ΤΕΙ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε

Ανάπτυξη Διαδικτυακού περιβάλλοντος διδασκαλίας των Τεχνολογιών JavaScript και AngularJS

Πτυχιακή Εργασία

Μαραγκουδάκης Νίκος
Α.Μ. 2011040

Επιβλέπον καθηγητής
Μπάρδης Γεώργιος

Σπάρτη
2016

ΤΕΙ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε

Ανάπτυξη Διαδικτυακού περιβάλλοντος διδασκαλίας των Τεχνολογιών JavaScript και AngularJS

Πτυχιακή Εργασία

Μαραγκουδάκης Νίκος
Α.Μ. 2011040

Επιβλέπον καθηγητής
Μπάρδης Γεώργιος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την .....

(Υπογραφή)	(Υπογραφή)	(Υπογραφή)
.....	.....	.....

Σπάρτη
2016

ΛΟΓΟΚΛΟΠΗ

Με πλήρη επίγνωση των συνεπειών του νόμου περί πνευματικών δικαιωμάτων, δηλώνω ενυπογράφως ότι είμαι αποκλειστικός συγγραφέας της παρούσας Πτυχιακής Εργασίας, για την ολοκλήρωση της οποίας κάθε βοήθεια είναι πλήρως αναγνωρισμένη και αναφέρεται λεπτομερώς στην εργασία αυτή. Έχω αναφέρει πλήρως και με σαφείς αναφορές, όλες τις πηγές χρήσης δεδομένων, απόψεων, θέσεων και προτάσεων, ιδεών και λεκτικών αναφορών, είτε κατά κυριολεξία είτε βάση επιστημονικής παράφρασης. Αναλαμβάνω την προσωπική και ατομική ευθύνη ότι σε περίπτωση αποτυχίας στην υλοποίηση των ανωτέρω δηλωθέντων στοιχείων, είμαι υπόλογος έναντι λογοκλοπής, γεγονός που σημαίνει αποτυχία στην Πτυχιακή μου Εργασία και κατά συνέπεια αποτυχία απόκτησης του Τίτλου Σπουδών, πέραν των λοιπών συνεπειών του νόμου περί πνευματικών δικαιωμάτων. Δηλώνω, συνεπώς, ότι αυτή η Πτυχιακή Εργασία προετοιμάστηκε και ολοκληρώθηκε από εμένα προσωπικά και αποκλειστικά και ότι, αναλαμβάνω πλήρως όλες τις συνέπειες του νόμου στην περίπτωση κατά την οποία αποδειχθεί, διαχρονικά, ότι η εργασία αυτή ή τμήμα της δε μου ανήκει διότι είναι προϊόν λογοκλοπής άλλης πνευματικής ιδιοκτησίας.

Όνομα και Επώνυμο Συγγραφέα (Με Κεφαλαία):

……

Υπογραφή (Ολογράφως, χωρίς μονογραφή):

……

Ημερομηνία (Ημέρα – Μήνας – Έτος):

……

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της πτυχιακής μου, κ. Μπάρδη Γεώργιο που πίστεψε σε εμένα δίνοντας μου την ευκαιρία αυτή, καθώς και για την αμέριστη βοήθεια και καθοδήγηση που μου παρείχε, ανοίγοντας μου νέους ορίζοντες.

Επίσης, θα ήθελα να ευχαριστήσω τους γονείς μου για τη δύναμη και την στήριξη που μου έδωσαν όλα αυτά τα χρόνια πιστεύοντας σε εμένα, καθώς χωρίς αυτούς δεν θα είχα την δυνατότητα να φτάσω μέχρι εδώ. Τέλος, δεν θα μπορούσα να μην ευχαριστήσω όλους τους συγγενείς και φίλους για την πολύτιμη βοήθειά τους με οποιονδήποτε τρόπο.

ΠΕΡΙΛΗΨΗ

Η παρούσα πτυχιακή εργασία σχετίζεται με τις νέες τεχνολογίες του διαδικτύου. Ο σκοπός της είναι η δημιουργία μιας πλατφόρμας ηλεκτρονικής μάθησης (e-lerning) μέσω της οποίας θα παρουσιάζονται έτοιμα software labs τόσο για τη JavaScript όσο και το AngularJS, μέσω βίντεο διαλέξεων. Οι ενότητες που καλύπτονται για τη JavaScript είναι Program flow, functions, Data and Objects, DOM, Events and Event handling. Για το AngularJS οι ενότητες που καλύπτονται είναι model view whatever, services and Dependency injection, Data Binding and Directives και Custom Directives and Services. Ο τελικός στόχος είναι οι συγκεκριμένες τεχνολογίες να είναι διαθέσιμες σε όλους τους φοιτητές της σχολής τεχνολογικών εφαρμογών του τμήματος μηχανικών πληροφορικής του ΑΤΕΙ Πελοποννήσου, μέσο ενός μαθήματος που θα δημιουργηθεί στο Moodle με σκοπό την προετοιμασία των αποφοίτων της σχολής μας για την αγορά εργασίας.

Η πτυχιακή εργασία χωρίζεται σε δύο μέρη. Στο πρώτο μέρος γίνεται αναφορά στην ηλεκτρονική μάθηση, στις πλατφόρμες διαχείρισης μάθησης και στο Moodle το οποίο και χρησιμοποιήθηκε, καθώς και στη JavaScript και το AngularJS. Στο δεύτερο μέρος, γίνεται παρουσίαση του μαθήματος [1] “Learn and Understand JavaScript & AngularJS”, χρησιμοποιώντας της σημειώσεις που δημιούργησα για τον σκοπό αυτό ενώ τα παραδείγματα από το κάθε κεφάλαιο παρατίθενται στο παράρτημα.

Λέξεις-κλειδιά: Σύγχρονη Τηλεκπαίδευση, Ασύγχρονη Τηλεκπαίδευση, JavaScript, AngularJS, Moodle, SPA.

ABSTRACT

This thesis is related with new web technologies. The main goal is to create an e-learning platform which shows ready software labs for JavaScript and AngularJS via video lectures. The sections covered for JavaScript are Program Flow, Functions, Data and Objects, DOM, Events and Event handling. For AngularJS the covered sections are Model View Whatever, Services and Dependency Injection, Data Binding and Directives, and Custom Directives and Services. The ultimate goal is to made these technologies available to all students of School of Applied Technologies of Department of Computer Engineering of Technological Educational Institute of Peloponnese by a course to be created in Moodle platform to prepare graduates for the job market.

The thesis is split into two part. The first refers to e-learning and its platforms, in Moodle which used, as well as JavaScript and AngularJS. In second part, there is a presentation of the course [1] "Learn and Understand JavaScript & AngularJS", which created using the notes I created for this purpose with examples of each chapter are listed in the appendix.

Key Words: Synchronous e-Learning, Asynchronous e-Learning, JavaScript, AngularJS, Moodle, SPA.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΩΜΕΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΕΙΚΩΝΩΝ

ΕΥΡΕΤΗΡΙΟ ΠΙΝΑΚΩΝ

1

ΕΙΣΑΓΩΓΗ

1.1 JavaScript

1.1.1 Εισαγωγή

Η JavaScript [2] [3] είναι μια από τις ποιο δημοφιλής γλώσσες προγραμματισμού, και συγκεκριμένα είναι μια υψηλού επιπέδου δυναμική scripting γλώσσα που δημιουργήθηκε έχοντας στόχο να κάνει τις ιστοσελίδες ποιο διαδραστικές. Τα τελευταία χρόνια χρησιμοποιείτε από τα σύγχρονα λειτουργικά συστήματα όπως τα Windows 8.1 και 10, αλλά και από διάφορες εφαρμογές με στόχο την αύξηση της ταχύτητας αλλά και της συμβατότητας.

1.1.2 Ιστορική Αναδρομή

Δημιουργήθηκε από έναν εργαζόμενο της Netscape Communications Corporation, τον Brendan Eich το 1995, θέλοντας μια ελαφριά ερμηνευόμενη γλώσσα που θα μπορούσε να συμπληρώσει την Java γίνοντας ποιο ελκυστική σε ερασιτέχνες και αρχάριους προγραμματιστές. Ενώ αρχικά ονομάστηκε livescript, από την έκδοση 2.0B3 μετονομάστηκε σε JavaScript δίνοντας την αίσθηση ότι η JavaScript ήταν βασισμένη στη Java, παρόλο που κάτι τέτοιο δεν συνέβαινε, καθώς είναι τελείως διαφορετικές γλώσσες. Το 1996 δίνεται στην ECMA International ώστε να δημιουργηθεί ένα πρότυπο, το οποίο ονομάζεται ECMAScript. Η τελευταία έκδοση είναι η 6 η οποία κυκλοφόρησε τον Ιούνιο του 2015.

1.1.3 Χαρακτηριστικά

Η JavaScript είναι ερμηνευόμενη scripting γλώσσα προγραμματισμού. Αυτό σημαίνει ότι κατά την εκτέλεση του κώδικα, υπάρχει ένας διερμηνέας ο οποίος εκτελεί τον κώδικα χωρίς να χρειάζεται να μεταγλωττιστεί πρώτα σε γλώσσα μηχανής.

Υποστηρίζει τον δομημένο προγραμματισμό, καθώς έχει όμοια σύνταξη με την c και τη c++. Μια διαφοροποίηση με αυτές είναι πως η JavaScript προσθέτει αυτόματα το ερωτηματικό στο τέλος δίνοντας την δυνατότητα στον προγραμματιστή να το παραλείπει.

Έχει χαλαρό σύστημα τύπων. Όπως στις περισσότερες scripting γλώσσες έτσι και στη JavaScript μια τιμή σχετίζεται με έναν τύπο. Με αυτόν τον τρόπο μια μεταβλητή μπορεί να είναι ένας αριθμός ενώ αργότερα να την κάνουμε αλφαριθμητικό. Επίσης δεν χρειάζεται ο προγραμματιστής να ορίσει τον τύπο μιας μεταβλητής όπως για παράδειγμα γίνεται στη c, καθώς αυτό γίνεται αυτόματα.

Είναι ανεξάρτητη από την πλατφόρμα. Εκτελείτε δηλαδή με τον ίδιο τρόπο, ανεξάρτητα από το λειτουργικό σύστημα που έχει ο υπολογιστής. Και αυτό γιατί η JavaScript στηρίζεται στο περιβάλλον όπου εκτελείτε. Ένα τέτοιο περιβάλλον εκτέλεσης μπορεί να είναι ένας Web browser.

Υποστηρίζει διάφορα χαρακτηριστικά όπως δομές επιλογής και επανάληψης, πίνακες και πίνακες συσχετίσεων, συναρτήσεις η οποίες λόγο της υποστήριξης του αντικειμενοστραφούς προγραμματισμού μπορούν να χρησιμοποιηθούν και ως μέθοδοι, χρησιμοποιεί πρωτότυπα καθώς και regular expressions.

Τέλος, η JavaScript παρόλο που αρχικά ήταν αποκλειστικά client based, δηλαδή ο κώδικας εκτελείτε στον υπολογιστή του χρήστη και όχι σε κάποιον διακομιστή, πλέον υπάρχουν διάφοροι τρόποι όπως με την χρήση του node.js να εκτελεστεί κώδικας JavaScript στη μεριά του Διακομιστή.

1.1.4 Πλεονεκτήματα

Τα βασικά πλεονεκτήματα που έχει η χρήση της JavaScript είναι ότι:

- Μειώνει την αλληλεπίδραση με τον διακομιστή, καθώς μπορούν κάποιοι από τους ελέγχους που γίνονται εκεί, να γίνονται μέσα από την JavaScript, μειώνοντας με αυτόν τον τρόπο την κίνηση και το φορτίο του του διακομιστή.
- Αυξάνει την διαδραστικότητα σε μια σελίδα, δημιουργώντας μενού πλοήγησης και διάφορα άλλα στοιχεία που μπορούν να αλληλοεπιδρούν με τον χρήστη.
- Δυνατότητα χρήσης ποιο εξειδικευμένων δυνατοτήτων όπως sliders προσθέτοντας μεγαλύτερη ευχρηστία.

1.1.5 Μειονεκτήματα

Μπορεί να είναι μια δυνατή γλώσσα, όμως υπάρχουν και μερικά μειονεκτήματα:

- Δεν επιτρέπει για λόγους ασφαλείας λειτουργείες που σχετίζονται με την επεξεργασία και το διάβασμα αρχείων.
- Δεν υποστηρίζει δυνατότητες για χρήση σε πολλά νήματα και πυρήνες καθώς και πολλούς επεξεργαστές.
- Δεν υποστηρίζει δυνατότητες δικτύωσης.

1.1.6 Χρήση

Η βασική χρήση της JavaScript είναι ως κομμάτι των ιστοσελίδων με σκοπό την αλληλεπίδραση με το Document Object Model (DOM) κάνοντας τες δυναμικές. Ένα τέτοιο παράδειγμα χρήσης είναι η επικύρωση μιας φόρμας για την έγκυρη εισαγωγή τιμών στα πεδία της, πριν γίνει η υποβολή των στοιχείων αυτών στον διακομιστή.

1.2 AngularJS

1.2.1 Εισαγωγή

Το AngularJS είναι μια ανοιχτού κώδικα JavaScript βιβλιοθήκη. Υποστηρίζεται από την Google και ο βασικός στόχος του είναι να διορθώσει το πρόβλημα της HTML ότι δεν είναι σχεδιασμένη για δυναμικές ιστοσελίδες, επεκτείνοντας την με νέα χαρακτηριστικά και δυνατότητες. Υποστηρίζει τόσο το μοντέλο MVC – Model View Controller όσο και το MVVM – Model View Viewmodel ενώ ενδείκνυται για την κατασκευή Single Page Application.

1.2.2 Γιατί AngularJS

Όπως είπαμε προηγούμενος, ο στόχος του AngularJS είναι να κάνει την html δυναμική. Αυτό αυτομάτως το κάνει να είναι μια αρκετά ελκυστική λύση για χρήση σε μεγάλα projects όπου χρειάζεται οι σελίδες να είναι δυναμικές αλλά ταυτόχρονα και γρήγορες. Ένα ακόμα πλεονέκτημα, είναι ότι μαζί με το AngularJS, μπορούν να χρησιμοποιηθούν και άλλες βιβλιοθήκες JavaScript όπως για παράδειγμα το jQuery, προσθέτοντας μεγαλύτερη ευελιξία αλλά και επιπρόσθετες λειτουργείες. Επίσης υποστηρίζει το μοντέλο MV* ή MVW (Model View Whatever) δίνοντας την δυνατότητα στον προγραμματιστή να επιλέξει εκείνος ποιο μοντέλο ανταποκρίνεται καλύτερα στις εκάστοτε ανάγκες, ενώ είναι κατάλληλο για την κατασκευή Single Page Application. Για να κάνει την HTML δυναμική, προσθέτει νέα στοιχεία και ιδιότητες σε αυτή, κάνοντας έτσι ευκολότερο τον χειρισμό του DOM. Τέλος, ως αποτέλεσμα όλων των παραπάνω, μπορεί να γίνει ποιο εύκολα η συντήρηση του κώδικα ολόκληρης τα σελίδας.

1.2.3 Χαρακτηριστικά

Μία από τις ποιο χρήσιμες δυνατότητες του AngularJS είναι το Two-way Data-Binding. Κάνοντας χρήση απλής JavaScript, όταν γίνεται μια αλλαγή στο μοντέλο, για να γίνει κάποια αλλαγή στο DOM, πρέπει να έχει μεριμνήσει ο προγραμματιστής ώστε να γίνεται σωστά. Ουσιαστικά λοιπόν, είναι η δυνατότητα συγχρονισμού μεταξύ του DOM και του μοντέλου.

Υποστηρίζει templates ή αλλιώς πρότυπα, τα οποία είναι το παραγόμενο αποτέλεσμα από τον controller και το μοντέλο. Τα template συνήθως είναι είτε ολόκληρα html αρχεία, είτε μεμονωμένα κομμάτια κώδικα html σε ξεχωριστά αρχεία html.

Υποστήριξη του μοντέλου MV* MVW. Το AngularJS υποστηρίζει ένα υβριδικό μοντέλο το οποίο δίνει την δυνατότητα να γίνει χρήση όποιο μοντέλο ανταποκρίνεται καλύτερα στις ανάγκες του εκάστοτε project. Στο Model View Controller υπάρχει το μοντέλο που είναι τα δεδομένα στην σελίδα, το view που είναι το HTML μαζί με το AngularJS και ο Controller που κάνει την σύνδεση ανάμεσα σε αυτά τα δυο. Από την άλλη στο Model View View-model, έχουμε το μοντέλο και το view, αλλά αντί για τον controller υπάρχει το viewmodel το οποίο είναι ένα αντικείμενο που παρέχει συγκεκριμένα δεδομένα και μεθόδους στα διάφορα views.

Ακόμα, ενδιαφέροντα χαρακτηριστικά είναι το Dependency Injection το οποίο έρχεται ενσωματωμένο στο AngularJS και δίνει την δυνατότητα να γίνεται ευκολότερη η δημιουργία, και ο έλεγχος των εφαρμογών, Deep Linking που σημαίνει ότι επιτρέπει την αποτύπωση μιας κατάστασης στο url, services, όπως το $http που υλοποιεί το XMLHttpRequest, και τέλος, το Routing που δίνει τη δυνατότητα για την εμφάνιση ενός view ανάλογα το url.

1.2.4 Πλεονεκτήματα

Υπάρχουν διάφορα πλεονεκτήματα έναντι άλλον JavaScript βιβλιοθηκών. Ας δούμε ποια είναι:

- Υποστηρίζεται από όλους τους σύγχρονους Web browserσ καθώς και από τα smartphones-tablets.
- Χρήση του Dependency injection.
- Δυνατότητα κατασκευής reusable components.
- Χρήση απλής HTML για το κομμάτι του view και απλής JavaScript στους controllers.
- Δυνατότητα κατασκευής εφαρμογής μιας σελίδας (single page application) με μεγάλη ευκολία.
- Δυνατότητα παροχής data binding, υποστηρίζοντας τόσο one-way όσο και two-way.
- Ο προγραμματιστής μπορεί να έχει περισσότερες δυνατότητες, γράφοντας ελάχιστο κώδικα.
- Ευκολία στη συντήρηση του κώδικα καθώς και στο debugging.

1.2.5 Μειονεκτήματα

Εκτός από τα παραπάνω πλεονεκτήματα, το AngularJS έχει και μερικά μειονεκτήματα:

Χρειάζεται ο χρήστης να μην έχει απενεργοποιήσει τη JavaScript από τον browser καθώς κάτι τέτοιο θα έχει ως αποτέλεσμα την εμφάνιση μιας απλής σελίδας χωρίς τα χαρακτηριστικά του AngularJS.

Δεν παρέχει περισσότερη ασφάλεια από όση παρέχει η χρήση απλής JavaScript. Αυτό έχει ως αποτέλεσμα να χρειάζεται ακόμα έλεγχος των δεδομένων και από τη μεριά του διακομιστή, για παράδειγμα αν χρειάζεται να γίνει ταυτοποίηση και αυθεντικοποίηση.

2

ΗΛΕΚΤΡΟΝΙΚΗ ΜΑΘΗΣΗ

2.1 Τηλεκπαίδευση

Η τηλεκπαίδευση ή αλλιώς εκπαίδευση από απόσταση είναι μια μορφή εκπαίδευσης όπου δεν παρευρίσκονται στον ίδιο χώρο εκπαιδευτής και εκπαιδευόμενος. Η μεταξύ τους επικοινωνία γίνεται μέσο μιας πλατφόρμας η οποία προσφέρει δυνατότητες είτε σύγχρονης είτε ασύγχρονης επικοινωνίας, ενώ το περιεχόμενο της διδασκαλίας μπορεί να παρέχεται τόσο σε φυσική μορφή, όσο και σε ψηφιακή μορφή παραδείγματος χάριν σημειώσεις σε ηλεκτρονική μορφή βίντεο παρουσιάσεις κλπ.

2.2 Μορφές Τηλεκπαίδευση

Η τηλεκπαίδευση ανάλογα με τον τρόπο που διεξάγεται, χωρίζεται κυρίως σε ασύγχρονη και σύγχρονη.

2.2.1 Ασύγχρονη Τηλεκπαίδευση

Στην διδασκαλία στην ασύγχρονη τηλεκπαίδευση, ο εκπαιδευόμενος είναι σε θέση να παρακολουθεί το εκπαιδευτικό πρόγραμμα με το δικό του ρυθμό, χωρίς να χρειάζεται να είναι στον ίδιο χώρο με τον εκπαιδευτή όπως επίσης και χωρίς να χρειάζεται να γίνεται σε πραγματικό χρόνο. Η επικοινωνία με τον εκπαιδευτή αλλά και τους υπόλοιπους εκπαιδευόμενους γίνεται ασύγχρονα μέσο ηλεκτρονικού ταχυδρομείου ή μέσω ομάδων συζήτησης. Τέλος το εκπαιδευτικό υλικό μπορεί να δοθεί σταδιακά στους εκπαιδευόμενους ενώ ενδείκνυται για άτομα που έχουν περιορισμένο χρόνο ή προβλήματα υγείας.

Πλεονεκτήματα

Μερικά από τα πλεονεκτήματα της ασύγχρονης τηλεκπαίδευσης είναι:

- Η ευελιξία στο χρόνο καθώς η μάθηση γίνεται σε μη πραγματικό χρόνο.
- Ο εκπαιδευόμενος έχει πρόσβαση στο εκπαιδευτικό υλικό οποιαδήποτε στιγμή το θελήσει.
- Υπάρχει αύξηση στον ρυθμό μεταφοράς της γνώσης.
- Δυνατότητας συμμετοχής από οποιοδήποτε σημείο του πλανήτη.
- Δυνατότητα συνεχούς εμπλούτισης του εκπαιδευτικού υλικού.

Μειονεκτήματα

Εκτός από τα πλεονεκτήματα, υπάρχουν και μερικά μειονεκτήματα:

- Υπάρχει μεγάλο κόστος για την αγορά και την συντήρηση του απαραίτητου εξοπλισμού.
- Απουσία της ζωντανής επικοινωνίας μεταξύ εκπαιδευτή και εκπαιδευόμενου.

2.2.2 Σύγχρονη Τηλεκπαίδευση

Στην σύγχρονη τηλεκπαίδευση, η διδασκαλία πραγματοποιείτε σε πραγματικό χρόνο χωρίς όμως να χρειάζεται να βρίσκονται στον ίδιο χώρο. Μπορεί να γίνει τόσο με τη μέθοδο της τηλεδιάσκεψης όσο και με την ανταλλαγή άμεσων μηνυμάτων, ενώ εκπαιδευτές και εκπαιδευόμενοι έχουν την δυνατότητα να αλληλοεπιδρούν μεταξύ τους.

Η μέθοδος της τηλεδιάσκεψης περιλαμβάνει μια εικονική αίθουσα διδασκαλίας όπου προσφέρει όλα τα πλεονεκτήματα της ζωντανής διδασκαλίας, μαζί με δυνατότητες όπως ανταλλαγή αρχείων, διαμοιρασμό επιφάνειας εργασίας καθώς και χρήση ηλεκτρονικού πίνακα, μικροφώνου και κάμερας μεταξύ εκπαιδευτών και εκπαιδευόμενων.

Στη μέθοδο με ανάλλαγή μηνυμάτων, γίνεται χρήση λογισμικού ανταλλαγής μηνυμάτων όπως για παράδειγμα το skype. Παρότι είναι ως μέθοδος μπορεί να γίνει αργή και

κουραστική, προσφέρει το πλεονέκτημα ότι ο εκπαιδευόμενος μπορεί να σκεφτεί πριν απαντήσει.

Πλεονεκτήματα

Τα πλεονεκτήματα που προσφέρει η σύγχρονη τηλεκπαίδευση είναι:

- Η δυνατότητα εγγραφής και επανάληψης του μαθήματος.
- Επικοινωνία σε πραγματικό χρόνο μεταξύ εκπαιδευτή και εκπαιδευόμενο.
- Χρήση ειδικών λογισμικών για την πραγματοποίηση εικονικών εργαστηρίων.
- Αλληλοεπίδραση μεταξύ εκπαιδευτή και εκπαιδευόμενο.

Μειονεκτήματα

Εκτός από τα πλεονεκτήματα, υπάρχουν και μερικά μειονεκτήματα:

- Αποξένωση των εκπαιδευόμενων.
- Χρειάζεται εκπαιδευτής και εκπαιδευόμενος να είναι εξοικειωμένοι με την τεχνολογία.
- Απαιτεί ισχυρό εξοπλισμό και υποστήριξη ακόμα και ειδικά εξοπλισμένες αίθουσες για μεγάλος πλήθος συμμετεχόντων.
- Απαιτεί πολύ καλή και γρήγορή διασύνδεση με το διαδίκτυο.

2.3 Συστήματα Διαχείρισης Μάθησης (LMS)

Με τον όρο Συστήματα διαχείρισης Μάθησης εννοούμε τις πλατφόρμες που έχουν ως στόχο τη διανομή αλλά και τη διαχείριση μαθημάτων. Μέσο μιας τέτοιας πλατφόρμας, δίνεται η δυνατότητα στον εκπαιδευτή να δημιουργεί μαθήματα ανεβάζοντας σημειώσεις, ηλεκτρονικά βιβλία, παρουσιάσεις ή ακόμα και αρχεία ήχου όπως φωνητικές σημειώσεις.

Χαρακτηριστικά

Τα κύρια χαρακτηριστικά ενός συστήματος διαχείρισης μάθησης είναι η διαδικτυακή ανάρτηση του εκπαιδευτικού υλικού, η δυνατότητα αξιολόγησης των εκπαιδευόμενων με διάφορους τρόπους και οι ομάδες συζήτησης όπου μπορούν να πραγματοποιηθούν συζητήσεις σχετικά με το μάθημα. Αναλυτικά, προσφέρει τις εξής δυνατότητες:

- Η εγγραφή μαθητών είναι αυτοματοποιημένη ενώ ελέγχεται η πρόσβαση που έχει στην πλατφόρμα και στα μαθήματα.
- Η δυνατότητα του εκπαιδευτή να δημιουργεί διαγωνίσματα, ερωτηματολόγια και εξετάσεις μέσα από την πλατφόρμα.
- Παροχή εργαλείων επικοινωνίας και παρακολούθησης για τη διαχείριση της τάξης.
- Εύκολη διαχείριση του εκπαιδευτικού υλικού, καθώς παρέχονται οδηγοί για την εισαγωγή και διαχείριση των μαθημάτων και του υλικού που τα συνοδεύει.
- Δυνατότητα παρακολούθησης της προόδου του κάθε εκπαιδευόμενου ξεχωριστά.
- Δυνατότητα παρακολούθησης των ενεργειών των χρηστών της πλατφόρμας, από την είσοδο μέχρι και την έξοδο τους από εκεί. Στα δεδομένα που καταγράφονται, έχουν πρόσβαση ο διαχειριστής της πλατφόρμας καθώς και οι εκπαιδευτές ενώ μερικά από αυτά τα δεδομένα είναι η συμμετοχή σε μαθήματα, η βαθμολογία εργασιών και διαγωνισμάτων και η συμμετοχή στις ομάδες συζήτησης.

2.4 Πλατφόρμες Συστημάτων Διαχείρισης Μάθησης

Υπάρχουν πλήθος από πλατφόρμες διαχείρισης μάθησης. Μερικές είναι εμπορικές και άλλες δωρεάν ανοικτού κώδικα. Λαμβάνοντας υπόψιν αυτό, ότι οι πλατφόρμες ανοικτού κώδικα εξελίσσονται ταχύτερα λόγο των περισσότερων ατόμων που ασχολούνται μαζί τους, αλλά και το κόστος συντήρησης της κάθε μίας από αυτές, για τη δημιουργία του μαθήματος στράφηκα στις μη εμπορικές.

2.4.1 Μη εμπορικές πλατφόρμες (δωρεάν)

Υπάρχουν διάφορες πλατφόρμες διαχείρισης μάθησης, οι οποίες είναι δωρεάν και ανοικτού κώδικα. Ας ρίξουμε μια ματιά σε μερικές από τις ποιο γνωστές.

Open Eclass

Η open eclass [4], είναι μια πλατφόρμα ολοκληρωμένης διαχείρισης μαθημάτων. Έχει σχεδιαστεί και αναπτυχθεί από το Ελληνικό Ακαδημαϊκό Διαδίκτυο, ενώ παρέχεται υπό την άδεια GNU GPLv2. Από το 2003 όπου και ξεκίνησε η διανομή της πρώτης έκδοσης μέχρι και σήμερα, χρησιμοποιείτε τόσο από τα Ελληνικά Ακαδημαϊκά Ιδρύματα όσο και από την πρωτοβάθμια και δευτεροβάθμια εκπαίδευση.

Moodle

Το moodle είναι ένα διαδικτυακό σύστημα διαχείρισης μαθημάτων. Δημιουργήθηκε αρχικά από τον Martin Dougiamas με την πρώτη έκδοση να κυκλοφορεί τον Αύγουστο του 2002. Είναι ανοικτού κώδικα ενώ από την τρίτη έκδοση παρέχεται υπό την άδεια GNU GPLv3.

2.4.2 Εμπορικές πλατφόρμες

Εκτός από τις δωρεάν πλατφόρμες, υπάρχουν διάφορες άλλες πλατφόρμες διαχείρισης μάθησης, οι οποίες παρέχονται είτε με κάποιο αντίτιμο είτε με τη παροχή κάποιας συνδρομής. Ας ρίξουμε μια ματιά σε μερικές από τις ποιο γνωστές.

Blackboard

Το Blackboard [5] είναι ένα ολοκληρωμένο σύστημα διαχείρισης μαθημάτων το οποίο έχει αναπτυχθεί για εκπαιδευτικά ιδρύματα. Είναι μια ευέλικτη πλατφόρμα με την οποία μπορούν εύκολα οι εκπαιδευτές να προσαρμόσουν ανάλογα το υλικό της μάθησης ή το μοντέλο διδασκαλίας που θέλουν να χρησιμοποιήσουν.

Docebo

Το Docebo [6] είναι ένα cloud based σύστημα διαχείρισης μαθημάτων. Έχει σχεδιαστεί να προσφέρεται ως υπηρεσία (Software as a Service) και παρέχει πλήθος από εργαλεία τα οποία ο κάθε χρήστης χρησιμοποιεί και πληρώνει μόνο όσα χρειάζεται.

2.5 Moodle

2.5.1 Τι είναι

Το Moodle [7] [8] [9] είναι ένα διαδικτυακό σύστημα διαχείρισης μαθημάτων (LMS) η οποία παρέχει σε εκπαιδευτές και εκπαιδευόμενους ένα ολοκληρωμένο, ισχυρό και ασφαλές σύστημα για την δημιουργία ενός προσωποποιημένου εκπαιδευτικού περιβάλλοντος. Το όνομά του αποτελείτε από τα αρχικά του Modular Object Developmental Learning Environment. Είναι ανοικτού κώδικα ενώ από την Τρίτη του έκδοση παρέχεται υπό την άδεια GNU GPLv3. Το περιβάλλον της πλατφόρμας αποτελείτε από διάφορα modules όπως ομάδες συζήτησης (forum), quiz κλπ ενώ το περιβάλλον είναι αντικειμενοστραφές με την έννοια ότι οι χρήστες ασκούν ενέργειες στα αντικείμενα του περιβάλλοντος κάνοντας το με αυτόν τον τρόπο ποιο φιλικό προς τον χρήστη. Ακόμα

παρέχεται η δυνατότητα της παρακολούθησης διαφόρων πληροφοριών την εκπαιδευόμενων όπως βαθμολογίες σε διαγωνίσματα και quiz καθώς και η πορεία τους ενώ είναι διαθέσιμο σε περισσότερες από 78 γλώσσες.

2.5.2 Ιστορική αναδρομή

Δημιουργήθηκε αρχικά από τον Martin Dougiamas το 1999 στο πλαίσιο της διδακτορικής του διατριβής, με την πρώτη έκδοση να κυκλοφορεί τον Αύγουστο του 2002 και την τελευταία μέχρι αυτή τη στιγμή να είναι η έκδοση 3.0.4 όπου κυκλοφόρησε τον Μάϊο του 2016. Ο σκοπός του ήταν να βοηθήσει τους εκπαιδευτικούς να δημιουργούν μαθήματα στο διαδίκτυο δίνοντας έμφαση στην αλληλεπίδραση και την ομαδική κατασκευή περιεχομένου, καθώς και τη συνεχή εξέλιξη του. Σήμερα, το moodle αναπτύσσετε από το moodle project το οποίο συντονίζεται από μια αυστραλιανή εταιρία, την Moodle HQ. Μέχρι αυτή τη στιγμή, έχει 73,703 ενεργές εγκαταστάσεις σε 229 χώρες, έχουν δημιουργηθεί 9,601,274 μαθήματα ενώ έχει 85,891,520 ενεργούς χρήστες και 251,681,237 εγγραφές.

2.5.3 Χαρακτηριστικά

Το Moodle χρησιμοποιείτε από τα μεγαλύτερα πανεπιστήμια ανάμεσα στα οποία είναι και το MIT. Στην Ελλάδα Χρησιμοποιείται από την πρωτοβάθμια μέχρι την τριτοβάθμια εκπαίδευση ενώ μεταξύ των ιδρυμάτων βρίσκονται το Εθνικό Μετσόβιο Πολυτεχνείο και το Ελληνικό Ανοικτό Πανεπιστήμιο.

Η πλατφόρμα διανέμεται δωρεάν κάτω από την άδεια GNU GPLv3 με αποτέλεσμα να μπορεί να χρησιμοποιηθεί ελεύθερα καθώς και να τροποποιηθεί ο κώδικάς του για την προσθήκη νέων χαρακτηριστικών.

Το Moodle επικεντρώνετε στην αποτελεσματική εκπαίδευση καθώς βασίζεται σε διάφορες παιδαγωγικές αρχές σε αντίθεση με άλλες εμπορικές πλατφόρμες.

Διαθέτει ενεργεί κοινότητα χρηστών που ασχολούνται τόσο με τη διόρθωση σφαλμάτων όσο και με την επίλυση προβλημάτων και αποριών.

Το εκπαιδευτικό υλικό μπορεί να οργανωθεί ανά εβδομάδα ή ανά θεματική ενότητα, ενώ οι εκπαιδευόμενοι μπορούν να υποβάλουν εργασίες ηλεκτρονικά μέσα σε καθορισμένο χρονικό πλαίσιο.

Οι εκπαιδευόμενοι αφού εγγραφούν μπορούν να δημιουργήσουν το προσωπικό τους προφίλ και στη συνέχεια να εγγραφούν στα μαθήματα που επιθυμούν.

Υπάρχει η δυνατότητα από τους εκπαιδευτές αξιολόγησης μέσω διαφόρων εργαλείων, ενώ οι εκπαιδευόμενοι ενημερώνονται άμεσα για τα αποτελέσματα.

2.5.4 Λειτουργίες του Moodle

Το Moodle παρέχει ένα πλήθος από λειτουργίες που αφορούν τόσο τους εκπαιδευτές όσο και τους εκπαιδευόμενους. Ας δούμε μερικές από τις βασικότερες.

Απουσιολόγιο

Με το απουσιολόγιο, ο εκπαιδευόμενος έχει τη δυνατότητα να παρακολουθεί τις απουσίες του, οι οποίες προκύπτουν από τη συμμέτοχή του σε κάποιο μάθημα. Η καταγραφή των απουσιών γίνετε είτε από τον εκπαιδευτή είτε αυτόματα από την πλατφόρμα.

Απορίες

Όλοι οι εκπαιδευόμενοι μπορούν να εκφράσουν τις απορίες τους μέσω της πλατφόρμας και να λάβουν είτε μια αυτοματοποιημένη απάντηση με βάση την ερώτηση είτε μια προσωποποιημένη από τον εκπαιδευτή.

Ασκήσεις

Ο κάθε εκπαιδευτής μπορεί να αναθέσει ασκήσεις στους εκπαιδευόμενους. Ο εκπαιδευτής μπορεί να ζητήσει την βελτίωση της άσκησης ενώ ο τελικός βαθμός προκύπτει από την βαθμολογία τόσο του εκπαιδευτή όσο και από του ίδιου του εκπαιδευόμενου.

Βιβλίο

Είναι ουσιαστικά το εκπαιδευτικό υλικό σε ηλεκτρονική μορφή.

Διάλογοι

Υπάρχει δυνατότητα επικοινωνίας μεταξύ των εκπαιδευόμενων αλλά και με τους εκπαιδευτές.

Επιλογές

Οι εκπαιδευτές μπορούν να δώσουν τη δυνατότητα στους εκπαιδευόμενους να αποφασίζουν για θέματα που τους αφορούν.

Εργασίες

Ο εκπαιδευτής μπορεί να ορίσει εργασίες στους εκπαιδευόμενους ενώ η λύση της ανεβαίνει από τους εκπαιδευόμενους.

Εργαστήρια

Μέσω των εργαστηρίων οι εκπαιδευόμενοι μπορούν να αξιολογήσουν τις εργασίες τους.

Quiz

Το quiz είναι ενός είδους test το οποίο περιέχει ερωτήσεις με τη μορφή πολλαπλής επιλογής, σωστού λάθους κλπ.

Ομάδες Συζητήσεων

Οι εκπαιδευόμενοι μπορούν να λαμβάνουν μέρος σε ομάδες συζητήσεων (forum) ανταλλάσσοντας απόψεις μεταξύ τους.

SCORM

Το SCORM είναι ένα σύστημα χρήσης μαθησιακού περιεχομένου και βρίσκεται στο διαδίκτυο ως αντικείμενο εκμάθησης του e-learning. Αποτελείται από γραφικά, ιστοσελίδες, προγράμματα JavaScript και γενικά οτιδήποτε λειτουργεί σε έναν Web Browser.

Ακόμα ενδιαφέρουσες λειτουργίες είναι η δημιουργία wikis, Λεξικών, ερευνών και συνομιλιών σε πραγματικό χρόνο.

2.5.5 Αρχιτεκτονική

Η πλατφόρμα μπορεί να τρέξει σε οποιαδίποτε σύστημα υποστηρίζει PHP ενώ χρειάζεται και μια βάση δεδομένων (MySQL, MSSQL, SQLite κλπ). Αποτελείτε από τον κατάλογο εφαρμογών όπου υπάρχουν τα διάφορα modules, ο κατάλογος δεδομένων που περιέχει όλα τα αρχεία που ανεβαίνουν στην πλατφόρμα και η βάση δεδομένων που περιέχει όλο το υλικό των μαθημάτων.

2.6 Γιατί το Moodle

Παρακάτω θα δούμε τους λόγους για τους οποίους επέλεξα την πλατφόρμα Ασύγχρονης Τηλεκπαίδευσης Moodle.

Το Moodle είναι μια ιδιαίτερα ασφαλές πλατφόρμα καθώς ο ρόλος που έχει ο κάθε χρήστης είναι συγκεκριμένος και ελέγχεται. Η πλατφόρμα παρέχει πάρα πολλές δυνατότητες οι οποίες είναι εύκολα αξιοποιήσιμες από τον χρήστη. Ένας ακόμη λόγος είναι το γεγονός ότι είναι μια ανοικτού κώδικα και ευέλικτη πλατφόρμα σε σύγκριση με άλλες εμπορικές.

Η υποστήριξη που παρέχει το συγκεκριμένο Σύστημα Διαχείρισης Μάθησης στους χρήστες και τους διαχειριστές του είναι πολύ καλά οργανωμένη λόγω του ότι έχει πολύ ενεργή κοινότητα, ενώ είναι το πιο διαδεδομένο LMS για τη δημιουργία ιστοσελίδων με εκπαιδευτικό περιεχόμενο, επειδή παρέχει πολλές πρόσθετες εκπαιδευτικές δραστηριότητες.

3

ΤΟ ΜΑΘΗΜΑ

3.1 Εισαγωγή

Στο παρόν κεφάλαιο, θα δούμε αναλυτικά τα κεφάλαια του μαθήματος [1]. Για την πραγματοποίηση και την αρτιότερη κατανόηση του, χρειάστηκε η ανάπτυξη κάποιων παραδειγμάτων. Τα παραδείγματα αυτά είναι φτιαγμένα με τέτοιον τρόπο ώστε να μπορεί να τα κατανοήσει ο εκπαιδευόμενος ακόμα και αν έχει μόνο τις βασικές γνώσεις της JavaScript. Σε κάθε ενότητα, υπάρχει μόνο η θεωρία. Ο πηγαίος κώδικας μαζί με την περιγραφή των παραδειγμάτων που υλοποιήθηκαν, τόσο των JavaScript αρχείων όσο και των HTML και CSS, υπάρχουν στο τελευταίο παράρτημα. Θα ξεκινήσουμε πρώτα με το κομμάτι της JavaScript και στη συνέχεια με του AngularJS.

3.2 Σκοπός του Μαθήματος

Ο σκοπός του εκπαιδευτικού προγράμματος είναι να δώσει τα απαραίτητα εφόδια και επιστημονικά προσόντα σε όλους τους ενδιαφερόμενους με μικρή προηγούμενη γνώση σε προγραμματισμό, μέσα από περιβάλλον που βασίζεται στα windows, το Moodle.

Το σεμινάριο προορίζεται για άτομα τα οποία δεν έχουν καθόλου προηγούμενη εμπειρία στον προγραμματισμό web με χρήση της JavaScript ή έχουν πολύ μικρή εμπειρία και θα ήθελαν να εμβαθύνουν περισσότερο και να βελτιωθούν.

3.3 Ολοκληρώνοντας το μάθημα

Μετά την ολοκλήρωση του μαθήματος, ο εκπαιδευόμενος θα έχει αποκτήσει τις απαραίτητες γνώσεις και δεξιότητες ώστε:

- Να γνωρίζει την γλώσσα προγραμματισμού JavaScript και το JavaScript Framework AngularJS.
- Να αναπτύξει κριτική σκέψη και πρωτοβουλία απλού σχεδιασμού και επίλυσης προβλημάτων ακολουθώντας απλές μεθόδους προγραμματισμού.
- Να είναι σε θέση να υλοποιεί απλές εφαρμογές χρησιμοποιώντας την JavaScript, και ποιο πολύπλοκες με το AngularJS.

3.4 Εισαγωγή στη JavaScript

3.4.1 Εισαγωγή

Σχετικά με την JavaScript

Η JavaScript είναι μια από τις ποιο δημοφιλής γλώσσες προγραμματισμού, και συγκεκριμένα είναι μια scripting γλώσσα που δημιουργήθηκε με στόχο να κάνει τις ιστοσελίδες ποιο διαδραστικές, ενώ ήδη χρησιμοποιείτε από τα σύγχρονα λειτουργικά συστήματα με στόχο την αύξηση της ταχύτητας και της συμβατότητας.

Χαρακτηριστικά

Η JavaScript είναι ερμηνευόμενη scripting γλώσσα προγραμματισμού. Αυτό σημαίνει ότι κατά την εκτέλεση του κώδικα, υπάρχει ένας διερμηνέας ο οποίος εκτελεί τον κώδικα χωρίς να χρειάζεται να μεταγλωττιστεί πρώτα σε γλώσσα μηχανής.

Υποστηρίζει τον δομημένο προγραμματισμό, καθώς έχει όμοια σύνταξη με την c και τη c++. Μια διαφοροποίηση με αυτές είναι πως η JavaScript προσθέτει αυτόματα το ερωτηματικό στο τέλος δίνοντας την δυνατότητα στον προγραμματιστή να το παραλείπει.

Έχει χαλαρό σύστημα τύπων. Όπως στις περισσότερες scripting γλώσσες έτσι και στη JavaScript μια τιμή σχετίζεται με έναν τύπο. Με αυτόν τον τρόπο μια μεταβλητή μπορεί να είναι ένας αριθμός ενώ αργότερα να την κάνουμε αλφαριθμητικό. Επίσης δεν χρειάζεται ο προγραμματιστής να ορίσει τον τύπο μιας μεταβλητής όπως για παράδειγμα γίνεται στη c, καθώς αυτό γίνεται αυτόματα.

Είναι ανεξάρτητη από την πλατφόρμα. Εκτελείτε δηλαδή με τον ίδιο τρόπο, ανεξάρτητα από το λειτουργικό σύστημα που έχει ο υπολογιστής. Και αυτό γιατί η JavaScript στηρίζεται στο περιβάλλον όπου εκτελείτε. Ένα τέτοιο περιβάλλον εκτέλεσης μπορεί να είναι ένας web browser.

Υποστηρίζει διάφορα χαρακτηριστικά όπως δομές επιλογής και επανάληψης, πίνακες και πίνακες συσχετίσεων, συναρτήσεις η οποίες λόγο της υποστήριξης του αντικειμενοστραφούς προγραμματισμού μπορούν να χρησιμοποιηθούν και ως μέθοδοι, χρησιμοποιεί πρωτότυπα καθώς και regular expressions.

Τέλος, η JavaScript παρόλο που αρχικά ήταν αποκλειστικά client based, δηλαδή ο κώδικας εκτελείτε στον υπολογιστή του χρήστη και όχι σε κάποιον διακομιστή, πλέον υπάρχουν διάφοροι τρόποι όπως με την χρήση του node.js να εκτελεστεί κώδικας JavaScript στη μεριά του διακομιστή.

Προαπαιτούμενα

Δεν χρειάζεται να υπάρχει κάποια προηγούμενη γνώση στον προγραμματισμό για την ολοκλήρωση του μαθήματος. Για την εκτέλεση των παραδειγμάτων χρειάζεται ένας οποιοσδήποτε browser και ένας οποιασδήποτε editor.

Βασικές αρχές της JavaScript

Αρχικά για να εισάγουμε κώδικα JavaScript μέσα σε ένα αρχείο html χρησιμοποιούμε το tag <script>. Ενώ σε παλαιότερες εκδώσεις χρειαζόταν να δηλώσουμε τον τύπο του script, στην Html5 δεν χρειάζεται.

Μπορούμε να εισάγουμε JavaScript σε οποιοδήποτε σημείο στον κώδικά μας ανάμεσα στα html tag. Καλό είναι όμως όταν σε ένα project έχουμε να εισάγουμε πολλά ή μεγάλα script να τα βάζουμε στο τέλος, πριν κλείσουμε το body tag, και αυτό διότι με αυτόν τον τρόπο καταφέρνουμε ο browser να φορτώσει πρώτα τα html/css και στο τέλος τα

JavaScript έτσι ώστε ο χρήστης να μην χρειάζεται να περιμένει να φορτώσουν πρώτα τα scripts για να δει την ιστοσελίδα.

Τέλος υπάρχουν 2 τρόποι εισαγωγής ενός JavaScript. Ο πρώτος είναι τοποθετώντας τον κώδικα μέσα στα script tags και ο δεύτερος σε ένα εξωτερικό αρχείο. Όποιον τρόπο και να διαλέξουμε θα έχουμε το ίδιο αποτέλεσμα, ωστόσο χρησιμοποιώντας τον 2ο έχουμε 3 πλεονεκτήματα: 1) ξεχωρίζουμε τον κώδικα από την html 2) μπορούμε να συντηρήσουμε τον κώδικά μας ευκολότερο και τέλος 3) να πετύχουμε καλύτερο χρόνο φόρτωσης της σελίδας λόγο του cache που κάνει ο browser στο εξωτερικό αρχείο.

Σχετικά με την σύνταξη, να πούμε ότι η JavaScript είναι case sensitive, δηλαδή είναι διαφορετική μεταβλητή η name με την Name. Στο τέλος κάθε εντολής, αν και δεν είναι υποχρεωτικό, καλό είναι να εισάγουμε το ";". Ακόμα έχουμε σχόλια μίας και πολλών γραμμών. Για να εμφανίσουμε δεδομένα στη JavaScript έχουμε 4 επιλογές: η 1η είναι με τη χρήση της μεθόδου alert που πετάει ένα popup στον χρήστη, η 2η είναι η document.write που γράφει στην έξοδο της html, η 3η είναι η innerhtml που γράφει μέσα σε ένα αντικείμενο της html και η τελευταία είναι η conslole.log που γράφει στη console του browser.

Δομικά Στοιχεία

Ας δούμε τώρα τα βασικά δομικά στοιχεία της JavaScript, τους τύπους.

Ας ξεκινήσουμε με το πρώτο στοιχείο που είναι οι αριθμοί. Στην JavaScript ένας αριθμός είναι απλά ένας αριθμός. Τι εννοούμε με αυτό. Μπορούμε να κάνουμε πράξεις μαζί τους χωρίς να χρειάζεται να δηλώσουμε αν είναι για παράδειγμα ακέραιος ή δεκαδικός, θετικός ή αρνητικός. Όταν δηλαδή θέλουμε να γράψουμε έναν αριθμό, απλά τον πληκτρολογούμε. Για παράδειγμα, ένας αριθμός μπορεί να είναι το 100 ή το 10,1.

Ας δούμε τώρα το επόμενο στοιχείο που είναι τα αλφαριθμητικά. Τα αλφαριθμητικά είναι ακολουθίες χαρακτήρων. Χαρακτήρες μπορεί να είναι τα γράμματα, οι αριθμοί, και τα κενά. Όταν θέλουμε δηλαδή να γράψουμε ένα αλφαριθμητικό, το εισάγουμε μέσα σε μονά ή διπλά αυτάκια. Για παράδειγμα ένα αλφαριθμητικό μπορεί να είναι το "ποδήλατο" αλλά και το "100". Ενδιαφέρων είναι πως άμα θέλουμε μπορούμε να κάνουμε ένωση χρησιμοποιώντας το "+".

Επόμενο στοιχείο είναι οι μεταβλητές. Οι μεταβλητές χρησιμοποιούνται για την αποθήκευση τιμών. Δηλώνονται βάζοντας μπροστά την λέξη var για παράδειγμα var number;

Υπάρχουν μερικοί κανόνες για την ονομασία των μεταβλητών οι οποίοι είναι:

- Το όνομα τους μπορεί να περιέχει γράμματα, αριθμούς, κάτω παύλα, $
- Πρέπει να ξεκινάει με γράμμα ή $,_ και όχι από αριθμό
- Είναι case-sensitive
- Δεν μπορεί να είναι κάποια από τις δεσμευμένες λέξεις

Επόμενο στοιχείο είναι οι τελεστές. Στους τελεστές έχουμε τους αριθμητικούς, ανάθεσης, συμβολοσειρών, σύγκρισης, λογικούς, δυαδικούς και τριαδικό που μοιάζει με μια if.

Αριθμητικοί	+	-	*	/	%	++	--	
Ανάθεσης	=	+=	-=	/=	%=			
Συμβολοσειράς	+	+=						
Σύγκρισης	==	===	!=	!==	>	<	>=	<=
Λογικοί	&&	||	!					
Διάδικοί	&	|	~	^	<<	>>		
Τριαδικός	var = (συνθήκη) ? τιμή1 : τιμή2;							

Εικόνα 1: Τελεστές

Αφού είπαμε και για τους τελεστές ας πούμε και για τις εκφράσεις. Μια έκφραση λοιπόν είναι οποιοδήποτε κομμάτι κώδικα που καταλήγει σε ένα αποτέλεσμα. Υπάρχουν 2 είδη εκφράσεων, αυτές που αναθέτουν μια τιμή σε μια μεταβλητή και αυτές που απλά έχουν μια τιμή. Ως παράδειγμα εδώ μπορούμε να πούμε το X=7 και το 3+4 αντίστοιχα για τις 2 περιπτώσεις.

Οι δηλώσεις ή statments είναι σετ από expressions. Είναι δηλαδή οι οδηγίες για το τι θα εκτελέσει ο browser. Εκτελούνται ένα προς ένα με την σειρά που υπάρχουν. Διαχωρίζονται από το “;”, ενώ ξεκινάνε με μια από τις παρακάτω δεσμευμένες λέξεις:

break	Terminates a switch or a loop
continue	Jumps out of a loop and starts at the top
debugger	Stops the execution of JavaScript, and calls (if available) the debugging function
do ... while	Executes a block of statements, and repeats the block, while a condition is true
for	Marks a block of statements to be executed, as long as a condition is true
function	Declares a function
if ... else	Marks a block of statements to be executed, depending on a condition
return	Exits a function
switch	Marks a block of statements to be executed, depending on different cases
try ... catch	Implements error handling to a block of statements
var	Declares a variable

Πίνακας 1: Δεσμευμένες λέξεις

Σχετικά τώρα με το κενό ή whitespace όπως ονομάζεται, όταν το εισάγουμε πάνω από 1 φορά σε ένα statement τότε αυτό υπολογίζεται ως ένα.

Τέλος έχουμε τις συναρτήσεις και τα αντικείμενα για τα οποία θα μιλήσουμε σε επόμενη ενότητα.

Βασικές μέθοδοι

Σε αυτή την ενότητα θα δούμε μερικές χρήσιμες μεθόδους και ιδιότητες που θα μας διευκολύνουν στο να κάνουμε διάφορες εργασίες.

Ας δούμε πρώτα μερικές μεθόδους που μπορούμε να χρησιμοποιήσουμε με όλους τους τύπους δεδομένων:

Number()
που μετατρέπει την παράμετρο σε αριθμό.

parseFloat()
που μετατρέπει την παράμετρο σε δεκαδικό.

parseInt()
που μετατρέπει την παράμετρο σε ακέραιο.

Ας δούμε τώρα μερικές μεθόδους που μπορούμε να χρησιμοποιήσουμε με τους αριθμούς:

toString()
που επιστρέφει έναν αριθμό ως συμβολοσειρά.

toExponential()
που επιστρέφει μια συμβολοσειρά με έναν αριθμό στρογγυλοποιημένο σε εκθετική μορφή. Για να ορίσουμε την στρογγυλοποίηση θα πρέπει να δώσουμε ως παράμετρο την θέση στην οποία θα γίνει.

toFixed()
επιστρέφει μια συμβολοσειρά με έναν αριθμό γραμμένο με συγκεκριμένα δεκαδικά ψηφία που έχουμε ορίσει ως παράμετρο.

toPrecision()
επιστρέφει μια συμβολοσειρά με έναν αριθμό γραμμένο με συγκεκριμένο μήκος μπορούμε να το ορίσουμε ως παράμετρο.

valueOf()
που επιστρέφει έναν αριθμό ως αριθμό.

Στα αλφαριθμητικά μπορούμε να χρησιμοποιήσουμε μερικές από τις εξής μεθόδους:

indexOf()
που επιστρέφει την θέση που βρίσκετε μια λέξη μέσα σε μια πρόταση.

lastIndexOf()
που επιστρέφει την θέση που βρίσκετε μια λέξη μέσα σε μια πρόταση με βάση το τελευταίο γράμμα της λέξης.

search()
που παρόμοια με την indexOf() με την διαφορά ότι η πρώτη παρέχει ποιο ισχυρές αναζητήσεις τιμών.

Ακόμα έχουμε τις slice(start, end), substring(start, end) και substr(start, length) όπου η πρώτη επιστρέφει από ένα string κάποιο τμήμα της, ενώ οι άλλες 2 δίνουν ίδιο αποτέλεσμα με την διαφορά ότι η 2η δεν δέχεται αρνητικές τιμές και η 3η παίρνει ως δεύτερη παράμετρο το μήκος που θέλουμε να έχει το η νέα συμβολοσειρά μας.

Boolean

Ας πούμε και 2 λόγια για τις δυαδικές μεταβλητές. Οι μεταβλητές αυτές έχουν 2 τιμές. True ή false. Και μας δίνουν τη δυνατότητα να τις χρησιμοποιούμε όποτε χρειαζόμαστε μια μεταβλητή να έχει 1 ή 2 τιμές. Ωραία. λοιπόν. Να πούμε εδώ ότι όποια τιμή περιέχει μια πραγματική τιμή, τότε αυτή είναι αληθής, ενώ οι υπόλοιπες είναι ψευδής. Στις ψευδής περιλαμβάνονται οι εξής περιπτώσεις. Όταν η τιμή μιας δυαδικής μεταβλητής είναι 0, -0, όταν έχει κενό αλφαριθμητικό, έχει απροσδιόριστη τιμή, είναι null ή NaN (Not a Number) καθώς και την τιμή false. Με αυτά τελειώσαμε και αυτό το κεφάλαιο. Στο επόμενο θα ασχοληθούμε με τις δομές ελέγχου.

3.4.2 Δομές ελέγχου

Στην JavaScript όπως και στις γλώσσες c++/Java όπου και έχει δανειστεί στοιχεία, έχουμε τις κλασσικές δομές επιλογής if/else if και switch καθώς και τις δομές επανάληψης for/while και do while.

Σύνταξη if/else if

```
if (συνθήκη){
    κώδικάς που θα εκτελεστεί αν ισχύει η συνθήκη
}
```

```
if (συνθήκη){
    κώδικάς που θα εκτελεστεί αν ισχύει η συνθήκη
} else {
    κώδικάς που θα εκτελεστεί αν δεν ισχύει η συνθήκη
}
```

Σύνταξη switch

```
switch(έκφραση) {
    case n:
       κώδικας;
        break;
    case m:
       κώδικας;
        break;
    default:
        κώδικας που θα εκτελεστεί αν δεν ισχύει καμία από τις παραπάνω
περιπτώσεις;
}
```

Σύνταξη for / for in

```
for (δήλωση 1;  δήλωση 2;  δήλωση 3) {
    κώδικας που θα εκτελεστεί
}
```

```
for (x in y) {
    κώδικας που θα εκτελεστεί
}
```

Σύνταξη while / do while

```
while (συνθήκη) {
    κώδικας που θα εκτελεστεί
}
```

```
do {
    κώδικας που θα εκτελεστεί
}
while (συνθήκη);
```

3.4.3 Συναρτήσεις

Σύνταξη while

Μια συνάρτηση είναι ένα κομμάτι κώδικα που σχεδιάστηκε για να κάνει μια συγκεκριμένη λειτουργία. Μπορούμε να δηλώσουμε μια συνάρτηση με 3 τρόπους. Ο 1ος είναι χρησιμοποιώντας την λέξη function ακολουθούμενη από ένα όνομα και ο 2ος είναι δηλώνοντας πρώτα μια μεταβλητή και στη συνέχεια δηλώνεται η συνάρτηση και ο 3ος χρησιμοποιώντας τον function constructor. Ας τώρα δούμε λίγο την σύνταξη.

1) function όνομα(παράμετρος1, παράμετρος2,…) { κώδικας που θα εκτελεστεί }
2) var μεταβλητή = function (παράμετρος1, παράμετρος2,…) κώδικας που θα εκτελεστεί};
3) var μεταβλητή = new Function("παράμετρος1",…, "κώδικας που θα εκτελεστεί");

Παράμετροι Συναρτήσεων

Ας αναφερθούμε τώρα στις παραμέτρους των συναρτήσεων. Πρώτον δεν χρειάζεται να δηλώσουμε τον τύπο των παραμέτρων που περνάμε μέσα σε μια συνάρτηση όπως επίσης δεν γίνεται και έλεγχος του τύπου τους. Ακόμα δεν ελέγχετε ο αριθμός των παραμέτρων που λαμβάνει η συνάρτηση. Αυτό σημαίνει ότι, αν κληθεί μια συνάρτηση με λιγότερες παραμέτρους από ότι έχει οριστεί, τότε οι τιμές τους γίνονται απροσδιόριστες (ή undefined), αντίθετα αν κληθεί με περισσότερες παραμέτρους από ότι έχει οριστεί, τότε αυτές μπορούμε να τις αξιοποιήσουμε χρησιμοποιώντας το αντικείμενο παραμέτρων (ή arguments object).

Ακόμα οι παράμετροι περνιούνται by value. Αυτό σημαίνει ότι η συνάρτηση πρώτον διαβάζει απλά τις τιμές των παραμέτρων και όχι το που βρίσκονται. Δεύτερον αν αλλάξει η τιμή μιας μεταβλητής μέσα σε μια συνάρτηση, τότε αυτή δεν θα περαστεί στην αρχική μεταβλητή μιας και ότι γίνετε μέσα σε μια συνάρτηση δεν είναι ορατό έξω από αυτή.

Εκτέλεση Συναρτήσεων

Όταν θέλουμε να εκτελέσουμε μια συνάρτηση, έχουμε 4 διαφορετικούς τρόπους. Ο 1ος είναι καλώντας την ως συνάρτηση, ο 2ος ως μέθοδο, ο 3ος με χρήση του function constructor και ο τελευταίος ως μέθοδο συνάρτησης.

1) Όνομα_συνάρτησης(παράμετρος1, παράμετρος2,…);
2) Αντικείμενο.όνομα_συνάρτησης();
3) var x = new όνομα_συνάρτησης("παράμετρος1",…);
4) Αντικείμενο = όνομα_συνάρτησης.call(Αντικείμενο, παράμετρος1, …);

Argument Object

Το Argument Object ουσιαστικά είναι ένας πίνακας ο οποίος ονομάζεται arguments, και εκεί κρατούνται όλοι οι παράμετροι την στιγμή που γίνετε κλήση μιας συνάρτησης.

3.4.4 Αντικείμενα

Εισαγωγή

Η JavaScript κάνει χρήση ενός προτύπου υποστήριξης προγραμματισμού με αντικείμενα, δίνοντας στον προγραμματιστή την δυνατότητα να δημιουργεί τα δικά του αντικείμενα καθώς και να χρησιμοποιεί υπάρχοντα.

Το πρώτο πράγμα που πρέπει να αναφέρουμε είναι ότι στη JavaScript όλα είναι αντικείμενα. Ή μάλλον σχεδόν όλα. Γιατί εξαιρούνται οι αριθμοί, τα αλφαριθμητικά, το true/false καθώς και οι τιμές null και undefined που μπορούν να γραφούν και ως αντικείμενα. Το δεύτερο είναι, ότι τα αντικείμενα είναι μεταβλητές που περιέχουν μεταβλητές. Σε αντίθεση με τις απλές μεταβλητές που μπορούν να περιέχουν μια τιμή, τα αντικείμενα μπορούν να περιέχουν πολλές, πίνακες ή και άλλα αντικείμενα.

Δημιουργία Αντικειμένων

Ας δούμε λίγο τι τρόπους έχουμε για να δημιουργήσουμε αντικείμενα. Έχουμε 3 διαφορετικούς τρόπους. Ο 1ος είναι δηλώνοντας και δημιουργώντας ένα αντικείμενο χρησιμοποιώντας την λέξη new, ο 2ος είναι δηλώνοντας μια μεταβλητή που θα είναι το αντικείμενο μας, και μέσα στις αγκύλες φτιάχνουμε τις ιδιότητες του, και ο 3ος χρησιμοποιώντας τον object constructor.

Literal

Ο τρόπος αυτός, παρότι έχει διαφορετική σύνταξη, παράγει το ίδιο αποτέλεσμα:

```
var myObject = {property1:value1,property2:value2,…};
```

1. Με χρήση του new Object

```
Var myObject = new Object();
myObject .property1=value1;
myObject.property2=value2…
```

2. Object constructor

Αυτός ο τρόπος, προσφέρετε όταν θέλουμε να δημιουργήσουμε πολλά αντικείμενα, που όμως θα έχουν τις ίδιες ιδιότητες. Αυτό ονομάζεται object type και για το δημιουργήσουμε χρησιμοποιούμε τον object constructor.

```
Function myObject(var1,var2,…){
        this.property1=var1;
        this.property2=var2;
}
```

Στον contructor, δηλώνουμε μέσα στις παρενθέσεις τις μεταβλητές που δέχεται και μέσα στις αγκύλες κάνουμε τις αρχικοποιήσεις. Το μόνο που χρειάζεται να προσέξουμε εδώ είναι οι μεταβλητές που είναι δεξιά από το = να υπάρχουν μέσα στις παρενθέσεις, ενώ δεν χρειάζεται να είναι ίδιες οι μεταβλητές αριστερά του = με αυτές δεξιά.

Constructors

Η JavaScript έχει κάποιους έτοιμους contructors που μπορούμε να χρησιμοποιήσουμε:

```
1. var x = new Object();
   Δημιουργεί 1 νέο αντικείμενο ενός αντικειμένου.
2. var x = new String();
   Δημιουργεί 1 νέο αντικείμενο ενός αλφαριθμητικού.
3. var x = new Number();
   Δημιουργεί 1 νέο αντικείμενο ενός αριθμού.
4. var x = new Boolean();
   Δημιουργεί 1 νέο αντικείμενο ενός δυαδικού.
5. var x = new Array();
   Δημιουργεί 1 νέο αντικείμενο ενός πίνακα.
6. var x = new RegExp();
   Δημιουργεί 1 νέο αντικείμενο ενός Regular expression.
7. var x = new Function();
   Δημιουργεί 1 νέο αντικείμενο μιας συνάρτησης.
8. var x = new Date();
   Δημιουργεί 1 νέο αντικείμενο μιας ημερομηνίας.
```

Εκτός από ειδικές περιπτώσεις, καλό είναι να μην χρησιμοποιούμε αυτούς τους contructors. Και αυτό γιατί όταν πάμε να δημιουργήσουμε ένα αλφαριθμητικό για παράδειγμα, τότε αν το κάνουμε με την χρήση του contructor τότε χάνουμε σε ταχύτητα ενώ κάνουμε και τον κώδικά μας ποιο περίπλοκο.

```
1. var x = {};
2. var x = “”;
3. var x = 0;
4. var x = false;
5. var x = [];
6. var x = /()/;
7. var x = function;
```

This

Το this λοιπόν είναι 1 αντικείμενο στο οποίο ανήκει ο κώδικας. Για πχ όταν χρησιμοποιηθεί μέσα σε μια συνάρτηση τότε το this είναι ένα αντικείμενο που του ανήκει η συνάρτηση ενώ αν χρησιμοποιηθεί μέσα σε ένα αντικείμενο τότε του ανήκει αυτό.

Ιδιότητες και Μέθοδοι

Ας δούμε πρώτα πως μπορούμε έχουμε πρόσβαση στις ιδιότητες ενός αντικειμένου. Έχουμε 3 τρόπος. Ο 1ος είναι γράφοντας το όνομα του αντικειμένου και το όνομα της ιδιότητας, ο 2ος γράφοντας το όνομα του αντικειμένου και σε αγκύλες το όνομα της ιδιότητας και ο 3ος είναι γράφοντας το όνομα του αντικειμένου και σε αγκύλες μια έκφραση.

1. objectName.property
πχ dog.name

2. bjectName["property"]
πχ dog["name”]

3. objectName[expression]
πχ x = "name"; dog[x]

Αν και ο τελευταίος τρόπος δεν φαίνεται να δίνει κάποιο προφανή πλεονέκτημα σε σχέση με τον 2ο, η χρησιμότητά του φαίνεται αν έχουμε μια επανάληψη.

Σχετικά με τις μεθόδους, είναι οι ενέργειες που μπορούν να κάνουν τα αντικείμενα. Μπορούμε να δημιουργήσουμε μια μέθοδο, γράφοντας το όνομα της, “:”, function() και μέσα στις αγκύλες τον κώδικα που θέλουμε να τρέχει όταν την καλέσουμε με τον εξής τρόπο:

```
var myObject = {
      property:value,
      methodName: function() {code…}};
```

ενώ μπορούμε να την καλέσουμε γράφοντας το όνομα του αντικειμένου, τελεία και μετά το όνομα της μεθόδου:

```
myObject. methodName();
```

Πίνακες

Οι πίνακες χρησιμεύουν στην αποθήκευση πολλών τιμών σε μία μεταβλητή. Για να δηλώσουμε έναν πίνακα, έχουμε 2 τρόπους. Ο 1ος είναι χρησιμοποιώντας την απλή σύνταξη και ο 2ος με τη χρήση του array constructor. Στον πρώτο τρόπο, δηλώνουμε πρώτα μια μεταβλητή και μετά μέσα σε αγκύλες γράφουμε τις τιμές που θέλουμε να έχει. Στον 2ο τρόπο δηλώνουμε πάλι μια μεταβλητή και μετά δημιουργούμε τον πίνακα με τη χρήση του array constructor περνώντας του μέσα στις παρενθέσεις τις τιμές του πίνακα. Και οι 2 τρόποι έχουν το ίδιο αποτέλεσμα. Και πάλι όμως είναι προτιμότερο να χρησιμοποιούμε τον 1ο τρόπο γιατί είναι γρηγορότερος στην εκτέλεση.

```
1. var myArray = [item1, item2, ...];
2. var myArray = new Array(item1, item2, ...);
```

Όπως και σε άλλες γλώσσες προγραμματισμού, έτσι και εδώ η πρόσβαση γίνεται με τη βοήθεια ενός δείκτη. Και στη JavaScript, το 1ο στοιχείο του πίνακα είναι το 0, το 2ο το 1 και ου το καθεξής.

```
for(var i=0;i<nums.length;i++)
{
   document.write(nums[i]+", ");
}
```

Η εμφάνιση των στοιχείων μπορεί να γίνει με τους εξής τρόπους:

- `document.write(nums[5]);`
- `document.write(nums);`

Για να προσθέσουμε άλλο ένα στοιχείο στον πίνακα, έχουμε 3 τρόπους. Ο πρώτος είναι με τη χρήση της μεθόδου push(), η οποία προσθέτει ένα νέο στοιχείο στο τέλος του πίνακα, ο 2ος με τη χρήση της ιδιότητας length και ο 3ος τοποθετώντας το σε μια συγκεκριμένη θέση μέσα στον πίνακα. Ο τελευταίος όμως καλό θα ήταν να αποφεύγεται διότι δεν είναι και ότι καλύτερο γιατί αν έχουμε πολλά στοιχεία και δώσουμε μεγάλη τιμή στον δείκτη, τότε το αποτέλεσμα θα είναι να έχουμε έναν πίνακα με κενά.

Μέθοδοι πινάκων

Υπάρχουν διάφορες μέθοδοι που μπορούμε να χρησιμοποιήσουμε στους πίνακες:

.push();

Προσθέτει ένα στοιχείου στο τέλος του πίνακα.

.pop();

Αφαιρεί το τελευταίο στοιχείο του πίνακα.

.sort();

Ταξινόμηση έναν πίνακα.

.reverse();

κάνει φθίνουσα ταξινόμηση στον πίνακα.

.slice();

Διαχωρίζει τα στοιχεία ενός πίνακα.

.shift();

Αφαίρει το 1ο στοιχείο του πίνακα.

.unshift();

Προσθέτει νέο στοιχείο στην αρχή του πίνακα.

.concat();

Κάνει ένωση δυο πινάκων.

.join();

Ενώνει τα στοιχεία ενός πίνακα σε ένα string χωρίς τα κόμματα.

.toString();

Μετατρέπει να στοιχεία του πίνακα σε strings.

Αλφαριθμητικά

Όπως έχουμε δει, αλφαριθμητικά μπορούμε να εισάγουμε μέσα σε μονά ή διπλά αυτάκια, οι χαρακτήρες διαφυγής ξεκινάνε με το "\" ενώ μπορούμε να ενώσουμε 2 ή και παραπάνω αλφαριθμητικά με το "+". Ακόμα είναι διαθέσιμες οι περισσότεροι μέθοδοι που έχουμε αναφέρει ποιο πριν.

\'	single quote
\"	double quote
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Πίνακας 2: Χαρακτήρες Διαφυγής

Regular Expressions

Τα regular expressions, είναι μια ακολουθία από χαρακτήρες που σχηματίζουν ένα μοτίβο αναζήτησης. Μπορεί να είναι ένας χαρακτήρας, μπορεί να είναι πολλοί, μπορεί να είναι αριθμοί, ή ακόμα και ένας πολύ πολύπλοκος συνδυασμός όλων αυτών. Χρησιμεύουν κυρίως για αναζητήσεις και αντικαταστάσεις γιατί μπορείς με τη χρήση του να περιγράψεις αυτό που θες να ψάξεις. Ενα regular expressions μπαίνει μέσα σε διπλή κάθετο.

Οι modifiers μας επιτρέπουν να κάνουμε κάποιες ενέργειες. Αυτοί είναι το I, το g και το m, όπου το πρώτο κάνει ταίριασμα χωρίς να κάνει διάκριση μεταξύ πεζών και κεφαλαίων. Το 2ο μας επιτρέπει να κάνουμε ένα γενικό ταίριασμα χωρίς να σταματάει η αναζήτηση στο 1ο αποτέλεσμα ενώ το m πραγματοποιεί ταίριασμα πολλών γραμμών.

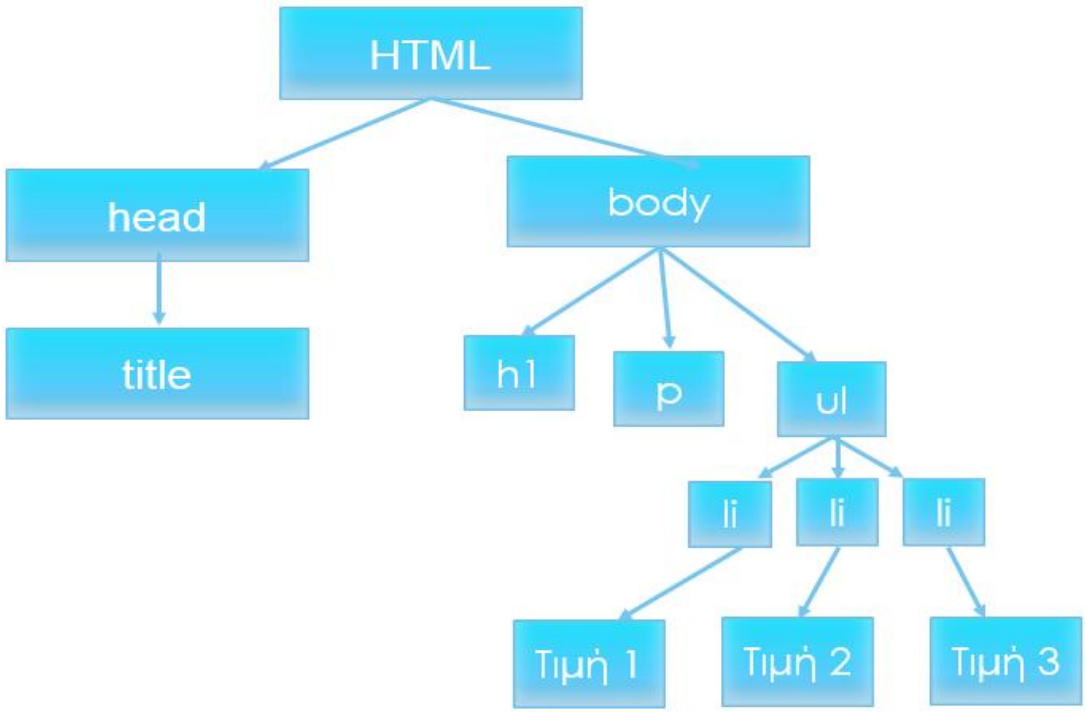
Για να αναζητήσουμε μερικά γράμματα, μπορούμε να χρησιμοποιήσουμε αγκύλες και μέσα σε αυτές τα γράμματα που θέλουμε να αναζητήσουμε. Μπορούμε να αναζητήσουμε είτε συνεχόμενα χρησιμοποιώντας την "-" είτε μεμονωμένα. Τέλος αν

χρησιμοποιήσουμε την μέθοδο test που μας επιστρέφει true ή false ανάλογα αν βρέθηκε ή όχι αυτό που αναζητάμε.

3.4.5 DOM, Events and Event handling

Document Object Model

Το DOM είναι η παρουσίαση όλων των στοιχείων που απαρτίζουν μια ιστοσελίδα. Είναι δηλαδή, το τελικό αποτέλεσμα του κώδικά μας. Υπάρχουν 2 κατηγορίες. Το core dom το xml dom και το html dom. Εμείς θα ασχοληθούμε με το τελευταίο. Δηλώνει τα στοιχεία της html, τις ιδιότητές της, τις μεθόδους και τα events που δίνουν πρόσβαση στα στοιχεία της.



Εικόνα 2: Η δομή του DOM

Ιδιότητες και μέθοδοι

Οι ιδιότητες είναι η τιμές που μπορούμε να πάρουμε από 1 html στοιχείο ή να δώσουμε ενώ οι μέθοδοι οι ενέργειες που μπορούμε να κάνουμε στα στοιχεία της html. Για παράδειγμα στο παρακάτω απόσπασμα html:

```
<p id=text></p> in script->
document.getelementbyid(text).innerHTML=”Hello!”;
```

το getelementbyid είναι η μέθοδος και το innerHTML η ιδιότητα. Εδώ λοιπόν, έχουμε ένα στοιχείο της html το p και έχουμε δώσει το id text. Με το getelementbyid βρίσκουμε το στοιχείο με id text και με το innerhtml έχουμε πρόσβαση στη τιμή του στοιχείου με αυτό το id.

Μπορούμε να έχουμε πρόσβαση στα στοιχεία της html. Αυτό το κάνουμε με τη βοήθεια του document object, ενώ αυτό μας παρέχει ένα πλήθος από μεθόδους και ιδιότητες.

- **document.write(text)**
  Γράφει στην έξοδο της html.
- **document.getElementById(id)**
  βρίσκει ένα στοιχείο της html με βάση το id.
- **document.getElementsByTagName(name)**
  βρίσκει ένα στοιχείο της html με βάση το όνομα tag.
- **document.getElementsByClassName(name)**
  βρίσκει ένα στοιχείο της html με βάση το όνομα της κλάσης.
- **element.innerHTML = new html content**
  αλλάζει το περιεχόμενο ενός στοιχείου
- **element.attribute = new value**
  αλλάζει την τιμή ενός χαρακτηριστικού
- **element.setAttribute(attribute, value)**
  αλλάζει την τιμή ενός χαρακτηριστικού
- **element.style.property = new style**
  αλλάζει το στυλ ενός στοιχείου.

- **document.createElement(element)**
  δημιουργεί 1 νέο στοιχείο.
- **document.removeChild(element)**
  αφαιρεί 1 στοιχείο
- **document.appendChild(element)**
  προσθέτει ένα στοιχείο παιδί σε γονικό στοιχείο
- **document.replaceChild(element)**
  αντικαθιστά ένα στοιχείο παιδί

Events

Τα events, είναι το αποτέλεσμα πάνω στα στοιχεία της html μετά από κάποια ενέργεια. Για παράδειγμα, μια τέτοια ενέργεια μπορεί να είναι το πάτημα ενός κουμπιού, η ακόμα και η ολοκλήρωση του φορτώματος μιας σελίδας. Συνήθως όταν θέλουμε να εκτελεστεί ένα κομμάτι κώδικα μετά από κάποια ενέργεια, τότε χρησιμοποιούμε events.

```
<h1 onclick="click1()" id="title">DOM - Events</h1>
```

Όπως καταλαβαίνουμε και από το όνομα του event, μόλις ο χρήστης κάνει κλικ στην επικεφαλίδα, τότε καλείτε η συνάρτηση που φτιάξαμε.

AddEventListener

Μπορούμε μέσο της μεθόδου αυτής, να ορίσουμε σε οποιοδήποτε στοιχείο του dom ένα event handler. Μπορούμε να προσθέσουμε πολλά events σε ένα στοιχείο καθώς και πολλά events του ίδιου τύπου στο ίδιο στοιχείο, για παράδειγμα δύο onclick σε κουμπί.

```
document.getElementById('par2').addEventListener('dblclick',hide);
```

Μέσα στην addeventlistener, έχουμε δύο παραμέτρους. Η 1η είναι το event, το οποίο όπως παρατηρούμε γράφεται χωρίς το “on” μπροστά ενώ η δεύτερη είναι η συνάρτηση που θέλουμε να κληθεί μόλις ενεργοποιηθεί το event. Στη 2η παράμετρο μπορούμε είτε να

γράψουμε ολόκληρη την συνάρτηση είτε μόνο το όνομά της. Για να προσθέσουμε δύο διαφορετικά events σε ένα στοιχείο της html μπορούμε να κάνουμε το εξής:

```
	document.getElementById('list1').addEventListener('mouseover',mouseover
);
	document.getElementById('list1').addEventListener('mouseout',mouseout);
```

Τέλος, όπως είπαμε στην αρχή, μπορούμε να ορίσουμε ένα event σε οποιοδήποτε στοιχείο του dom.

```
window.addEventListener('load',page);
```

3.5 Εισαγωγή στο AngularJS

3.5.1 Εισαγωγή

Έχοντας ολοκληρώσει το κομμάτι της JavaScript, πάμε στο 2ο μέρος που είναι το AngularJS. Το AngularJS είναι ένα client side JavaScript framework που προσθέτει στη html, νέα χαρακτηριστικά και ιδιότητες κάνοντας την δυναμική.

Για να προχωρήσουμε στο AngularJS, χρειάζεται να γνωρίζουμε html και JavaScript ενώ χρήσιμες είναι οι γνώσεις css και βάσεων δεδομένων. Δεν χρειάζεται να γνωρίζουμε κάποιο άλλο JavaScript framework όπως το jQuery καθώς και οποιαδήποτε τεχνολογία back-end όπως php κα Node.js κ.α.

Ας δούμε όμως γιατί να επιλέξουμε το AngularJS. Όπως είπαμε, κάνει την html δυναμική, οπότε χρησιμοποιείτε σε μεγάλες κι δυναμικές σελίδες. Επίσης μπορεί να γίνει χρήση μαζί και άλλου framework όπως του bootstrap jQuery. Άλλο ένα σημαντικό πλεονέκτημα που προσφέρει είναι ότι υποστηρίζει τις αρχιτεκτονικές mvc – model view controller, mvvm – model view view-model καθώς και το RESTful – representational state transfer. Άλλο ένα πλεονέκτημα είναι ότι μπορούμε να χειριστούμε ευκολότερα το DOM γράφοντας λιγότερες γραμμές κώδικα ενώ όπως είπαμε και ποιο πριν, προσθέτει νέα στοιχεία και ιδιότητες στην html. Ως αποτέλεσμα όλων των παραπάνω, μπορεί να γίνεται ευκολότερη συντήρηση του κώδικά μας.

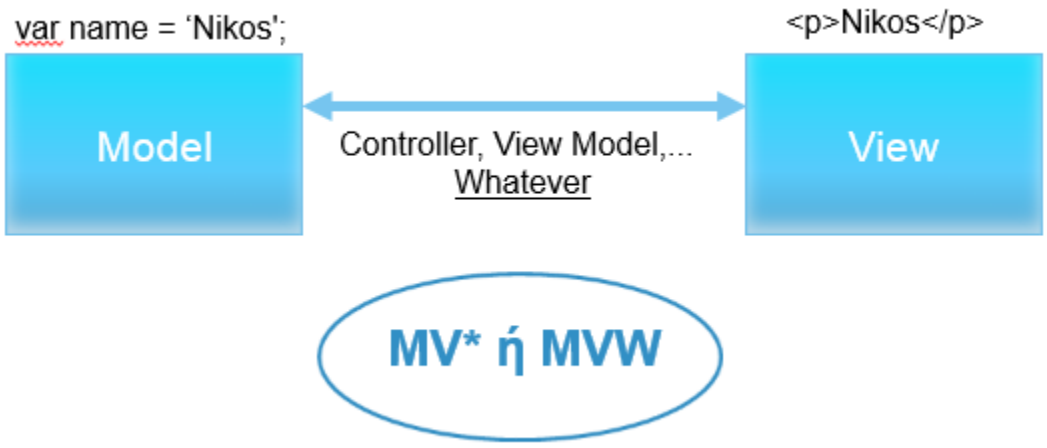
Τέλος, για να μπορέσουμε να το χρησιμοποιήσουμε, χρειάζεται να προσθέσουμε στο html την βιβλιοθήκη του AngularJS.

```
https://ajax.googleapis.com/ajax/libs/angularjs/1.5.5/angular.min.js
```

3.5.2 Model View Whatever

MV*

Το AngularJS υποστηρίζει διάφορες αρχιτεκτονικές. Για αυτόν τον λόγο, το AngularJS υποστηρίζει το MV* ή MVW όπου το W σημαίνει whatever. Και αυτό είναι που δίνει στον προγραμματιστή τη δυνατότητα να κάνει χρήση όποιας αρχιτεκτονικής πιστεύει ότι ταιριάζει καλύτερα στο εκάστοτε project.



*Εικόνα 3: Η Αρχιτεκτονική MV**

Όπως βλέπουμε στην παραπάνω εικόνα, το μοντέλο, είναι ο κώδικας (μεταβλητές, αντικείμενα JavaScript), και το view είναι η html. Αυτό λοιπόν που κάνει την διαφορά, είναι ο τρόπος που θα γίνει η σύνδεση μεταξύ αυτών των δυο. Και αυτό είναι το mv* ή MVW. Όπου το "*" ή το "w" αναπαριστά αυτή τη δυνατότητα.

Module, Apps and Controllers

Αφού έχουμε προσθέσει το AngularJS, μπορούμε να χρησιμοποιήσουμε όλα τα χαρακτηριστικά του. Το πρώτο πράγμα που φτιάχνουμε σε μια εφαρμογή βασισμένη στο AngularJS είναι το module. Φτιάχνουμε μια global μεταβλητή δίνοντας ένα όνομα το οποίο θα είναι και το όνομα της εφαρμογής μας. Η μεταβλητή αυτή θα είναι ίση με ένα αντικείμενο, το οποίο είναι το global angular object. Αυτό το αντικείμενο, έχει μία μέθοδο

την module. η οποία παίρνει δυο ορίσματα. Το πρώτο είναι ένα όνομα, που είναι το όνομα του που έχουμε δώσει στην εφαρμογή μας, και η δεύτερη παράμετρος είναι ένας πίνακας.

```
var myApp = angular.module('myApp', []);
```

Τέλος κάνουμε τη σύνδεση μεταξύ model και view.

```
<html lang="" ng-app="myApp">
```

Χρησιμοποιώντας από εδώ και πέρα μόνο το αντικείμενο myApp ή γενικά το αντικείμενο με το όνομα του module, μπορούμε να δημιουργήσουμε έναν controller με το εξής τρόπο:

```
myApp.controller('mainController', function(){
});
```

Η μέθοδος controller του AngularJS, δέχεται ένα όνομα, όπου είναι ουσιαστικά το όνομα του controller, και μια συνάρτηση.

Τέλος πρέπει να αντιστοιχίσουμε τον controller με κάποιο κομμάτι του view:

```
<div ng-controller="mainController"> <h1>Hello world!</h1></div>
```

3.5.3 Services and Dependency Injection

Dependency Injection

Το Dependency Injection είναι μια τεχνική σχεδιασμού λογισμικού, όπου ουσιαστικά σπάει η εξάρτηση για παράδειγμα ανάμεσα σε μια συνάρτηση και ένα αντικείμενο.

The Scope Service

Το Scope Service, είναι ένα από τα βασικότερα κομμάτια της σύνδεσης που γίνεται μεταξύ view και model. Το στοιχείο αυτό είναι ένα αντικείμενο που ονομάζεται scope ή scope service. Το οποίο αντικείμενο αυτό εμπλέκει το Dependency Injection. Το AngularJS δηλώνει το αντικείμενο αυτό στην αρχή, και το κάνει inject. Το αντικείμενο αυτό, μπορούμε αν θέλουμε να το χρησιμοποιήσουμε όπως και τα υπόλοιπα. Δηλαδή μπορούμε να προσθέσουμε ιδιότητες ή μεθόδους σε αυτό.

```
$scope.name=’Nikos’;
```

Dependency Injection in Action

Το AngularJS έχει μια μέθοδο που κάνει το injection, την injector. Η injector πάλι, έχει μια άλλη μέθοδο, την annotate.

```
console.log(angular.injector().annotate(test));
```

Μέσα στην annotate, περνάμε μια μεταβλητή, για παράδειγμα την test. Ουσιαστικά γίνεται parse το string που υπάρχει στη μεταβλητή αυτή σε έναν πίνακα που περιέχει τα ονόματα των παραμέτρων που δέχεται η συνάρτηση. Οπότε, το dependency injection είναι ουσιαστικά το πέρασμα ενός αντικειμένου που θέλουμε ως παράμετρο σε μια συνάρτηση, αυτόματα και χωρίς να χρειάζεται να το δημιουργήσουμε μέσα σε αυτό. Και αυτό είναι ένα πολύ δυνατό χαρακτηριστικό του AngularJS, γιατί 1ον όλο αυτό το κάνει από μόνο του

χωρίς να χρειάζεται να επέμβουμε εμείς, και 2ον, λόγο του πρώτου, απλουστεύεται ο κώδικας μας κάνοντας τον ποιο εύκολο στη συντήρηση και στο debigging.

Other Services

Αρχικά. Όλα τα services ξεκινάνε με το "$" μπροστά. Εκτός από το scope, υπάρχουν και άλλα services που χρησιμοποιούν dependency injection. Για να χρησιμοποιήσουμε κάποιο service, απλά πάμε και το προσθέτουμε στη συνάρτηση, δίνοντας μας τη δυνατότητα να το χρησιμοποιούμε μαζί με τις μεθόδους και τις ιδιότητές του.

```
myApp.controller('mainController',        function($scope,        $log){
```

Για να προσθέσουμε ένα service το οποίο όμως δεν παρέχετε από το AngularJS, πρέπει να προσθέσουμε το JavaScript αρχείο στο html, όπως ακριβώς με του AngularJS. Όμως πρέπει εκτός από την προσθήκη στο html και στη συνάρτηση, πρέπει να προστεθεί και στον πίνακα του module. Σε αυτόν τον πίνακα, μπαίνουν τα εξωτερικά modules, δηλαδή αυτά που εξαρτώνται από κάποιο άλλο αρχείο. Για παράδειγμα αν θέλαμε να προσθέσουμε το service ngResource θα έπρεπε πρώτα να το κάναμε κάπως έτσι:

```
var myApp = angular.module('myApp', ['ngResource']);
myApp.controller('mainController',    function($scope,    $resource){…}
```

Στον πίνακα αυτόν μπορούμε να προσθέσουμε όσα modules θέλουμε, και αφού κάναμε την προσθήκη αυτή, έχουμε πρόσβαση σε όλα τα χαρακτηριστικά του.

Dependency Injection & Minification

Γενικά, προτιμάμε να εισάγουμε το AngularJS χρησιμοποιώντας το συρρικνωμένο αρχείο. Από το αρχείο αυτό, έχουν αφαιρεθεί διάφορα που θα έχουν ως αποτέλεσμα τη μείωση του μεγέθους του αρχείου για πχ αφαίρεση των κενών και απλοποίηση μεταβλητών. Η διαδικασία λοιπόν για να γίνει ένα αρχείο JavaScript συρρικνωμένο ονομάζεται minification. Έχουμε 2 τρόπους. Ο ένας είναι χρησιμοποιώντας κάποιον online minifier

και ο 2ος είναι χρησιμοποιώντας κάποιο πρόσθετο σε κάποιον editor. Στο google υπάρχουν διάφορες σελίδες που κάνουν minify JavaScript αλλά και css. Όμως παρόλο που αυτός ο τρόπος λειτουργεί, στο AngularJS πρέπει να κάνουμε και κάτι ακόμα, γιατί οι υπηρεσίες που θέλουμε να χρησιμοποιήσουμε, τις βάζουμε μέσα στη συνάρτηση ως strings. Κάνοντας όμως minify τον controller, γίνεται αντικατάσταση του Scope για παράδειγμα, και όταν πάει να γίνει το injection, δεν βρίσκει τις νέες μεταβλητές και παράγεται το σφάλμα. Οπότε για να μπορέσουμε να κάνουμε με επιτυχία minify θα πρέπει να κάνουμε μια αλλαγή. Αντί για την συνάρτηση, μπορούμε να περνάμε στον controller έναν πίνακα, καθώς, σε έναν πίνακα μπορούμε να περάσουμε διαφορετικούς τύπους δεδομένων, ακόμα και μια συνάρτηση.

```
myApp.controller('mainController', ['$scope','$log',function
($log,$scope) {…}]);
```

Με αυτόν τον τρόπο, καταφέρνουμε να περάσουμε τα services χωρίς να γίνει αντικατάσταση τους με ποιο απλή μεταβλητές, επειδή κατά το minification δεν γίνεται αντικατάσταση των strings. Και αυτό, δουλεύει γιατί ο injector βλέπει ότι αντί για την συνάρτηση υπάρχει πίνακας. Τα στοιχεία αυτά τα κάνει αντιστοιχία με τα ορίσματα τις συνάρτησης. Δηλαδή το πρώτο string με το πρώτο όρισμα, και το δεύτερο με το 2ο όρισμα. Το μόνο πράγμα που πρέπει να προσέχουμε εδώ, είναι η σειρά των strings να είναι ίδια με την σειρά των παραμέτρων της συνάρτησης. Και αυτό γιατί πλέον όπως είπαμε γίνεται αντιστοίχιση.

3.5.4 Data Binding and Directives

Scope and interpolation

Το interpolation είναι η τεχνική που δημιουργούμε ένα string, ενώνοντας άλλα strings. Ουσιαστικά είναι αυτό που γίνεται στη JavaScript με το “+”.

```
$scope.name = ‘Nick’;
```

Για να εμφανίσουμε τη μεταβλητή στο html, τη βάζουμε μέσα διπλά άγκιστρα.

```
<h1>Hello {{ name + '. How are you?'}}</h1>
```

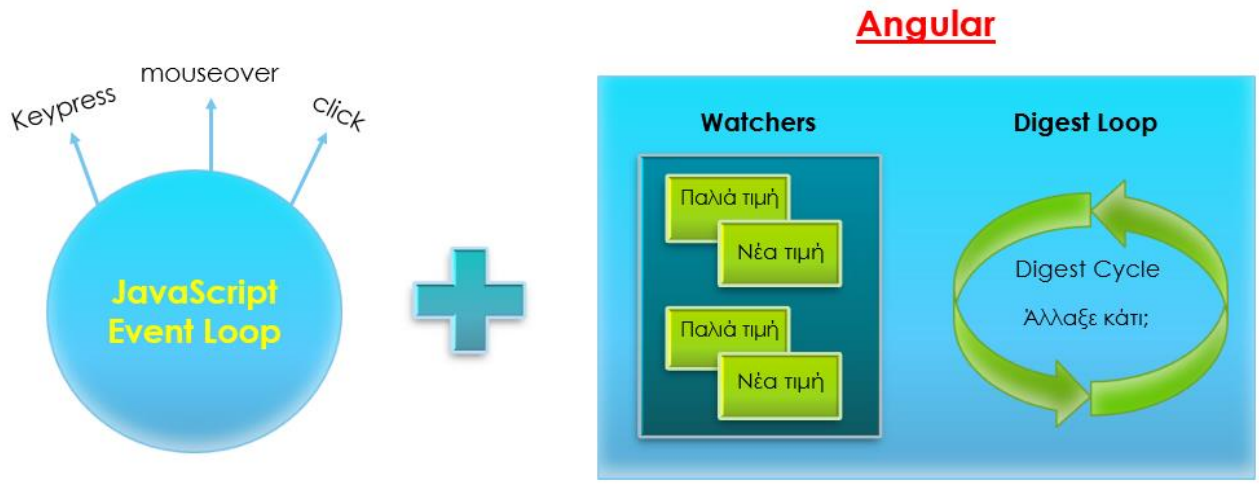
Αυτό που κάνει λοιπόν το AngularJS, είναι να κοιτάει στο scope αν υπάρχει αυτή η μεταβλητή.

Directives and two-way data binding

Τα directives είναι ουσιαστικά οι εντολές που κάνουν το AngularJS να χειρίζεται το DOM. Τέτοιες εντολές μπορεί να είναι για παράδειγμα, να προστεθεί μια κλάση. Ένα τέτοιο στο AngularJS είναι το ng-model, το οποίο γίνεται ίσο με μια μεταβλητή και ουσιαστικά ενώνεται το view με το model. Και αν έχουμε παραδείγματος χάριν ένα input, τότε ενώνεται και με αυτό με αποτέλεσμα να γίνεται two-way binding.

Watchers and the digest loop

Η JavaScript, έχει διάφορα events. Είχαμε δει επίσης για παράδειγμα το πώς μπορούμε να χρησιμοποιήσουμε την addeventlistener. Όλη αυτή η διαδικασία για να γίνει η παρακολούθηση των αλλαγών του χρήστη, ονομάζεται event loop.



Εικόνα 4: Event Loop, Watchers και Digest Loop

Έχουμε λοιπόν το event loop που περιέχει διάφορα events. Το AngularJS τώρα, όταν προσθέτουμε μια μεταβλητή ή κάποιο binding, προσθέτει ένα watcher. Η δουλειά ενός watcher είναι να παρακολουθεί την παλιά και την νέα τιμή. Αυτό γίνεται κάθε φορά για παράδειγμα όταν πάμε να γράψουμε σε ένα πεδίο. Πατώντας ένα πλήκτρο, αλλάζει η τιμή ενώ αν ξαναπατήσουμε τότε ξανά αλλάζει. Η διαδικασία που γίνεται λοιπόν για να καταλάβει αν έχει γίνει κάποια αλλαγή ονομάζεται digest loop. Με απλά λόγια, κοιτάει κάθε μεταβλητή μέσα στη watch list και συγκρίνει την παλιά τιμή με την νέα. Αν κάποια από τις 2 τιμές έχει αλλάξει, τότε την ενημερώνει παντού. Ουσιαστικά λοιπόν το AngularJS κοιτάει τα φυσικά events σε μια σελίδα. Έχει στη watch list οτιδήποτε μπορεί να αλλάξει και όταν ενεργοποιηθεί κάποιο event, τότε ψάχνει οτιδήποτε μπορεί να έχει αλλάξει. Και αν έχει αλλάξει τότε ενημερώνει το dom, και το ανάποδο.

Common Directives

Ας δούμε μερικά ακόμα αρκετά χρήσιμα directives.

ng-if

Το ng-if δέχεται JavaScript μέσα, που μπορεί να είναι σύγκριση ή κλήση μια συνάρτησης. Γενικά κάτι που μπορεί να επιστρέψει true/false. Ουσιαστικά το directive αυτό λέει στο dom τι να κάνει κάτω από ορισμένες συνθήκες.

ng-show

Αυτό λειτουργεί παρόμοια. Αν ισχύει η έκφραση, είναι αληθής δηλαδή, τότε εμφανίζεται το στοιχείο στο οποίο είναι μέσα.

ng-hide

Θα εμφανίζεται το περιεχόμενο μόνο όταν ισχύει η συνθήκη.

ng-class

Το ng-class είναι ένα αντικείμενο JavaScript. Το οποίο μοιάζει πολύ με JSON. Μέσα στις αγκύλες, παίρνει το όνομα μια κλάσεις μέσα σε αυτάκια, και μετά άνω κάτω τελεία και την τιμή η οποία θα το προσθέτει.

```
ng-class="{'alert-warning':username.length < characters}"
```

ng-repeat

Το ng-repeat αυτό που κάνει είναι να αναπαράγει ένα στοιχείο της html για κάθε αντικείμενο που έχει ο πίνακάς μας, ή αν υπήρχαν σε κάποια βάση, στη βάση.

```
<li ng-repeat="rule in rules">
```

Αυτός ο τρόπος χρησιμοποιείτε κυρίως όταν θέλουμε να εμφανίσουμε σε μια εφαρμογή στοιχεία από μια βάση δεδομένων.

ng-click

Η ng-click μέσα στα αυτάκια παίρνει μέσα μια συνάρτηση που έχουμε στο scope. Και η οποία θα εκτελεστεί όταν πατήσουμε πάνω στο κουμπί.

ng-cloak

Αυτό το μόνο που κάνει είναι να κρύβει το στοιχείο αυτό από το dom, μέχρι το AngularJS να φορτωθεί. Όταν θέλουμε να εμφανίσουμε μια μεταβλητή από τον controller στο view, χρησιμοποιούμε τα διπλά άγκιστρα. Αν όμως ένας χρήστης έχει αργή σύνδεση στο διαδίκτυο, ή αργό υπολογιστή, τότε μέχρι να φορτώσει το AngularJS, ο χρήστης θα δει τις αγκύλες με την μεταβλητή μέσα έως ότου ολοκληρωθεί το φόρτωμα της σελίδας.

JS: The XMLHTTPRequest Object

Υπάρχει ένα στοιχείο που ουσιαστικά είναι ο τρόπος που όλες οι σύγχρονες διαδικτυακές εφαρμογές λειτουργούν. Αυτό το στοιχείο ονομάζεται XMLHTTPRequest object και το χρησιμοποιούν όλα τα σύγχρονα frameworks όπως και το AngularJS. Το αντικείμενο αυτό υποστηρίζεται από όλους τους browsers με τον ίδιο τρόπο, καθώς είναι κομμάτι αυτών και της JavaScript. Και ο σκοπός του είναι να κάνει internet requests. Επειδή είναι λίγο περίεργη η σύνταξή του, το jquery, το AngularJS αλλά και τα άλλα frameworks χρησιμοποιούν δικό τους wrapper γύρο από το αντικείμενο αυτό.

```
var rulesrequest = new XMLHTTPRequest();
rulesrequest.onreadystatechange = function(){
  if(rulesrequest.readystate == 4 && rulesrequest.status == 200){
      result = JSON.parse(rulesrequest.responseText);
      }
}
```

Το πρώτο event που χρησιμοποιείτε είναι το onreadystatechange, το οποίο μας δείχνει ότι κάτι έχει γίνει ή αν έγινε κάποιο request και τελείωσε. Στην if κοιτάμε αν δεν υπάρχει κάποιο πρόβλημα και το status code είναι 200, το οποίο σημαίνει πως αυτό που ζητάμε στο διαδίκτυο βρέθηκε. Μετά, χρησιμοποιούμε το responsetext, το οποίο ουσιαστικά είναι το περιεχόμενο που θα πάρουμε από τη βάση δεδομένων. Γίνεται parse

ως json με αποτέλεσμα να έχουμε έναν πίνακα με το περιεχόμενο μας που έρχεται από τη βάση. Τελευταίο βήμα είναι να διαβάσουμε το json από κάποιον διακομιστή.

```
rulesrequest.open('GET','some_api.php',true);
rulesrequest.send();
```

Για να το κάνουμε αυτό, χρησιμοποιούμε πάλι το rulesrequest και την ιδιότητα open και βάζουμε εκεί το url που βρίσκεται το php αρχείο του api μας. Τέλος επειδή όλα αυτά δεν γίνονται μέσα στο AngularJS, πρέπει να χρησιμοποιηθεί πρώτα η μέθοδος apply του AngularJS ώστε να ξεκινήσει το digest loop.

External Data and $http

Ας δούμε τώρα πως μπορούμε να κάνουμε το ίδιο, χρησιμοποιώντας την προσέγγιση του AngularJS. Για να γίνει αυτό, θα χρησιμοποιήσουμε το service $http. Αφότου το έχουμε εισάγει στον controller, μπορούμε να πάρουμε το json από το api με τον εξής τρόπο:

```
$http.get('api.php');
```

Όπως παρατηρούμε, ο κώδικας έχει ήδη απλουστευθεί. Αμέσως μετά, θα χρησιμοποιώντας την μέθοδο success, η οποία δέχεται μια συνάρτηση που θα εκτελεστεί αν πάρουμε με επιτυχία τα δεδομένα από τον διακομιστή. Η συνάρτηση δέχεται μια παράμετρο η οποία θα είναι τα δεδομένα που θα πάρουμε από το api.

```
success(function(result){ … })
```

Μετά, χρησιμοποιούμε την μέθοδο error για την περίπτωση που προκύψει κάποιο σφάλμα. Μέσα σε αυτή μπαίνει μια συνάρτηση η οποία δέχεται ως ορίσματα μια μεταβλητή για τα δεδομένα που μπορεί να επιστραφούν και μια για το http status.

```
.error(function(data,status){ … });
```

Για να στείλουμε δεδομένα στον διακομιστή, μπορούμε να χρησιμοποιήσουμε την μέθοδο post του αντικειμένου $http, χρησιμοποιώντας και πάλι τις μεθόδους success και error.

```
$http.post('add_api.php',{…})
```

3.5.5 Custom Services & Directives

Creating a service

Για να δημιουργήσουμε δικά μας services, το πρώτο που πρέπει να κάνουμε είναι να χρησιμοποιήσουμε τη global μεταβλητή που έχουμε δημιουργήσει και να χρησιμοποιήσουμε την μέθοδο service. Η μέθοδος αυτή μέσα της δέχεται ένα όνομα όπου είναι το όνομα του service και μια συνάρτηση η οποία περιέχει τον κώδικα που θα εκτελείτε όταν το χρησιμοποιούμε.

```
myApp.service('name', function(){
      //κώδικας
});
```

Για να έχουμε πρόσβαση σε κάποιο service μέσα από κάποιον controller, πρέπει πρώτα να προστεθεί στον controller όπως και τα υπόλοιπα services, χωρίς όμως το δολάριο μπροστά. Το πλεονέκτημα αυτού του τρόπου είναι η δυνατότητα να μοιράζονται στοιχεία ανάμεσα στις σελίδες ενός spa, αφού δεν γίνεται ανανέωση κατά την περιήγησή σε αυτές.

Reusable components

Με τον όρο reusable components, αναφερόμαστε στη λύση ενός από τα προβλήματα της html, που είναι ότι δεν υπάρχει ένας γρήγορος τρόπος να δημιουργούμε δικά μας tags. Για παράδειγμα σε μια σελίδα που εμφανίζει τα αποτελέσματα μιας αναζήτησης μπορούμε να έχουμε τον εξής κώδικα:

```
<searchResult></searchResult>
ή
<div searchResult></div>
```

Variable names & Normalization

Με τον όρο normalization ή στα ελληνικά κανονικοποίηση εννοούμε να κάνουμε κάτι, για παράδειγμα μια μεταβλητή, να ακολουθεί κάποια πρότυπα.

```
<search-result result-link-href = "#"></search-result>
```

Όλο αυτό ακολουθεί ένα πρότυπο. Η παύλα μεταξύ των λέξεων καθώς και ότι είναι όλα γραμμένα με μικρά γράμματα. Το AngularJS χρησιμοποιεί αυτή τη τεχνικής, επειδή στη JavaScript η παύλα είναι για την αφαίρεση. Αυτό που κάνει είναι να μετατρέπει τα ονόματα σε μια συγκεκριμένη μορφή η οποία ονομάζεται camelcase. Το camelcase είναι όταν το πρώτο γράμμα μιας λέξης εκτός της πρώτης γράφεται με κεφαλαίο και δεν γίνετε χρήση του κενού καθόλου. Άρα ο παραπάνω κώδικας θα γίνει κάπως έτσι:

```
<search-result resultLinkHref = "#"></search-result>
```

Creating a directive

Για να δημιουργήσουμε δικά μας services, το πρώτο που πρέπει να κάνουμε είναι να χρησιμοποιήσουμε τη global μεταβλητή που έχουμε δημιουργήσει και να χρησιμοποιήσουμε την μέθοδο directive. Η μέθοδος αυτή μέσα της δέχεται ένα όνομα όπου είναι το όνομα του directive και μια συνάρτηση η οποία περιέχει τον κώδικα που θα εκτελείτε όταν το χρησιμοποιούμε.

```
      myApp.directive("searchResult", function(){
            template: ‘κώδικας html’
});
```

Το όνομα του directive γράφεται με camelcase ενώ μέσα στη συνάρτηση, χρησιμοποιώντας την ιδιότητα template ή templateUrl, ορίζουμε τι html θα επιστρέφει όταν το χρησιμοποιήσουμε, εισάγοντας το ως string μέσα σε αυτάκια. Λόγο της κανονικοποίησης, στο html το directive θα πρέπει να γραφτεί με την εξής μορφή. Όπου υπάρχει κεφαλαίο γράμμα μπαίνει μπροστά παύλα ενώ όλα τα κεφαλαία γράμματα γίνονται μικρά. Με αυτόν τον τρόπο, το AngularJS θα ψάξει για ένα directive με το ίδιο όνομα αλλά γραμμένο με camelcase. Τέλος υπάρχουν τέσσερις διαφορετικοί τρόποι για την εμφάνιση του directive, με τους δύο τελευταίους να χρειάζεται να βάλουμε στη ιδιότητα restrict το c ή το M.

- `<search-result></search-result>`
- `<div search-result></div>`
- `<div class="search-result">`
- `<!-- directive: search-result-->`

Σε περίπτωση που χρειαζόμαστε να αναπαράγουμε αυτό το στοιχείο, μπορούμε να το κάνουμε χρησιμοποιώντας το ng-repeat.

```
<search-result ng-repeat="person in people"></search-result>
```

3.5.6 Bonus

Nasted Controllers, clean code & Controller as

Από την έκδοση 1.2 του AngularJS, έχει εισαχθεί μια νέα σύνταξη, η controller as. Η οποία είναι εναλλακτική του $scope.

```
<div ng-controller="parent2Controller as parent2">
      {{parent2.message}}
      <div ng-controller="child2Controller as child2">
          {{parent2.message}}
          <br>
          {{child2.message}}
          <br>
          <input type="text" ng-model="parent2.message">
      </div>
</div>
```

Το μόνο πραγματικό πλεονέκτημα που παρέχει ο τρόπος αυτός είναι ότι κάνει τον κώδικα ποιο καθαρό και ποιο κατανοητό.

ES6, AtScipt, and more

Η έκδοση 2 του AngularJS αυτή τη στιγμή βρίσκετε σε beta κατάσταση ενώ θα κυκλοφορήσει κάποια στιγμή, μέσα στο 2016. Το AngularJS 2, στοχεύει στους σύγχρονους browsers και στις mobile συσκευές. Όμως επειδή είναι ακόμα νωρίς και τα χαρακτηριστικά του δεν έχουν οριστικοποιηθεί, ας πούμε λίγα λόγια για το ECMAScript 6 και για το ATSctipt.

Το ECMAScript 6 είναι ουσιαστικά η νέα έκδοση της JavaScript. Το AngularJS 2 λοιπόν έχει στραφεί προς το ES6, καθώς προσφέρει νέες δυνατότητες και χαρακτηριστικά. Για παράδειγμα, προσθέτει κανονικές κλάσεις, στο μοτίβο του δομημένου προγραμματισμού.

Το ATScipt είναι μια transpiled γλώσσα προγραμματισμού που πρόκειται να έρθει μαζί με το AngularJS 2. Η έννοια transpiled σημαίνει τη μετατροπή του πηγαίου κώδικα από μια γλώσσα σε μια άλλη. Αυτό σημαίνει ότι γράφοντας ATScipt αυτό θα γίνεται

JavaScript που θα τρέχει κανονικά στον browser. Το ATScipt θα είναι κατά κάποιον τρόπο ένα πρόσθετο σετ σε μια άλλη γλώσσα, την TypeScript.

ΠΑΡΑΡΤΗΜΑ 1

Regular Expressions

Modifiers

i	ταίριασμα πεζών κεφαλαίων
g	συνεχίζει μετά το πρώτο ταίριασμα
m	ταίριασμα μεταξύ πολλών γραμμών

Πίνακας 3: Modifiers

Αναζήτηση εύρους χαρακτήρων

[0-9]	Αναζήτηση όλων των αριθμών μέσα στις αγκύλες
[abc]	Αναζήτηση όλων των χαρακτήρων μέσα στις αγκύλες
[^0-9]	Αναζήτηση όλων των αριθμών έξω από τις αγκύλες
[^abc]	Αναζήτηση όλων των χαρακτήρων έξω από τις αγκύλες
[x|y]	Αναζήτηση τουλάχιστον ενός από τους δύο

Πίνακας 4: Παράμετροι για την αναζήτηση εύρους χαρακτήρων

Ειδικοί χαρακτήρες

.	Βρίσκει ένα χαρακτήρα, εκτός του newline και line terminator
\0	Βρίσκει τον χαρακτήρα null
\b	Βρίσκει αντιστοιχία στην αρχή και το τέλος μιας λέξης
\B	Βρίσκει αντιστοιχία εκτός αρχής και τέλους μιας λέξης
\d	Βρίσκει έναν αριθμό
\D	Βρίσκει ένα χαρακτήρα εκτός από αριθμό
\f	Βρίσκει έναν χαρακτήρα page break
\n	Βρίσκει έναν χαρακτήρα αλλαγής σελίδας
\r	Βρίσκει έναν χαρακτήρα επιστροφής
\s	Βρίσκει έναν whitespace χαρακτήρα
\S	Βρίσκει έναν μη whitespace χαρακτήρα
\t	Βρίσκει έναν χαρακτήρα tab
\uxxx	Βρίσκει έναν Unicode χαρακτήρα (δεκαεξαδικό)
\v	Βρίσκει έναν χαρακτήρα οριζόντιου tab
\w	Βρίσκει ένα χαρακτήρα γράμμα
\W	Βρίσκει ένα χαρακτήρα εκτός από γράμμα

\xdd	Βρίσκει ένα χαρακτήρα ορισμένο στο δεκαεξαδικό
\xxx	Βρίσκει ένα χαρακτήρα ορισμένο στο οκταδικό
n+	Ταιριάζει ένα αλφαριθμητικό που περιέχει τουλάχιστον ένα n
n*	Ταιριάζει ένα αλφαριθμητικό που περιέχει κανένα ή περισσότερα n
n?	Ταιριάζει ένα αλφαριθμητικό που περιέχει κανένα ή ένα n
n{X}	Ταιριάζει ένα αλφαριθμητικό που περιέχει μια ακολουθία από X n
n{X,Y}	Ταιριάζει ένα αλφαριθμητικό που περιέχει μια ακολουθία από X έως Y n
n{X,}	Ταιριάζει ένα αλφαριθμητικό που περιέχει μια ακολουθία από τουλάχιστον X n
n$	Ταιριάζει ένα αλφαριθμητικό που περιέχει ένα n στο τέλος
^n	Ταιριάζει ένα αλφαριθμητικό που περιέχει ένα n στην αρχή
?=n	Ταιριάζει ένα αλφαριθμητικό που ακολουθείτε από μια αλφαριθμητικό n
?! n	Ταιριάζει ένα αλφαριθμητικό που δεν ακολουθείτε από μια αλφαριθμητικό n

Πίνακας 5: Ειδικοί παράμετροι για την αναζήτηση

Ιδιότητες

.global	Ελέγχει αν έχει οριστεί ο modifier g
.ignoreCase	Ελέγχει αν έχει οριστεί ο modifier i
.lastIndex	Ορίζει το δείκτη για το ξεκίνημα του επόμενου ταιριάσματος
.multiline	Ελέγχει αν έχει οριστεί ο modifier m
.source	Επιστρέφει το κείμενο του μοτίβου

Πίνακας 6: Ιδιότητες

Μέθοδοι

exec()	Ελέγχει για ένα ταίριασμα σε ένα string και επιστρέφει το πρώτο
test()	Ελέγχει για ένα ταίριασμα σε ένα string και επιστρέφει true ή false
toString()	Επιστρέφει την τιμή της έκφρασης

Πίνακας 7: Μέθοδοι

ΠΑΡΑΡΤΗΜΑ 2

DOM

Document Methods

- document.write(text)
- document.getElementById(id)
- document.getElementsByTagName(name)
- document.getElementsByClassName(name)
- element.innerHTML = new html content
- element.attribute = new value
- element.setAttribute(attribute, value)
- element.style.property = new style
- document.createElement(element)
- document.removeChild(element)
- document.appendChild(element)
- document.replaceChild(element)
- Element.hasChildNodes()
- Element.replaceChild()

Document Properties

- document.anchors
- document.baseURI
- document.body
- document.cookie
- document.doctype
- document.documentElement
- document.documentMode
- document.documentURI
- document.domain
- document.embeds
- document.forms

- document.head
- document.images
- document.implementation
- document.inputEncoding
- document.lastModified
- document.links
- document.readyState
- document.referrer
- document.scripts
- document.strictErrorChecking
- document.title
- document.URL

ΠΑΡΑΡΤΗΜΑ 3

Events

Mouse Events

- onclick
- oncontextmenu
- ondblclick
- onmousedown
- onmouseup
- onmouseenter
- onmouseleave
- onmousemove
- onmouseover
- onmouseout

Keyboard Events

- onkeydown
- onkeypress
- onkeyup

Object Events

- onabort
- onbeforeunload
- onerror
- onhashchange
- onload
- onpageshow
- onpagehide
- onresize
- onscroll
- onunload

Form Events

- onblur
- onchange
- onfocus
- onfocusin
- onfocusout
- oninput
- oninvalid
- onreset
- onsearch
- onselect
- onsubmit

Drag Events

- ondrag
- ondragend
- ondragcenter
- ondragleave
- ondragover
- ondragstart
- ondrop

Clipboard Events

- oncopy
- oncut
- onpaste

Print Events

- onafterprint
- onbeforeprint

Media Events

- onabord
- oncanplay
- oncanplaythrough
- ondurationchange
- onemptied
- onended
- onerror
- onloaddata
- onloadedmetadata
- onloadedmetadata
- onloadstart
- onpause
- onplay
- onplaying
- onprogress
- onratechange
- onseeked
- onseeking
- oninstall
- onsuspend
- ontimeupdate
- onvolumechange
- onwaiting

Animation Events

- animationend
- animationiteration
- animationstart

Transition Events

- transitionend

Διακομιστή Sent Events

- onerror
- onmessage
- Onopen

Other Events

- onmessage
- ononline
- onoffline
- onpopstate
- onshow
- onstarage
- ontoggle
- onwheel

Touch Events

- ontouchcancel
- ontouchend
- ontouchmove
- ontouchstart

ΠΑΡΑΡΤΗΜΑ 4

Παραδείγματα

Παραδείγματα Ενότητας 3.4.2

If / else if

```
var age = prompt("Δώστε την ηλικία για τον υπολογισμό του εισιτηρίου:");
var timh = 12;
var telikh_timh = 0;
var ekptwsh = 0;

if(age <= 6)
{
    telikh_timh = 0;
    document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh + "€");
}
else if(age > 6 && age <= 12)
{
    ekptwsh = (50/100)*timh;
    telikh_timh = timh-ekptwsh;
    document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh + "€");
}
else if(age > 12 && age <= 18)
{
    ekptwsh = (25/100)*timh;
    telikh_timh = timh-ekptwsh;
    document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh + "€");
}
else if(age > 60)
{
    ekptwsh = (50/100)*timh;
    telikh_timh = timh-ekptwsh;
    document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh + "€");
}
else
{
```

```
    document.write("Το κόστος του εισιτηρίου είναι: " + timh + "€");
}
```

Switch

```
var age = prompt("Δώστε την ηλικία για τον υπολογισμό του εισιτηρίου:");
var timh = 12; var telikh_timh = 0; var ekptwsh = 0;

switch(true)
{
    case age <= 6:
        {  telikh_timh = 0;
            document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh +
"€");
            break; }
    case age > 6 && age <= 12:
        {  ekptwsh = (50/100)*timh;
            telikh_timh = timh-ekptwsh;
            document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh +
"€");
            break; }
    case age > 12 && age <= 18:
        {  ekptwsh = (25/100)*timh;
            telikh_timh = timh-ekptwsh;
            document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh +
"€");
            break; }
    case age > 60:
        {   ekptwsh = (50/100)*timh;
            telikh_timh = timh-ekptwsh;
            document.write("Το κόστος του εισιτηρίου είναι: " + telikh_timh +
"€");
            break; }
    default:
        { document.write("Το κόστος του εισιτηρίου είναι: " + timh + "€");  }
}
```

for / for in

```
var persons = prompt("Δώστε αριθμό ατόμων");
var age = 0;var timh = 12;var telikh_timh = 0;
var ekptwsh = 0;var sum = 0;

for(var i = 0; i < persons;i++)
{
    age = prompt("Δώστε την ηλικία για τον υπολογισμό του εισιτηρίου:");
    switch(true)
    {
        case age <= 6:
            {
                telikh_timh = 0;
                sum += telikh_timh;
                document.write("Το κόστος του εισιτηρίου είναι: " +
telikh_timh + "€");
                break;
            }
        case age > 6 && age <= 12:
            {
                ekptwsh = (50/100)*timh;
                telikh_timh = timh-ekptwsh;
                sum += telikh_timh;
                document.write("Το κόστος του εισιτηρίου είναι: " +
telikh_timh + "€");
                break;
            }
        case age > 12 && age <= 18:
            {
                ekptwsh = (25/100)*timh;
                telikh_timh = timh-ekptwsh;
                sum += telikh_timh;
                document.write("Το κόστος του εισιτηρίου είναι: " +
telikh_timh + "€");
                break;
            }
        case age > 60:
            {
                ekptwsh = (50/100)*timh;
                telikh_timh = timh-ekptwsh;
                sum += telikh_timh;
                document.write("Το κόστος του εισιτηρίου είναι: " +
telikh_timh + "€");
                break;}
        default:
            {  document.write("Το κόστος του εισιτηρίου είναι: " + timh +
"€");
                sum += timh;}
```

```
    }
    document.write("<br>");}
document.write("Το συνολικό κόστος είναι: " + sum + "€");
```

while / do while

```
var value = 10;
var i = 0;
do
{
    value--;i++;
}
while(value > 0)
document.write("Έγιναν " + i + " Επαναλήψεις");
```

Παραδείγματα Ενότητας 3.4.3

Πράξεις με χρήση των συναρτήσεων 1

```
function add(a,b)
{
	return a+b;
}
var sum = add(5,6);
document.write(sum);
```

Πράξεις με χρήση των συναρτήσεων 2

```
var x = function(a,b)
{
	Return a+b;
}
var sum = x(4,3);
document.write(sum);
```

Πράξεις με χρήση των συναρτήσεων 3

```
var myFunction = new Function(“a”, “b”, “return a+b”);
var sum = myFunction(5,3);
document.write(sum);
```

Κλήση συνάρτησης με λιγότερες παραμέτρους

```
var sum=0;
function add(a, b){
	if(a===undefined){
		a=0;
	}
	else if(b===0){
		b=0;
	}
return a + b;
}
sum = add(5);
document.write(sum);
```

Κλήση συνάρτησης με περισσότερες παραμέτρους

```
var sum=0; var i=0;
function add(a, b){
      if(a===undefined){
            a=0;
      }
      else if(b===0){
            b=0;
      }
      For(i=0;i<arguments.length;i++)
      {
            Sum+=arguments[i];
}
return sum;
}
var sum2 = add(5,6,7,8);
document.write(sum2);
```

Υπολογισμός Υποτείνουσας με χρήση εμφωλευμένων συναρτήσεων α΄ τρόπος

```
function hypotenuse()
{
      var side1 = 3;
      var side2 = 4;

      function square_side1()
      {
            return side1 * side1;
      }
      function square_side2()
      {
            return side2 * side2;
      }
      return Math.sqrt(square_side1() + square_side2());
}

document.write(hypotenuse());
```

Υπολογισμός Υποτείνουσας με χρήση εμφωλευμένων συναρτήσεων β΄ τρόπος

```
function hypotenuse(a,b)
{
      function square(x)
      {
            return x * x
      }

      return Math.sqrt(square(a) + square(b));
}
document.write(hypotenuse(3,4));
```

Υπολογισμός παραγοντικού με Αναδρομή

```
function factorial(n) {
    if (n < 0) {
    console.log("Δεν επιτρέπονται αρνητικές τιμές.");
    return;
  }
  if (n === 0) {
    return 1;
  }
  return n * factorial(n-1);
}
factorial(6);
```

Παραδείγματα Ενότητας 3.4.4

Εμφάνιση χαρακτηριστικών ενός σκύλου με χρήση ενός αντικειμένου 1ος τρόπος

```
var dog = {
      name: “max”,
      breed: “setter”,
      color: “black”,
      weight: 26,
      info: function(){
            return this.name + “ is a ” + this.color + “ ” + this.breed;
      }
}
document.write(dog.info());
```

Εμφάνιση χαρακτηριστικών ενός σκύλου με χρήση ενός αντικειμένου 2ος τρόπος

```
var dog = new Object();
dog.name = “max”;
dog.breed = “setter”;
dog.color = “black”;
dog.weight = 26;
document.write(dog.name + “ ” + dog.breed + “ ” + dog.color);
```

Εμφάνιση χαρακτηριστικών ενός σκύλου με χρήση ενός αντικειμένου 3ος τρόπος

```
function dog(name, breed, color, weight)
{
      this.name = name;
      this.breed = breed;
      this.color = color;
      this.weight = weight;
}
var dog1 = new dog(“max”, “setter”, “black”, 26);
var dog2 = new dog(“regina”, “spaniel”, “brown”, 24);
```

Παράδειγμα με τη χρήση της split σε αλφαριθμητικά

```
var string1 = “th,is i,s a \”st,ring\” “;
var string2 = ‘and this is a 2nd “string”’;
var string3 = ‘and |this is a| 3|rd string’;

var komma = string1.split(“,”);
var space = string2.split(“ ”);
var pipe = string3.split(“|”);

document.write(“<br><br>” + komma + “<br>” + space + “<br>” + pipe);
```

Παράδειγμα με τη χρήση Regular Expression

```
var dateTime = /\d\d-\d\d-\d\d\d\d \d\d:\d\d/;
document.write(dateTime.test("10-05-2016 15:23"));
document.write("<br>");
document.write(dateTime.test("30-jan-2016 15:23"));
```

Παραδείγματα Ενότητας 3.4.5

Παράδειγμα χρησιμοποιώντας τις δυνατότητων του DOM σε λίστα HTML

HTML

```
<!DOCTYPE html>
<html lang="">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>body{background-color:#e6e6e6;}.color{color:deepskyblue;}</style>
</head>
<body>
    <h1 id="title" onclick="click1()">DOM - Events</h1>
    <br>
    <p id="par">Λίστα</p>
    <ul id="list1">
            <li>Πορτοκάλι</li>
            <li>Μήλο</li>
            <li>Αχλάδι</li>
            <li>Ανανάς</li>
        </ul>
    <p id="par2"> </p>
        <ul id="list2">
            <li>Λάχανο</li>
            <li>Μαρούλι</li>
        </ul>
    <br><br>

    <div style="border:solid; border-color:gray; width:50%"
onmousedown="mousedown(this)" onmouseup="mouseup(this)">
        <span id="mouse_event"> </span>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

JavaScript

```
var elem = document.getElementById('par');
document.getElementById('par2').innerHTML = elem.innerHTML + ' Λαχανικών';
elem.innerHTML = 'Λίστα Φρούτων';

document.getElementsByTagName('p')[0].setAttribute('class','color');
```

```
var node = document.createElement('li');
var textnode = document.createTextNode('Μπρόκολο');
node.appendChild(textnode);
document.getElementById('list2').appendChild(node);

var list = document.getElementById('list2');
list.removeChild(list.childNodes[1]);

document.body.style.backgroundColor = 'lightgreen';

function click1()
{
    alert(document.getElementById('title').innerHTML);
}

function mouseover()
{
    document.getElementById('mouse_event').innerHTML = 'Ο δείκτης είναι μέσα
στη λίστα';
}

function mouseout()
{
    document.getElementById('mouse_event').innerHTML = 'Ο δείκτης είναι έξω
απο τη λίστα';
}

function mousedown(obj)
{
    obj.style.backgroundColor = 'gray';
}
function mouseup(obj)
{
    obj.style.backgroundColor = '';
}

function hide()
{
    var list = document.getElementById('list2');
    if(list2.offsetParent === null){
        list2.style.display = 'block';
    }
    else{
        list2.style.display = 'none';
    }
}

document.getElementById('par2').addEventListener('mouseover',mouseover);
document.getElementById('list1').addEventListener('mouseout',mouseout);
```

```
document.getElementById('list1').addEventListener('dblclick',hide);
document.getElementById('list1').removeEventListener('dblclick',hide);

window.addEventListener('load',click1);
```

Έλεγχος εγκυρότητας σε φόρμα εγγραφής χρήστη

HTML

```
<!DOCTYPE html>
<html lang="">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <style>body{background-
color:#e6e6e6;}.color{color:deepskyblue;}label{margin-right: 3%;}body {
        background-color: #eee;
        }
        *[role="form"] {
            max-width: 530px;
            padding: 15px;
            margin: 0 auto;
            background-color: #fff;
            border-radius: 0.3em;
        }
        *[role="form"] h2 {
            margin-left: 5em;
            margin-bottom: 1em;
        }</style>
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
integrity="sha384-
1q8mTJOASx8j1Au+a5WDVnPi2lkFfwwEAa8hDDdjZlpLegxhjVME1fgjWPGmkzs7"
crossorigin="anonymous">
</head>
<body>
    <div class="container">
        <form class="form-horizontal" role="form">
            <h2>Registration Form Demo</h2>
            <div class="form-group">
                <label for="firstName" class="col-sm-3 ">Πλήρες Όνομα</label>
                <div class="col-sm-9">
                    <input type="text" id="firstName" placeholder="Full Name"
class="form-control" autofocus>
                </div>
            </div>
            <div class="form-group">
                <label for="email" class="col-sm-3 ">Email</label>
```

```
                <div class="col-sm-9">
                    <input type="email" id="email" placeholder="Email"
class="form-control">
                </div>
            </div>
            <div class="form-group">
                <label for="password" class="col-sm-3 ">Password</label>
                <div class="col-sm-9">
                    <input type="password" id="password"
placeholder="Password" onblur="empty()" class="form-control">
                </div>
            </div>
            <div class="form-group">
                <label for="birthDate" class="col-sm-3 ">Ημ. Γέννησης</label>
                <div class="col-sm-9">
                    <input type="date" id="birthDate" class="form-control">
                </div>
            </div>
            <div class="form-group">
                <label for="country" class="col-sm-3 ">Περιοχή</label>
                <div class="col-sm-9">
                    <select id="country" class="form-control">
                        <option>Ανταρκτική</option>
                        <option>Ασία</option>
                        <option>Αφρική</option>
                        <option>Βόρεια Αμερική</option>
                        <option>Νότια Αμερική</option>
                        <option>Ευρώπη</option>
                        <option>Ωκεανία</option>
                    </select>
                </div>
            </div>
            <div class="form-group">
                <label class="col-sm-3">Φύλο</label><br><br>
                <div class="col-sm-6">
                    <div class="row">
                        <div class="col-sm-4">
                            <label class="radio-inline">
                                <input type="radio" id="femaleRadio"
name="gender" value="Female">Female
                            </label>
                        </div>
                        <div class="col-sm-4">
                            <label class="radio-inline">
                                <input type="radio" id="maleRadio"
name="gender" value="Male">Male
                            </label>
                        </div>
                        <div class="col-sm-4">
```

```
                            <label class="radio-inline">
                                <input type="radio" id="uncknownRadio"
name="gender" value="Unknown">Unknown
                            </label>
                        </div>
                    </div>
                </div>
            </div> <!-- /.form-group -->
            <div class="form-group">

            </div> <!-- /.form-group -->
            <div class="form-group">
                <div class="col-sm-9 ">
                    <div class="checkbox">
                        <label>
                            <input type="checkbox">I accept <a
href="#">terms</a>
                        </label>
                    </div>
                </div>
            </div> <!-- /.form-group -->
            <div class="form-group">
                <div class="col-sm-9 col-sm-offset-3">
                    <button type="submit" class="btn btn-primary btn-block"
onsubmit="empty()">Register</button>
                </div>
            </div>
        </form> <!-- /form -->
    </div> <!-- ./container -->

    <script src="script.js"></script>
</body>
</html>
```

JavaScript

```
function empty()
{
    var pass = document.getElementById('password');
    if(pass.value == "")
    {
        pass.style.borderColor = "red";
    }
    else
    {
        pass.style.borderColor = "";
    }
}
```

Παραδείγματα Ενότητας 3.5.3

Δημιουργία module και controller

HTML

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">

        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style>
            html, body
            {
                font-size: 1.1em;
            }
        </style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.js"></script>
    </head>
    <body>
        <header>
            <nav class="navbar navbar-default">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">AngularJS</a>
                </div>

                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
                </ul>
            </div>
            </nav>
        </header>

        <div class="container">
            <div ng-controller="mainController">
                <h1>Hello world!</h1>
```

```
            </div>
        </div>
    </body>
</html>
```

JavaScript

```
var myApp = angular.module('myApp', []);
myApp.controller('mainController', ['$scope','$log',function ($log,$scope) {
    $log.info($scope);
}]);
```

Παραδείγματα Ενότητας 3.5.4

Data Binding και εφαρμογή φίλτρων

HTML

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style>
            html, body{
                font-size: 1.1em;
            }
        </style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.js"></script>
    </head>
    <body>
        <header>
            <nav class="navbar navbar-default">
```

```
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">AngularJS</a>
                </div>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
                </ul>
            </div>
            </nav>
        </header>
        <div class="container">
            <div ng-controller="mainController">
                <div>
                    <label>Facebook username:</label>
                    <input type="text" ng-model="username" />
                </div>
                <hr>
               <h1>facebook.com/{{username|uppercase|limitTo:10}}</h1>
            </div>
        </div>
    </body>
</html>
```

App.js

```
var myApp = angular.module('myApp', []);

myApp.controller('mainController', ['$scope','$filter', function
($scope,$filter) {
    $scope.username='';
}]);
```

Παράδειγμα με χρήση Watchers και του Digest Loop

HTML

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">
```

```
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style>
            html, body{
                font-size: 1.1em;
            }
        </style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.js"></script>
    </head>
    <body>
        <header>
            <nav class="navbar navbar-default">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">AngularJS</a>
                </div>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
                </ul>
            </div>
            </nav>
        </header>
        <div class="container">
            <div ng-controller="mainController">
                <div>
                    <label>Facebook username:</label>
                    <input type="text" ng-model="username" />
                </div>
                <hr>
              <h1>facebook.com/{{username|lowercase|limitTo:10}}</h1>
            </div>
        </div>
    </body>
</html>
```

App.js

```
var myApp = angular.module('myApp', []);

myApp.controller('mainController', ['$scope','$filter','$timeout', function
($scope,$filter,$timeout) {
    $scope.username='';
```

```
    $scope.$watch('username', function(newValue,oldValue){
        console.info('Changed!');
        console.log('Old: '+oldValue);
        console.log('New: '+newValue);
    });

    $timeout(function(){
            $scope.username = 'text';
            console.log('Scope Changed!');
    },3000);
}]);
```

Εφαρμογή κανόνων σε επιλογή όνομα χρήστη με χρήση Two-Way Data Binding και ελέγχων στην HTML

HTML

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style> html, body{ font-size: 1.1em; } </style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.js"></script>
    </head>
    <body>
        <header>
            <nav class="navbar navbar-default">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">AngularJS</a>
                </div>

                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#"><i class="fa fa-home"></i> Home</a></li>
                </ul>
            </div>
            </nav>
```

```
        </header>

        <div class="container">
            <div ng-controller="mainController">
                <div>
                    <label>Facebook username:</label>
                    <input type="text" ng-model="username" />
                </div>
                <div class="alert" ng-class="{'alert-warning':username.length
< characters,'alert-danger':username.length > characters}" ng-
show="username.length !== characters">

                    <div ng-show="username.length < 8">
                        Κάτω από 8 χαρακτήρες!
                    </div>
                    <div ng-show="username.length > 8">
                        Πάνω από 8 χαρακτήρες!
                    </div>
                </div>
                <hr>
                <h1>facebook.com/{{lowercaseUsername()}}</h1>
                <h3>Κανόνες</h3>
                <ul>
                    <li ng-repeat="rule in rules" >{{rule.rulename}}</li>
                </ul><br>
                <input class="btn btn-primary" type="button" value="Click me"
ng-click="alertClick()">
            </div>
        </div>
    </body>
</html>
```

App.js

```
var myApp = angular.module('myApp', []);

myApp.controller('mainController', ['$scope','$filter', function
($scope,$filter) {
    $scope.username='';
    $scope.lowercaseUsername = function(){
        return $filter('lowercase')($scope.username);
    };
    $scope.characters = 8;
    $scope.rules = [
        {rulename: "Πρέπει να είναι 8 χαρακτήρες"},
        {rulename: "Πρέπει να μην έχει χρησιμοποιηθεί αλλού"},
        {rulename: "Πρέπει να είναι ωραίο"}
    ];
    $scope.alertClick = function(){
        alert("Clicked!");
    }
}]);
```

Τροποποίηση του παραδείγματος με την εφαρμογή των κανόνων όνομα χρήστη, ώστε οι κανόνες να διαβάζονται από βάση δεδομένων καθώς και να προστίθενται νέοι χωρίς να χρειάζεται να γίνει ανανέωση της σελίδας

HTML

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style> html, body{ font-size: 1.1em; }</style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.js"></script>
    </head>
    <body>
        <header>
```

```
            <nav class="navbar navbar-default">
                <div class="container">
                    <div class="navbar-header">
                        <a class="navbar-brand" href="/">AngularJS</a>
                    </div>

                    <ul class="nav navbar-nav navbar-right">
                        <li><a href="#"><i class="fa fa-home"></i>
Home</a></li>
                    </ul>
                </div>
            </nav>
        </header>

        <div class="container">
            <div ng-controller="mainController">
                <div>
                    <label>Facebook username:</label>
                    <input type="text" ng-model="username" />
                </div>
                <div class="alert" ng-class="{'alert-warning':username.length
< characters,'alert-danger':username.length > characters}" ng-
show="username.length !== characters">

                    <div ng-show="username.length < 8">
                        Κάτω από 8 χαρακτήρες!
                    </div>
                    <div ng-show="username.length > 8">
                        Πάνω από 8 χαρακτήρες!
                    </div>
                </div>
                <hr>
                <h1>facebook.com/{{lowercaseUsername()}}</h1>
                <h3>Κανόνες</h3>
                <ul>
                    <li ng-repeat="rule in rules" >{{rule.RuleName}}</li>
                </ul>
                <br>
                <label>Add rule: </label><input type="text" ng-
model="newrule" >
                <br><br>
                <input class="btn btn-primary" type="button" value="Add Rule"
ng-click="AddRule()">
            </div>
        </div>
    </body>
</html>
```

App.js

```
var myApp = angular.module('myApp', []);

myApp.controller('mainController', ['$scope','$filter','$http', function
($scope,$filter,$http) {

    $scope.username='';

    $scope.lowercaseUsername = function(){
        return $filter('lowercase')($scope.username);
    };

    $scope.characters = 8;

    $http.get('http://83.212.116.174/nikos.93/web/api/index.php')
        .success(function(result){
        $scope.rules = result;
    })
        .error(function(data,status){
        console.log(data);
    });

    $scope.newrule = '';
    $scope.AddRule = function(){
$http.post('http://83.212.116.174/nikos.93/web/api/add.php',{newrule:
$scope.newrule})
            .success(function(result){
            $scope.rules = result;
            $scope.newrule = '';
            console.log(result);
        })
            .error(function(data,status){
            console.log(data);
        });
    };
}]);
```

Παραδείγματα Ενότητας 3.5.5

Δημιουργία ενός SPA

index.html

```
<!DOCTYPE html>
<html lang="" ng-app="myApp">
    <head>
        <title>Introduction to AngularJS</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">

        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style> html, body{ font-size: 1.1em; }</style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="app.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-route.min.js
"></script>
    </head>
    <body>
        <header>
            <nav class="navbar navbar-default">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">AngularJS</a>
                </div>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#/"><i class="fa fa-home"></i> Home</a></li>
                    <li><a href="#/second"><i class="fa fa-home"></i>
Second</a></li>
                </ul>
            </div>
            </nav>
        </header>
        <div class="container">
            <div ng-view="mainController">
            </div>
        </div>
    </body>
</html>
```

app.js

```
var myApp = angular.module('myApp', ['ngRoute']);

myApp.config(function($routeProvider){
    $routeProvider

    .when('/',{
        templateUrl: 'templates/main.html',
        controller: 'mainController'
    })
    .when('/second/:num',{
        templateUrl: 'templates/second.html',
        controller: 'secondController'
    })
});

myApp.controller('mainController', ['$scope','$log', function ($scope,$log) {
    $scope.name = 'main';
}]);
myApp.controller('secondController', ['$scope','$log','$routeParams',
function ($scope,$log,$routeParams) {
    $scope.num = $routeParams.num || 1;
}]);
```

main.html

```
<h1>Main page</h1>
<h3>Scope value: {{name}}</h3>
```

second.html

```
<h1>Second page</h1>
<h3>Scope route: {{num}}</h3>
```

Παραδείγματα Ενότητας 3.5.6

Δημιουργία ενός service

app.js

```
var myApp = angular.module('myApp', ['ngRoute']);

myApp.config(function($routeProvider){
    $routeProvider

    .when('/',{
        templateUrl: 'templates/main.html',
        controller: 'mainController'
    })
    .when('/second/:num',{
        templateUrl: 'templates/second.html',
        controller: 'secondController'
    })
});

myApp.service('nameService', function(){
   var self = this;
    this.name = 'Nick';
    this.namelength = function(){
        return self.name.length;
    };
});

myApp.controller('mainController', ['$scope','$log','nameService', function
($scope,$log,nameService) {

    $scope.name = nameService.name;
    $scope.$watch('name', function(){
        nameService.name = $scope.name;
    });

    $log.log(nameService.name);
    $log.log(nameService.namelength());

}]);
myApp.controller('secondController',
['$scope','$log','$routeParams','nameService', function
($scope,$log,$routeParams,nameService) {

    $scope.num = $routeParams.num || 1;
    $scope.name = nameService.name;
```

```
    $scope.$watch('name', function(){
        nameService.name = $scope.name;
    });
}]);
```

Δημιουργία ενός directive

app.js

```
var myApp = angular.module('myApp', ['ngRoute']);

myApp.config(function($routeProvider){
    $routeProvider

    .when('/',{
        templateUrl: 'templates/main.html',
        controller: 'mainController'
    })
    .when('/second/:num',{
        templateUrl: 'templates/second.html',
        controller: 'secondController'
    })
});

myApp.controller('mainController', ['$scope','$log', function ($scope,$log) {
    $scope.name = 'Main';
}]);
myApp.controller('secondController', ['$scope','$log','$routeParams',
function ($scope,$log,$routeParams) {
    $scope.num = $routeParams.num || 1;
}]);

myApp.directive("searchResult", function(){
   return {
       restrict: '',
       templateUrl: 'directives/searchresult.html',
       replace: false
   }
});
```

directives/searchresult.html

```
<a href="#" class="list-group-item">
    <h4 class="list-group-item-heading">Γιάννης, Παπαδόπουλος</h4>
    <p class="list-group-item-text">Αθηνάς 5, Αθήνα, 12345</p>
</a>
```

main.html

```
<label>Search</label>
<input type="text" value="Γιάννης">

<h3>Search Results</h3>
<div class="list-group">
    <search-result></search-result>
    <div search-result></div>
    <div class="search-result"></div>
    <!-- directive: search-result-->
</div>
```

Αναπαραγωγή ενός directive με transclude

app.js

```
var myApp = angular.module('myApp', ['ngRoute']);

myApp.config(function($routeProvider){
    $routeProvider

    .when('/',{
        templateUrl: 'templates/main.html',
        controller: 'mainController'
    })
    .when('/second/:num',{
        templateUrl: 'templates/second.html',
        controller: 'secondController'
    })
});

myApp.controller('mainController', ['$scope','$log', function ($scope,$log) {
    $scope.name = 'Main';
    $scope.people = [
        {
        name: 'Γιάννης, Παπαδόπουλος',
        address: 'Αθηνάς 5',
        city: 'Αθήνα',
        zip: '12345'
```

```
        },
        {
        name: 'Τάσος, Παπαδόπουλος',
        address: 'Αθηνάς 15',
        city: 'Αθήνα',
        zip: '22222'
        },
        {
        name: 'Γιώργος, Παπαδόπουλος',
        address: 'Αθηνάς 25',
        city: 'Νίκαια',
        zip: '11111'
        }
    ];

    $scope.fullAddress = function(person){
        return person.address + ', ' + person.city + ' ' + person.zip;
    };

}]);
myApp.controller('secondController', ['$scope','$log','$routeParams',
function ($scope,$log,$routeParams) {

    $scope.num = $routeParams.num || 1;

}]);

myApp.directive("searchResult", function(){
   return {
       restrict: '',
       templateUrl: 'directives/searchresult.html',
       replace: false,
       scope: {
           personObject: "=",
           fullAddressFunction: "&"
       },
       transclude: true
   }
});
```

directives/searchresult.html

```
<a href="#" class="list-group-item">
    <h4 class="list-group-item-heading">{{personObject.name}}</h4>
    <p class="list-group-item-text">{{fullAddressFunction({p:
personObject})}}</p>
    <small><ng-transclude></ng-transclude></small>
</a>
```

main.html

```
<label>Search</label>
<input type="text" value="Γιάννης">

<h3>Search Results</h3>
<div class="list-group">
    <search-result person-object="person" full-address-
function="fullAddress(p)" ng-repeat="person in people">*Το αποτέλεσμα μπορεί
να μην είναι έγκυρο</search-result>

</div>
```

Παράδειγμα Ενότητας 3.5.7

Δημιουργία ενός SPA για την πρόγνωση του καιρού

index.html

```
<!DOCTYPE html>
<html lang="" ng-app="weatherApp">
    <head>
        <title>Weather Forecast SPA</title>
        <meta http-equiv="X-UA-Compatible" content="IE=Edge">
        <meta charset="UTF-8">
        <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/css/bootstrap.min.css"
/>
        <style> html, body {font-size: 1.1em;} </style>
        <script
src="https://ajax.googleapis.com/ajax/libs/angularjs/1.5.0/angular.min.js"></
script>
        <script src="https://code.angularjs.org/1.5.0/angular-
route.min.js"></script>
        <script src="https://code.angularjs.org/1.5.0/angular-
resource.min.js"></script>
        <script src="app.js"></script>
        <script src="routes.js"></script>
        <script src="services.js"></script>
        <script src="controllers.js"></script>
        <script src="directives.js"></script>
    </head>
    <body>
```

```
        <header>
            <nav class="navbar navbar-default">
            <div class="container">
                <div class="navbar-header">
                    <a class="navbar-brand" href="/">Weather Forecast SPA</a>
                </div>
                <ul class="nav navbar-nav navbar-right">
                    <li><a href="#/"><i class="fa fa-
home"></i>Αρχική</a></li>
                </ul>
            </div>
            </nav>
        </header>

        <div class="container">
            <div ng-view> </div>
        </div>

    </body>
</html>
```

app.js

```
var weatherApp = angular.module('weatherApp', ['ngRoute', 'ngResource']);
```

controllers.js

```
weatherApp.controller('homeController', ['$scope','$location','cityService',
function($scope,$location,cityService){

    $scope.city = cityService.city;

    $scope.$watch('city', function(){
        cityService.city = $scope.city;
    });

    $scope.submit = function(){
        $location.path("/forecast");
    }

}]);

weatherApp.controller('forecastController',
['$scope','weatherService','cityService','$routeParams',
function($scope,weatherService,cityService,$routeParams){

    $scope.city = cityService.city;
```

```
    $scope.days = $routeParams.days || 2;

    $scope.weatherResults = weatherService.GetWeather($scope.city,
$scope.days);

    $scope.convertKelvinToCelsius = function(kelvin){
        return kelvin-273.15;
    }

    $scope.convertDate = function(dt){
        return new Date(dt*1000);
    }

}]);
```

directives.js

```
weatherApp.directive('weatherReport', function(){
    return {
        restrict: 'E',
        templateUrl: 'directives/weatherReport.html',
        replace: true,
        scope: {
            weatherDay: "=",
            convTemp: "&",
            convDay: "&",
            dateFormat: "@"
        }
    }
});
```

routes.js

```
weatherApp.config(function($routeProvider){

    $routeProvider

        .when('/', {
        templateUrl: 'pages/home.html',
        controller: 'homeController'
    })
        .when('/forecast', {
        templateUrl: 'pages/forecast.html',
        controller: 'forecastController'
    })
        .when('/forecast/:days', {
        templateUrl: 'pages/forecast.html',
```

```
        controller: 'forecastController'
    })
});
```

services.js

```
weatherApp.service('cityService', function(){

    this.city = 'Athens, GR';
});

weatherApp.service('weatherService', ['$resource', function($resource){

    this.GetWeather = function(city, days){
        var weatherApi =
$resource("http://api.openweathermap.org/data/2.5/forecast/daily", {callback:
"JSON_CALLBACK"}, {get: {method: "JSONP"}});

        return weatherApi.get({q: city, cnt: days, APPID:
'2cacd4cbc0c4013026f278c0990c71c5'});
    };
}]);
```

home.html

```
<div class="row">
    <div class="col-md-6 col-md-offset-3">
        <h4>Πρόγνωση με βάση την ημέρα</h4>
        <form ng-submit="submit()">
            <div class="form-group">
                <input type="text" ng-model="city" class="form-control">
            </div>
            <input type="submit" class="btn btn-primary" value="Πρόγνωση">
        </form>
    </div>
</div>
```

forecast.html

```
<p>
    <a href="#/">Επιστροφή</a>
</p>
Πρόγνωση για {{city}}
<hr>
Ημέρες: <a href="#/forecast/3" ng-class="{'bg-primary': days === '3'}">3</a>
| <a href="#/forecast/5" ng-class="{'bg-primary': days === '5'}">5</a> | <a
href="#/forecast/7" ng-class="{'bg-primary': days === '7'}">7</a>
<hr>
<div ng-repeat="weather in weatherResults.list">
    <div class="row">
        <div class="col-md-12">
            <weather-report weather-day="weather" conv-
temp="convertKelvinToCelsius(temp)" conv-day="convertDate(dt)" date-
format="EEE d MMM y"></weather-report>
        </div>
    </div>
</div>
```

weatherReport.html

```
<div class="panel panel-default">
    <div class="panel-heading">
        <h3 class="panel-title">{{convDay({dt: weatherDay.dt}) | date:
dateFormat}}</h3>
    </div>
    <div class="panel-body">
        Θερμοκρασία: {{convTemp({temp: weatherDay.temp.day}) |
limitTo:4}}&deg;C
    </div>
</div>
```

ΕΥΡΕΤΗΡΙΟ

ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ ΑΚΡΩΝΥΜΙΑ

MV*	Model View Star
SPA	Single Page Application
GNU	GNU's Not Unix
GPL	General Public Licence
HTML	HyperText Markup Language
APP	Application
DOM	Document Object Model
HTTP	HyperText Transfer Protocol
AJAX	Asynchronous JavaScript and XML

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Ν. Μαραγκουδάκης, *Learn and Understand JavaScript & AngularJS,* Αθήνα: DGA, 2016.

[2] MDN, «JavaScript,» 2016. [Ηλεκτρονικό]. Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[3] W3Schools, «JavaScript Tutorial,» 2016. [Ηλεκτρονικό]. Available: http://www.w3schools.com/js.

[4] Α. Δ. (GUnet), 2016. [Ηλεκτρονικό]. Available: http://www.openeclass.org/.

[5] B. Inc., 2016. [Ηλεκτρονικό]. Available: http://uki.blackboard.com/.

[6] docebo, 2016. [Ηλεκτρονικό]. Available: https://www.docebo.com/.

[7] M. community, «About Moodle,» [Ηλεκτρονικό]. Available: https://docs.moodle.org/30/en/About_Moodle.

[8] M. community, «Moodle forum,» [Ηλεκτρονικό]. Available: https://moodle.org/mod/forum/.

[9] t. f. e. Wikipedia, «Moodle,» [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Moodle.

[10] W. C. Geeks, AngularJS Programming Cookbook, Exelixis Media P.C., 2015.

[11] D. Wahlin, AngularJS in 60 Minutes, Wahlin Consulting, 2013, 2014.

[12] Google, «AngularJS,» 2016. [Ηλεκτρονικό]. Available: https://angularjs.org/.

[13] «angularjs,» 2016. [Ηλεκτρονικό]. Available: http://www.tutorialspoint.com/angularjs.

[14] Wikipedia, «Educational technology,» 2016. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Educational_technology.

[15] P. Dillenbourg, «Virtual Learning Environments,» σε *EUN CONFERENCE 2000*, UNIVERSITY OF GENEVA, 2000.

[16] Wikipedia, «Learning management system,» 2016. [Ηλεκτρονικό]. Available: https://en.wikipedia.org/wiki/Learning_management_system.

[17] A. Buchner, Moodle Administration: An administrator's guide to configuring, securing, customizing and extending Moodle, Birmingham: Packt, 2008.

[18] Α. ΑΣΤΕΡΙΟΣ, *ΚΑΛΕΣ ΠΡΑΚΤΙΚΕΣ ΕΦΑΡΜΟΓΗΣ,* ΘΕΣΣΑΛΟΝΙΚΗ, 2013.