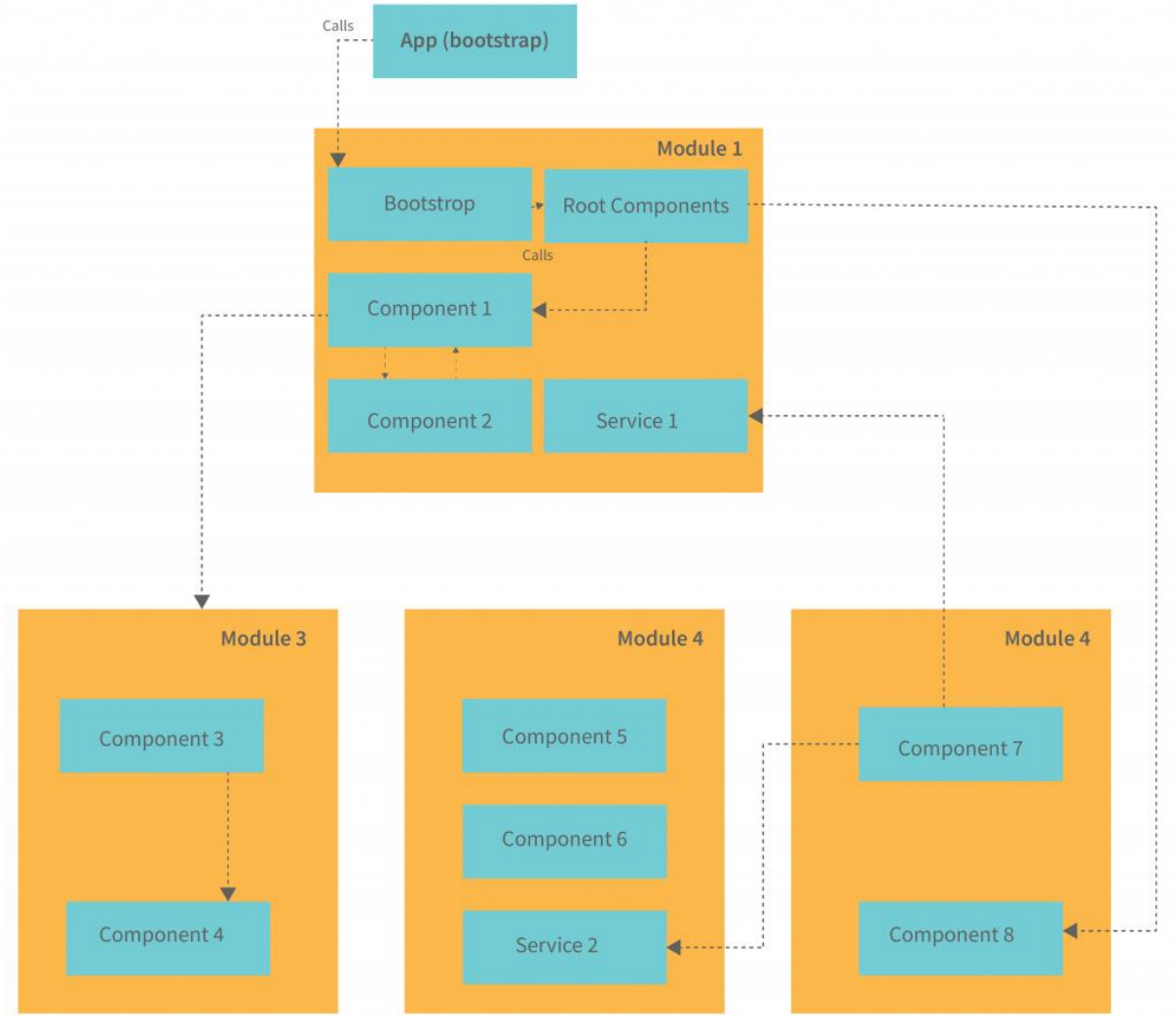
Η Angular ακολούθησε και αυτή την λογική των modules της JavaScript. Στην Angular πλέον ονομάζονται NgModules. Ένα NgModule ορίζει ένα σύνολο από Components, συσχετίζοντάς τα για να σχηματίσει λειτουργικές μονάδες. Κάθε εφαρμογή Angular έχει ένα Root Module, που ονομάζεται συμβατικά AppModule, το οποίο παρέχει τον μηχανισμό bootstrap που εκκινεί την εφαρμογή.




**Εικόνα 21: Συσχέτιση Agular εφαρμογής με NgModule (Ganatan, 2024)**

Πριν συνεχίσουμε με την ανάλυση των NgModules, ας δούμε εν συντομία πώς χρησιμοποιούνται κατά την εκκίνηση της εφαρμογής. Το Root Module της εφαρμογής μας είναι αυτό που κάνουμε bootstrap για να ξεκινήσει η εφαρμογή. Όπως μπορούμε να δούμε στην παραπάνω εικόνα, κατά την εκκίνηση της εφαρμογής γίνεται bootstrapping το root AppModule (το οποίο ονομάζετε entryComponent). Η διαδικασία bootstrapping δημιουργεί τα Components που αναφέρονται στον πίνακα «bootstrap» και εισάγει το καθένα στο DOM του προγράμματος περιήγησης. Έτσι, κάθε Component είναι η βάση του δικού του δέντρου από Components.

Στην εικόνα που ακολουθεί μπορούμε να δούμε πώς ένα Module επηρεάζει τα Component αλλά και πώς αυτά με τη σειρά τους επηρεάζουν το Module.




**Εικόνα 22: Αλληλεπίδραση Component με Module**

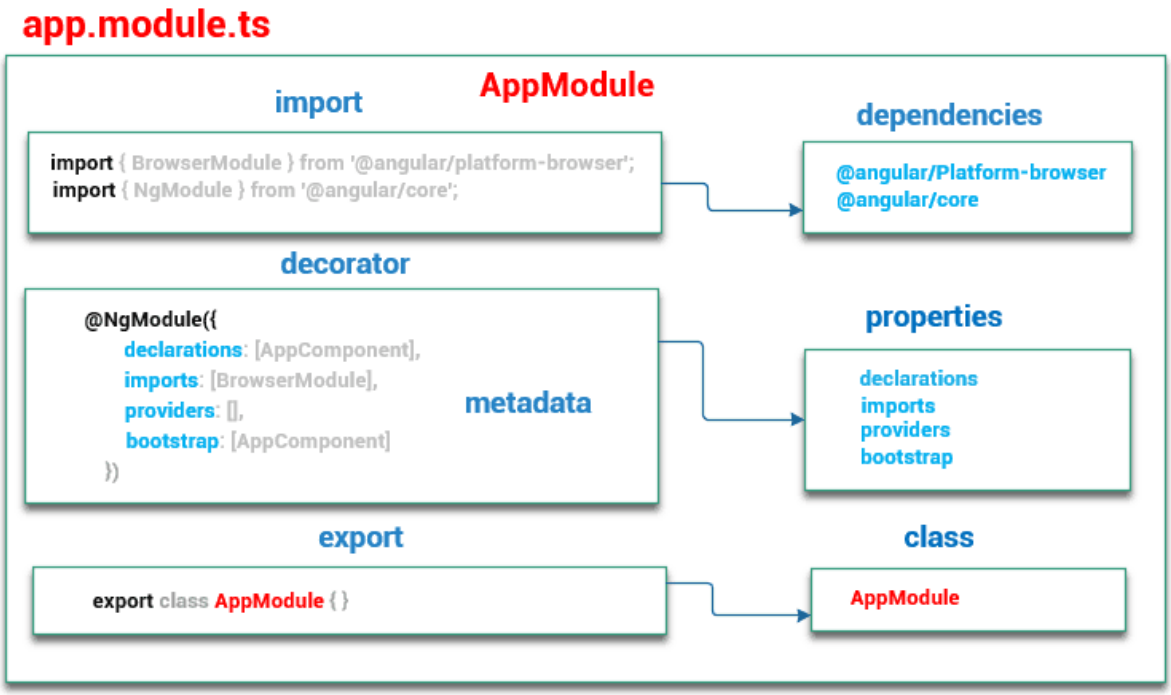
Τα NgModules, ακόμα και αν δεν έχουν κάποια μεταξύ τους συσχέτιση, μπορούν να εισάγουν λειτουργικότητα από άλλες μονάδες NgModules και ταυτόχρονα να επιτρέπουν την εξαγωγή και χρήση της δικής τους λειτουργικότητας από άλλα NgModule. Αυτό που είναι γνωστό ως Angular Libraries είναι τα NgModules.

Για να δηλώσουμε ένα NgModule θα πρέπει να χρησιμοποιήσουμε τον decorator «@NgModule» στην κλάση μας. Αυτός ο decorator είναι ένα αντικείμενο μεταδεδομένων του οποίου οι ιδιότητες περιγράφουν τη λειτουργική μονάδα. Οι πιο σημαντικές ιδιότητες, είναι οι εξής:

- Declarations: Components, Directives και Σωλήνες που ανήκουν στο NgModule

- Exports: το υποσύνολο των δηλώσεων που θα πρέπει να είναι ορατό και χρησιμοποιήσιμο στα Components των άλλων NgModules
- Imports: άλλες ενότητες των οποίων οι εξαγόμενες κλάσεις απαιτούνται από Components που δηλώνονται στο NgModule
- Providers: λίστα με τα απαραίτητα Services που, επειδή παρατίθενται εδώ, γίνονται διαθέσιμες σε όλη την εφαρμογή
- Bootstrap: το κύριο View της εφαρμογής, που ονομάζεται Root Component, το οποίο φιλοξενεί όλα τα άλλα view της εφαρμογής (Bors, 2018).

Όλα τα παραπάνω μπορούν να απεικονιστούν καλύτερα στην εικόνα που ακολουθεί:




**Εικόνα 23: Δομή του app.module.ts (Ganatan, 2024)**

Ακολουθεί ένα παράδειγμα ενός NgModule:

```
// Importing Angular Libraries

import { BrowserModule } from '@angular/platform-browser';

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

```
import { FormsModule } from '@angular/forms';
import { HttpModule } from '@angular/http';
// Importing the AppComponent
import { AppComponent } from './app.component';
// Importing a custom feature module
import { CustomFeatureModule } from './custom-feature-module/custom-feature-module.module';
// Declaring the Module
@NgModule({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule,
        FormsModule,
        HttpModule,
        CustomerDashboardModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

### 3.3.2 Component

Το Angular Component είναι το θεμελιώδες δομικό στοιχείο για τη δημιουργία εφαρμογών στην Angular. Χρησιμοποιώντας την καλά ορισμένη αρχιτεκτονική τους, η Angular στοχεύει να παρέχει την κατάλληλη δομή για την οργάνωση του έργου σε διαχειρίσιμα, καλά οργανωμένα μέρη με σαφείς ευθύνες, έτσι ώστε ο κώδικάς που παράγει ο προγραμματιστής να είναι διατηρήσιμος και επεκτάσιμος.

Ένα Component ελέγχει μια ενημερωμένη έκδοση κώδικα οθόνης που ονομάζεται Προβολή. Η λογική ενός Component ορίζεται μέσα σε μια κλάση. Η Angular δημιουργεί, ενημερώνει και καταστρέφει Component καθώς ο χρήστης μετακινείται στην εφαρμογή.

Ένα Component προσδιορίζεται από τον decorator «@Component» που έχει ένα σύνολο ιδιοτήτων. Οι βασικές ιδιότητες είναι οι ακόλουθες:

- Selector: λέει πώς το Component αναφέρεται από την HTML, δηλαδή αντιστοιχεί σε μια ετικέτα HTML.
- templateUrl: δίνει τη διαδρομή του προτύπου HTML. Το Component έχει ένα View και αυτό ορίζεται μέσω ενός προτύπου HTML. Αυτό το αρχείο HTML περιέχει επίσης ορισμένα συντακτικά στοιχεία που περιλαμβάνονται στην Angular. Θα μπορούσαμε αντί να χρησιμοποιήσουμε τη συγκεκριμένη ιδιότητα, να χρησιμοποιήσουμε την ιδιότητα template και στη συνέχεια να πληκτρολογήσουμε την HTML που θέλουμε να συμπεριληφθεί.
- Providers: μια σειρά από Dependency Injection Providers για υπηρεσίες που απαιτεί το στοιχείο.

Ακολουθεί ένα παράδειγμα ενός Component με τις βασικές του ιδιότητες

```
@Component({
  selector:    'my-component',
  templateUrl: './my-component.component.html',
  providers:  [ MyService ]
```

```
})
export class MyComponent implements OnInit {
   // Some code
}
```

https://www.scholarhat.com/tutorial/angular/angular-components

### 3.3.3 Template

Στην Angular, ένα πρότυπο είναι μια προβολή της οποίας ο σκοπός είναι να εμφανίζει δεδομένα και να τα ενημερώνει όταν ενεργοποιηθεί κάποιο συμβάν. Η HTML είναι η προεπιλεγμένη γλώσσα προτύπου. Το επίπεδο προβολής διαχωρίζεται από το υπόλοιπο πλαίσιο με πρότυπα, κάτι που επιτρέπει στους προγραμματιστές να τροποποιούν το επίπεδο προβολής χωρίς να δημιουργούν προβλήματα στην εφαρμογή.

Κάθε πρότυπο Angular στην εφαρμογή είναι ένα κομμάτι HTML που θα εμφανίσει το πρόγραμμα περιήγησης ως μέρος της σελίδας. Όπως το συμβατικό HTML, ένα πρότυπο Angular HTML δημιουργεί μια προβολή ή μια διεπαφή χρήστη στο πρόγραμμα περιήγησης, αλλά με πολύ περισσότερες δυνατότητες.

Όταν χρησιμοποιηθεί το Angular CLI για τη δημιουργία μιας εφαρμογής Angular, το προεπιλεγμένο πρότυπο είναι το app.component.html, το οποίο περιέχει το σύμβολο κράτησης θέσης HTML.

Τα πρότυπα μπορούν να οριστούν με δύο τρόπους.

- Ενσωματωμένο πρότυπο (Inline Template)
- Εξωτερικό πρότυπο (External Template)

Θα ξεκινήσουμε παρουσιάζοντας τον τρόπο ορισμένου του προτύπου ως ενσωματωμένο πρότυπο.

Χρησιμοποιώντας την ιδιότητα template του decorator @Component, μπορούμε να δημιουργήσουμε ένα πρότυπο απευθείας στην κλάση στοιχείων. Στο παράδειγμα που ακολουθεί μπορούμε να δούμε τη χρήση της ιδιότητας template, σε κάποιο αρχείο Component.ts, στην οποία ανατίθεται συγκεκριμένη τιμή με κώδικα html.

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


@Component({
  selector: 'app-hi',
  template: `
  <h1> Welcome </h1>
  <h2> Name: {{ Name }}</h2>
`,
  styleUrls: ['./hi.component.css']
})
export class HiComponent implements OnInit {
  Name : string = "XYZ";
  constructor() { }


  ngOnInit(): void {
  }
}
```

Το εξωτερικό πρότυπο χρησιμοποιείται από προεπιλογή από το Angular CLI. Χρησιμοποιεί την επιλογή templateUrl και με τον τρόπο αυτό συνδέει ένα πρότυπο με ένα στοιχείο.

Μέχρι στιγμής παρουσιάσαμε το πώς μπορούμε να δηλώσουμε στην Angular ένα template. Θα συνεχίσουμε αναφέροντας τα ποιο σημαντικά στη σύνταξη των template, αναφέροντας ότι όλη η σύνταξη HTML είναι έγκυρη σύνταξη προτύπου. Εκτός όμως από την ετικέτα <script>, η οποία έχει αφαιρεθεί εξαλείφοντας έτσι τον κίνδυνο επιθέσεων script injection.

Θα ξεκινήσουμε παρουσιάζοντας την παρεμβολή (Interpolation). Η παρεμβολή αναφέρεται στην ενσωμάτωση εκφράσεων σε επισημασμένο κείμενο. Από προεπιλογή, η παρεμβολή χρησιμοποιεί τα διπλά άγκιστρα {{ και }} ως οριοθέτες. Με το παράδειγμα που ακολουθεί μπορούμε να δούμε πώς λειτουργεί η παρεμβολή, δίνοντας μια τιμή σε μια μεταβλητή και μετά χρησιμοποιούμε την παρεμβολή για να εμφανίσουμε την τιμή της μεταβλητής σε ένα template. Η Angular αντικαθιστά το currentCustomer με την τιμή συμβολοσειράς της αντίστοιχης ιδιότητας στοιχείου, όπου στην περίπτωσή μας είναι το 'Maria':

```
currentCustomer = 'Maria';

<h3>Current customer: {{ currentCustomer }}</h3>
```

Θα συνεχίσουμε με τη σύνδεση δεδομένων (data binding). Το data binding διατηρεί αυτόματα τη σελίδα ενημερωμένη με βάση την κατάσταση της εφαρμογής σας. Η κύρια χρήση του είναι για να καθοριστούν διάφορα αντικείμενα όπως η προέλευση (src) μιας εικόνας, η κατάσταση ενός κουμπιού ή ta δεδομένα για έναν συγκεκριμένο χρήστη.

Οι τύποι σύνδεσης μπορούν να ομαδοποιηθούν σε τρεις κατηγορίες που διακρίνονται από την κατεύθυνση της ροής δεδομένων:

- source-to-view
- view-to-source
- view-to-source-to-view

Ο πίνακας που ακολουθεί περιέχει για την κάθε κατηγορία παράδειγμα σύνταξης.

**Πίνακας 4: Παραδείγματα σύνταξης για κάθε κατηγορία σύνδεσης δεδομένων (Angular, 2024)**

| Κατηγορία | Σύνταξη | Τύπος |
|---|---|---|
| source-to-view<br>(Μονόδρομος) | {{expression}}<br>[target]="expression" | Interpolation<br>Property<br>Attribute<br>Class<br>Style |
| view-to-source<br>(Μονόδρομος) | (target)="statement" | Event |
| Αμφίδρομη | [(target)]="expression" | Αμφίδρομη |

Για να γίνει καλύτερα κατανοητή η χρήση ας δούμε κάποια παραδείγματα. Θα ξεκινήσουμε από το source-to-vies και τον τύπο Property binding, το οποίο μας βοηθά να ορίσουμε τιμές για ιδιότητες στοιχείων HTML ή οδηγιών. Χρησιμοποιήστε το Property binding για εναλλαγή λειτουργιών κουμπιών, ορισμός διαδρομών μέσω προγραμματισμού και κοινή χρήση τιμών μεταξύ στοιχείων. Κατά τη χρήση του Property binding θα μετακινηθεί μια τιμή προς μία κατεύθυνση, από την ιδιότητα ενός στοιχείου σε μια ιδιότητα στοιχείου προορισμού.

Αν θέλουμε να ορίσουμε την ιδιότητα ενός στοιχείου, θα πρέπει να την περικλείσουμε σε αγκύλες, [], οι οποίες προσδιορίζουν την ιδιότητα ως ιδιότητα προορισμού. Μια ιδιότητα προορισμού είναι η ιδιότητα DOM στην οποία θέλουμε να εκχωρήσουμε μια τιμή. Για παράδειγμα αν θέλουμε να εκχωρήσουμε μια τιμή σε μια ιδιότητα προορισμού src του στοιχείου εικόνας, θα γράψουμε το ακόλουθο:

```
<img alt="item" [src]="itemImageUrl">
```

Αν για παράδειγμα θέλουμε να απενεργοποιήσουμε ένα button, μπορούμε να ορίσουμε την ιδιότητα disabled στην τιμή της μεταβλητής isUnchanged η οποία έχει τιμή ίση με true.

```
<button type="button" [disabled]="isUnchanged">Disabled Button</button>
```

Ας δούμε τώρα τον τρόπο χρήσης του τύπου view-to-source και συγκεκριμένα τη σύνδεση με ένα Event. Αυτή η σύνταξη αποτελείται από ένα όνομα συμβάντος στόχου μέσα σε παρένθεση στα αριστερά ενός συμβόλου = και μια δήλωση προτύπου στα δεξιά. Ας δούμε το ακόλουθο παράδειγμα όπου το όνομα συμβάντος προορισμού είναι click και η δήλωση προτύπου είναι το onSave().

```
<button (click)="onSave()">Save</button>
```

Η σύνδεση συμβάντος ακούει τα συμβάντα κλικ του κουμπιού και καλεί τη μέθοδο onSave() του στοιχείου κάθε φορά που γίνεται κλικ (Angular, 2024).

### 3.3.4 Directive

Τα Directives είναι κλάσεις που προσθέτουν επιπλέον συμπεριφορά σε στοιχεία στις εφαρμογές Angular. Τα Directives μπορούν να χρησιμοποιηθούν για να διαχειριστούμε φόρμες, λίστες, στυλ και το τί βλέπουν οι χρήστες. Υπάρχουν τρεις διαφορετικοί τύποι από Directives, όπως μπορούμε να δούμε στον πίνακα που ακολουθεί:

**Πίνακας 5: Τύποι Directive (Angular, 2024)**

| Τύποι Directives | Λεπτομέρειες |
|---|---|
| Components | Χρησιμοποιούνται με ένα Template. Αυτός ο τύπος Direcive είναι ο πιο κοινός τύπος από τους τρεις. |
| Attribute directives | Για αλλαγή της εμφάνισης ή τη συμπεριφορά ενός στοιχείου, ενός Component ή ενός άλλου Directive. |

| | |
|---|---|
| Structural directives | Για αλλαγή της διάταξης του προσθέτοντας και αφαιρώντας στοιχεία DOM. |

Τα Attribute directive ακούν και τροποποιούν τη συμπεριφορά άλλων στοιχείων, ιδιοτήτων, ιδιοτήτων και στοιχείων HTML. Πολλές οδηγίες χαρακτηριστικών ορίζονται μέσω λειτουργικών μονάδων όπως το CommonModule, το RouterModule και το FormsModule. Οι πιο κοινές οδηγίες Attribute directive είναι οι εξής:

**Πίνακας 6: Οι πιο κοινές οδηγίες Attribute directive (Angular, 2024)**

| Common directives | Λεπτομέρειες |
|---|---|
| NgClass | Προσθέτει και αφαιρεί ένα σύνολο κλάσεων CSS. |
| NgStyle | Προσθέτει και αφαιρεί ένα σύνολο στυλ HTML. |
| NgModel | Προσθέτει αμφίδρομη σύνδεση δεδομένων σε ένα στοιχείο φόρμας HTML. |

Χρησιμοποιώντας το NgClass μπορούμε να προσθέσουμε ή αφαιρέσουμε πολλές κλάσεις CSS ταυτόχρονα (για την προσθήκη ή αφαίρεση μιας κλάσης καλύτερα να χρησιμοποιηθεί class binding αντί της NgClass).

Για να χρησιμοποιηθεί το NgClass, θα πρέπει να εισαχθεί το CommonModule και να προστεθεί στη λίστα εισαγωγών του Component, όπως στο παράδειγμα που ακολουθεί.

```
import {CommonModule} from '@angular/common';
/* . . . */
@Component({
  standalone: true,
  /* . . . */
  imports: [
```

```
  CommonModule, // <-- import into the component
  /* . . . */
 ],
})
export class AppComponent implements OnInit {
 /* . . . */
}
```

Για να ορίσουμε στυλ σε ένα στοιχείο, αρκεί να προσθέσουμε το NgClass και να το ορίσουμε ίσο με μια έκφραση. Για παράδειγμα, έστω ότι το isSpecial είναι ένα boolean που ορίζεται σε true στο app.component.ts. Επειδή το isSpecial είναι αληθές, το ngClass εφαρμόζει την κλάση special στο  στο παράδειγμα που ακολουθεί:

```
<!-- toggle the "special" class on/off with a property -->
<div [ngClass]="isSpecial ? 'special' : """>This div is special</div>
```

Όμως η χρήση του NgClass δεν περιορίζεται σε αυτή που δείξαμε παραπάνω. Μπορεί να χρησιμοποιηθεί σε μια μέθοδο, όπως στο ακόλουθο παράδειγμα, η setCurrentClasses() ορίζει την ιδιότητα currentClasses με ένα αντικείμενο που προσθέτει ή αφαιρεί τρεις κλάσεις με βάση την αληθή ή ψευδή κατάσταση τριών άλλων ιδιοτήτων συστατικού. Κάθε κλειδί του αντικειμένου είναι ένα όνομα κλάσης CSS. Εάν ένα κλειδί είναι αληθές, το ngClass προσθέτει την κλάση. Εάν ένα κλειδί είναι ψευδές, το ngClass αφαιρεί την κλάση.

```
currentClasses: Record<string, boolean> = {};
/* . . . */
setCurrentClasses() {
```

```
  // CSS classes: added/removed per current state of component properties
  this.currentClasses = {
    saveable: this.canSave,
    modified: !this.isUnchanged,
    special: this.isSpecial,
  };
}
```

Στο Template, προσθέτουμε την ιδιότητα ngClass με τιμή ίση το currentClasses για να ορίσουμε τις κλάσεις του στοιχείου:

```
<div [ngClass]="currentClasses">This div is initially saveable, unchanged, and special.</div>
```

Για να χρησιμοποιηθεί το NgStyle, θα πρέπει να εισαχθεί το CommonModule και να το προσθέσουμε στη λίστα εισαγωγών του Component, όπως κάναμε παραπάνω και για το ngClass. Μπορούμε να χρησιμοποιήσουμε το NgStyle για να ορίσουμε πολλαπλά ενσωματωμένα στυλ ταυτόχρονα, με βάση την κατάσταση του Component. Για να χρησιμοποιήσουμε το NgStyle, προσθέτουμε μια μέθοδο στην κλάση Component. Στο παράδειγμα που ακολουθεί, η setCurrentStyles() ορίζει την ιδιότητα currentStyles με ένα αντικείμενο που ορίζει τρία στυλ, με βάση την κατάσταση τριών άλλων ιδιοτήτων συστατικού.

```
currentStyles: Record<string, string> = {};
/* . . . */
setCurrentStyles() {
  // CSS styles: set per current state of component properties
  this.currentStyles = {
```

```
    'font-style': this.canSave ? 'italic' : 'normal',
    'font-weight': !this.isUnchanged ? 'bold' : 'normal',
    'font-size': this.isSpecial ? '24px' : '12px',
  };
}
```

Για να ορίσουμε το στυλ ενός element αρκεί να προσθέσουμε την ιδιότητα ngStyles θέτοντάς τη ίση με την τιμή currentStyles:

```
<div [ngStyle]="currentStyles">
  This div is initially italic, normal weight, and extra large (24px).
</div>
```

Οι structural directives είναι υπεύθυνες για τη διάταξη HTML. Διαμορφώνουν ή αναδιαμορφώνουν τη δομή του DOM, συνήθως προσθέτοντας, αφαιρώντας και ελέγχοντας τα στοιχεία του host στα οποία είναι συνδεδεμένα. Στον πίνακα που ακολουθεί μπορούμε να δούμε τις πιο κοινές structural directives:

**Πίνακας 7: Πιο συχνές structural directives (Angular, 2024)**

| Structural directives | Λεπτομέρειες |
|---|---|
| NgIf | Δημιουργεί ή απορρίπτει υπό όρους δευτερεύουσες προβολές από το πρότυπο. |
| NgFor | Για την επανάληψη ενός κόμβου για κάθε στοιχείο σε μια λίστα. |
| NgSwitch | Ένα σύνολο από directive που εναλλάσσονται μεταξύ εναλλακτικών προβολών. |

Για να χρησιμοποιηθεί κάποιο από τα παραπάνω, θα πρέπει να εισαχθεί το CommonModule και να το προσθέσουμε στη λίστα εισαγωγών του Component, όπως κάναμε νωρίτερα και για το ngClass.

Ας δούμε ένα παράδειγμα χρήσης τους ngIf για να προσθέσουμε ή να αφαιρέσουμε ένα στοιχείο. Όταν το NgIf είναι false, η Angular αφαιρεί ένα στοιχείο και τους απογόνους του από το DOM. Στη συνέχεια, η Angular απορρίπτει τα components τους, κάτι που απελευθερώνει μνήμη και πόρους. Για να προσθέσουμε ή να αφαιρέσουμε ένα στοιχείο, αρκεί να συνδέσουμε το *ngIf σε μια έκφραση συνθήκης όπως το isActive στο ακόλουθο παράδειγμα.

```
<app-item-detail *ngIf="isActive" [item]="item"></app-item-detail>
```

Στο παράδειγμα αυτό όταν η έκφραση isActive επιστρέφει αληθή τιμή, το NgIf προσθέτει το ItemDetailComponent στο DOM. Όταν η έκφραση είναι ψευδής, το NgIf αφαιρεί το ItemDetailComponent από το DOM και απορρίπτει το στοιχείο και όλα τα components του.

Θα συνεχίσουμε παρουσιάζοντας ένα παράδειγμα για τη χρήση της NgFor για να παρουσιάσουμε μια λίστα στοιχείων. Θα πρέπει να οριστεί ένα μπλοκ HTML που καθορίζει πώς η Angular αποδίδει ένα μεμονωμένο στοιχείο. Για να εμφανιστούν τα στοιχεία, θα αντιστοιχίσουμε το συντομογραφικό στοιχείο let item of items στο *ngFor.

```
<div *ngFor="let item of items">{{item.name}}</div>
```

Η συμβολοσειρά "let item of items" δίνει εντολή στην Angular να κάνει τα ακόλουθα βήματα:

- Να αποθηκεύσει κάθε στοιχείο στον πίνακα στοιχείων στη μεταβλητή βρόχου τοπικού στοιχείου
- Να κάνει κάθε στοιχείο διαθέσιμο στο πρότυπο HTML για κάθε επανάληψη
- Να μεταφράσει το «let item of items» σε ένα <ng-template> γύρω από το στοιχείο
- Να επαναλάβει το <ng-template> για κάθε στοιχείο της λίστας

Θα ολοκληρώσουμε παρουσιάζοντας το ngSwitch. Όπως η δήλωση switch της JavaScript, το NgSwitch εμφανίζει ένα στοιχείο από πολλά πιθανά στοιχεία, με βάση μια συνθήκη. Η Angular τοποθετεί μόνο το επιλεγμένο στοιχείο στο DOM. Το NgSwitch είναι ένα σύνολο τριών οδηγιών όπως μπορούμε να δούμε στον πίνακα που ακολουθεί:

| NgSwitch directives | Λεπτομέρειες |
|---|---|
| NgSwitch | Μια οδηγία χαρακτηριστικών που αλλάζει τη συμπεριφορά των συνοδευτικών οδηγιών της. |
| NgSwitchCase | Structural directive που προσθέτει το στοιχείο του στο DOM όταν η δεσμευμένη τιμή του ισούται με την τιμή του switch και αφαιρεί την δεσμευμένη τιμή του όταν δεν ισούται με την τιμή του switch. |
| NgSwitchDefault | Structural directive που προσθέτει το στοιχείο της στο DOM όταν δεν υπάρχει επιλεγμένο NgSwitchCase. |

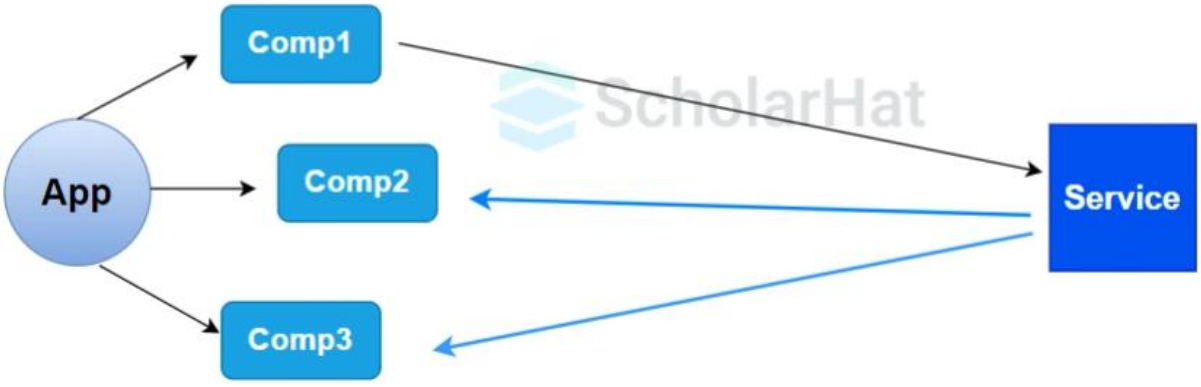
Σε ένα στοιχείο, όπως ένα , προσθέτουμε το ngSwitch, το οποίο να λαμβάνει τιμή από μια παράσταση. Στη συνέχεια συνδέουμε με τα *ngSwitchCase και *ngSwitchDefault τα στοιχεία, όπως στο παράδειγμα που ακολουθεί (Angular, 2024):

```
<div [ngSwitch]="currentItem.feature">
  <app-stout-item    *ngSwitchCase="'stout'"    [item]="currentItem"></app-stout-item>
  <app-device-item   *ngSwitchCase="'slim'"     [item]="currentItem"></app-device-item>
  <app-lost-item     *ngSwitchCase="'vintage'"  [item]="currentItem"></app-lost-item>
  <app-best-item     *ngSwitchCase="'bright'"   [item]="currentItem"></app-best-item>
<!-- . . . -->
  <app-unknown-item  *ngSwitchDefault           [item]="currentItem"></app-unknown-item>
</div>
```

### 3.3.5 Service

Ένα Service είναι χρήσιμο για να ορίσει κάποια δεδομένα – στοιχεία, τα οποία δεν χωρούν ή δεν πρέπει να οριστούν σε ένα Component. Ένα Service είναι μια κλάση με έναν καλά καθορισμένο σκοπό. Για παράδειγμα, θα πρέπει να δημιουργήσουμε ένα Service όταν δύο ή περισσότερα Component ή άλλα στοιχεία πρέπει να έχουν πρόσβαση στα ίδια δεδομένα ή εάν θέλουμε να ενσωματώσουμε τις αλληλεπιδράσεις με έναν διακομιστή ιστού ή εάν θέλουμε να ορίσουμε τον τρόπο επικύρωσης των δεδομένων που εισάγει ο χρήστης.

Τα Service είναι Singleton, επομένως υπάρχει μόνο μία παρουσία για κάθε Service που ορίζουμε. Είναι stateless objects που μπορούν να κληθούν από οποιοδήποτε στοιχείο. Σκοπός τους είναι να μας βοηθήσουν να χωρίσουμε την εφαρμογή μας σε μικρές, διαφορετικές λογικές μονάδες που μπορούν να επαναχρησιμοποιηθούν.




**Εικόνα 24: Angular Service (ScholarHat, 2024)**

Ακολουθεί ένα παράδειγμα ενός Service (Bors, 2018):

```
export class Logger {
  log(msg: any)   { console.log(msg); }
  error(msg: any) { console.error(msg); }
  warn(msg: any)  { console.warn(msg); }
```

```
}
```

### 3.3.6 Dependency injection

Συχνά οι προγραμματιστές έχουν έρθει αντιμέτωποι με την ανάγκη χρήσης ενός εξωτερικού αντικειμένου από ένα μπλοκ κώδικα, το οποίο δημιουργεί μια εξάρτηση. Οι εξαρτήσεις αυτές μπορούν να δημιουργηθούν στο ίδιο μπλοκ όπου χρειάζονται, αλλά αυτό θα οδηγήσει το συγκεκριμένο μπλοκ κώδικα να εξαρτάται από έναν εξωτερικό αντικείμενο. Εάν υπάρξει αλλαγή στον τρόπο δημιουργίας του αντικειμένου, το μπλοκ που το χρησιμοποιεί πρέπει να τροποποιηθεί.

Έτσι, η δημιουργία εξάρτησης όπου χρειάζεται καθιστά τον κώδικα εύθραυστο και δύσκολο να διατηρηθεί. Καθιστά επίσης δύσκολο τον έλεγχο του κώδικα, καθώς οι εξαρτήσεις που δημιουργούνται σε αυτόν δεν μπορούν να χλευαστούν.

Το Dependency Injection στην Angular, που ονομάζεται επίσης DI, είναι ένα Design Pattern στο οποίο μία ή περισσότερες εξαρτήσεις (Services) εγχέονται σε ένα εξαρτημένο αντικείμενο (Πελάτης). Με τον τρόπο αυτό οι προγραμματιστές μπορούν να εφαρμόσουν μια χαλαρά συνδεδεμένη αρχιτεκτονική, διαχωρίζοντας τη δημιουργία των εξαρτήσεων ενός πελάτη από τη συμπεριφορά τους.

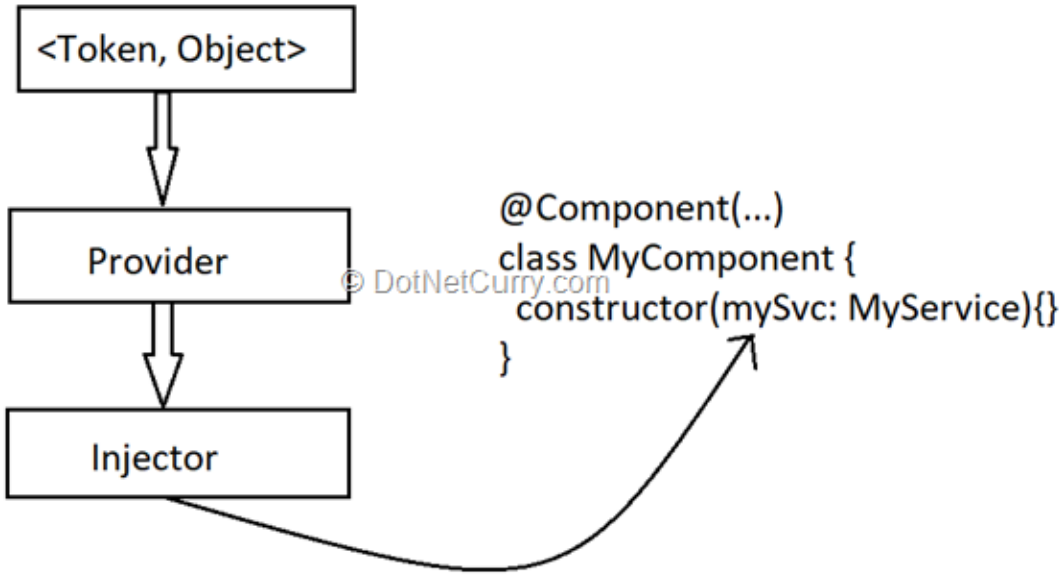
Το Pattern DI χρησιμοποιείται συνήθως για την εφαρμογή της Αρχής Αντιστροφής του Ελέγχου, η οποία με λίγα λόγια διαχωρίζει το «τι πρέπει να συμβεί» από το «όταν πρέπει να συμβεί». Με άλλα λόγια, πρόκειται να αφήσουμε κάποιον άλλο να χειριστεί τη ροή του ελέγχου.

Η Angular χρησιμοποιεί το δικό του πλαίσιο Dependency Injection το οποίο αποτελείται από τα ακόλουθα:

- Injector: που εκθέτει τα API. Είναι υπεύθυνος για τη δημιουργία Service instances και την έγχυσή τους σε κλάσεις.
- Provider: που ενημερώνει και καθοδηγεί στον Injector πώς να δημιουργήσει ένα instance of a dependency.

- Dependency: ο τύπος της οποίας πρέπει να δημιουργηθεί ένα αντικείμενο.

Ο τρόπος λειτουργίας του Dependency, του Provider και του Injector απεικονίζεται στην εικόνα που ακολουθεί.




**Εικόνα 25: Dependency injection στην Angular (Kiran, 2018)**

Η Angular διαθέτει σύστημα Injection Ιεραρχικής Εξάρτησης. Υπάρχει ένα δέντρο Injectors που είναι παράλληλο με το δέντρο Component μιας εφαρμογής. Μια εφαρμογή μπορεί να έχει πολλαπλούς Injectors. Αυτό σημαίνει ότι μπορούμε να διαμορφώσουμε Παρόχους σε διαφορετικά επίπεδα:

- Για όλη την εφαρμογή κατά την εκκίνηση. Όλα τα δευτερεύοντα Injectors θα βλέπουν τον Πάροχο και θα μοιράζονται το στιγμιότυπο.
- Για ένα συγκεκριμένο Component και όλα τα sub-Component . Τα άλλα Component δεν θα βλέπουν τον Πάροχο.
- Για Services. Χρησιμοποιούν ένα από τα Injectors από το στοιχείο που καλεί την αλυσίδα Service.

Όταν χρησιμοποιούμε το DI με το Angular, θα δούμε κυρίως τον decorator «@Injectable». Αυτός ο decorator επισημαίνει μια τάξη ως διαθέσιμη στο Injector για δημιουργία (Bors, 2018).

Ας δούμε ένα παράδειγμα, όπου θέλουμε να δημιουργήσουμε ένα βοηθό στην κουζίνα. Ο βοηθός αυτό (RecipeService), θέλουμε να βρίσκει συνταγές στο διαδίκτυο ή από μια τοπική βάση δεδομένων. Ξεκινάμε με την υλοποίηση του RecipeService.

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root' // This makes it a single assistant for the whole app
})
export class RecipeService {

  constructor(private http: HttpClient) { }

  getRecipes(): Promise<Recipe[]> {
    return this.http.get<Recipe[]>('https://api.example.com/recipes');
  }
}

interface Recipe {
  id: number;
  name: string;
  ingredients: string[];
}
```

Το RecipeService του έχει τον decorator @Injectable, λέγοντας στην Angular ότι είναι μια υπηρεσία στην οποία μπορούμε να βασιστούμε.

Το provideIn με τιμή 'root', μετατρέπει το RecipeService σε μια υπηρεσία singleton, το οποίο σημαίνει ότι υπάρχει μόνο μία παρουσία της σε όλη την εφαρμογή. Η υπηρεσία εισάγει το HttpClient για να πραγματοποιήσει κλήσεις σε ένα API για εύρεση συνταγής. Τέλος υπάρχει η μέθοδος getRecipes που ανακτά μια σειρά συνταγών από το API.

Το επόμενο βήμα είναι η δημιουργία του RecipeComponent που εμφανίζει πληροφορίες συνταγής. Αυτό το στοιχείο βασίζεται στην υπηρεσία RecipeService για τη λήψη δεδομένων συνταγών:

```
import { Component, OnInit } from '@angular/core';
import { RecipeService }  from './recipe.service'; // Import the service

@Component({
  selector: 'app-recipe-list',
  templateUrl: './recipe-list.component.html',
  styleUrls: ['./recipe-list.component.css']
})
export class RecipeListComponent implements OnInit { // We'll get to OnInit later

  recipes: Recipe[] = [];

  constructor(private recipeService: RecipeService) { } // Inject in constructor

  ngOnInit() {
    this.recipeService.getRecipes() // Call the service method
```

```
    .then(recipes => this.recipes = recipes); // Store fetched recipes
  }
}
```

Αρχικά εισάγουμε το RecipeService επειδή το Component χρειάζεται τη βοήθειά του. Στην συνέχεια στον κατασκευαστή του Component, χρησιμοποιούμε dependency injection για να πούμε στην Angular ότι το RecipeListComponent χρειάζεται μια υπηρεσία RecipeService. Με τη χρήση του «private recipeService: RecipeService», εισάγουμε την παρουσία υπηρεσίας στο Component.

Η ανάκτηση δεδομένων θα γίνει στο ngOnInit καλώντας τη μέθοδο getRecipes από το injected recipeService. Μόλις ληφθούν οι συνταγές, τις αποθηκεύουμε στη σειρά συνταγών του Component (AngularMinds, 2024).

### 3.3.7 Router

Σε μια εφαρμογή μιας σελίδας (single-page application), αλλάζουμε αυτό που βλέπει ο χρήστης εμφανίζοντας ή αποκρύπτοντας τμήματα της οθόνης που αντιστοιχούν σε συγκεκριμένα στοιχεία, χωρίς την ανάγκη της επικοινωνίας με το διακομιστή για να λάβουμε μια νέα σελίδα. Κατά τη διάρκεια πλοήγησής τους οι χρήστες, μετακινούνται μεταξύ των διαφορετικών προβολών που έχουμε ορίσει.

Για να χειριστούμε την πλοήγηση από τη μια προβολή στην άλλη, θα χρησιμοποιήσουμε το Angular Router. Ο δρομολογητής ενεργοποιεί την πλοήγηση ερμηνεύοντας μια διεύθυνση URL του προγράμματος περιήγησης ως οδηγία για την αλλαγή της προβολής.

Για να χρησιμοποιήσουμε σωστά το Router, πρέπει να προσθέσουμε ένα στοιχείο «βάσης» στο αρχείο «index.html». Πρέπει επίσης να εισαγάγουμε το Router Module. Στο αρχείο μας "app.module.ts", μπορούμε να κάνουμε το εξής:

```
import { RouterModule, Routes } from '@angular/router';

const appRoutes: Routes = [
```

```
  { path: 'characters',         component: CharactersComponent },
  { path: 'character/:id',      component: CharacterDetailComponent },
  { path: '', redirectTo: '/characters', pathMatch: 'full' },
  { path: '**',                 component: PageNotFoundComponent }
];
@NgModule({
  imports: [
    RouterModule.forRoot(appRoutes)
  ]
})
export class AppModule { }
```

Όπως μπορούμε να δούμε, ορίζουμε τις διαδρομές πλοήγησής μας στον πίνακα "appRoutes" και μεταβιβάζουμε αυτόν τον πίνακα στο "RouterModule". Μπορούμε τώρα να χρησιμοποιήσουμε την Οδηγία "RouterOutlet", που μαρκάρει το σημείο όπου ο δρομολογητής εμφανίζει μια προβολή, για να δημιουργήσουμε κάποιο είδος μενού πλοήγησης (Bors, 2018):

```
<nav>
  <a routerLink="/characters" routerLinkActive="active">Characters</a>
</nav>
<router-outlet></router-outlet>
```

### 3.3.8 State management

Η διαχείριση κατάστασης (State Management) στην Angular περιλαμβάνει τη διαχείριση των καταστάσεων δεδομένων και διεπαφής χρήστη (UI) εντός της εφαρμογής. Η διαχείριση αυτή είναι ιδιαίτερα σημαντική σε μεγάλες και πολύπλοκες εφαρμογές όπου τα Components πρέπει να μοιράζονται και να συγχρονίζουν δεδομένα. Υπάρχουν διάφορες τεχνικές διαχείρισης κατάστασης, θα παρουσιάσουμε όμως τις τρεις πιο δημοφιλείς προσεγγίσεις:

- Component State
- Services and RxJS
- NgRx Store

Το Component State είναι η απλούστερη μορφή διαχείρισης κατάστασης στην Angular. Περιλαμβάνει την αποθήκευση δεδομένων σε μεμονωμένα στοιχεία. Κάθε Component έχει τη δική του κατάσταση και τα δεδομένα μεταβιβάζονται μέσω της ιεραρχίας των Component χρησιμοποιώντας δεσμεύσεις εισόδου και εξόδου. Αυτή η προσέγγιση είναι κατάλληλη για μικρές έως μέτρια πολύπλοκες εφαρμογές όπου τα δεδομένα δεν χρειάζεται να μοιράζονται εκτενώς. Ακολουθεί ένα απλό παράδειγμα Component State:

```
// app.component.ts

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


@Component({

  selector: 'app-root',

  template: `

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

    <app-child [message]="message" (updateMessage)="updateMessage($event)"></app-child>  `

})
```

```
export class AppComponent {
  title = 'Component State Example';
  message = 'Hello from AppComponent!';
  updateMessage(newMessage: string) {
    this.message = newMessage;
  }
}
// app-child.component.ts
import { Component, Input, Output, EventEmitter } from '@angular/core';

@Component({
  selector: 'app-child',
  template: `
    <div>{{ message }}</div>
    <button (click)="changeMessage()">Change Message</button>   `
})
export class AppChildComponent {
  @Input() message: string;
  @Output() updateMessage = new EventEmitter<string>();

  changeMessage() {
    const newMessage = 'Updated message from AppChildComponent!';
    this.updateMessage.emit(newMessage);
  }
}
```

Σε αυτό το παράδειγμα, το AppComponent έχει τη δική του κατάσταση (title και message) και μεταβιβάζει το μήνυμα στο AppChildComponent. Όταν συμβεί το συμβάν κλικ στο κουμπί στο θυγατρικό στοιχείο, τότε θα εκπέμψει ένα συμβάν που ενημερώνει το μήνυμα στο γονικό στοιχείο.

Το Component State είναι μια απλή προσέγγιση, αλλά έχει περιορισμούς όσον αφορά την κοινή χρήση δεδομένων ή τη διαχείριση ασύγχρονων λειτουργιών.

Τα Services και το RxJS (Reactive Extensions for JavaScript) χρησιμοποιούνται συχνά μαζί για τη διαχείριση κατάστασης σε εφαρμογές Angular. Οι υπηρεσίες λειτουργούν ως κεντρικά σημεία για δεδομένα εφαρμογών και το RxJS παρέχει έναν ισχυρό τρόπο χειρισμού ασύγχρονων λειτουργιών και ροών δεδομένων. Ακολουθεί ένα παράδειγμα χρήσης υπηρεσιών και RxJS για διαχείριση κατάστασης:

```
// data.service.ts
import { Injectable } from '@angular/core';
import { BehaviorSubject } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class DataService {
  private messageSubject = new BehaviorSubject<string>('Hello from DataService');
  message$ = this.messageSubject.asObservable();
  updateMessage(newMessage: string) {
    this.messageSubject.next(newMessage);
  }
}
// app.component.ts
```

```
import { Component } from '@angular/core';
import { DataService } from './data.service';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{ title }}</h1>
    <div>{{ message$ | async }}</div>
    <button (click)="changeMessage()">Change Message</button>  `
})
export class AppComponent {
  title = 'Services and RxJS Example';
  constructor(private dataService: DataService) {}
  changeMessage() {
    const newMessage = 'Updated message from AppComponent!';
    this.dataService.updateMessage(newMessage);
  }
}
// app-child.component.ts
import { Component } from '@angular/core';
import { DataService } from './data.service';

@Component({
  selector: 'app-child',
  template: `
```

```
    <div>{{ message$ | async }}</div> `
})
export class AppChildComponent {
  constructor(private dataService: DataService) {}
}
```

Σε αυτό το παράδειγμα, έχουμε ένα DataService που χρησιμοποιεί ένα BehaviorSubject για να αποθηκεύει και να εκπέμπει το μήνυμα. Τόσο το AppComponent όσο και το AppChildComponent εγγράφονται για να λαμβάνουν ενημερώσεις όταν αλλάζει το μήνυμα. Αυτή η προσέγγιση επιτρέπει την αποτελεσματική κοινή χρήση δεδομένων και συγχρονισμό μεταξύ των στοιχείων.

Οι υπηρεσίες και το RxJS είναι ένας ευέλικτος και ισχυρός τρόπος διαχείρισης κατάστασης στο Angular, ειδικά για εφαρμογές μεσαίας έως μεγάλης κλίμακας με πολύπλοκες ροές δεδομένων.

Η Τρίτη προσέγγιση που θα παρουσιάσουμε είναι το NgRx, μια state management βιβλιοθήκη, εμπνευσμένη από το Redux για εφαρμογές Angular. Εισάγει μια μονοκατευθυντική ροή δεδομένων και έναν κεντρικό χώρο αποθήκευσης για τη διαχείριση της κατάστασης της εφαρμογής. Το NgRx Store είναι το πλέον κατάλληλο για μεγάλες και πολύπλοκες εφαρμογές όπου η διαχείριση κατάστασης γίνεται κρίσιμη. Ας δούμε ένα παράδειγμα χρήσης του NgRx Store για διαχείριση κατάστασης:

```
// app.state.ts
import { createAction, createReducer, on, createSelector } from '@ngrx/store';

export interface AppState {
  message: string;
}
```

```
export const initialAppState: AppState = {
  message: 'Hello from NgRx Store'
};
export const updateMessage = createAction(
  '[App] Update Message',
  (newMessage: string) => ({ newMessage })
);

export const appReducer = createReducer(
  initialAppState,
  on(updateMessage, (state, { newMessage }) => ({ ...state, message: newMessage }))
);

// Selector
export const selectMessage = (state: AppState) => state.message;
export const getMessage = createSelector(selectMessage, (message) => message);
// app.component.ts
import { Component } from '@angular/core';
import { Store } from '@ngrx/store';
import { updateMessage, getMessage } from './app.state';

@Component({
  selector: 'app-root',
  template: `
    <h1>{{ title }}</h1>
```

```
    <div>{{ message$ | async }}</div>
    <button (click)="changeMessage()">Change Message</button>  `
})
export class AppComponent {
  title = 'NgRx Store Example';
  message$ = this.store.select(getMessage);
  constructor(private store: Store) {}
  changeMessage() {
    const newMessage = 'Updated message from AppComponent!';
    this.store.dispatch(updateMessage({ newMessage }));
  }
}
// app-child.component.ts
import { Component } from '@angular/core';
import { Store } from '@ngrx/store';
import { getMessage } from './app.state';

@Component({
  selector: 'app-child',
  template: `
    <div>{{ message$ | async }}</div>   `
})
export class AppChildComponent {
  message$ = this.store.select(getMessage);
  constructor(private store: Store) {}
```

```
}
```

Σε αυτό το παράδειγμα, ορίζουμε μια διεπαφή AppState, ενέργειες, reducers και selectors για τη διαχείριση της κατάστασης. Τόσο το AppComponent όσο και το AppChildComponent χρησιμοποιούν το NgRx Store για να επιλέξουν και να ενημερώσουν το μήνυμα. Το NgRx Store παρέχει έναν δομημένο και επεκτάσιμο τρόπο χειρισμού κατάστασης σε μεγάλες εφαρμογές Angular (chintanonweb, 2023).

### 3.3.9 HTTP Client

Οι περισσότερες εφαρμογές front-end πρέπει να επικοινωνούν με έναν διακομιστή μέσω του πρωτοκόλλου HTTP, για λήψη ή αποστολή δεδομένων και πρόσβαση σε άλλες υπηρεσίες υποστήριξης. Η Angular παρέχει ένα API το HTTP client για εφαρμογές Angular, την κλάση υπηρεσιών HttpClient στο @angular/common/http.

Για να μπορέσουμε να χρησιμοποιήσουμε το HttpClient, πρέπει να το προσθέσουμε στο root dependency injector της εφαρμογής. Οι περισσότερες εφαρμογές το κάνουν στον πίνακα Provider του ApplicationConfig στο app.config.ts

```
import { ApplicationConfig } from '@angular/core';
import { HttpClientModule } from '@angular/common/http';
import { importProvidersFrom } from '@angular/core';

export const appConfig: ApplicationConfig = {
  providers: [
    importProvidersFrom(HttpClientModule),
  ]
};
```

Για να ανακτήσουμε δεδομένα από έναν διακομιστή, θα χρησιμοποιήσουμε τη μέθοδο HttpClient.get(). Αυτή η ασύγχρονη μέθοδος στέλνει ένα αίτημα HTTP και επιστρέφει ένα Observable που εκπέμπει τα ζητούμενα δεδομένα όταν λαμβάνεται η απάντηση. Η μέθοδος get(url, options) παίρνει σαν είσοδο δύο ορίσματα, τη διεύθυνση URL του σημείου από την οποία γίνεται ανάκτηση και ένα αντικείμενο προαιρετικών επιλογών για τη διαμόρφωση του αιτήματος (Angular, 2024).

## 3.4 Δημιουργία μιας Angular εφαρμογής

Για την δημιουργία μιας εφαρμογής σε Angular θα πρέπει να ξεκινήσουμε εγκαθιστώντας το Angular CLI με την ακόλουθη εντολή:

```
npm install - g @angular/cli
```

Στη συνέχεια θα μετακινηθούμε στον κατάλογο που θέλουμε να δημιουργήσουμε τη νέα εφαρμογή μας και θα εισάγουμε:

```
ng new angular-tour-of-heroes
```

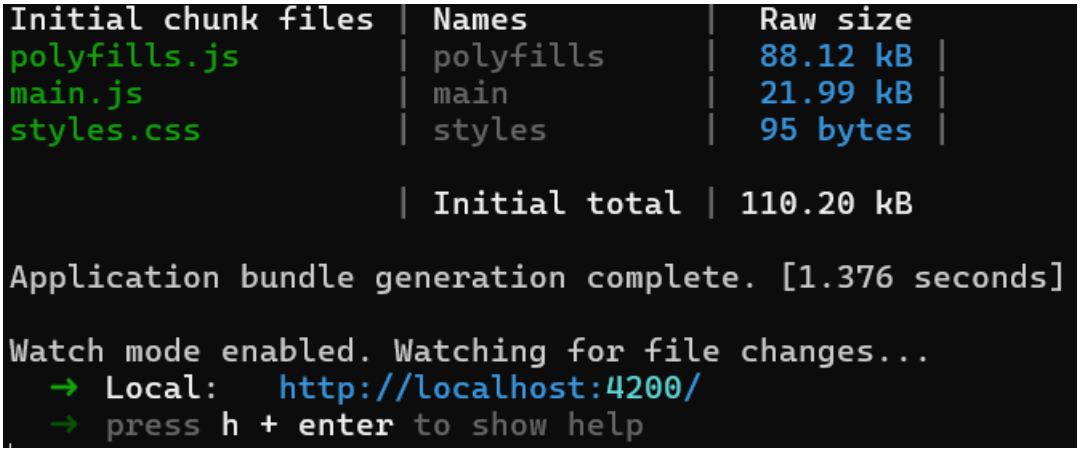
Το ng new ζητά πληροφορίες σχετικά με τις δυνατότητες που πρέπει να συμπεριληφθούν στο έργο. Μπορούμε να αποδεχτούμε τις προεπιλογές πατώντας το πλήκτρο Enter. Το ng new εγκαθιστά τα απαραίτητα πακέτα npm και άλλες εξαρτήσεις που απαιτεί η Angular. Η διαδικασία αυτή διαρκεί κάποιο χρόνο και αφού ολοκληρωθεί θα έχει δημιουργηθεί ένας κατάλογος με όνομα angular-tour-of-heroes.

Μέσα στον κατάλογο αυτό (και συγκεκριμένα μέσα στον υποκατάλογο src/app) θα υπάρχει το αρχικό έργο καθώς και σχετικά αρχεία διαμόρφωσης.

Αν μεταβούμε στον κατάλογο του έργου που δημιουργήθηκε και δώσουμε:

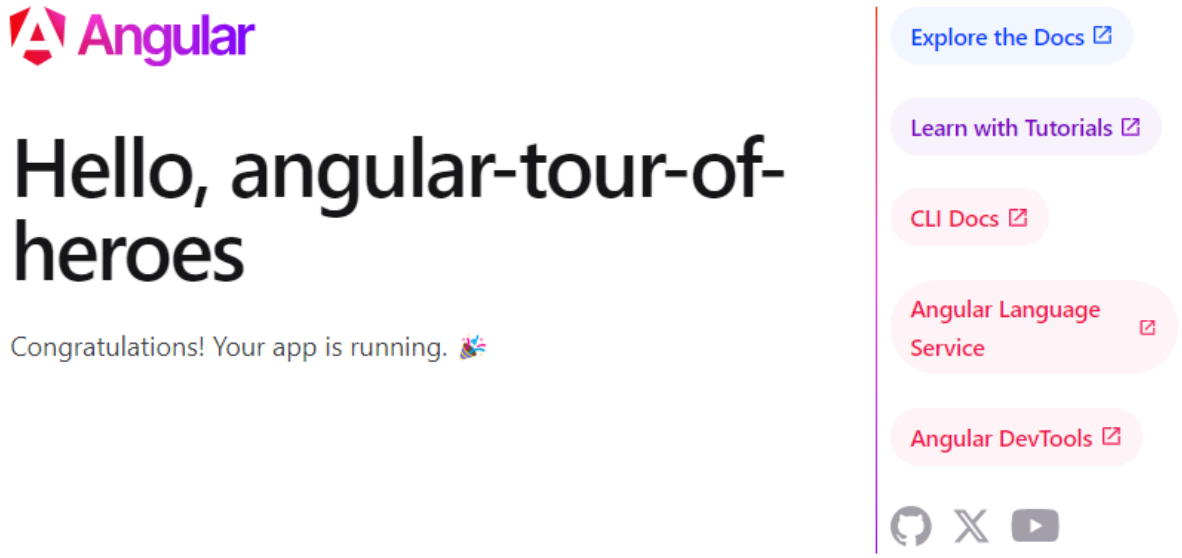
```
ng serve
```

θα ξεκινήσει η διαδικασία του build και θα εμφανιστεί η ακόλουθη εικόνα, στην οποία ενημερωνόμαστε ότι έχει ολοκληρωθεί το build και πλέον μπορούμε να δούμε το έργο μας στη διεύθυνση localhost:4200.




**Εικόνα 26: Αποτέλεσμα ng serve**

Αν πληκτρολογήσουμε τη διεύθυνση αυτή σε κάποιον φυλλομετρητή θα δούμε την ακόλουθη εικόνα:




**Εικόνα 27: Αποτέλεσμα της πρώτης εφαρμογής**

### 3.5 Δομή μιας Angular εφαρμογής

Όταν εισέλθουμε στον κατάλογο της εφαρμογής θα παρατηρήσουμε το αρχείο index.html που είναι το πρώτο που φορτώνει η Angular. Ας δούμε πώς είναι το αρχείο αυτό:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>AngularTourOfHeroes</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
```

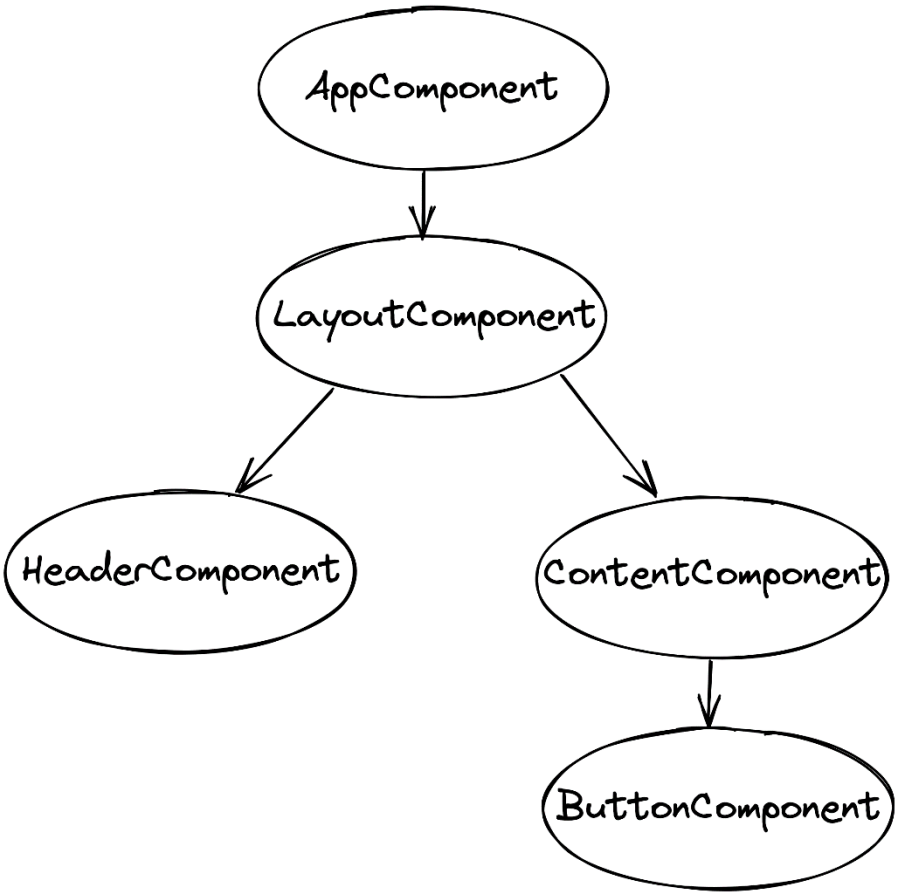
```
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Όπως παρατηρούμε είναι ένα έγγραφο HTML αρκετά τυπικής εμφάνισης. Το μόνο περίεργο είναι η ετικέτα <app-root> την οποία δεν έχουμε συναντήσει. Η συγκεκριμένη ετικέτα ορίζει το βασικό στοιχείο της εφαρμογής το οποίο υπάρχει στο src/app/app.component.ts

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


@Component({
  selector: 'app-root',
  standalone: true,
  imports: [RouterOutlet],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'angular-tour-of-heroes';
}
```

Αν προσέξουμε στην ιδιότητα selector θα δούμε το όνομα της ετικέτας που χρησιμοποιήσαμε στο index.html. Το στοιχείο αυτό καθορίζεται από την ιδιότητα templateUrl όπου υπάρχει ο html κώδικας που θα αντικαταστήσει την ετικέτα. Ο κύριος ρόλος του ριζικού μας στοιχείου (AppComponent) είναι να χρησιμεύσει ως δοχείο για την εφαρμογή μας. Σε γενικές γραμμές, σχεδόν όλες οι εφαρμογές Angular έχουν ένα ενιαίο στοιχείο ρίζας και μέσα σε αυτό το ριζικό στοιχείο θα υπάρχουν περισσότερα στοιχεία, τα οποία μπορεί επίσης να έχουν περισσότερα συστατικά, συνθέτοντας ένα δέντρο ένθετων στοιχείων:




**Εικόνα 28: Δέντρο ένθετων στοιχείων (AngularStart, 2024)**

Κάθε ένα Component μιας Angular εφαρμογής έχει μια δομή παρόμοια με το ακόλουθο:

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

```
@Component({
  selector: "app-item",
  standalone: true,
  // the following metadata specifies the location of the other parts of the component
  templateUrl: "./item.component.html",
  styleUrls: ["./item.component.css"],
})
export class ItemComponent {
  // code goes here
}
```

Το συγκεκριμένο Component ονομάζεται ItemComponent και ο selector του είναι το app-item. Η λογική του selector είναι παρόμοια όπως των κανονικών ετικετών HTML. Αν τοποθετήσουμε τον selector <app-item></app-item> σε ένα template τότε το πρόγραμμα περιήγησης θα κάνει render το πρότυπο αυτού του στοιχείου κάθε φορά που συναντάται μια παρουσία του selector.

Το template καθορίζεται με την επιλογή templateUrl, ενώ το css που θέλουμε να χρησιμοποιηθεί θα καθοριστεί με το styleUrls.

# ΚΕΦΑΛΑΙΟ 4

# Σχεδιασμός και Ανάπτυξη εφαρμογής

## 4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγράψουμε τη διαδικασία για το σχεδιασμό και την ανάπτυξη της εφαρμογής. Θα ξεκινήσουμε με το σχεδιασμό όπου θα αναφέρουμε τις απαιτήσεις που θέλουμε να έχει η εφαρμογή που θα υλοποιήσουμε. Στη συνέχεια θα περιγράψουμε τα βήματα της υλοποίησης της εφαρμογής.

## 4.2 Καταγραφή απαιτήσεων

Η εφαρμογή που θα υλοποιήσουμε θα είναι μια εισαγωγική εφαρμογή στην Angular, με τη βοήθεια της οποίας θα ξεκινήσουμε να χρησιμοποιούμε τις βασικές δυνατότητές της. Η εφαρμογή μας θα εμφανίζει μια λίστα από βιβλία και ο χρήστης θα μπορεί να επιλέξει κάποιο βιβλίο από τη λίστα και να διαβάσει τις αναλυτικές πληροφορίες του. Επίσης ο χρήστης θα μπορεί να φιλτράρει τα βιβλία που εμφανίζονται με βάση τον τίτλο τους.

## 4.3 Δομή της εφαρμογής

Η εφαρμογή μας υλοποιήθηκε σε Angular έκδοσης 17. Για να εγκαταστήσουμε την συγκεκριμένη έκδοση της angular δώσαμε την εντολή

```
npm install -g @angular/cli@17
```

Για να δημιουργήσουμε την εφαρμογή μας θα γράψουμε:

```
ng new books-app
```

Αφού επιλέξουμε τις default τιμές στις ερωτήσεις, μετά από λίγη ώρα θα έχει εγκατασταθεί η εφαρμογή. Περισσότερα για τη δομή μπορούμε να δούμε στην ενότητα 3.5. Στο σημείο αυτό θα αναφερθούμε μόνο στη δημιουργία των κατάλληλων οντοτήτων για τη λειτουργία του προγράμματος.

Αρχικά θα δημιουργήσουμε ένα interface με όνομα book-info το οποίο θα περιέχει τη δομή που θέλουμε να έχουμε για ένα βιβλίο. Επιλέξαμε τα ακόλουθα πεδία για να χρησιμοποιήσουμε για τα βιβλία:

- id:string;
- title: string;
- authors: string[];
- publisher: string;
- description: string;
- pageCount: number;
- printType: string;
- categories: string[];
- smallThumbnail: string;
- thumbnail: string;

Τα βιβλία που θα χρησιμοποιήσουμε θα τα αντλήσουμε από ένα api που παρέχει η google (https://developers.google.com/books/docs/v1/using#WorkingVolumes). Με το api αυτό μπορούμε να υποβάλουμε διάφορες ερωτήσεις για να αντλήσουμε πληροφορίες για βιβλία και εμείς επιλέξαμε να φέρουμε τα βιβλία του συγγραφέα Jo Nesbo που είναι στα ελληνικά.

Από ολόκληρη τη λίστα με τα δεδομένα που μας επέστρεψε η google εμείς κρατήσαμε μόνο τα παραπάνω δεδομένα τα οποία αποθηκεύσαμε σε έναν πίνακα με όνομα bookInfoList ο οποίος είναι μέσα σε ένα άλλο service με όνομα BookactionService. Στο ίδιο service υπάρχουν και δύο συναρτήσεις:

- getAllBookInfoList: για την αναζήτηση και επιστροφή όλων των βιβλίων από τη λίστα.

- getBookInfoByTitle: η οποία δέχεται σαν παράμετρο το όνομα (τίτλο) ενός βιβλίου και επιστρέφει μια λίστα από βιβλία που έχουν τίτλο ίσο με τον τίτλο του βιβλίου.

Στη συνέχεια στο Component BookComponent εμφανίζουμε μια φόρμα όπου ο χρήστης θα μπορεί να πληκτρολογήσει τον τίτλο που θέλει να αναζητήσει στη λίστα των βιβλίων. Στη φόρμα υπάρχει button που όταν γίνει click θα κληθεί η συνάρτηση filterResults (η οποία είναι δηλωμένη στο BookComponent. Η συνάρτηση αυτή δέχεται σαν είσοδο το κείμενο που έχει πληκτρολογήσει ο χρήστης. Αν ο χρήστης δεν εισάγει κανένα χαρακτήρα τότε θα εμφανιστούν όλα τα βιβλία διαφορετικά θα φιλτράρουμε τα αποτελέσματα με βάση τον τίτλο του βιβλίου (αφού τον μετατρέψουμε σε πεζά).

Στη συνέχεια υπάρχει μια ngFor η οποία θα κινηθεί σε όλα τη λίστα που θα λάβει σαν είσοδο και θα εμφανίσει τα βιβλία. Η εμφάνιση των βιβλίων γίνεται με τη χρήση του Component BookDataComponent. Το συγκεκριμένο Component θα εμφανίσει

- την εικόνα του βιβλίου,
- το «Περισσότερα» που λειτουργεί σαν routerLink για τις λεπτομέρειες του βιβλίου
- τον τίτλο του βιβλίου
- τους συγγραφείς του βιβλίου με τη βοήθεια της ngFor

Αν ο χρήστης επιλέξει το «Περισσότερα» θα κληθεί το Component DetailsComponent το οποίο εμφανίζει αναλυτικά τις πληροφορίες του βιβλίου. Ο κατασκευαστής του συγκεκριμένου Component είναι αυτός που θα αρχικοποιήσει τις μεταβλητές του βιβλίου που θέλουμε να εμφανίσουμε με βάση αυτό που έχει επιλέξει ο χρήστης (καλώντας τη συνάρτηση getBookInfoByTitle).

# ΚΕΦΑΛΑΙΟ 5

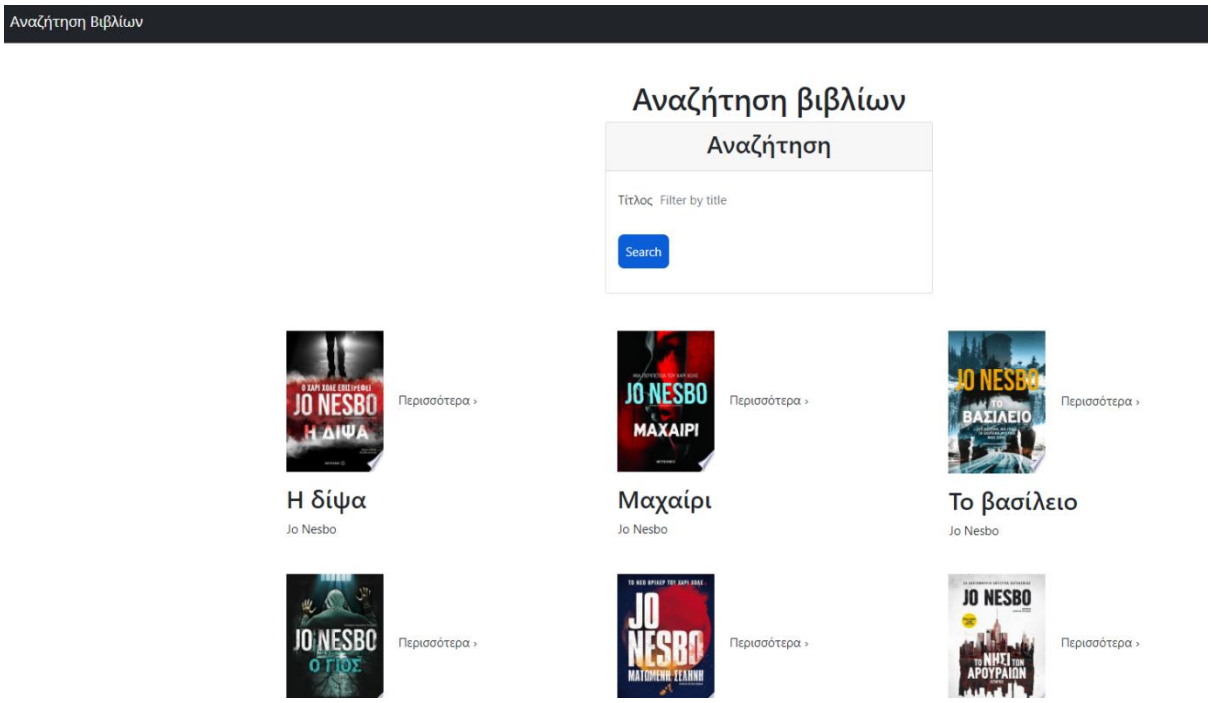
# Λειτουργία εφαρμογής

## 5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε τη λειτουργία της εφαρμογής δείχνοντας βήμα προς βήμα τις δυνατότητες που παρουσιάστηκαν κατά την καταγραφή απαιτήσεων.

## 5.2 Αναλυτική παρουσίαση του τρόπου λειτουργίας

Όταν ο χρήστης επισκεφθεί τη σελίδα http://localhost:4200/ (η συγκεκριμένη σελίδα ενεργοποιείται όταν δώσουμε ng serve από command line όταν είμαστε μέσα στο directory της εφαρμογής) θα δει:




**Εικόνα 29: Αρχική σελίδα εφαρμογής**

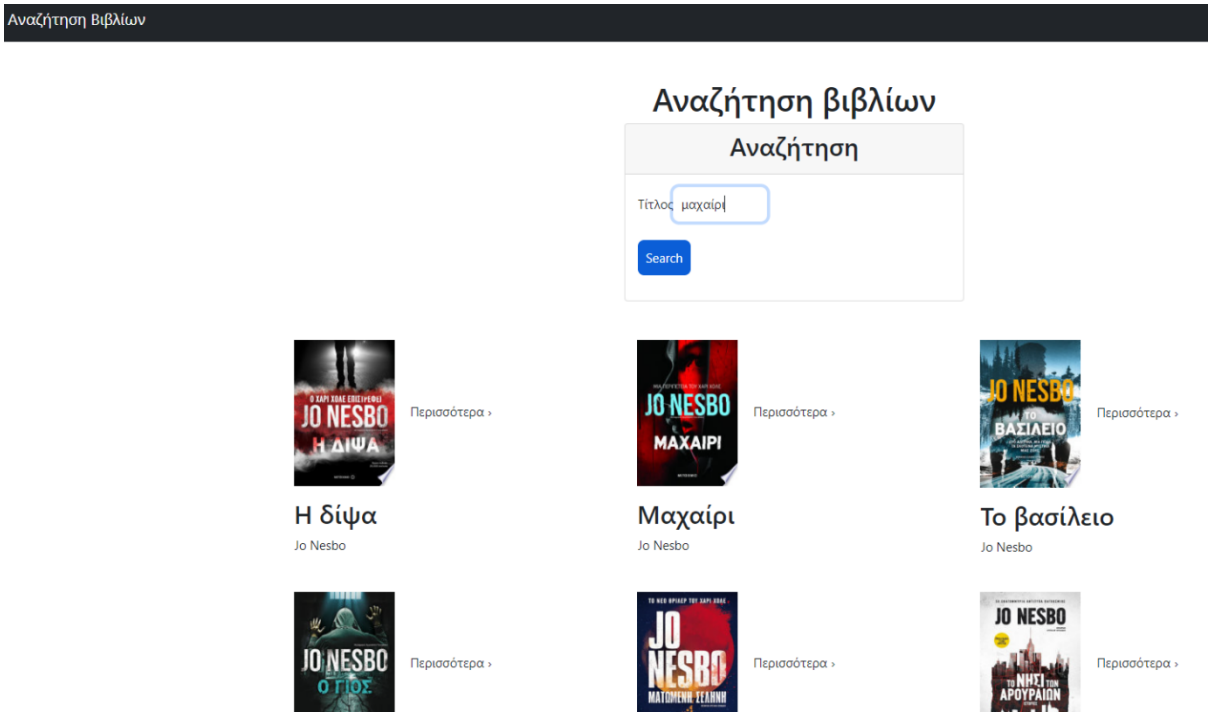
Ο χρήστης μπορεί να πατήσει το Περισσότερα (δίπλα από το βιβλίο που επιθυμεί) π.χ. από το βιβλίο Δίψα και η εφαρμογή θα εμφανίσει τη σελίδα του βιβλίου δείχνοντας αναλυτικά τις πληροφορίες του:




**Εικόνα 30: Εμφάνιση πληροφοριών βιβλίου**

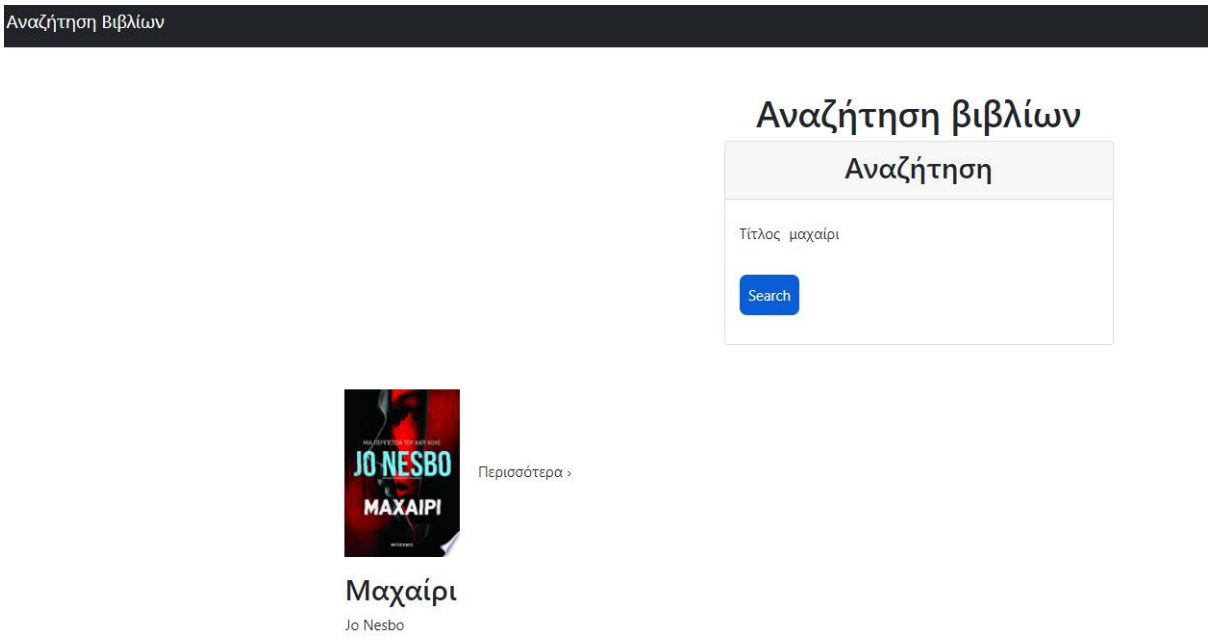
Για την επιστροφή στην αρχική σελίδα μπορούμε να επιλέξουμε το σύνδεσμο «Αναζήτηση Βιβλίων» στην πάνω αριστερή γωνία της εφαρμογής που λειτουργεί ως ανακατεύθυνση στην αρχική σελίδα.

Η επόμενη λειτουργία είναι όταν ο χρήστης στην αρχική σελίδα εισάγει το όνομα ενός βιβλίου και πατήσει το πλήκτρο Search (όπως στην εικόνα που ακολουθεί:




**Εικόνα 31: Εισαγωγή ονόματος προς αναζήτηση**

Το αποτέλεσμα που θα δει ο χρήστης είναι το βιβλίο που αναζήτησε. Αν θέλει μπορεί να επιλέξει το «Περισσότερα» για να δει τις αναλυτικές πληροφορίες του βιβλίου.




**Εικόνα 32: Αποτέλεσμα αναζήτησης με βάση τον τίτλο**

Αν όμως ο χρήστης δεν εισάγει κανένα κείμενο για αναζήτηση και πατήσει το Search τότε το σύστημα θα εμφανίσει την αρχική οθόνη του με όλα τα βιβλία.

# ΚΕΦΑΛΑΙΟ 6

## Συμπεράσματα

Ολοκληρώνοντας τη συγκεκριμένη πτυχιακή, πιστεύουμε ότι αποκτήσαμε αρκετές θεωρητικές γνώσεις σχετικά με τα JavaScript Framework. Μελετώντας την Angular και υλοποιώντας ένα απλό σχετικά παράδειγμα χρήσης της κατανοήσαμε καλύτερα τον τρόπο εμφάνισης των δεδομένων με τη χρήση της Angular. Επίσης κατανοήσαμε τη χρήση Service, Component και το Routing για την εμφάνιση των αποτελεσμάτων που θέλουμε σε μια Single Page Application.

Οι δυνατότητες που θα μπορούσαν να προστεθούν σε μια βελτίωση της εφαρμογής είναι:

- Χρήση δυναμικά ενός json για άντληση δεδομένων κατευθείαν από ένα api και να μην τα έχουμε μέσα στην εφαρμογή μας.
- Παροχή δυνατότητας στο χρήστη για κλήση του api δίνοντας το όνομα του συγγραφέα που θέλει να αναζητήσει.
- Προσθήκη δυνατότητας Εγγραφής και Σύνδεσης χρηστών έτσι ώστε να έχουν τις λειτουργίες:
    - Προσθήκη αγαπημένων βιβλίων
    - Καταγραφή σχολίων για βιβλία και προβολή τους
    - Δυνατότητα δημιουργίας καλαθιού αγορών.

# ΒΙΒΛΙΟΓΡΑΦΙΑ

Α. ΞΕΝΟΓΛΩΣΣΗ


Abaas A. 2023. Top 20 Features Of Angular [online] Διαθέσιμο από https://medium.com/@aqeelabbas3972/modular-development-angular-uses-a-modular-approach-to-development-which-helps-developers-create-7e9686a51eb4 [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Angular. 2024, Introduction to the Angular docs [online] Διαθέσιμο από https://v17.angular.io/docs [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

AngularMinds, 2024. Concepts of Dependency Injection in Angular [online] Διαθέσιμο από https://www.angularminds.com/blog/concepts-of-dependency-injection-in-angular [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

AngularStart, 2024. Structure of an Angular application [online] Διαθέσιμο από https://angularstart.com/modules/basic-angular-concepts/2/#structure-of-an-angular-application [Ημερομηνία πρόσβασης 30 Αυγούστου 2024

Bors M. L. 2018. An overview of Angular [online] Διαθέσιμο από https://medium.com/@mlbors/an-overview-of-angular-3ccd2950648e [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Chintanonweb, 2023. Angular State Management: A Comparison of the Different Options Available [online] Διαθέσιμο από https://hackernoon.com/angular-state-management-a-comparison-of-the-different-options-available [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Deshpande C. 2024. What Is Angular? A Guide to the Angular Framework, [online] Διαθέσιμο από https://www.simplilearn.com/tutorials/angular-tutorial/what-is-angular

Ganatan D. 2024. Understanding Modules with Angular 18 [online] Διαθέσιμο από https://www.ganatan.com/tutorials/modules-with-angular [Ημερομηνία πρόσβασης 30 Αυγούστου 2024] [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

GeeksForGeeks, 2024a. JavaScript Libraries and Frameworks [online] Διαθέσιμο από https://www.geeksforgeeks.org/javascript-libraries-and-frameworks/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

GeeksForGeeks 2024b. React Introduction [online] Διαθέσιμο από https://www.geeksforgeeks.org/reactjs-introduction/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

GSAP, 2024. GSAP [online]. Διαθέσιμο από https://gsap.com/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Hempenius K. and Osmani A., 2024. What's new in JavaScript Frameworks [online] Διαθέσιμο από https://developer.chrome.com/blog/frameworks-may-2024 [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Javatpoint. 2024, TypeScript Tutorial [online] Διαθέσιμο από https://www.javatpoint.com/typescript-tutorial [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Kiran R. 2018. Dependency Injection in Angular: Introduction and DI Techniques [online] Διαθέσιμο από https://www.dotnetcurry.com/angularjs/1446/dependency-injection-di-angular [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Kramer N. 2024. 10 Best UI Animation Libraries for Beginners 2024 [online] Διαθέσιμο από https://daily.dev/blog/10-best-ui-animation-libraries-for-beginners-2024 [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

Krishnakumar, 2023. Typescript Tutorial [online] Διαθέσιμο από https://medium.com/@krishsurya1249/typescript-tutorial-f8d45ce5766b [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
NgDevelop, 2024. Angular Architecture [online] Διαθέσιμο από https://www.ngdevelop.tech/angular/architecture/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Robie J. n.a. What is the Document Object Model? [online] Διαθέσιμο από https://www.w3.org/TR/WD-DOM/introduction.html [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Sakthi S. 2024. Angular Versions History – Evolution Of Angular Releases [online] Διαθέσιμο από https://www.calibraint.com/blog/the-evolution-of-angular-versions#Angular_17 [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Sarma S. 2023. Angular Features [online] Διαθέσιμο από https://www.educba.com/angular-features/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Scholarhat. 2024. What is Angular Services - Types of services in Angular with Examples [online] Διαθέσιμο από https://www.scholarhat.com/tutorial/angular/services-in-angular-example [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Sencha, 2024. Choosing the Best JavaScript Framework In 2024 [online] Διαθέσιμο από https://www.sencha.com/blog/how-to-choose-the-best-javascript-frameworks-in-2023/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]
Wappalyzer, 2024. JavaScript frameworks. [online] Διαθέσιμο από https://www.wappalyzer.com/technologies/javascript-frameworks/ [Ημερομηνία πρόσβασης 1 Σεπτεμβρίου 2024]
ZobiWebSolutions, 2020, Introduction to AngularJS [online] Διαθέσιμο από https://www.zobiwebsolutions.com/blog/introduction-to-angularjs/ [Ημερομηνία πρόσβασης 30 Αυγούστου 2024]

# ΠΑΡΑΡΤΗΜΑ Α

Index.html

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>BooksApp</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
EVSTQN3/azprG1Anm3QDgpJLIm9Nao0Yz1ztcQTwFspd3yD65VohhpuuCOmLASjC"
crossorigin="anonymous">
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.2/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
MrcW6ZMFYlzcLA8Nl+NtUVF0sA7MsXsP1UyJoMp4YLEuNSfAP+JcXn/tWtIaxVXM"
crossorigin="anonymous"></script>
</head>
<body>
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
       <a class="navbar-brand" href="/index.html">Αναζήτηση Βιβλίων</a>
       <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-
expanded="false" aria-label="Toggle navigation">
          <span class="navbar-toggler-icon"></span>
       </button>

    </div>
</nav>
<br /><br />
<div class="container">
       <div class="row">

          <h1 class="text-center"> Αναζήτηση βιβλίων </h1>

       <app-root></app-root>

    </div>
</div>


</body>
```

```
</html>
```

app.component.ts

```
import { Component } from '@angular/core';
import { BookComponent } from './book/book.component';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [BookComponent, RouterModule],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'books-app';
}
```

book-info.ts

```
export interface BookInfo {

  id:string;

  title: string;
    authors: string[];
    publisher: string;
    description: string;
    pageCount: number;
    printType: string;
    categories: string[];
    smallThumbnail: string;
    thumbnail: string;

}
```

bookaction.service.ts

```
import { Injectable } from '@angular/core';
import { BookInfo } from './book-info';
```

```
@Injectable({
  providedIn: 'root'
})
export class BookactionService {

  constructor() { }


  readonly baseUrl = 'https://angular.io/assets/images/tutorials/faa';

  getAllBookInfoList(): BookInfo[] {
    return this.bookInfoList;
  }

  getBookInfoByTitle(title: string): BookInfo | undefined {
    return this.bookInfoList.find(housingLocation => housingLocation.title === title);
  }



  bookInfoList : BookInfo[] = [
    {
      "id": "EROJDgAAQBAJ",
        "title": "Η δίψα",
        "authors": [
          "Jo Nesbo"
        ],
        "publisher": "Metaichmio Publications",
        "description": "Η δίψα αποτελεί τη συνέχεια του μπεστ σέλερ Η αστυνομία (του δέκατου βιβλίου στη σειρά Χάρι Χόλε), στο οποίο ο βετεράνος επιθεωρητής παλεύει να προστατέψει τους συναδέλφους του από έναν δολοφόνο που επιζητά εκδίκηση από την αστυνομία. Στη Δίψα ο Χάρι θα ξαναενταχθεί στις τάξεις της αστυνομίας του Όσλο για να εντοπίσει έναν σίριαλ κίλερ που σκοτώνει χρήστες του κοινωνικού δικτύου γνωριμιών Tinder. Ο τρόπος του και η μέθοδος του δολοφόνου θα οδηγήσουν τον Χάρι στο κυνήγι μιας νέμεσής του από το παρελθόν. Πλινκ: Ο ήχος του ταιριάσματος στο Tinder είναι ο τελευταίος που θα ακούσει η Ελίζε Χέρμανσεν πριν πέσει θύμα δολοφονίας μες στο διαμέρισμά της. Το νεκρό κορμί της θα βρεθεί από την αστυνομία σημαδεμένο και στραγγισμένο από το αίμα. Όταν μια ακόμα γυναίκα δολοφονείται, τα αντανακλαστικά του πρώην επιθεωρητή και νυν καθηγητή στην Αστυνομική Ακαδημία Χάρι Χόλε, θρυλικού διώκτη σίριαλ κίλερ, ενεργοποιούνται… Άλλωστε ένας επαναλαμβανόμενος εφιάλτης στοιχειώνει τον ύπνο του, φωνές κατακλύζουν κάθε τόσο το μυαλό του: υπάρχει ένας δολοφόνος που ο Χάρι δεν κατάφερε να συλλάβει. Μήπως ήρθε η ώρα να αναμετρηθούν ξανά;",
        "pageCount": 583,
        "printType": "BOOK",
```

```
        "categories": [
          "Fiction"
        ],
       "smallThumbnail":
"http://books.google.com/books/content?id=EROJDgAAQBAJ&printsec=frontcover&img=1
&zoom=5&edge=curl&source=gbs_api",
        "thumbnail":
"http://books.google.com/books/content?id=EROJDgAAQBAJ&printsec=frontcover&img=1
&zoom=1&edge=curl&source=gbs_api"
    },
    {
      "id": "ejKZDwAAQBAJ",
        "title": "Μαχαίρι",
        "authors": [
          "Jo Nesbo"
        ],
        "publisher": "Metaichmio Publications",
        "description": "Η δωδέκατη περιπέτεια του αγαπημένου ήρωα από τον ροκ σταρ της
αστυνομικής ή κατά κόσμον Jo Nesbo. Η πλοκή ξεκινάει εκεί που τελείωσε η Δίψα και
συνδετικός κρίκος της υπόθεσης είναι, τι άλλο, τα μαχαίρια, με χαρακτηριστικότερο το Κρις,
ένα παραδοσιακό μαχαίρι από την Ιάβα…",
        "pageCount": 589,
        "printType": "BOOK",
        "categories": [
          "Fiction"
        ],
          "smallThumbnail":
"http://books.google.com/books/content?id=ejKZDwAAQBAJ&printsec=frontcover&img=1
&zoom=5&edge=curl&source=gbs_api",
          "thumbnail":
"http://books.google.com/books/content?id=ejKZDwAAQBAJ&printsec=frontcover&img=1
&zoom=1&edge=curl&source=gbs_api"
      },
      {
        "id": "QMnyDwAAQBAJ",
          "title": "Το βασίλειο",
          "authors": [
            "Jo Nesbo"
          ],
          "publisher": "Metaichmio Publications",
          "description": "Το συγκλονιστικό νέο θρίλερ του ροκ σταρ της αστυνομικής
λογοτεχνίας που είναι το ίδιο εκρηκτικό και πολυεπίπεδο όσο τα μυθιστορήματα της σειράς
Χάρι Χόλε.",
          "pageCount": 527,
          "printType": "BOOK",
          "categories": [
            "Fiction"
```

```
        ],
          "smallThumbnail":
"http://books.google.com/books/content?id=QMnyDwAAQBAJ&printsec=frontcover&img=
1&zoom=5&edge=curl&source=gbs_api",
          "thumbnail":
"http://books.google.com/books/content?id=QMnyDwAAQBAJ&printsec=frontcover&img=
1&zoom=1&edge=curl&source=gbs_api"

    }
    , {
      "id": "VCqzBgAAQBAJ",
        "title": "Ο γιος",
        "authors": [
          "Jo Nesbo"
        ],
        "publisher": "Metaichmio Publications",
        "description": "Ο συγγραφέας της σειράς μπεστ σέλερ με τον Χάρι Χόλε γράφει αυτή
τη φορά ένα εξαιρετικά δυνατό μυθιστόρημα με σκηνικό έναν λαβύρινθο διαφθοράς στα
υψηλότερα κυβερνητικά κλιμάκια. Ο Σόνι είναι ένας παράξενα χαρισματικός και μακάριος
νεαρός άνδρας που έχει περάσει το μεγαλύτερο μέρος της ζωής του στη φυλακή (δώδεκα
χρόνια), οι συγκρατούμενοί του επιζητούν να βρεθούν για λίγο στο κελί μαζί του χάρη στην
απόκοσμη ικανότητά του να παρηγορεί και να κατευνάζει τις ψυχικές τους εντάσεις. Δεν
τους ενδιαφέρει φυσικά που ο Σόνι είναι ηρωινομανής ούτε από πού προμηθεύεται
ναρκωτικά ούτε ότι εκτίει την ποινή για εγκλήματα που δεν έχει διαπράξει ο ίδιος...",
        "pageCount": 480,
        "printType": "BOOK",
        "categories": [
          "Fiction"
        ],

          "smallThumbnail":
"http://books.google.com/books/content?id=VCqzBgAAQBAJ&printsec=frontcover&img=1
&zoom=5&edge=curl&source=gbs_api",
          "thumbnail":
"http://books.google.com/books/content?id=VCqzBgAAQBAJ&printsec=frontcover&img=1
&zoom=1&edge=curl&source=gbs_api"

    },
    {
      "id": "uCnCEAAAQBAJ",
        "title": "Ματωμένη σελήνη",
        "authors": [
          "Jo Nesbo"
        ],
        "publisher": "Metaichmio Publications",
        "description": "Το 13ο βιβλίο της σειράς με πρωταγωνιστή τον Χάρι Χόλε.",
```

```
        "pageCount": 522,
        "printType": "BOOK",
        "categories": [
          "Fiction"
        ],

          "smallThumbnail":
"http://books.google.com/books/content?id=uCnCEAAAQBAJ&printsec=frontcover&img=1
&zoom=5&edge=curl&source=gbs_api",
          "thumbnail":
"http://books.google.com/books/content?id=uCnCEAAAQBAJ&printsec=frontcover&img=1
&zoom=1&edge=curl&source=gbs_api"

      },
      {
        "id": "-Nx5EAAAQBAJ",
          "title": "Το νησί των αρουραίων - Ιστορίες",
          "authors": [
            "Jo Nesbo"
          ],
          "publisher": "Metaichmio Publications",
          "description": "Η δεύτερη συλλογή ιστοριών του Jo Nesbo μετά τον Άρχοντα της
ζήλιας. Πέντε ιστορίες που διαδραματίζονται στο εγγύς μέλλον.",

          "pageCount": 371,
          "printType": "BOOK",
          "categories": [
            "Fiction"
          ],

            "smallThumbnail": "http://books.google.com/books/content?id=-
Nx5EAAAQBAJ&printsec=frontcover&img=1&zoom=5&edge=curl&source=gbs_api",
            "thumbnail": "http://books.google.com/books/content?id=-
Nx5EAAAQBAJ&printsec=frontcover&img=1&zoom=1&edge=curl&source=gbs_api"

      },
      {
        "id": "KcRAEAAAQBAJ",
          "title": "Ο άρχοντας της ζήλιας - Ιστορίες",
          "authors": [
            "Jo Nesbo"
          ],
          "publisher": "Metaichmio Publications",
          "description": "Ιστορίες γεμάτες απληστία, αδίστακτους έρωτες και σκοτεινά
μυστικά. Το μοναδικό χάρισμα του Nesbo να δημιουργεί ατμόσφαιρα και να δίνει
απρόβλεπτη τροπή στις ιστορίες του καθιστά αυτή τη συλλογή πραγματική απόλαυση για
τους λάτρεις του αστυνομικού.",
```

```
          "pageCount": 248,
          "printType": "BOOK",
          "categories": [
            "Fiction"
          ],
            "smallThumbnail":
"http://books.google.com/books/content?id=KcRAEAAAQBAJ&printsec=frontcover&img=
1&zoom=5&edge=curl&source=gbs_api",
            "thumbnail":
"http://books.google.com/books/content?id=KcRAEAAAQBAJ&printsec=frontcover&img=
1&zoom=1&edge=curl&source=gbs_api"
        },

        {
          "id": "D85aDwAAQBAJ",

          "title": "Μάκβεθ",
          "description": "Σε μια ζοφερή αποβιομηχανοποιημένη πόλη που σπαράσσεται από τη
διαφθορά και το έγκλημα, μια απόπειρα σύλληψης εμπόρων ναρκωτικών μετατρέπεται σε
λουτρό αίματος. Την ευθύνη να ξεκαθαρίσει την υπόθεση αναλαμβάνει ο επιθεωρητής
Μάκβεθ και η ομάδα του – είναι ο καλύτερος μπάτσος της πόλης αλλά τον στοιχειώνει το
παρελθόν του. Για την επιτυχία του θα του δοθούν απλόχερα χρήματα, εξουσία, σεβασμός,
γιατί όμως νιώθει πως δεν θα πάρει ό,τι δικαιωματικά του ανήκει παρά μόνο αν σκοτώσει;
Ένα συγκλονιστικό θρίλερ από τον ροκ σταρ της σύγχρονης αστυνομικής λογοτεχνίας
βασισμένο στον σαιξπηρικό Μάκβεθ.",
          "authors": [
            "Jo Nesbo"
          ],
          "publisher": "Metaichmio Publications",

          "pageCount": 570,
          "printType": "BOOK",
          "categories": [
            "Fiction"
          ],
             "smallThumbnail":
"http://books.google.com/books/content?id=D85aDwAAQBAJ&printsec=frontcover&img=1
&zoom=5&edge=curl&source=gbs_api",
            "thumbnail":
"http://books.google.com/books/content?id=D85aDwAAQBAJ&printsec=frontcover&img=1
&zoom=1&edge=curl&source=gbs_api"
          }

  ];


}
```

routes.ts

```
import { Routes } from '@angular/router';
import { BookComponent } from './book/book.component';
import { DetailsComponent } from './details/details.component';

const routeConfig: Routes = [
    {
      path: '',
      component: BookComponent,
      title: 'Home page'
    },
    {
      path: 'details/:title',
      component: DetailsComponent,
      title: 'Book details'
    }
  ];

  export default routeConfig;
```

details.component.ts

```
import { Component, inject } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ActivatedRoute } from '@angular/router';
import { BookactionService } from '../bookaction.service';
import { BookInfo } from '../book-info';


@Component({
  selector: 'app-details',
  standalone: true,
  imports: [CommonModule],
  template: `
  <article>
    <img class="listing-photo" [src]="bookInfo?.thumbnail"
      alt="Exterior photo of {{bookInfo?.title}}"/>
    <section class="listing-description">
      <h2 class="listing-heading">{{bookInfo?.title}}</h2>
    </section>
    <section class="listing-description">
      <p>{{bookInfo?.description}}</p>
    </section>
    <section class="listing-features">
    <h3>Συγγραφείς</h3>
```

```
    <ol *ngFor="let p of authors">
       <li>{{p}}</li>
     </ol>
    </section>
    <section class="listing-features">
    <h3>Κατηγορία βιβλίου</h3>
    <ol *ngFor="let p of categories">
       <li>{{p}}</li>
     </ol>
    </section>
    <section class="listing-description">
     <p><b>Τύπος</b>: {{bookInfo?.printType}} με {{bookInfo?.pageCount}} σελίδες </p>
    </section>
    <section class="listing-description">
     <p><b>Εκδοτικός Οίκος</b>: {{bookInfo?.publisher}} </p>
    </section>
  </article>
  `,
  styleUrl: './details.component.css'
})
export class DetailsComponent {
  bookActionService = inject(BookactionService);
  bookInfo: BookInfo | undefined;
  authors : string[] = [];
  categories : string[] = [];
  route: ActivatedRoute = inject(ActivatedRoute);
  title = "";
  constructor() {
     this.title = this.route.snapshot.params['title'];
     this.bookInfo = this.bookActionService.getBookInfoByTitle(this.title);
     if (this.bookInfo?.authors != null )
       this.authors = this.bookInfo?.authors;
     if (this.bookInfo?.categories != null )
       this.categories = this.bookInfo?.categories;
  }
}
```

book-data.component.ts

```
import { Component, Input } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BookInfo } from '../book-info';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-book-data',
```

```
  standalone: true,
  imports: [CommonModule,
    RouterModule],
  templateUrl: './book-data.component.html',
  styleUrl: './book-data.component.css'
})
export class BookDataComponent {
  @Input() bookInfo!: BookInfo;


}
```

book-data.component.html

```
<section class="listing">
<!-- σύντομες πληροφορίες για ένα βιβλίο -->
    <p>
    <img [src]="bookInfo.thumbnail"  >
    <a [routerLink]="['/details', bookInfo.title]">Περισσότερα</a>
    </p>
    <h2>{{bookInfo.title}}</h2>
    <tr *ngFor="let p of bookInfo.authors">
      <td>
        {{p}}
        </td>
        </tr>


  </section>
```

book.component.ts

```
import { Component, inject  } from '@angular/core';
import { CommonModule } from '@angular/common';
import { BookInfo } from '../book-info';
import {BookDataComponent} from '../book-data/book-data.component';
import {BookactionService} from '../bookaction.service'

@Component({
  selector: 'app-book',
  standalone: true,
  imports: [CommonModule, BookDataComponent],
  templateUrl: './book.component.html',
  styleUrl: './book.component.css'
})
export class BookComponent {
  //readonly baseUrl = 'https://angular.io/assets/images/tutorials/faa';
```

```
  bookInfoList : BookInfo[] = [];
  filteredLBookInfoList: BookInfo[] = [];
  bookactionService : BookactionService = inject(BookactionService);

  constructor() {
    this.bookInfoList = this.bookactionService.getAllBookInfoList();
    this.filteredLBookInfoList = this.bookInfoList;
  }

  filterResults(text: string) {
    if (!text) {
      this.filteredLBookInfoList = this.bookInfoList;
      return;
    }
    this.filteredLBookInfoList = this.bookInfoList.filter(
      bookRec => bookRec?.title.toLowerCase().includes(text.toLowerCase())
    );
  }
}
```

book.component.html

```
<section>

  <div class="col-md-4 offset-md-4">
       <div class="card">
         <div class="card-header">
           <h2 class="text-center">Αναζήτηση</h2>
       </div>
       <div class="card-body">
         <form class="form-horizontal">
           <div class="form-group mb-4">
             <label class="control-label">Τίτλος</label>
           <input type="text" class="form-control"  placeholder="Filter by title" #filter>
         </div>
         <div class="form-group mb-3">
           <button class="btn btn-primary" type="button"
(click)="filterResults(filter.value)">Search</button>
         </div>
         </form>
         </div>
         </div>
         </div>


  </section>
  <section class="results">
```

```
  <!-- επανάληψη για εμφάνιση όλων των βιβλίων -->
  <app-book-data
       *ngFor="let bookInfo of filteredLBookInfoList" [bookInfo]="bookInfo">
  </app-book-data>



</section>
```