# ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ Ψηφιακής Τεχνολογίας

ΤΜΗΜΑ Πληροφορικής και Τηλεματικής

ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ Πληροφορική και Τηλεματική

ΚΑΤΕΥΘΥΝΣΗ Τηλεπικοινωνιακά Δίκτυα και Υπηρεσίες Τηλεματικής


**Ιστορική αναδρομή και Σύγχρονες τάσεις στην Ανάπτυξη εφαρμογών σε JavaScript**

Διπλωματική Εργασία


**Νόελ Θέμις Κουτλής**




Αθήνα, 2017



# ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ Ψηφιακής Τεχνολογίας
ΤΜΗΜΑ Πληροφορικής και Τηλεματικής
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ Πληροφορική και Τηλεματική
ΚΑΤΕΥΘΥΝΣΗ Τηλεπικοινωνιακά Δίκτυα και Υπηρεσίες Τηλεματικής

**Τριμελής Εξεταστική Επιτροπή**

**Επιβλέπων Κωνσταντίνος Τσερπές**
**Επίκουρος Καθηγητής, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο**

**Μαλβίνα Βαμβακάρη, Αναπληρώτρια Καθηγήτρια, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο**

**Δημοσθένης Αναγνωστόπουλος, Τμήμα Πληροφορικής και Τηλεματικής, Χαροκόπειο Πανεπιστήμιο**

Ο Νόελ Θέμις Κουτλής

δηλώνω υπεύθυνα ότι:

**1)** Είμαι ο κάτοχος των πνευματικών δικαιωμάτων της πρωτότυπης αυτής εργασίας και από όσο γνωρίζω η εργασία μου δε συκοφαντεί πρόσωπα, ούτε προσβάλει τα πνευματικά δικαιώματα τρίτων.

**2)** Αποδέχομαι ότι η ΒΚΠ μπορεί, χωρίς να αλλάξει το περιεχόμενο της εργασίας μου, να τη διαθέσει σε ηλεκτρονική μορφή μέσα από τη ψηφιακή Βιβλιοθήκη της, να την αντιγράψει σε οποιοδήποτε μέσο ή/και σε οποιοδήποτε μορφότυπο καθώς και να κρατά περισσότερα από ένα αντίγραφα για λόγους συντήρησης και ασφάλειας.

**ΕΥΧΑΡΙΣΤΙΕΣ**

ιδιαίτερα στους καθηγητές μου Κώστα Τσερπέ, Μαλβίνα Βαμβακάρη, Χρήστο Μιχαλακέλη, Γιώργο Δημητρακόπουλο, Γιώργο Μπράβο, Περικλή Λουκόπουλο, και όσους με στήριξαν σε αυτή τη προσπάθεια, τον Διονύση Γκρέκα, Λεωνίδα Φέγγο, Φωτεινή Γρόμπου, Δημήτρη Κοντίνο, Ζαχαρία Καρασάββα, Σταμάτη Αρκούλη, Παναγιώτη Ματσή και ιδιαίτερα τους συμφοιτητές μου Γιώργο και Σταμάτη Πανουτσακόπουλο

# ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

## Περίληψη στα Ελληνικά

Η γλώσσα JavaScript δημιουργήθηκε πριν 20 χρόνια. Τα τελευταία όμως 10 χρόνια από γλώσσα του browser μεταλλάχθηκε σε μια πλήρη γλώσσα για προγραμματισμό εφαρμογών διαδικτύου τόσο στο front end όσο και στο back end. Η εργασία παρουσιάζει την ιστορική αναδρομή της γλώσσας, τις σύγχρονες τάσεις προγραμματισμού στο Web με JavaScript frameworks, βάσεις δεδομένων, το stack της JavaScript και ένα παράδειγμα εφαρμογής Internet of Things σε JavaScript.



**Λέξεις κλειδιά:** ES6, EcmaScript 2016, JavaScript, βιβλιοθήκες JavaScript FrameWorks, IoT, full stack JavaScript



## Abstract

JavaScript was created before 20 years as a scripting language for the browsers. In the last 10 years the language evolved in a full stack language for the web, used in the front end and the back end. In this thesis, we present the historical moments that changed JavaScript, JavaScript frameworks, databases and the current trends in JavaScript, and a small example in the area of Internet of Things.



**Keywords:** ES6, EcmaScript 2016, JavaScript, JavaScript Frameworks, full stack JavaScript, IoT

# ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ - ΦΩΤΟΓΡΑΦΙΩΝ

# ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ



# ΚΑΤΑΛΟΓΟΣ ΣΧΗΜΑΤΩΝ

# ΣΥΝΤΟΜΟΓΡΑΦΙΕΣ

| | |
|---|---|
| ΕΛ/ΛΑΚ | Ελεύθερο Λογισμικό/Λογισμικό Ανοικτού Κώδικα |
| IoT | Internet Of Things |
| ECMA | European Computer Manufacturers Association |
| JS | JavaScript |
| LLVM | Low Level Virtual Machine |
| JIT | Just In Time |
| GPU | Graphics Processing Unit |
| JSON | JavaScript Object Notation |
| GWT | Google Web Toolkit |
| MEAN | MongoDB, Express.js, Angular, and Node.js |
| LAMP | Linux, Apache HTTP Server, MySQL, PHP/Perl/Python |
| WEBGL | Web Graphics Library |
| MVC | Model View Controller pattern |
| MVVM | Model View – View Model pattern |
| DDP | Distributed Data Protocol |
| BPAP | Bluetooth phonebook access profile |
| API | Application Programming Interface |
| BLE | Bluetooth Low Energy |
| IDE | Integrated Development Environment |

# ΕΙΣΑΓΩΓΗ

Η εργασία αυτή γράφτηκε το 2016 με την ευκαιρία της συμπλήρωσης των 20 ετών από τη δημιουργία της JavaScript (youtu.be/UZetViiy4jo). Όπως θα αναφερθεί στην ιστορική αναδρομή της γλώσσας, ήδη από το 1995 η γλώσσα αναπτύχθηκε ως εργαλείο του browser, δηλαδή ως απλή γλώσσα για χειρισμό ορισμένων λειτουργιών στο front end ενώ στο back end θα υπήρχε η αντίστοιχη σοβαρότερη γλώσσα Java. Τα χρόνια αυτά η ίδια η Java αντιμετωπίστηκε ως "toy language" δίπλα στη C και τη C++. Μια απλοποιημένη μορφή της παραδοσιακής γλώσσας των λειτουργικών συστημάτων χωρίς pointers και memory allocation και με αυτόματο garbage collection. Χρειάστηκαν πολλά χρόνια μέχρι η Java να κατακτήσει την νούμερο #1 θέση στις γλώσσες προγραμματισμού, να αποκτήσει οπαδούς, μεγάλη τεκμηρίωση και ιδιαίτερα, μεγάλο απόθεμα ελεύθερου πηγαίου κώδικα. Η μεγάλη της ώθηση ειδικά με το Android σκίασε την JavaScript που ακόμα και σήμερα θεωρείται ίδια γλώσσα παρά τις δομικές διαφορές που έχουν. Και οι δύο γλώσσες παρουσιάστηκαν το 1995 όμως η σχεδίαση της Java ξεκίνησε από το 1991 ενώ η JavaScript σχεδιάστηκε σε μόλις 10 μέρες το 1995. Το μόνο κοινό σημείο της JavaScript από τότε είναι η βιβλιοθήκη της ημερομηνίας και των μαθηματικών (Date, Math). Είχα τη τύχη να παρακολουθήσω την εξέλιξη και των δύο γλωσσών και έτυχε να παρευρεθώ στην δορυφορική παρουσίαση της JAVA σε κεντρικό ξενοδοχείο στην Ελλάδα από τον τότε αντιπρόσωπό της. Από τότε και για 20 χρόνια οι δύο γλώσσες εξελίσσονται ανεξάρτητα, η Java είναι ήδη στην έκδοση 8 ενώ η JavaScript είναι στην 6η (ES6) έκδοση από τον Ιούνιο του 2015, μάλιστα και οι δύο γλώσσες έχουν προσθέσει χαρακτηριστικά η μία της άλλης. Οι ίδιες οι γλώσσες, ίσως από τη μόδα και τις τάσεις της εποχής αλλάζουν ενώ οι προγραμματιστές που τις φτιάχνουν προσπαθούν να τις βελτιώσουν, είτε σε open sourcce projects είτε ως εταιρικά projects. Για λόγους συμβατότητας κρατούνται και λάθη στη σχεδίαση που αναγκαστικά γίνονται features μιας γλώσσας όπως συνέβη και στην JavaScript ενώ πολλές φορές μπαίνει μια διαχωριστική γραμμή (στη Javascript ορίστηκε με το strict) ή φτιάχνεται μια νέα γλώσσα στη θέση της προηγούμενης (περίπτωση Apple Swift).

Κατά τη γνώμη μου η διαφοροποίηση μιας γλώσσας από "toy language" σε business language είναι καθαρά στο μυαλό του προγραμματιστή. Χρειάζεται βέβαια αρκετή έρευνα για να αποδειχθεί αν μια γλώσσα θεωρείται κατάλληλη για ένα project, όταν μάλιστα κανείς δεν την αντιλαμβάνεται, αφού στο τέλος της μέρας αυτό που έχει σημασία είναι να λειτουργεί -και

μάλιστα χωρίς προβλήματα- όπου και να βρίσκεται αυτό το back end, αφού σήμερα αυτό είναι κατανεμημένο σε άγνωστους (γεωγραφικά τουλάχιστον) servers.

Σε καμιά περίπτωση η εργασία δε προωθεί τη JavaScript ως μοναδική γλώσσα που πρέπει να ξέρει ένας προγραμματιστής. Παρότι η Java και η JavaScript γεννήθηκαν την ίδια περίπου εποχή, η Java παραμένει η πρώτη σε χρήση γλώσσα και εγκατεστημένη βάση παγκοσμίως μαζί με παλαιότερες γλώσσες που ακολουθούν το ίδιο συντακτικό (C, C++). Όπως είχε πει παλαιότερα ο Bjarne Stroustrup «κανείς δεν πρέπει να καλεί τον εαυτό του επαγγελματία αν ξέρει μόνο μία γλώσσα» (youtu.be/NvWTnIoQZj4).

# ΚΕΦ.1: Ιστορική Αναδρομή


Φωτογραφία του browser ViolaWWW

## 1.1. Γλώσσες προγραμματισμού ενσωματωμένες στον browser

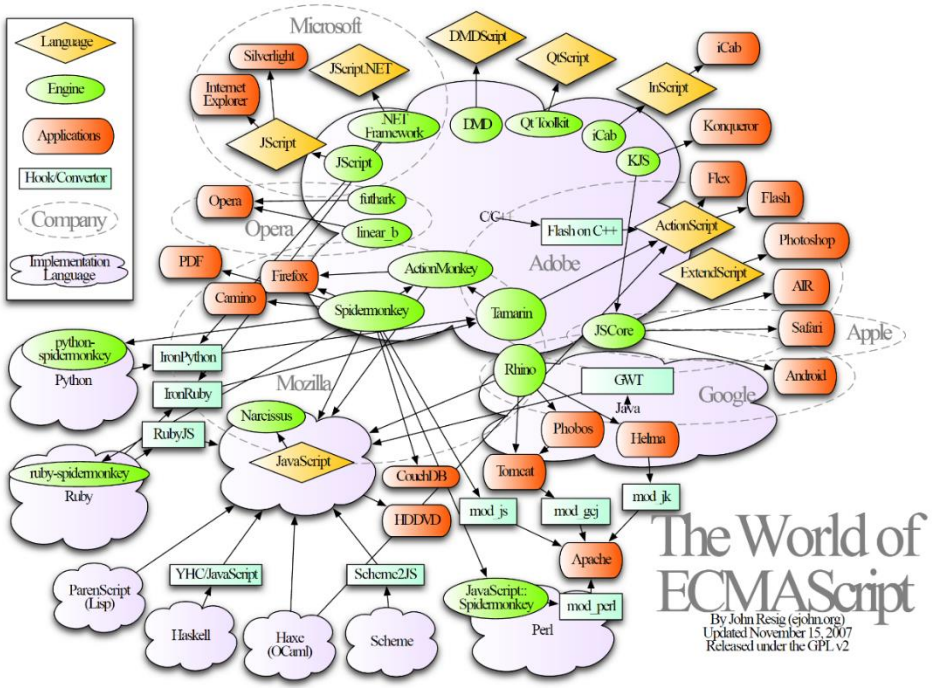
Πολλές φορές έχει γραφτεί η ιστορία του Internet και του www όπως ξεκίνησε στις αρχές της δεκαετίας του 90 από τον Sir Tim Berners-Lee ως μια λύση για διασύνδεση κειμένων. Παλαιότερες αναφορές για τεχνολογίες hypertext φτάνουν ακόμα πιο πίσω, στο 1960 με το Project Xanadu του Ted Nelson, το σύμφωνα με το περιοδικό Wired, μακροβιότερο vapor-ware[103]. Η ιστορία λοιπόν μας διδάσκει ότι πολλά projects μένουν σε ιδέες χωρίς ποτέ να υλοποιηθούν ή υλοποιούνται πολύ αργά (το Xanadu έφτασε να υλοποιείται 40 χρόνια μετά και έγινε ευρέως διαθέσιμο 54 χρόνια μετά) όταν η τεχνολογία ή η μόδα τα έχει ξεπεράσει.

Στο χώρο των browsers το 1992 εμφανίστηκε ο ViolaWWW, ο πρώτος browser (φωτογραφία επάνω) που είχε inline γραφικά, πίνακες, frames και stylesheets πριν αυτά καν γίνουν γνωστά από τη Netscape. Η εταιρεία Eolas Technologies που είχε τα πνευματικά δικαιώματα του browser μαζί με το Πανεπιστήμιο της Καλιφόρνιας έφτασαν να μηνύσουν μεταξύ άλλων την Microsoft, Google και Yahoo και να απαιτήσουν μισό δισεκατομμύριο δολάρια για τις πατέντες που είχαν, μια εκ των οποίων ήταν το scripting, η δυνατότητα που έδιναν στις html σελίδες να περιέχουν

εκτελέσιμο κώδικα, κάνοντας έτσι το www μια νέα πλατφόρμα διανομής και εκτέλεσης εφαρμογών.

Ο σχεδιαστής της εφαρμογής Viola (Visually Interactive Object-oriented Language and Application) Pei-Yuan Wei [104] είχε δηλώσει ότι εμπνεύστηκε από το HyperCard του Macintosh. Πολλά χρόνια αργότερα, το 2012, ο δημιουργός της JavaScript Brendan Eich δήλωσε ότι κι αυτός χρησιμοποίησε ιδέες του τύπου .onclick από το HyperCard του Bill Atkinson.

Από τη δεκαετία του 60 του Xanadu, τα demo του Douglas Engelbart το 68 που παρουσίαζαν το ποντίκι, τα παράθυρα, τις βιντεοσυνομιλίες, τους επεξεργαστές κειμένου, το hypertext, και όσα ενέπνευσαν τα project του Xerox PARC, πέρασαν δεκαετίες εξέλιξης από εκατοντάδες εταιρείες και χιλιάδες επιστημονικές εργασίες για να έχουμε σήμερα τις τεχνολογίες που θα παρουσιάσουμε στα επόμενα κεφάλαια.




Φωτογραφία ο κόσμος της JavaScript το 2007

## 1.2. Η ιστορία της JavaScript

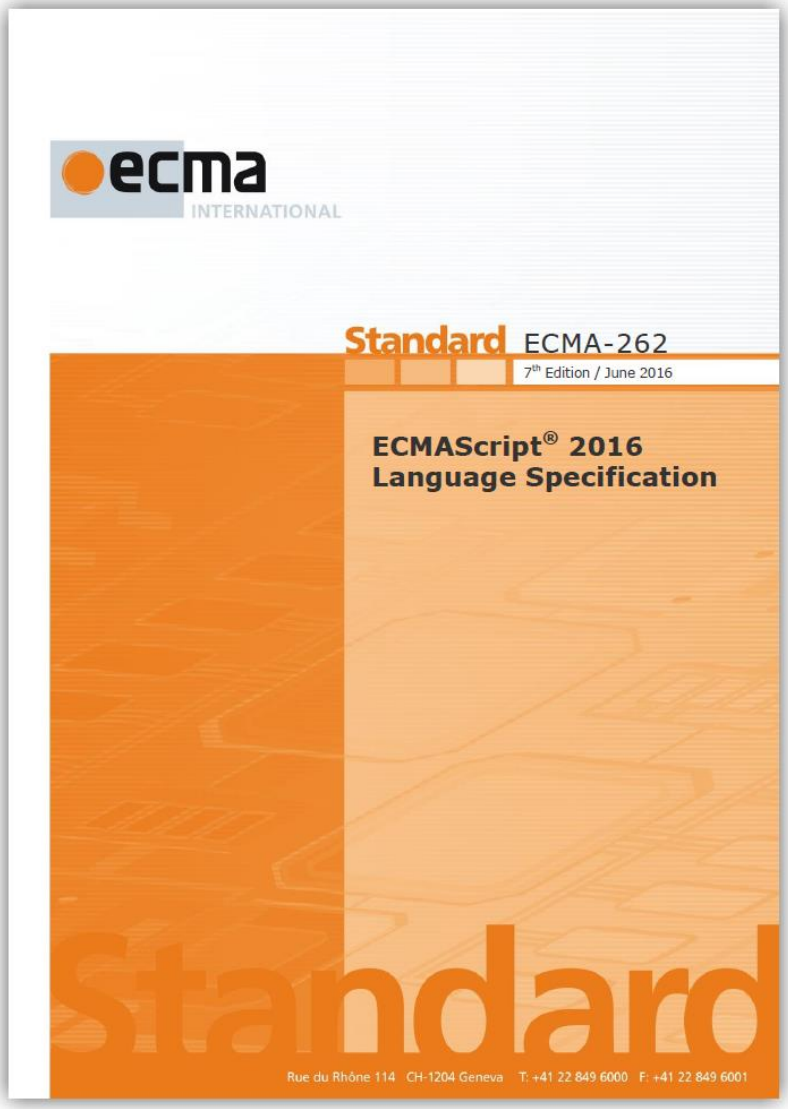
Η JavaScript εμφανίστηκε επίσημα ως γλώσσα το Δεκέμβριο του 1995 στην 3η beta έκδοση του browser Netscape Navigator 2.0. Να σημειώσουμε ότι τη χρονιά εκείνη συνδεμένοι στο Internet ήταν περίπου 16 εκατομμύρια χρήστες. Η επίσημη ιστορία γράφτηκε στο blog[66] του (34χρονου τότε) δημιουργού της γλώσσας, Brendan Eich προσπαθώντας να απαντήσει σε φήμες που είχαν από καιρό μετατραπεί σε ιστορία. Παραθέτουμε τα λόγια του μεταφρασμένα «προσλήφθηκα στην Netscape στις 4 Απριλίου του 1995 με το σκοπό να ενσωματώσω τη γλώσσα προγραμματισμού Scheme (συναρτησιακή γλώσσα, διάλεκτος της Lisp) στον browser. Το management αποτελούμενο από τους Tom Paquin, Michael Toy, Rick Schell, και Marc Andreessen είχε αποφασίσει ότι η Netscape έπρεπε να ενσωματώσει μια γλώσσα σε μορφή πηγαίου κώδικα, μέσα στην HTML. Χρειαζόμασταν ένα demo, ένα proof of concept και εκείνο τον καιρό μεταφέρθηκα στο τμήμα των προϊόντων server για λόγους πολιτικής.

Εκείνες τις μέρες η γλώσσα Oak μετονομάστηκε σε Java και η Netscape ήταν σε συζητήσεις με τη SUN για να ενσωματώσει τη γλώσσα στον Navigator.

Μέσα στη Netscape το κεντρικό θέμα των συζητήσεων ήταν για ποιο λόγο να ενσωματωθούν δύο γλώσσες. Χρειαζόταν μία γλώσσα για όσους έγραφαν components που εκείνη την εποχή οι προγραμματιστές έγραφαν σε C++ και ελπίζαμε να γράψουν σε Java και μια γλώσσα για ερασιτέχνες και επαγγελματίες, μία γλώσσα scripting που θα μπορούσε να μπει μαζί με την HTML.

Το management ήθελε μια γλώσσα που να μοιάζει με Java και έτσι αποκλείστηκε Perl, Python, Tcl αλλά και η Scheme. Δεν είμαι περήφανος αλλά ευτυχής που διάλεξα μια συναρτησιακή γλώσσα τύπου Scheme με prototypes. Με τον Bill Joy συζητήσαμε και αναπτύξαμε τον garbage collector και ο Bill συμφωνούσε ότι θέλαμε μια απλή scripting γλώσσα που να αναλογεί στη χρήση της VisualBasic σε σχέση με την C++. Από τον Απρίλιο μέχρι τον Μάϊο του 1995 με τον Kipp Hickman μελετήσαμε τη Java όπου την εποχή εκείνη ο Kipp έγραφε το δικό του JVM. Μαζί γράψαμε το NSPR (API στο οποίο πατάει και το JVM) και αρχές Μαϊου [η σχεδίαση της γλώσσας έγινε σε μόλις 10 μέρες] άρχισα να φτιάχνω το prototype της Mocha [το όνομα δόθηκε από τον ίδιο τον Marc Andreessen].» Μέχρι το Σεπτέμβριο η

γλώσσα είχε μετονομαστεί σε LiveScript, όνομα με το οποίο πρωτοεμφανίστηκε στις beta εκδόσεις του Navigator 2.0. Μέχρι το Δεκέμβριο είχαν ευδοκιμήσει και οι συζητήσεις με την SUN (ιδιοκτήτης της JAVA) και οι δύο εταιρείες κατέληξαν σε συνεργασία να προωθήσουν την JAVA στην πλευρά του Server και τη νέα γλώσσα στην πλευρά του Browser. Ετσι με δελτίο τύπου[67] η Netscape μαζί με την SUN ανακοίνωσαν την JavaScript ως συμπληρωματική γλώσσα της JAVA με υποστήριξη από 28 εταιρείες. Το σήμα κατατεθέν (trademark) του ονόματος πέρασε αυτόματα στην ORACLE όταν αυτή εξαγόρασε την SUN μαζί με τις υπόλοιπες τεχνολογίες της (και την JAVA) ενώ οι τεχνολογίες της Netscape πέρασαν στην AmericaOnLine (AOL) όταν αυτή εξαγόρασε την Netscape και αργότερα (1998) στον οργανισμό Mozilla ρόλο που σήμερα έχει το Mozilla Foundation.




Το εξώφυλλο του επίσημου οδηγού της γλώσσας, ECMA-262

## 1.1. Η συνεχιζόμενη ανάπτυξη της γλώσσας

Σήμερα η ανάπτυξη της γλώσσας έχει περάσει στον Ευρωπαϊκό οργανισμό κατασκευαστών υπολογιστών ECMA όπου έχει λάβει το επίσημο όνομα ECMAScript (κωδικός στον οργανισμό: ECMA-262). Ετσι η γλώσσα είχε φτάσει στη δεύτερη έκδοση το 1998, στην 3η το 1999 [68] . Στην υλοποίηση της 4ης έκδοσης συμμετείχαν και η Microsoft με τη δική της διάλεκτο που τότε ονομαζόταν Jscript (ο δημιουργός της TurboPascal και του Delphi, Anders Hejlsberg (είχε προσληφθεί στην Microsoft το 1996 για να φτιάξει την δική του διάλεκτο της Java, τη γλώσσα J) και η Jscript θα ήταν η αντίστοιχή της γλώσσα στον browser [ο ίδιος αργότερα ανέπτυξε το .net και την C#]. Αντίστοιχα την ίδια περίοδο η Macromedia που είχε φτιάξει την τεχνολογία Flash ήθελε να συνεισφέρει στην 4η έκδοση της JavaScript, καθώς στο Flash ενσωμάτωνε τη γλώσσα ActionScript ως παραλλαγή της HyperTalk (η γλώσσα που είχε το προϊόν HyperCard της Apple) και η οποία εξελίχθηκε ως επίσημα αναγνωρισμένη διάλεκτος της ECMAScript. Τελικά τα συνεργαζόμενα μέλη ποτέ δε συμφώνησαν στο τι θα πρέπει να περιέχει και έτσι η 4η έκδοση ποτέ δεν υλοποιήθηκε. Σήμερα όλοι οι browsers υποστηρίζουν την έκδοση 5.1 που παρουσιάστηκε τον Ιούνιο του 2011 ενώ Τον Ιούνιο του 2015 επισημοποιήθηκε η έκδοση 6 «ES6 Harmony» και αποφασίστηκε όλες οι νέες εκδόσεις να παίρνουν το όνομα της χρονιάς, με πιο πρόσφατη έγκριση στις 14 Ιουνίου 2016 της έκδοσης “ECMAScript® 2016”[94] (586 σελίδες)

Μερικά ακόμα ενδιαφέροντα στοιχεία για τη δημιουργία της JavaScript έδωσε ο Brendan Eich στη συνέντευξή του στο IEEE τον Ιανουάριο του 2012 (διαθέσιμη στο youtu.be/IPxQ9kEaF8c) όπως:

Φωτογραφία του Brendan Eich κατά τη συνέντευξη

- χρησιμοποίησα ιδέες του τύπου .onclick από το HyperCard του Bill Atkinson (γλώσσα της Apple)
- ήξερα ότι θα υπάρχουν λάθη στη γλώσσα γι' αυτό την έφτιαξα εύπλαστη, ώστε ο κάθε προγραμματιστής να την προσαρμόζει όπως θέλει
- αν έβαζα κλάσεις στην JavaScript το 1995 θα έλεγαν ότι μοιάζει με τη Java ή ότι η JavaScript ανταγωνίζεται την Java. Εκτελούσα εντολές του marketing να τη φτιάξω να μοιάζει με τη Java αλλά να μην είναι μεγάλη, να μοιάζει με το μικρό χαζό αδερφό της!


Το HyperCard στο System 7 της Apple (1991) από το οποίο εμπνεύστηκε η JavaScrip (emulator jamesfriend.com.au/pce-js/)

Στα 20 αυτά χρόνια, εκτός από την εξέλιξη της γλώσσας πολλές άλλες τεχνολογίες δημιουργήθηκαν για να διευκολύνουν τον προγραμματισμό ως προς την JavaScript και τη διαχείριση του DOM στον browser. Τον Απρίλιο του 2014 στο συνέδριο PyCON της γλώσσας Python (τη γλώσσα που έφτιαξε ο Guido van Rossum στις Χριστουγεννιάτικες διακοπές του το 1989) παρουσιάστηκε σε μια ομιλία «η γέννηση και ο θάνατος της JavaScript» από τον Gary Bernhardt[50]. Η ενδιαφέρουσα ομιλία επικεντρώνεται στην ιστορία της γλώσσας από την γέννησή της, τη μόδα που επικρατεί σήμερα και την άποψη του πως θα είναι η γλώσσα σε 10 και 20 χρόνια και πως από γλώσσα που θεωρούταν ότι θα καταργηθεί το 2001 μετά το dot-com bubble (μαζί με τις χιλιάδες των εταιρειών της φούσκας του internet που έκλεισαν).

Είναι προφανές ότι το βασικό host περιβάλλον της γλώσσας, ο browser -λόγω και του πλουραλισμού των εκδόσεων- δεν ακολουθεί πάντα τις εξελίξεις της γλώσσας, με αποτέλεσμα να υπάρχει πληθώρα από ασυμβατότητες τις οποίες καλούνται να λύσουν βιβλιοθήκες τρίτων (παράδειγμα Babel JS https://babeljs.io/). Οι βιβλιοθήκες αυτές είτε εξετάζουν ποιες function δεν υπάρχουν και τις προσθέτουν ή λειτουργούν ως transpilers μετατρέποντας τον κώδικα στο specification παλαιότερης έκδοσης.

Μερικά ενδιαφέροντα στοιχεία που προστέθηκαν στη γλώσσα στην τελευταία έκδοση:

- arrow functions που μοιάζουν με λάμδα functions δανεισμένες από άλλες γλώσσες όπως η CoffeeScript και ομορφαίνουν το συντακτικό στις περιπτώσεις που χρειαζόμασταν μια ανώνυμη function.
- κλάσεις οι οποίες έχουν constructors, getters και setters
- νέες συλλογές (maps, sets κλπ) από αντικείμενα
- iterators και generators για να περιτρέχουμε τις συλλογές με προγραμματιζόμενους τρόπους
- promises για καλύτερη διαχείριση ασύγχρονων λειτουργειών

Η JavaScript είναι από τις λίγες γλώσσες που στοιχεία από δημοφιλής βιβλιοθήκες καταλήγουν να γίνονται μέρος της επόμενης έκδοσης της γλώσσας, εμπλουτίζοντάς την στο διηνεκές.

## 1.2. Σημαντικοί σταθμοί στην εξέλιξη της JavaScript

1. Η ανακοίνωση των τεχνολογιών AJAX[69] το Φεβρουάριο του 2005 ως αποτέλεσμα της χρήσης τους σχεδόν ταυτόχρονα από το Gmail[54] (που είχε αρχίσει να κάνει την πρώτη beta εμφάνιση) και το Google Maps (που πρόσφατα είχε εξαγοραστεί από την Google). Η τεχνολογία είχε ξεκινήσει να εμφανίζεται από το 1999 όταν η Microsoft ενσωμάτωσε το XMLHTTP ActiveX control και αμέσως μετά υιοθετήθηκε από τη Mozilla το Safari και την Opera ως XMLHttpRequest. Η τεχνολογία αυτή επέτρεπε τη συμπεριφορά των web

σελίδων σα να ήταν κανονικές desktop εφαρμογές δημιουργώντας ένα νέο μοντέλο διασποράς εφαρμογών και πώλησής τους ως υπηρεσία πράγμα που έδωσε μεγάλη ώθηση στο cloud computing.

2. Η εμφάνιση βιβλιοθηκών που επέκτειναν ή διευκόλυναν τη χρήση της γλώσσας ειδικά για την κατασκευή μεγάλων "πλούσιων" εφαρμογών. Πολύ καλό παράδειγμα ήταν η βιβλιοθήκη JQuery που παρουσιάστηκε τον Ιανουάριο του 2006 και χρησιμοποιείται σήμερα στους πιο δημοφιλής ισοτόπους (>60% παγκοσμίως) και θα συζητηθεί εκτενέστερα στο κεφάλαιο 3.5
3. Η βελτίωση της ταχύτητας της JavaScript, ειδικά με την παρουσίαση της μηχανής V8 της Google τον Σεπτέμβριο του 2008 που κάνει compile σε κώδικα μηχανής τον κώδικα πριν την εκτέλεσή του αντί να λειτουργεί ως interpreter. Σύμφωνα με τους Financial Times[58] ο Δανός προγραμματιστής Lars Bak θεωρείται η μεγαλοφυΐα πίσω από αυτή τη μηχανή, καθόλου τυχαία, αφού ήταν ο ίδιος που έφτιαξε το 1994 τα virtual machines που έτρεχε η Smalltalk και η Java (την εταιρεία του εξαγόρασε η SUN το 1997 ενώ ο ίδιος ξεκίνησε να εργάζεται για την Google το 2004)[59]
4. Η έλευση το 2009 τεχνολογιών για χρήση της JavaScript σε εκτός browser περιβάλλοντα, αρχικά με το CommonJS[55] και αμέσως αργότερα τον Μάιο με το Node.js[56] του οποία η εκρηκτική εξάπλωση δεν έχει ακόμα σταματήσει.

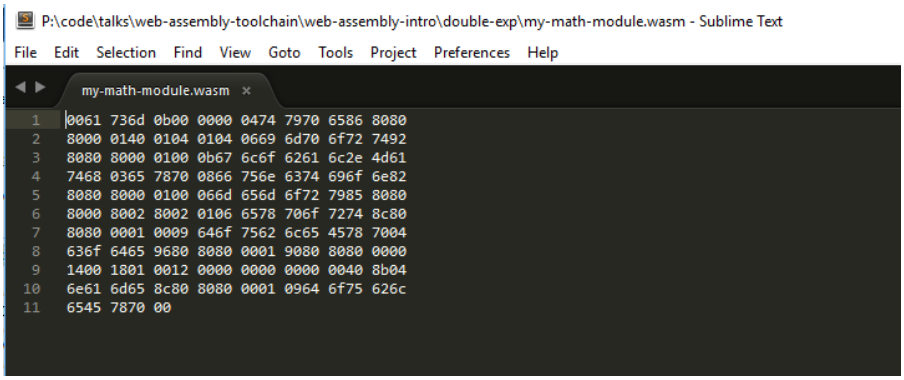
Φωτογραφία: Ο JavaScript υπολογιστής ESPRUINO PICO μεγέθους 33mm x 15mm με επεξεργαστή 84Mhz ARM Cortex M4, 384kB μνήμη Flash, 96kB RAM και USB

Τον Μάρτιο του 2013 εμφανίστηκε μια νέα βιβλιοθήκη με το όνομα asm.js η οποία περιορίζει τη JavaScript σε συγκεκριμένες εντολές που εκτελούνται σχεδόν σε ταχύτητα κώδικα μηχανής. Χάρη στην asm.js αναπτύχθηκε μια νέα γενιά από compilers (αρχικά C και C++) όπου ο εκτελέσιμος κώδικας είναι σε asm.js και ως αποτέλεσμα πολλά μεγάλα projects μετατρέπονται σε JavaScript και εκτελούνται σε πολύ καλές ταχύτητες μέσα στον browser με project σταθμό την μετατροπή της μηχανής Unreal[60] και πολλών Virtual Machines και emulators[61].

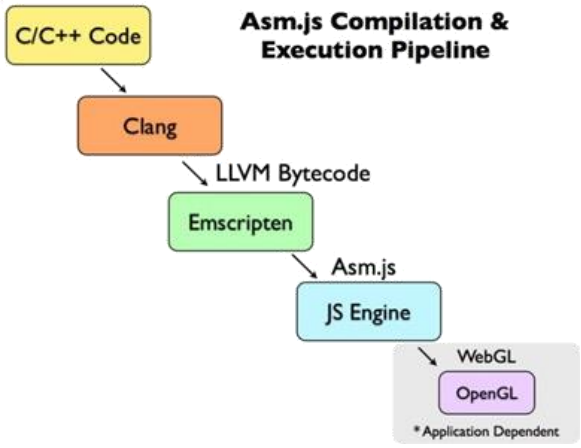
παράδειγμα μιας function γραμμένη σε asm.js [96]

```
function MyMathModule(global) {
    "use asm";
    var exp = global.Math.exp;
    function doubleExp(value) {
        value = +value;
        return +(+exp(+value) * 2.0);
    }
    return { doubleExp: doubleExp };
}
```

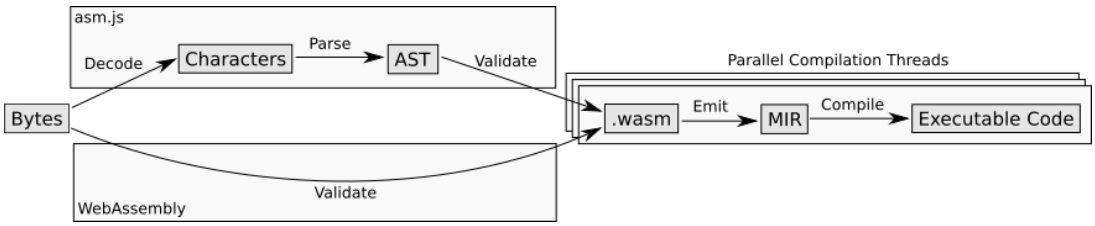
Wasm bytecode του παραπάνω παραδείγματος




Τον Ιούνιο του 2015[68] ανακοινώθηκε από τον Brendan Eich[70] ότι η Google, Microsoft, Mozilla και οι μηχανικοί του WebKit συνεργάζονται για τη δημιουργία ενός νέου προτύπου με την ονομασία WebAssembly (βασισμένο στις τεχνολογίες asm.js, simd.js και sharedarraybuffer) που θα φορτώνει στις ιστοσελίδες binary κώδικα precompiled που μετά θα τρέχει με JIT compilation (webassembly.github.io)




Κεντρικό ρόλο στο χώρο του WebAssembly έχει ο transpiler Emscripten (διαθέσιμος στο kripken.github.io/emscripten-site) που μετατρέπει τον κώδικα της C και της C++ σε asm.js Χρησιμοποιώντας τον Emscripten έγινε και η μετατροπή της μηχανής Unreal 3, Quake 3, Doom, Dune II, Autodesk FormIt, των γλωσσών Python, Ruby, Lua, Perl και άλλων πολύ-διαφημιζόμενων projects. Τον Δεκέμβριο του 2014 το Internet Archive ανακοίνωσε μία έκδοση του emulator DOSBox που φτιάχτηκε με το Emscripten για την εκτέλεση εντός browser εφαρμογών MS-DOS[77] (archive.org/details/softwarelibrary_msdos_games).



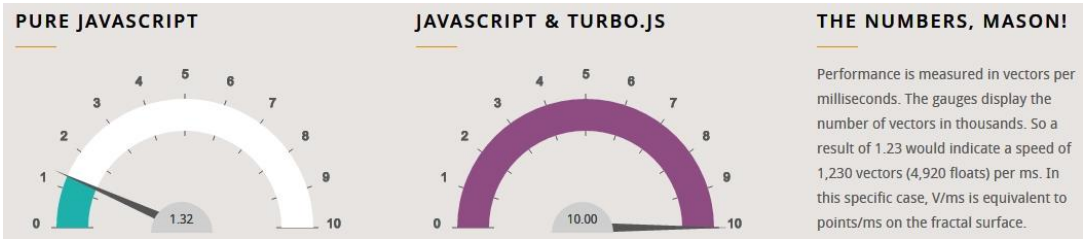

Η WebAssembly και το asm.js θα παίξουν μεγάλο ρόλο στο μέλλον ως νέο format διανομής εφαρμογών από το 2016 και μετά. Η πρώτη ανακοίνωση για πειραματική υποστήριξη της WebAssembly στο Chrome έγινε στα μέσα Μαρτίου 2016[84]

Παραδείγματα ορισμένων από τους πάρα πολλούς emulators που φτιάχτηκαν κυρίως μετατρέποντας κώδικα C και C++ σε JavaScript:

- Apple2JS - **Apple II** (www.scullinsteel.com/apple2)
- pdp11-js - **PDP-11** UNIX V6 emulator (pdp11.aiju.de)
- PCjs – το αυθεντικό **IBM PC Model 5150** (www.pcjs.org)

- Virtual x86 - **x86 emulator** (github.com/copy/v86) με παραδείγματα για να δούμε στον browser παλιά λειτουργικά και εφαρμογές (πχ Linux 2.6, OpenBSD, **Windows 1.01**)
- ElkJS - **Acorn Electron** emulator (elkjs.azurewebsites.net)
- Intel 8080 CPU Emulator – τρέχει **CP/M** στον browser (www.tramm.li/i8080)
- JSBeeb - JavaScript BBC **Micro emulator** (bbc.godbolt.org)
- Visual 6502 - JavaScript simulator της 6502 CPU (www.visual6502.org/JSSim)
- jor1k - **OpenRISC** OR1K JavaScript emulator που τρέχει Linux στον browser (s-macke.github.io/jor1k)
- JSLinux – **PC Emulator που τρέχει Linux** (jslinux.org)

Εκτός από την λύση της WebAssembly για αύξηση της ταχύτητας επεξεργασίας υπάρχει πάντα και η δυνατότητα χρήσης της GPU με θεαματικά αποτελέσματα όπως βλέπουμε στη βιβλιοθήκη turbo.js (turbo.github.io) με θεαματικά αποτελέσματα (ακόμα και x10, με χρήση ακόμα και σε κινητά τηλέφωνα που έχουν GPU)




Φωτογραφία: αύξηση ταχύτητας τρέχοντας κώδικα στην GPU

## 1.3. JSON: Το φορμάτ των αντικειμένων της JavaScript

Το φορμάτ JSON (JavaScript Object Notation) που αυτή τη στιγμή ξεπερνάει σε δημοτικότητα το XML είναι ο εγγενής τρόπος καταχώρησης δεδομένων στα αντικείμενα της JavaScript. Πήρε την ονομασία JSON από τον Douglas Crockford στις αρχές του 2000 όταν άρχισε να το χρησιμοποιεί για να περάσει την κατάσταση του browser μεταξύ συνδέσεων http. Ο ίδιος καταχώρησε την ονομασία json.org το 2002, τον Ιούλιο του 2006 έδωσε τα specifications με το RFC 4627 ενώ από τον Οκτώβριο του 2013 το μοντέλο JSON αποτελεί και αυτό στάνταρτ της Ecma International με τον κωδικό ECMA-404[71] και πλέον χρησιμοποιείται σε όλες τις γλώσσες προγραμματισμού για την ανταλλαγή δεδομένων. Ο Douglas Crockford έχει συντελέσει ιδιαιτέρως στην ανάπτυξη της JavaScript καθώς το βιβλίο του " JavaScript: The Good Parts" θεωρείται η βίβλος της γλώσσας.

**Δείγμα Αντικειμένου JSON**

```
var myObject = {
 "first": "John",
 "last": "Doe",
 "age": 39,
 "sex": "M",
 "salary": 70000,
 "registered": true
};
```

**Δείγμα Πίνακα JSON**

```
var myArray = [
  { "name": "John Doe", "age": 29 },
  { "name": "Anna Smith", "age": 24 },
  { "name": "Peter Jones", "age": 39 }
];
```

Πηγή: www.json.com

Τον Οκτώβριο του 2015 στην OSCON 2015 στο Αμστερνταμ ο Douglas Crockford παρουσίασε μια νέα καινοτομία στο χώρο με την ονομασία SEIF. Το Seif Project (βρίσκεται κάτω από την τοποθεσία της paypal στο github.com/paypal/seifnode) αποτελεί την αρχή ενός νέου πρωτόκολλου ασφαλής και κρυπτογραφημένης μεταφοράς δεδομένων μέσω JSON το οποίο όπως αναφέρει ο εμπνευστής του θα

δώσει ακόμα περισσότερη ώθηση στην διανομή εφαρμογών μέσω web όπως αναφέρθηκε και στην προηγούμενη παράγραφο.

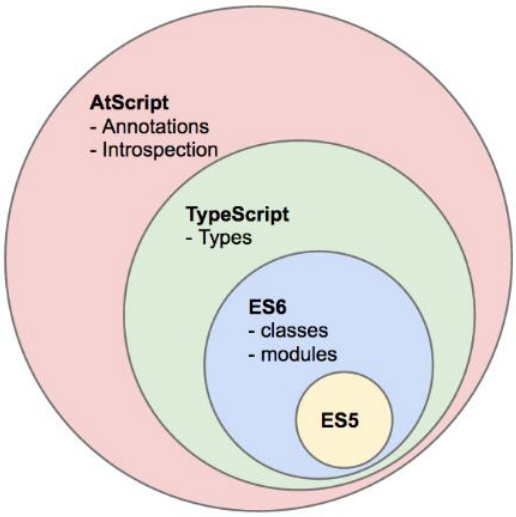
## 1.4. Γλώσσες που κάνουν “compile” σε JavaScript

Η Google τον Μάϊο του **2006** ανακοίνωσε το **GWT** (Google Web Toolkit) ένα toolkit με το οποίο επέτρεπε στους χιλιάδες JAVA προγραμματιστές της να φτιάχνουν JavaScript εφαρμογές για το web. Η συγκεκριμένη μόδα έδωσε την ώθηση να φτιαχτούν νέες γλώσσες προγραμματισμού οι οποίες κάνουν “compile” σε JavaScript, πολλές από τις οποίες είναι πιο απλές στη γραφή και πιο παραγωγικές στον προγραμματιστή. Μια πλήρης λίστα γλωσσών που έχουν άμεση σχέση (κυρίως κάνουν compile σε JavaScript) υπάρχει στο GitHub[62] (http://goo.gl/nW330g) παρόλ’ αυτά θα θέλαμε να κάνουμε μια σύντομη αναφορά στην παράγραφο αυτή σε γλώσσες που αυτή τη στιγμή ανήκουν στο παγκόσμιο hot-spot[63]:

- Objective-J (2008) είναι μια παραλλαγή της Objective-C της Apple ενώ είναι υπερσύνολο της JavaScript το οποίο σημαίνει ότι ένα πρόγραμμα JavaScript τρέχει ως έχει στην Objective-J, κάτι που συμβαίνει και στην TypeScript. Σε συνδυασμό με το Cappuccino που θα αναφερθεί στο κεφάλαιο 3.5 αποτελεί τη λύση της Apple για δημιουργία εφαρμογών web που συμπεριφέρονται σαν desktop εφαρμογές.
- CoffeeScript (Δεκέμβριος του 2009) πρόκειται για τη πρώτη γλώσσα που κάνει compile σε JavaScript και μπαίνει στο top 100 του δείκτη TIOBE τον Μάρτιο του 2015 (θέση 64 με κάτω από 1%). Ηδη υποστηρίζεται από πολλά IDE χωρίς επεκτάσεις. Σαν γλώσσα δανείζεται πολλά στοιχεία από την Ruby, την Python και την Haskell και είναι σχετικά νέα (Δεκέμβριος του 2009) ενώ ο προγραμματιστής μπορεί να γράψει και JavaScript ταυτόχρονα στον ίδιο κώδικα
- TypeScript (Οκτώβριος 2012) η γλώσσα που έφτιαξε ο δημιουργός της Turbo Pascal, του Delphi και της C# Anders Hejlsberg ο οποίος προσπάθησε (και το κατάφερε) να φτιάξει μια νέα γλώσσα-επέκταση της JavaScript που να προσφέρει κλάσεις, τύπους, modules και interfaces και τελικά να την

κάνει μια γλώσσα τύπου Java. Ολο το project είναι ανοιχτού κώδικα αλλά η σημαντική ώθηση που θα δούμε σύντομα αφορά στην ανακοίνωση που έγινε τον Μάρτιο του 2015[64] ότι η δεύτερη έκδοση της Angular κατασκευάστηκε με TypeScript. Το Angular όπως θα δούμε στην επόμενη ενότητα αποτελεί το σημαντικότερο και πρώτο σε χρήση framework της JavaScript και η κίνηση αυτή έχει τη δική της σημασία μιας που το Angular ανήκει στην Google η οποία είχε και η ίδια μια δική της επέκταση της JavaScript (και της TypeScript) που ονομαζόταν AtScript (προς το παρόν εγκαταλειμμένη από τον Μάρτιο του 2015)

- Dart (Νοέμβριος 2013) άλλη μια γλώσσα της Google από τον Lars Bak και τον Kasper Lund την οποία η Google ήθελε να κάνει μέρος του ίδιου του Chrome browser της και η οποία έχει τον δικό της dart2js JavaScript compiler. Η ίδια η γλώσσα είναι επηρεασμένη από τις C, C#, Java, διαθέτει συντακτικό που μοιάζει στην ALGOL ενώ έχει επιρροές και από την Smalltalk. Παρότι κανένας άλλος browser δεν είναι θετικός στην υιοθεσία της γλώσσας, ο κώδικάς της τρέχει στο 78% της ταχύτητας που θα έτρεχε μια καθαρογραμμένη JavaScript ενώ στο πειραματικό περιβάλλον του Chrome τρέχει 21% ταχύτερα από τον αντίστοιχο κώδικα στη μηχανή V8. Τον Οκτώβριο του 2016 η γλώσσα έκανε την επανεμφάνισή της με strong types και μια παραλλαγή της Angular με την ονομασία AngularDart 2.0 με την Google να δηλώνει ότι η γλώσσα χρησιμοποιείται σε πολλά νέα projects. Μια αντίστοιχη κίνηση έχει γίνει πάντως και από τον οργανισμό Mozilla με τη γλώσσα Sweet.js



- Στα βήματα της TypeScript και της Dart και της AtScript υπάρχουν και άλλα πειραματικά project που δίνουν type safety στη JavaScript, όπως τα πρόσφατα SoundScript (2015, Google), JS++ (2016, Onux) και Flow (2014, Facebook)
- Opal (2011) πρόκειται για γλώσσα που μεταφράζει από Ruby (δεν πρέπει να συγχέεται με τη Γερμανική functional γλώσσα Opal Project)
- Processing (2001), μια πολύ ενδιαφέρουσα γλώσσα του πανεπιστημίου MIT με δικό της περιβάλλον IDE που φτιάχτηκε για τον καλλιτεχνικό κόσμο, με αρκετές εντολές για εύκολα γραφικά (2D/3D), ήχο και πάνω από 100 βιβλιοθήκες που την επεκτείνουν. Επιπλέον χαρακτηριστικό, εκτός του ότι κάνει compile σε JavaScript κάνει compile σε Java και βγάζει native android εφαρμογές.
- Fable (F# |> BABEL) είναι μια ενδιαφέρουσα πρόσφατη (2016) προσπάθεια μεταγλώττισης της functional γλώσσας F# σε JavaScript (fsprojects.github.io/Fable/)

Φυσικά για τους λάτρεις των παλαιότερων γλωσσών, για όλες οι σημαντικές γλώσσες των τελευταίων ετών (C/C++, BASIC, LISP, PASCAL, PERL, JAVA, PYTHON) υπάρχουν μεταφραστές σε JavaScript, ακόμα και για COBOL https://github.com/ajlopez/CobolScript. Από τα πιο πρόσφατα project το JavaPoly.js (βασισμένο στο doppiojvm.org) είναι ένα κανονικό Java Virtual Machine σε JavaScript που τρέχει εντός του browser.

Ενας μεγάλος κατάλογος από γλώσσες που κάνουν compile ή επεκτείνουν την JavaScript βρίσκεται στο github.com/jashkenas/coffeescript/wiki/list-of-languages-that-compile-to-js

## 1.5. Functional Programming στην JavaScript

Όπως είπαμε και στο ιστορικό της γλώσσας (2.1) όταν ο Brendan Eich ξεκίνησε στη Netscape ο σκοπός του ήταν να φτιάξει μια γλώσσα αντίστοιχη της Scheme. Ετσι η γλώσσα έχει ήδη ορισμένα χαρακτηριστικά από συναρτησιακές γλώσσες. Στην αντίστοιχη περιοχή awesome του github (github.com/stoeffel/awesome-fp-js)

υπάρχει ένας πλήρης κατάλογος από βιβλιοθήκες που επιτρέπουν την εξερεύνηση στο συναρτησιακό προγραμματισμό χρησιμοποιώντας την JavaScript, ενδεικτικά θα αναφέρουμε ότι οι πιο δημοφιλής βιβλιοθήκες για το σκοπό αυτό είναι η Ramda, ενώ όπως προαναφέραμε, αρκετές functional γλώσσες κάνουν compile σε JavaScript (πχ Scala.js και ClojureScript)

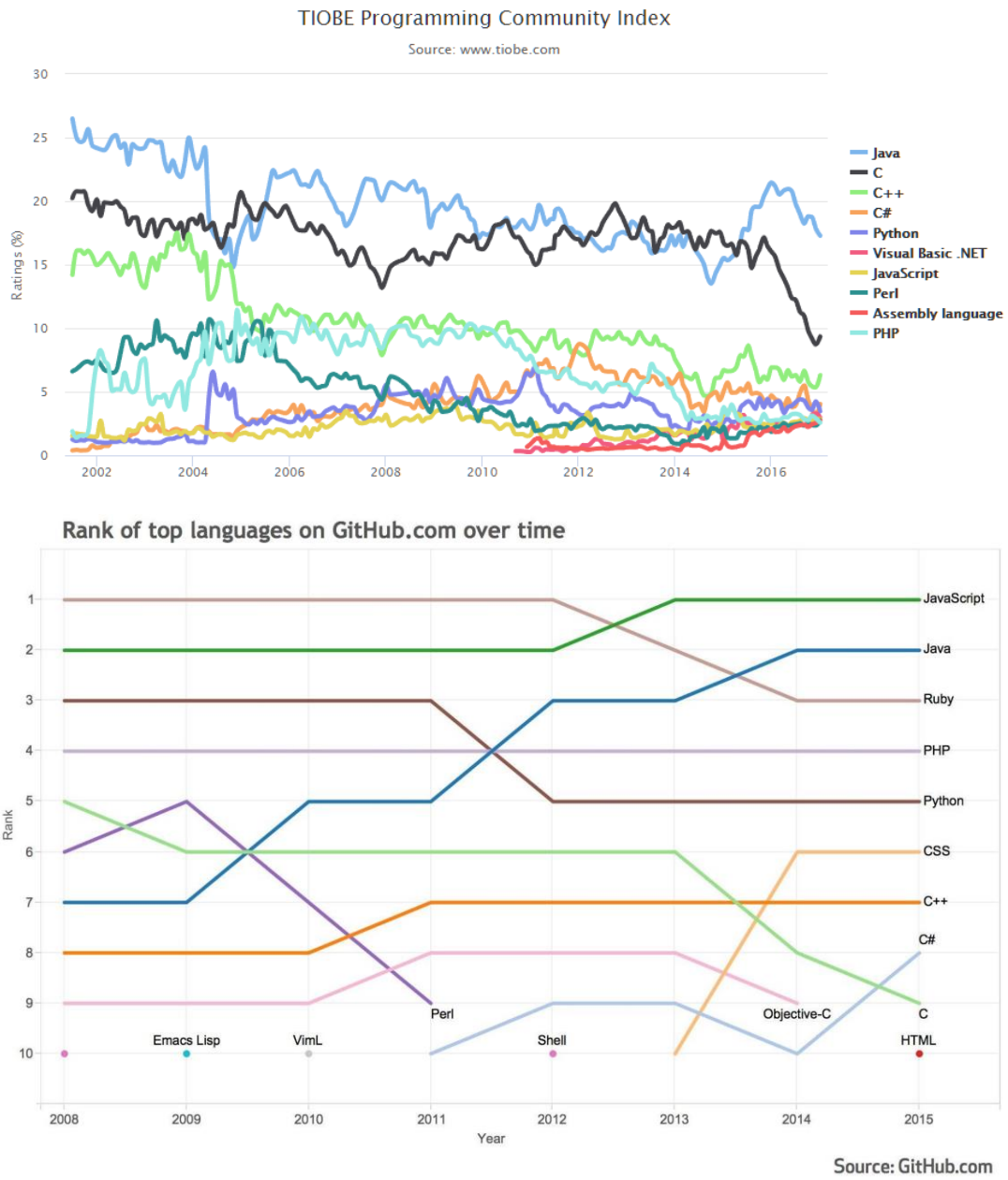
## 1.6. Reactive Programming στην JavaScript

Αντίστοιχα με το συναρτησιακό τρόπο προγραμματισμού, ένας άλλος τρόπος που επιτυγχάνεται με βιβλιοθήκες στη JavaScript είναι ο Reactive. Στην περίπτωση αυτή οι ίδιες οι μεταβλητές αλλάζουν ενώ ο κώδικας τρέχει ενώ ταυτόχρονα και αλλάζουν και οι τιμές που περιέχουν τις μεταβλητές αυτές. Πιο γνωστή βιβλιοθήκη που όμως δεν περιορίζεται στην JavaScript είναι η RxJS (github.com/Reactive-Extensions/RxJS) η οποία βασίζεται στο reactivex.io Τη λογική του reactive programming θα τη δούμε αρκετές φορές είτε με τη βιβλιοθήκη meteor είτε με το react είτε με τα observables που γίνονται μέρος της ίδιας της γλώσσας.

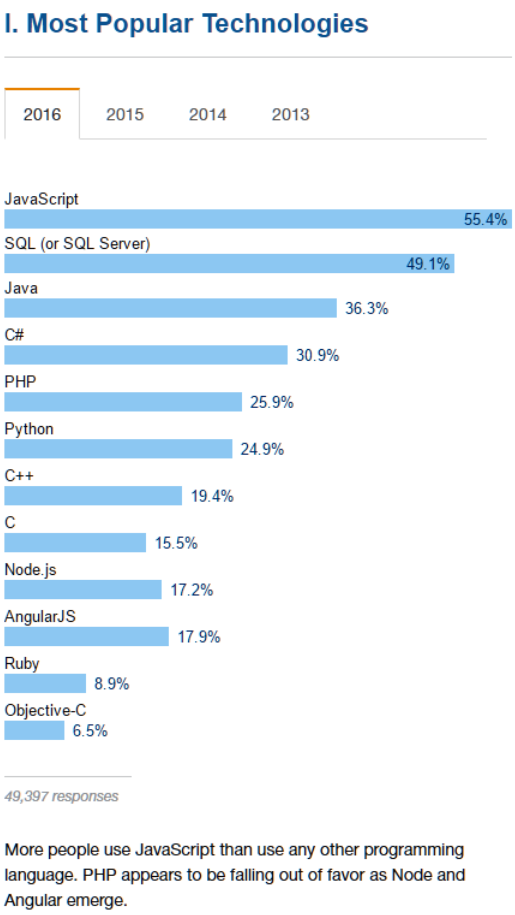
## 1.7. Παγκόσμια κατάταξη της JavaScript

Αν και είναι δύσκολο να κατατάξουμε την JavaScript λόγω του περίεργου τρόπου με τον οποίο αυτή χρησιμοποιείται, ο έγκυρος δείκτης TIOBE που εξετάζει ετήσια πάνω από 500 εκατομμύρια γραμμές κώδικα, την κατατάσσει το 2016 7η γλώσσα σε χρήση στο κόσμο[51] ενώ σύμφωνα με το δείκτη RedMonk[52] είναι η #1 σε χρήση γλώσσα και στο GitHub, κυρίως λόγω της εκρηκτικής ανόδου στη χρήση των JavaScript frameworks[53] που θα αναλυθούν περισσότερο παρακάτω.

| Ιαν 2017 | Ιαν 2016 | Αλλαγη | Γλώσσα Προγραμματισμού | Δημοτικότητα | Αλλαγή |
|---|---|---|---|---|---|
| 1 | 1 | | Java | 17278 | -4.19% |
| 2 | 2 | | C | 9349 | -6.69% |
| 3 | 3 | | C++ | 6301 | -0.61% |
| 4 | 4 | | C# | 4039 | -0.67% |
| 5 | 5 | | Python | 3465 | -0.39% |
| 6 | 7 | ^ | Visual Basic .NET | 2960 | +0.38% |
| 7 | 8 | ^ | JavaScript | 2850 | +0.29% |
| 8 | 11 | ^ | Perl | 2750 | +0.91% |
| 9 | 9 | | Assembly language | 2701 | +0.61% |
| 10 | 6 | vv | PHP | 2564 | -0.14% |
| 11 | 12 | ^ | Delphi/Object Pascal | 2561 | +0.78% |
| 12 | 10 | v | Ruby | 2546 | +0.50% |
| 13 | 54 | ^^ | Go | 2325 | +2.16% |
| 14 | 14 | | Swift | 1932 | +0.57% |
| 15 | 13 | v | Visual Basic | 1912 | +0.23% |
| 16 | 19 | ^ | R | 1787 | +0.73% |
| 17 | 26 | ^^ | Dart | 1720 | +0.95% |
| 18 | 18 | | Objective-C | 1617 | +0.54% |
| 19 | 15 | vv | MATLAB | 1578 | +0.35% |
| 20 | 20 | | PL/SQL | 1539 | +0.52% |

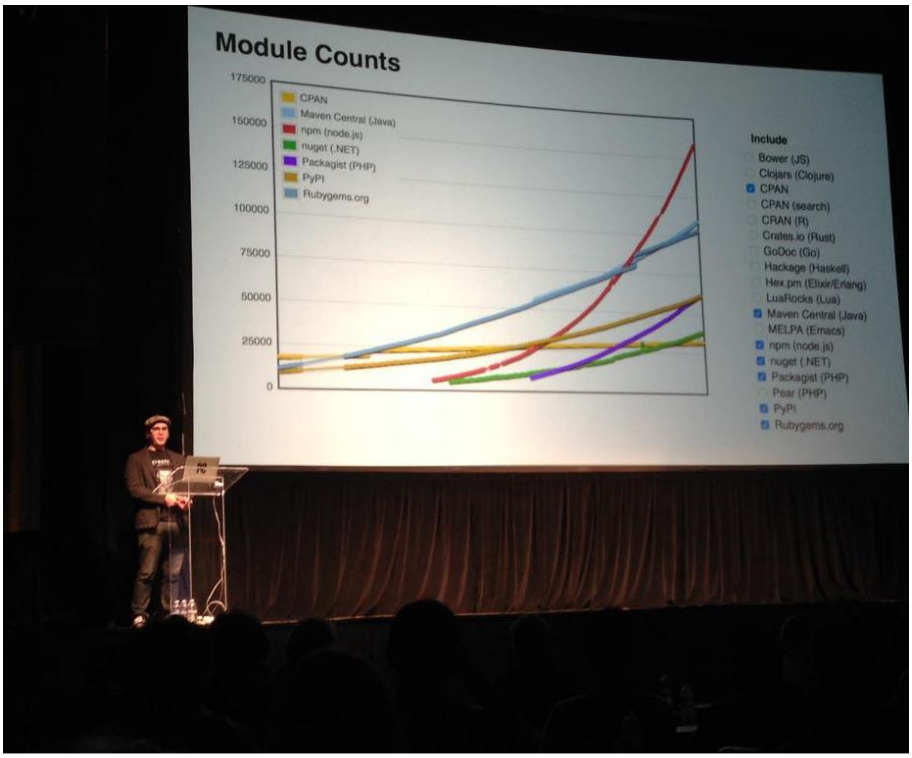



Όπως ανέφερε ο Matt Mullenweg, ο δημιουργός του WordPress (η πλατφόρμα που χρησιμοποιούν το 60% των sites που βασίζονται σε CMS, και σχεδόν το ¼ των sites συνολικά παγκοσμίως, η οποία ξαναγράφεται σε JavaScript) «η JavaScript δεν είναι απλά το μέλλον του WordPress αλλά και το μέλλον του web, θα επιτρέψει το WordPress να ευδοκιμήσει για τα επόμενα 13 χρόνια[89].




Παραπάνω βλέπουμε την έρευνα που έγινε από το stackoverflow με 49397 (!) απαντήσεις. Όπως θα διαπιστώσουμε και από το ιστορικό της γλώσσας, όλες οι γλώσσες προγραμματισμού ακολουθούν μόδες και σημαντικά γεγονότα επηρεάζουν τη χρήση τους, τις εφαρμογές τους και τη γενικότερη συνεχιζόμενη υποστήριξή τους. Θα αναλύσουμε αυτά τα γεγονότα στην περίπτωση της JavaScript και θα διαπιστώσουμε ότι καθόλου τυχαία η JavaScript είναι πρώτη στις προτιμήσεις τα 4 τελευταία χρόνια

# ΚΕΦ.2: Full Stack JavaScript




**freeCodeCamp** @freeCodeCamp · 24 Jun 2015
Node.js is the biggest open source ecosystem that has ever existed. @mikeal #jQuerySF

Φωτογραφία: Παρουσίαση του npm στο freeCodeCamp: «το node.js είναι το μεγαλύτερο οικοσύστημα που κατασκευάστηκε ποτέ»

## 2.1. ο ρόλος του Node

Η ιδέα του να τρέχει η JavaScript σε περιβάλλον εκτός browser (όπου δεν υπάρχει το DOM) ξεκίνησε από το Δεκέμβριο του 1995 όταν η Netscape παρουσίασε τον Enterprise Server 2.0[72]. Όμως η μεγάλη ώθηση στην ιδέα ήρθε σχεδόν 15 χρόνια μετά, το 2009, όταν δημιουργήθηκε το project ServerJS (το οποίο μετονομάστηκε σε CommonJS) που είναι ένα σετ από τεχνολογίες που ορίζουν πως θα λειτουργεί η JavaScript από την πλευρά του server. Από το project αυτό γεννήθηκαν διάφορες εφαρμογές, μερικές από τις οποίες θα παρουσιαστούν στο επόμενο κεφάλαιο 3.6 όπως οι βάσεις CouchDB, MongoDB και η Wakanda.

Την ίδια εποχή η Google παρουσίαζε τον δικό της browser Chrome με τη νέα της μηχανή V8 (Σεπτέμβριο του 2008, https://code.google.com/p/v8/) η οποία βελτίωνε θεαματικά την ταχύτητα εκτέλεσης εφαρμογών JavaScript.

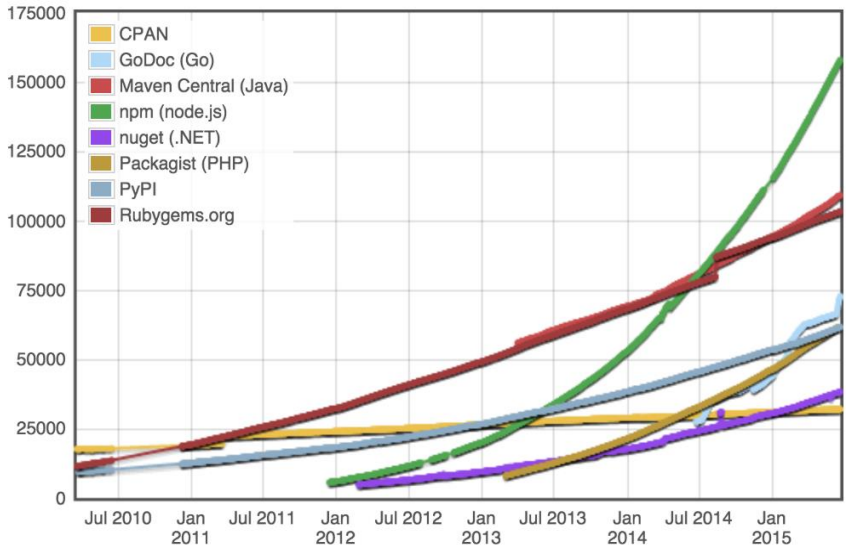
Αρχες του 2009 ο μαθηματικός Ryan Dahl ο οποίος έφτιαχνε modules για τον NGINX webserver έψαχνε τρόπο για να φτιάξει ένα σύστημα που θα απαντούσε σε http ερωτήματα χωρίς να περιμένει το αποθηκευτικό μέσο να απαντήσει (το σκληρό δίσκο). Πρόκειται για το σημαντικότερο πρόβλημα που κάνει αργούς τους web servers, και η απλούστερη λύση που βρήκε ήταν η JavaScript με τα events που εκτελούνται ασύγχρονα.

Όπως ο ίδιος ανέφερε στην παρουσίαση του Node.js τον Νοέμβριο του 2009 (youtu.be/ztspvPYybIY) ο σκοπός του ήταν να τρέχει non-blocking κώδικας, δηλαδή το πρόγραμμα να μη σταματά περιμένοντας τα δεδομένα να έρθουν από το αργό μέσο, και έτσι ο web server να απαντά άμεσα στο χρήστη. Όπως ο ίδιος ανέφερε[73] «αρχικά δεν ήθελα να χρησιμοποιήσω JavaScript, εκείνη την εποχή παρουσιάστηκε η V8 και είχα μια μάλλον ξαφνική επιφοίτηση ότι η Javascript ήταν η τέλεια γλώσσα για αυτό που ήθελα».


Φωτογραφία: διαμοιράζοντας 34 δισσεκατομύρια πακέτα λογισμικού JavaScript μέσα σε ένα χρόνο

## 2.2. o Node Package Manager

Αρχες του 2010 υπήρξε μια ακόμα σημαντική προσθήκη στον κόσμο της JavaScript και του Node ειδικότερα, που ακόμα περισσότερο εκτόξευσε τη χρήση του. Ο Isaac Z. Schlueter που τότε εργαζόταν στην Yahoo! στο team του YUI, ενθουσιασμένος από τη χρήση του node ξεκίνησε να φτιάχνει έναν package manager που να διευκολύνει τη χρήση των modules. Η έκδοση 0.1.0 παρουσιάστηκε τον Μάιο του 2010 αλλά η πραγματική επανάσταση ξεκίνησε το 2012 όπως βλέπουμε από το γράφημα:

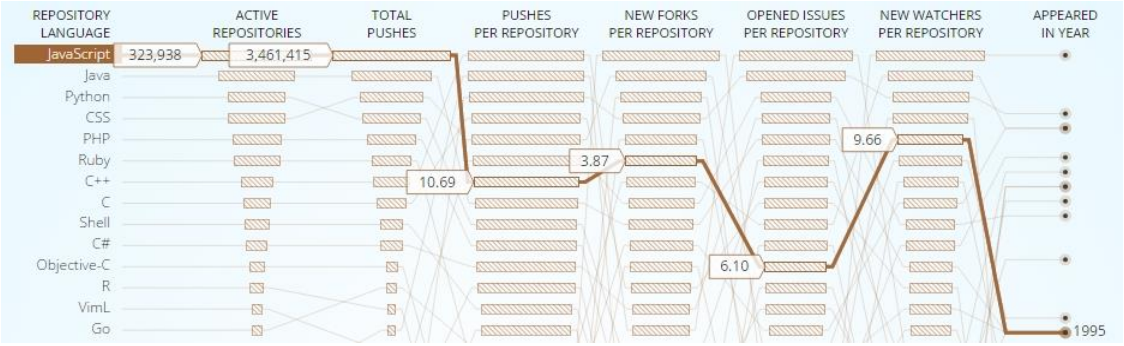



Αυτή τη στιγμή 380 χιλιάδες βιβλιοθήκες με περισσότερα από **3 δισεκατομμύρια downloads το μήνα** αποτελεί τη δημοφιλέστερη πλατφόρμα ανάπτυξης εφαρμογών στην πλευρά του server δημιουργώντας μια νέα κατηγορία προγραμματιστών «full stack» που κατανοούν και αναπτύσσουν σε όλο το εύρος της υποδομής: στο front end με JavaScript, στον server με Node (JavaScript) και με βάσεις δεδομένων που αποθηκεύουν δεδομένα και μοντέλα σε JSON. Η τελευταία εξέλιξη με την παρουσίαση του Browserify (2011) είναι να μετατρέπονται οι βιβλιοθήκες του node για εκτέλεση στον browser. Εφαρμογές Javascript που τρέχουν είτε στον browser είτε στον server το τελευταίο διάστημα ονομάζονται ισομορφικές (Isomorphic JavaScript).

Το 2011 ήταν και το έτος που το LinkedIn στήριξε όλη τη mobile εφαρμογή τους στο Node. Ενώ μεγαλύτερη στροφή εταιρείας στο Node.js έχει κάνει το PayPal δεδομένης και της πρόσληψης του Douglas Crockford το 2012.

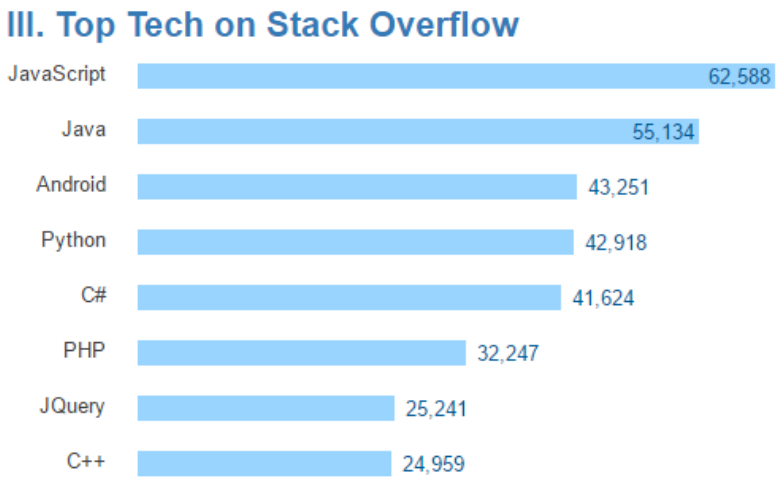
Το αποτέλεσμα της ανάπτυξης του οικοσυστήματος του NodeJS στο GitHub

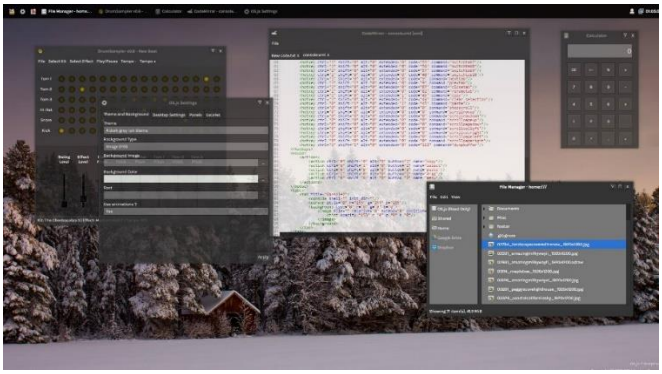
| | Language | Active Repositories | | Language | Active Repositories |
|---|---|---|---|---|---|
| **1** | JavaScript | 323,938 | **6** | Ruby | 132,848 |
| **2** | Java | 222,852 | **7** | C++ | 86,505 |
| **3** | Python | 164,852 | **8** | C | 73,075 |
| **4** | CSS | 164,585 | **9** | Shell | 65,670 |
| **5** | PHP | 138,771 | **10** | C# | 56,062 |



Φωτογραφία: από githut.info)

Σύμφωνα με τα επίσημα αποτελέσματα του Developer Survey του 2016 στο StackOverflow[85], ανάμεσα σε 56 χιλιάδες προγραμματιστές από 173 χώρες, το 28% δηλώνει Full Stack developers, εκ των οποίων το 85% σε JavaScript με 2η γλώσσα να έρχεται η C# και η PHP με λιγότερο από 37%, ενώ 27% καταλαμβάνει από μόνο του το Node. Η JavaScript είναι η πιο δημοφιλής γλώσσα στο StackOverflow με 55% (36% η Java, 30% η C#, 25% η PHP και η Python) ενώ το framework React είναι η τεχνολογία με τη μεγαλύτερη άνοδο (311%) από το 2015

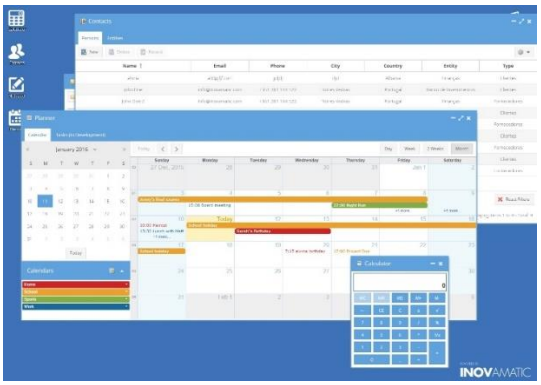




Φωτογραφία του OS.js το Νοέμβριο του 2014

## 2.3. **NodeOS – λειτουργικό σε Node**

Δεδομένης της γεωμετρικής εξάπλωσης του NodeJS, είναι εύκολο να υποθέσουμε ότι ένα λειτουργικό σύστημα θα εμφανιζόταν βασισμένο στο Node ώστε να εκμεταλλευτεί τις χιλιάδες βιβλιοθήκες του npm. Ετσι τον Ιούλιο του 2013 ο Jacob Groundwater ξεκίνησε το NodeOS (github.com/NodeOS) χρησιμοποιώντας τον πυρήνα του linux ενώ στα τέλη του 2014 ο Sergii Iefremov ξεκίνησε την ανάπτυξη του Runtime.JS (github.com/runtimejs) το οποίο αναμένεται να γίνει το kernel του NodeOS στο μέλλον. Το μόνο κομμάτι του πυρήνα που είναι γραμμένο σε C++ είναι αυτό που χρειάζεται προσπέλαση στον επεξεργαστή, τη μνήμη και τη μηχανή V8.



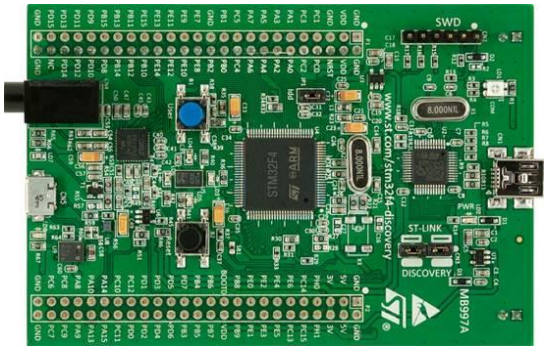
Φωτογραφία του webtop από το wtdemo.inovamatic.com

## 2.4. **OS.js – λειτουργικό στον browser**

Αντίθετα με το NodeOS ένα πιο παλιό project, από τον Νοέμβριο του 2011 με την αρχική ονομασία desktop.js φιλοδοξούσε να γίνει ένα ακόμα λειτουργικό σύστημα που τρέχει μέσα στον browser.

Στην σημερινή του μορφή το OS.js του Νορβηγού Anders Evenrud βρίσκεται στο github.com/andersevenrud/OS.js-v2 όταν και ξαναγράφτηκε από την αρχή δύο χρόνια αργότερα, τον Νοέμβριο του 2013. Τον Ιούλιο του 2015 ο προγραμματιστής έφτιαξε έναν δικό του Google Mail client ο οποίος τρέχει μέσα στο web λειτουργικό. Στο github υπάρχει και ο κώδικας (Node) για να τρέξει το λειτουργικό στο Arduino Yun των 75 ευρώ, με βάση το linux και την MySQL για authentication. Στο εκτενές documentation υπάρχει API για client (os.js.org/doc/client) και server (os.js.org/doc/server).

Παρόμοια ενδιαφέροντα αλλά commercial projects είναι τα zeropc.com και webtop της Πορτογαλλικής Innovate




Φωτογραφία του STM32F4DISCOVERY

## 2.5. **Internet Of Things**

Ένα open source project της Samsung με την ονομασία IoT.js απευθύνεται σε όσους θέλουν να γράψουν με JavaScript σε συσκευές με περιορισμένη μνήμη και επεξεργαστική ισχύ. Η πλατφόρμα (samsung.github.io/iotjs/) χρησιμοποιεί μια πολύ μικρή JavaScript μηχανή (χωράει σε μόλις 200KB ROM, με ανάγκες μικρότερες από 64KB RAM) που η Samsung ονομάζει JerryScript (github.com/Samsung/jerryscript). Παρόλα αυτά από ότι αναφέρει η ίδια η Samsung[87] το περιβάλλον είναι συμβατό με Node καθώς ακολουθεί το specification CommonJS.

Το IoT.js ήδη τρέχει στα λειτουργικά Linux και NuttX (realtime) σε πλακέτες Raspberry Pi και STM32F4DISCOVERY της STMicroelectronics. Ένα παρόμοιο project έχει ξεκινήσει και η Microsoft (με τη νέα μηχανή της Chakra) για το Windows 10 IoT, υιοθετώντας όμως το Node σε επεξεργαστές ARM[88].

Φωτογραφία του puck.js από το KickStarter

Άλλο ένα επίσης ενδιαφέρον project της Espruino (κατηγορίας Physical Web / Beacon) είναι το Puck.js (φωτογραφία κάτω) με ενσωματωμένη JavaScript, Web Bluetooth Smart 5.0 χαμηλής ενέργειας με εμβέλεια 80 μέτρα, NFC, 64kB

RAM/512kB Flash, αισθητήρα θερμοκρασίας, φωτός, μαγνητόμετρο, πυξίδα, ενώ εκπέμπει πακέτα στις κοντινές συμβατές συσκευές (iPhone, Android κλπ) ανοίγοντας ένα νέο κόσμο εφαρμογών


Φωτογραφία του mCookie πάνω σε αυτοκίνητο LEGO™

## 2.6. JavaScript Robotics

Ένα βήμα παραπέρα από το IoT μας προχωρούν οι βιβλιοθήκες Johnny-Five και Cylon.js που απευθύνονται σε ρομποτικές συσκευές. Η Cylon μάλιστα υποστηρίζει 43 πλατφόρμες (cylonjs.com) που περιλαμβάνουν από λάμπες μέχρι μικρά ρομποτάκια.

```
var Cylon = require('cylon');
Cylon.robot({
  connections: { bluetooth: {adaptor: 'central', uuid:
‘d03972a24e55', module: 'cylon-ble'}},  devices: {mip: {driver:
'mip'}},  work: function(my) {
    my.mip.setHeadLED(2, 2, 2, 2);
    after((2).seconds(), function() {
      my.mip.driveDistance(0, 10, 0, 0);
    });
    after((3).seconds(), function() {
      my.mip.setHeadLED(1, 1, 1, 1);
    });  }
}).start();
```

## 2.7. Πρωτόκολλο Ηλεκτρονικών Συναλλαγών (Interledger Protocol)

Πρόκειται για μια νέα προσπάθεια (2015) από τους Stefan Thomas, Evan Schwartz και Adrian Hope-bailie, να δημιουργηθεί ένα νέο πρωτόκολλο με την υποστήριξη του W3C για να γίνονται ηλεκτρονικές συναλλαγές μεταξύ λογαριασμών[101] με ασφαλή τρόπο ασχέτως ποιος οργανισμός παρεμβάλλεται. Η τεχνολογία από τον Οκτώβριο του 2016 αποτελεί μέρος του JS Foundation.

Παράδειγμα αποστολής χρημάτων από interledger.org και github.com/interledgerjs/ilp

```
import WalletClient from 'five-bells-wallet-client'

const client = new WalletClient({
  username: 'alice@red.ilpdemo.org',
  password: 'secret'
})

setInterval(() => {
  client.send({
    destination: 'bob@blue.ilpdemo.org',
    destinationAmount: '0.01',
    message: 'Still love you!'
  })
}, 1500)
```

# ΚΕΦ.3: JavaScript Frameworks


Φωτογραφία: Δημοφιλή Frameworks του 2016

## 3. JavaScript Frameworks

### 3.1. Σύντομος κατάλογος frameworks

Όπως περιγράψαμε και στο ην αρχή της ενότητας, η εκρηκτική άνοδος στη χρήση της JavaScript οφείλεται ιδίως στα JavaScript frameworks. Τα frameworks στην JavaScript είναι κάτι παραπάνω από βιβλιοθήκες που επεκτείνουν τη χρήση της γλώσσας. Σε πολλές περιπτώσεις αλλάζουν την ίδια τη γλώσσα, τον τρόπο προγραμματισμού, τη χρήση design patterns, τον τρόπο διαμόρφωσης των CSS αλλά και του DOM και σε μερικές περιπτώσεις φέρνουν την JavaScript πιο κοντά στην Java ή άλλες γλώσσες προγραμματισμού όπως γίνεται σε μεγάλο βαθμό πλέον με το γνωστότερο framework Angular. Μπορούμε να πούμε ότι η JavaScript εξελίσσεται ως γλώσσα και εξαιτίας των βιβλιοθηκών που δημιουργούνται για τη γλώσσα.

Παρακάτω παρουσιάζεται μια λίστα με τα σημαντικότερα Frameworks

- **Express**: θεωρείται η βάση για όσους θέλουν να ξεκινήσουν με το Node για να φτιάχνουν εφαρμογές στο λεγόμενο MEAN stack (κεφάλαιο 3.7)
- **jQuery**: ίσως η πιο διαδεδομένη βιβλιοθήκη στο internet αυτή τη στιγμή, αφού χρησιμοποιείται από όλους τους προγραμματιστές web εφαρμογών

ανεξαρτήτως της γλώσσας στον server. Είναι από τις πρώτες βιβλιοθήκες (Αύγουστος 2006) που χρησιμοποίησαν την τεχνολογία AJAX για να απλοποιήσουν διαδικασίες.

- **AngularJS**: Πρωτοεμφανίστηκε το 2009 από την Brat Tech η οποία εξαγοράστηκε από την Google η οποία συνεχίζει να το εξελίσσει μάλιστα σε συνεργασία με την Microsoft αφού η νέα έκδοσή του κάνει χρήση της TypeScript. Για το 2015 είναι το πιο δημοφιλές framework για ανάπτυξη εφαρμογών τύπου MVC σε JavaScript ενώ η νέα του έκδοση 2.0 στηρίζεται ιδιαίτερα και από την Microsoft καθώς είναι γραμμένο κυρίως σε TypeScript που είναι γλώσσα της Microsoft όπως αναφέρθηκε παραπάνω. Εκτός του ότι είναι το πιο δημοφιλές framework σε χρήση σήμερα, το Angular είναι με μέρος του MEAN stack που θα αναφερθεί περισσότερο στο επόμενο κεφάλαιο.
- **Aurelia:** Φτιαγμένο από τον Rob Eisenberg της εταιρείας Durandal το 2014-2015, ο οποίος ήταν μέλος της ομάδας ανάπτυξης της Angular. Όπως είχε δηλώσει ο ίδιος[97] θεώρησε ότι η Angular γίνεται πολύ "βαριά". Η έκδοση 1.0 εμφανίστηκε επίσημα τέλη Ιουλίου του 2016
- **Ember.js**: Πρωτοεμφανίστηκε το 2007 από τον Yehuda Katz και σύντομα υιοθετήθηκε από τη Yahoo και το Groupon. Θεωρείται ανταγωνιστικό της Angular για δημιουργία εφαρμογών τύπου MVC ενώ η 2$^{η}$ έκδοσή του εμφανίστηκε μόλις τον Αύγουστο του 2015 με αρκετές ομοιότητες με το React.
- **ReactJS**: Πρόσφατο (2013) δημιούργημα του Facebook που αφορά κυρίως το View κομμάτι (γρήγορο manipulation μέσω ενός Virtual DOM), στηρίζει εκτός από το δημοφιλέστερο κοινωνικό δίκτυο και το Instagram της ίδιας εταιρείας αλλά και τα Netflix και AirBNB. Σημαντική εξέλιξη ότι το Yahoo εγκαταλείπει το δικό του framework (YUI) και αρχικά για το Yahoo Mail αγκαλιάζει το React (Subramanyan Murali, Engineering Manager του Yahoo, Σεπτέμβριος 2014). Συνδυάζεται με άλλα frameworks για το MVC κομμάτι, με υλοποιήσεις μαζί με το Angular, το Backbone και άλλα παρότι το ίδιο το Facebook προτείνει ένα νέο μοντέλο ανάπτυξης με την ονομασία Flux για αντικαταστάτη του MVC. Είναι σημαντικό να αναφέρουμε ότι ήδη τώρα που γράφονται αυτές οι γραμμές το React στο github έχει 46000 αστέρια σε σχέση με τα 14000 της Angular, 34000 του Meteor, 16000 του Ember, 26000 του Express, ανακηρύσσοντάς το 6$^{ο}$ πιο σημαντικό project σε όλο το GitHub. Το ReactJS εισάγει μια νέα μορφή αρχείων

(JSX) αποσυνδέοντας τη JavaScript από το HTML που πλέον παράγεται από αυτό (περισσότερα στο JSX in Depth: **goo.gl/52xYb5**). Το React είναι ήδη κομμάτι αρκετών μεγαλύτερων framework σε μόλις 3 χρόνια, ενώ *αποτελεί κεντρικό κομμάτι του ReactNative που χρησιμοποιεί το Facebook για την ανάπτυξη των mobile εφαρμογών του*.

- **Meteor**: Ένα πολλά υποσχόμενο framework βασισμένο στο Node.js και την MongoDB. Βασικό χαρακτηριστικό της η reactive σχέση browser και server. Στον browser υπάρχει μια μικρή βάση δεδομένων που συνεχώς συγχρονίζεται με την βάση δεδομένων στον server με αποτέλεσμα πολλά ανοιχτά site να δείχνουν άμεσα τις αλλαγές που προέρχονται από τα συνδεμένα site. Με το Meteor θα ασχοληθούμε περισσότερο στο κεφάλαιο 4.
- **Meatier**: αποτελεί το παράδειγμα ενός project που εμπνεύστηκε από το Meteor (αρκετά πρόσφατο, 2016) αντικαθιστώντας τα βασικά components από άλλα πιο πρόσφατα αλλά όχι τόσο διαδεδομένα ακόμα. Για παράδειγμα χρησιμοποιεί την RethinkDB αντί την MongoDB, το GraphQL και το React (του Facebook) αντί τα Collections και το Blaze κλπ. Η αναφορά γίνεται κυρίως για να δείξουμε ότι όλο και περισσότερα frameworks θα εστιάζουν πλέον σε λογικές *reactive*, δηλαδή site που είναι συνεχώς online και διαμορφώνονται δυναμικά χωρίς να γίνεται ποτέ refresh μια σελίδα.
- **VUE**: πρόκειται για νέο (Φεβρουάριος 2014) αλλά γρήγορα ανερχόμενο templating framework εναλλακτικό του react (για front end UI) με στοιχεία reactivity, routing, modules κλπ. Γραμμένο από τον Evan You πρώην προγραμματιστή του Meteor, με την έκδοση 2 (2016) η χρήση του εκτοξεύτηκε.
- **Knockout**: ένα ακόμα δημοφιλές framework που ξεκίνησε το 2010 από τον Steve Sanderson, υπάλληλο της Microsoft, για να υιοθετήσει το μοντέλο ανάπτυξης MVVM στην JavaScript.
- **Backbone.js και Underscore.js**: Με κύρια συνεισφορά από τον Jeremy Ashkenas, τον σχεδιαστή της γλώσσας CoffeeScript. Το backbone ξεκίνησε το 2010 (είναι βασικό στοιχείο του WordPress, 75 εκατομμύρια sites βασίζονται σ αυτό) με μοντέλο ανάπτυξης MVP και Actor, ενώ το underscore αποτελεί κυρίως μια βιβλιοθήκη με functions που εκτελούν διάφορες λειτουργίες, επεκτείνοντας τη γλώσσα. Μη ξεχνάμε ότι η JavaScript θεωρείται και functional

γλώσσα. Μαζί με το **Prototype.js** θεωρούνται οι σημαντικότερες βιβλιοθήκες με χρήση περισσότερη από το 5% όλων των sites παγκοσμίως.

- **Socket.IO**: πρόκειται για δύο βιβλιοθήκες, μία στον browser και μία στον server, που επιτρέπουν την μεταξύ τους επικοινωνία για *realtime* εφαρμογές με κλασικό παράδειγμα το chat.
- **Capuccino**: η βιβλιοθήκη της Apple για ανάπτυξη web εφαρμογών που θα συμπεριφέρονται όπως οι αντίστοιχες desktop Mac OS X εφαρμογές. Η Apple την συνδυάζει με τη δική της γλώσσα Objective-J (class-based) η οποία κάνει compile σε JavaScript.
- **D3.js**: είναι ***η δημοφιλέστερη*** JavaScript βιβλιοθήκη για παραγωγή στατιστικών (και όχι μόνο) γραφημάτων στο Internet ενώ αποτελεί και βάση για άλλες βιβλιοθήκες γραφικών. Χρησιμοποιείται στο front end ανεξαρτήτως της γλώσσας στο backend. Η ανάπτυξη της ξεκίνησε το 2011 στο Stanford βασισμένη σε προηγούμενη δουλειά του 2009 για τη βιβλιοθήκη Protovis που επίσης είχε βασιστεί σε προηγούμενη δουλειά του καθηγητή τους Jeff Herr το 2005 όταν σχεδίαζε το Prefuse (για τον κόσμο της Java).
- **Ext JS**: πρόκειται για framework στο οποίο βασίζονται πάρα πολλές εφαρμογές τύπου LOB (line of business) σε επιχειρηματικά περιβάλλοντα, δηλαδή πολλές φόρμες με πολλές διαδικασίες. Το Ext JS είχε ξεκινήσει το 2007 ως βιβλιοθήκη-επέκταση του YUI, του βασικού framework του Yahoo του οποίου η εξέλιξη σταμάτησε το 2014.
- **Handlebars**: ξεκίνησε από τον Yehuda Katz του Ember.js ως ένα σύστημα για templating στο front end και χρησιμοποιείται τόσο από το ember όσο και από άλλα frameworks όπως το meteor και το derby.
- **Konva:** βασισμένο στο **Kinetic.js** που έπαψε να συντηρείται στα τέλη του 2014, πρόκειται για βιβλιοθήκη στην οποία στηρίζονται άλλες βιβλιοθήκες για animation
- **Kendo UI**: της πολύ γνωστής εταιρείας Telerik, μια βιβλιοθήκη με έτοιμα εργαλεία για front end widgets όπως διαγράμματα, λίστες, ημερολόγια, editors, στατιστικά διαγράμματα κλπ η οποία χρησιμοποιείται σε συνδυασμό με πολλές γλώσσες προγραμματισμού στο backend για ανάπτυξη LOB εφαρμογών.

- **MooTools**: σχεδόν παλιά όσο το jQuery (Σεπτέμβριος 2006) αλλά όχι τόσο διαδεδομένη βιβλιοθήκη που διευκολύνει την ανάπτυξη αντικειμενοστραφούς JavaScript με κλάσεις κλπ χωρίς όμως να υπάρχει και το αντίστοιχο του jQueryUI front end interface.
- **Durandal** και **Aurelia:** πρόκειται για frameworks που υποστηρίζουν τρόπους ανάπτυξης MVC, MVP και MVVM σε JavaScript. Πολλοί τα βλέπουν ως αντίπαλους της Angular δεδομένης μάλιστα της ανάπτυξης του Aurelia από τον Rob Eisenberg ο οποίος ήταν στην ομάδα ανάπτυξης της Angular 2 την οποία εγκατέλειψε κατηγορώντας ότι το project ξέφευγε από τον αρχικό σκοπό.
- **Polymer**: πολλά υποσχόμενο και πολυδιαφημισμένο project της Google που βασίζεται στην τεχνολογία των web components. Λόγω της μη-υποστήριξης από την Microsoft (Internet Explorer, Edge) ενδέχεται η τεχνολογία στο μέλλον να ενσωματωθεί στην Angular. Μαζί με την τεχνολογία αυτή η Google εισήγαγε και το λεγόμενο **Material Design** για το UI κομμάτι.
- **Wakanda**: πρόκειται για το framework της παλιάς (δεκαετία του 80) Γαλλικής εταιρείας 4D που ειδικεύεται σε βάσεις δεδομένων. Παρόμοια με το Polymer χρησιμοποιεί μια αντίστοιχη τεχνολογία των "widgets" τα οποία στην περίπτωση της Wakanda συνδέονται με τη δική της βάση δεδομένων WakandaDB. Η ίδια η βάση απαντάει σε ερωτήματα που έχουν σύνταξη τύπου javascript και όχι SQL.
- **SAP OpenUI5**: Η JavaScript βιβλιοθήκη της SAP (SapUI5) που έγινε πρόσφατα (τέλη 2013) opensource (github.com/SAP/openui5). βασισμένη κυρίως στο jQuery, η βιβλιοθήκη προσφέρει εκτός από πολλά controls και binding με πιο μονολιθικά πρωτόκολλα όπως XML αλλά και OData που η SAP είναι ο μεγαλύτερος υποστηρικτής της.
- **Sails**: Γραμμένο πριν 2 χρόνια από τον Mike McNeil και αρκετά παρόμοιο με το Meteor, με τη διαφορά ότι το ORM του υποστηρίζει πολλές περισσότερες βάσεις δεδομένων (MySQL, Mongo, Postegre, Reids) και δυνατότητα να δημιουργεί αυτόματα REST API.
- **Falcor**: Από τα πιο πρόσφατα frameworks ανεπτυγμένο από την εταιρεία Netflix. Η εταιρεία το αναφέρει κυρίως ως middleware ανάμεσα στην εφαρμογή

και τη βάση, ουσιαστικά παρουσιάζει μια παραλλαγή του JSON με το όνομα JSON Graph με το οποίο περνάει ερωτήματα προς τη βάση.

- **InfernoJS**: Πολύ πρόσφατη React like βιβλιοθήκη την οποία αναφέρουμε λόγω υποστήριξης από τον δημιουργό της JavaScript Brendan Eich

Τα παραπάνω frameworks μπορεί να είναι στη μόδα την εποχή που γράφεται η εργασία (2015-2016) χωρίς να σημαίνει ότι θα είναι και σε ένα χρόνο από σήμερα. Όχι μόνο εξελίσσονται με απίστευτους ρυθμούς κυρίως λόγω του τρόπου του open source development (GitHub) αλλά δημιουργούν και το λεγόμενο “JavaScript fatigue” από την κούραση που προκαλεί κάθε επιπλέον νέο framework. . Παρόλα αυτά η επιλογή ενός από τα παραπάνω αποτελεί στρατηγική επιλογή για μια εταιρεία software καθώς σε μεγάλο βαθμό “δένεται” με αυτό και τις δυνατότητες που προσφέρει.


Φωτογραφία: διάγραμμα χρήσης των JavaScript frameworks (InfoQ)

### 3.2. Ερευνα της InfoQ «JavaScript frameworks in the Real World»)

Για την καλύτερη κατανόηση της τρέχουσας κατάστασης θα θέλαμε να αναφέρουμε την online έρευνα που επαναλήφθηκε μετά από 2 χρόνια στον ιστότοπο InfoQ[65] από 586 προγραμματιστές (μέσα Ιουνίου του 2015) ως προς την αξία χρήσης και το βαθμό της υιοθεσίας τους. Όπως βλέπουμε και από το γράφημα (και τα στατιστικά στο τέλος της παραγράφου) το πιο δημοφιλές Framework είναι το Angular. Παρόλ' αυτά τις μέρες που γράφεται αυτή η εργασία το framework που απολαμβάνει της μέγιστης προσοχής των προγραμματιστών είναι το ReactJS (ανάπτυξη >40% από μήνα σε μήνα).

Τα περισσότερα frameworks που δεν εμφανίζουν ποσοστά το 2013 είναι πρωτο-εμφανιζόμενα. Αντίστοιχα πολλά frameworks του 2013 εξαφανίστηκαν στο συγκεκριμένο ερωτηματολόγιο, πράγμα που δείχνει την τάση (μόδα) της εποχής. Το AngularJS παραμένει στην κορυφή παρότι η Google έχει παρουσιάσει και το νέο της (δικό της, όχι αγορασμένο) framework polymer. Τα frameworks που δεν εμφανίστηκαν στο ερωτηματολόγιο του 2015 είναι τα Agility, Epitome, Maria, PlastronJS, rAppid,

Sammy, Serenade, soma, Spine, Stapes, CanJS. Το πιο πιθανό είναι σε μερικά χρόνια να έχουν επικρατήσει νέα και ορισμένα από τα παρακάνω να είναι πλέον άγνωστα:

| Framework | Θέση | Θέση 2013 | Έτοιμοι να το υιοθετήσουν | Ήταν το 2013 | Αξία Πρότασης | Ήταν το 2013 | Ψήφοι |
|---|---|---|---|---|---|---|---|
| AngularJS | 1 | 1 | 79% | 81% | 81% | 84% | 447 |
| jQuery | 2 | | **91%** | | 83% | | 363 |
| ReactJS | 3 | | 69% | | 77% | | 233 |
| Backbone.js | 4 | 2 | 77% | 78% | 72% | 74% | 221 |
| D3.js | 5 | | 77% | | 77% | | 209 |
| Underscore.js | 6 | | 81% | | 80% | | 206 |
| Ember.js | 7 | 4 | 68% | 69% | 73% | 74% | 179 |
| Knockout | 8 | 3 | 73% | 74% | 72% | 73% | 154 |
| Handlebars | 9 | | 76% | | 70% | | 143 |
| Socket.io | 10 | | 73% | | 79% | | 142 |
| Ext JS | 11 | 5 | 72% | 71% | 66% | 70% | 138 |
| Meteor | 12 | | 62% | | 71% | | 131 |
| Google Web Toolkit | 13 | | 70% | | 60% | | 130 |
| Kendo | 14 | 6 | 65% | 64% | 71% | 73% | 90 |
| Aurelia | 15 | - | 53% | | 66% | | 78 |
| MooTools | 16 | | 62% | | 56% | | 72 |
| Durandal | 17 | | 54% | | 67% | | 62 |
| Capuccino | 18 | | 56% | | 47% | | 61 |
| Kinetic.js | 19 | | 56% | | 57% | | 61 |
| JavaWebToolkit | 20 | | 57% | | 49% | | 58 |
| Echo | 21 | | 54% | | 54% | | 56 |
| Rialto | 22 | | 47% | | 59% | | 39 |

# ΚΕΦ.4: Βάσεις δεδομένων για την JavaScript

## 4. Βάσεις δεδομένων για την JavaScript

Τίποτα από τα παραπάνω δε θα ήταν ολοκληρωμένο εάν δεν υπήρχε τρόπος να αποθηκευτούν τα αντικείμενα της JavaScript σε κάποιο [μόνιμο] αποθηκευτικό μέσο, δηλαδή μια βάση δεδομένων. Μια από τις πρώτες βάσεις δεδομένων που χρησιμοποίησε την JavaScript ως γλώσσα ερωτημάτων προς τη βάση (μέσω http) ήταν η CouchDB. Αν και η CouchDB είναι γραμμένη σε Erlang τα δεδομένα αποθηκεύονται σε JSON σε μορφή documents χωρίς να υπάρχουν σχέσεις μεταξύ αυτών. Η ανάπτυξη της CouchDB ξεκίνησε το 2005 από τον Damien Katz, πρώην προγραμματιστής του Lotus Notes που (ως Lotus Domino) ήταν από τις πιο χρησιμοποιημένες document databases στο κόσμο.

### 4.1. Σύντομος κατάλογος Βάσεων Δεδομένων

Παρακάτω βλέπουμε μια μικρή λίστα (από τις πάνω από 150) βάσεις δεδομένων που κατά κύριο λόγω λειτουργούν χρησιμοποιώντας το φορμάτ JSON και έχουν ευρεία χρήση στο backend των εφαρμογών του web, χωρίς απαραίτητα αυτές να έχουν γραφτεί σε JavaScript

- **MongoDB**: γραμμένη σε C και C++ θεωρείται στις μέρες μας το ανάλογο της βάσης MySQL για την PHP. Εκτός του ότι αποθηκεύει documents με τη μορφή JSON (αναφέρονται ως BSON) τα ερωτήματα προς τη βάση έχουν και αυτά μορφή JavaScript όπως πχ

```
db.inventory.find( { 'producer.company': 'ABC123' } )
```

Για τις ανάγκες του Meteor γράφτηκε σε JavaScript μια μικρή εκδοχή της Mongo με την ονομασία MiniMongo η οποία χρησιμοποιείται ως in-memory database στην πλευρά του browser. Η Mongo όπως θα δούμε και παρακάτω διαδραματίζει πλέον

πολύ σημαντικό ρόλο σε σειρά από υλοποιήσεις παγκοσμίως. Σύμφωνα με το db-engines θεωρείται πλέον (db-engines.com/en/ranking) η #4 βάση παγκοσμίως μετά την Oracle, MySQL και MS SQL και η #1 NOSQL βάση δεδομένων. Μάλιστα αξίζει να σημειωθεί ότι υπάρχουν πλέον drop-in replacements (όπως έχει γίνει με την MariaDB και την MySQL) από τρίτες εταιρείες όπως η **Percona** την οποία προτιμούν μεγάλες επιχειρήσεις.

- **PouchDB**: παρόμοια με την MiniMongo, πρόκειται για βάση δεδομένων γραμμένη σε JavaScript η οποία μπορεί να ενσωματωθεί πλήρως στο front end (συμπιεσμένο μέγεθος μικρότερο από 50K) αρκεί ο browser να είναι συμβατός με ES5 (Firefox από 29, chrome από 34, Android) ενώ τρέχει και σε Node.js. Σημαντική η χρήση της εάν θέλουμε να έχουμε εφαρμογές που λειτουργούν και off line κρατώντας τα data στον browser και αργότερα τα συγχρονίζουν όταν ο χρήστης βρεθεί online.
- **LokiJS**: θεωρείται από τις ταχύτερες βάσεις δεδομένων γραμμένες σε JavaScript (1,1 εκατομμύρια operations το δευτερόλεπτο). Η LokiJS μπορεί να αντικαταστήσει την SQLite στην περίπτωση εφαρμογών γραμμένων στο framework Cordova/PhoneGap (για mobile εφαρμογές).

- **RethinkDB**: γραμμένη σε C++ πρόκειται για μια βάση δεδομένων που προωθεί αντικείμενα JSON στην εφαρμογή (push) ανανεώνοντας συνεχώς τα δεδομένα της εφαρμογής. Η γλώσσα των ερωτημάτων είναι η ReQL (από το Reactive) και η σύνταξή της είναι παρόμοια με τη JavaScript παράδειγμα:

```
r.tableCreate('movies');
r.table('movies').insert(r.http('http://rethinkdb.com/sample/top-250-ratings.json'))
```

Η RethinkDB εκτός του ότι έχει αποκτήσει momentum σε πολλές γλώσσες προγραμματισμού γιατί χρησιμοποιεί τη σχετικά δεδομένη query γλώσσα ReQL (η οποία σημειωτέων έχει και joins) με επίσημη υποστήριξη για πολλές γλώσσες εκτός JavaScript, όπως εξάλλου και η Mongo. Τον Οκτώβριο του 2016 η συγκεκριμένη βάση έγινε community project στο github.com /rethinkdb/rethinkdb

- **NeDB:** (Node embedded database) πρόκειται για άλλη μια βάση δεδομένων γραμμένη σε JavaScript με δυνατότητα να τρέξει στο backend (node.js) ή ακόμα και μέσα στον browser όπου μπορεί να χρησιμοποιηθεί και ως in-memory datastore. Το API της είναι υποσύνολο της MongoDB. (https://github.com/louischatriot/nedb)

- **DocumentDB**: η εναλλακτική πρόταση της Microsoft[105] που παρουσιάστηκε το 2014 είναι βασισμένη σε JSON αντικείμενα και στην οποία τα ερωτήματα γράφονται με παρόμοιο με την SQL τρόπο, πλήρως όμως εναρμονισμένα για JavaScript. Μάλιστα και οι stored procedures που εκτελούνται στον server γράφονται κι αυτές σε JavaScript. Η συγκεκριμένη λύση δίνεται μέσω του Azure (cloud) και δεν είναι δωρεάν (αν και υπάρχει ένας "emulator" για δοκιμές σε development επίπεδο) έχει κόστος με το GB με το μήνα αλλά η Microsoft υπόσχεται εύκολη υλοποίηση και απαντήσεις σε λιγότερα από 10ms ανεβάζοντας αυτόματα τους πόρους για να πετυχαίνει αυτούς τους χρόνους.

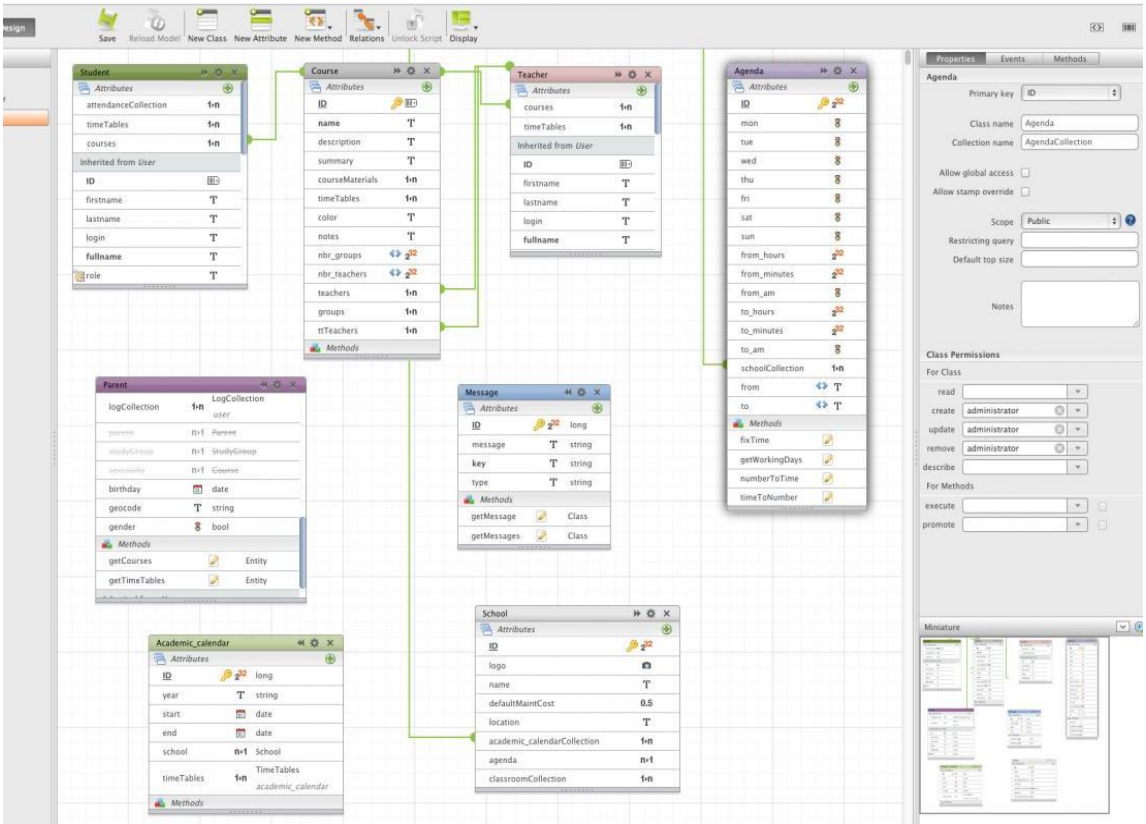
- **CouchDB**: εμπνευσμένο project (2005) από το document storage model του Lotus Domino και φτιαγμένο από τον πρώην developer του συστήματος Damien Katz, η Couch (cluster of unreliable commodity hardware) όχι απλά αποθηκεύει τα αντικείμενά της σε JSON αλλά έχει και views που γράφονται σε JavaScript. Η ίδια η open source (couchdb.apache.org) βάση είναι γραμμένη σε Erlang ενώ όπως παλαιότερα και το Domino έχει multiversion concurrency control κρατώντας conflict documents που η εφαρμογή μετά αποφασίζει ποια θα διατηρήσει. Τη βάση χρησιμοποιεί και το NPM για τη registry των πακέτων.

- **Couchbase**: παρότι έχει παρόμοιο όνομα με την CouchDB (και ορισμένους πρώην developers της) πρόκειται για διαφορετική βάση, επίσης opensource (github.com/couchbase), NoSQL, document-oriented που αποθηκεύει JSON αντικείμενα. Γραμμένη το 2010 σε C++ και C, αργότερα Erlang και Go πρόκειται για πολύ δημοφιλές project λόγω του ότι τα queries γράφονται παρόμοια με SQL (N1QL) ενώ έχει και embeddable έκδοση για κινητά με δυνατότητα συγχρονισμού. Πολύ ενδιαφέρον ότι το 2012 την έκδοση 2 ανέλαβε ο Damien Katz ανακοινώνοντας[106] ότι φτιάχνει από την αρχή ότι δεν του άρεσε στην CouchDB

- **WakandaDB**: πρόκειται στην ουσία για web application server γραμμένο σε C++ που αποθηκεύει αντικείμενα στη μορφή JSON και μέσω http δέχεται ερωτήματα σε JavaScript. Η Γαλλική εταιρεία που το έχει αναπτύξει το παρουσιάζει ως πλήρες (full stack) περιβάλλον ανάπτυξης εφαρμογών σε JavaScript βάζοντας το ως αντίπαλο του Node. Η γλώσσα ερωτημάτων είναι αρκετά απλή πχ:

```
ds.Company.query('countryName == USA').compute('revenues')
```

ενώ οι functions που γράφονται στα ερωτήματα γράφονται σε JavaScript που εκτελείται στον server. Ενδιαφέρον είναι ότι η κατασκευάστρια εταιρεία 4D έχει αναπτύξει και περιβάλλον ανάπτυξης εφαρμογών με γραφικό εργαλείο σχεδίασης των βάσεων.



εικόνα από το περιβάλλον ανάπτυξης της Wakanda

- **OrientDB**: άλλη μια βάση δεδομένων που χρησιμοποιεί **γράφους και documents** (με references αντί για joins) και έχει το JSON ως φορμάτ ανταλλαγής δεδομένων. Λόγω της χρήσης των references μπορούμε να μεταβούμε μεταξύ των κόμβων ενός γράφου

ή ενός δέντρου ενώ τα ερωτήματα μπορούν να γίνουν και υπό μορφής SQL. Η ίδια η βάση μπορεί να προσπελασθεί εύκολα από το frontend της JavaScript αλλά και μέσω Node.js

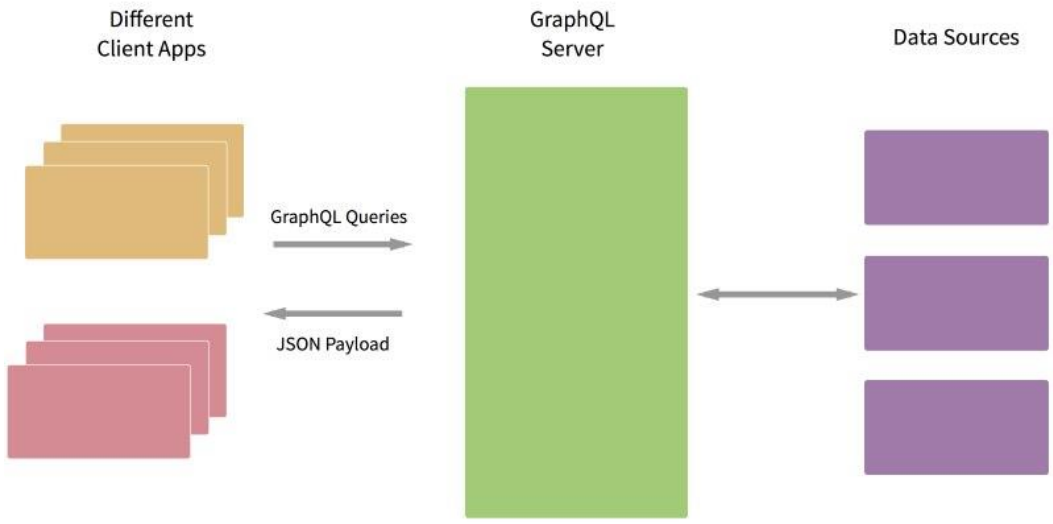
- **ArangoDB**: Μια βάση δεδομένων που δεν είναι φτιαγμένη απαραίτητα για την αποθήκευση αντικειμένων σε JSON ούτε για χρήση αποκλειστικά από JavaScript προγραμματιστές. Η ArangoDB είναι μια σχετικά πρόσφατη (2011) βάση δεδομένων που επιτρέπει την αποθήκευση πολλών τύπων δεδομένων (multi-model): εκτός από JSON η βάση αποθηκεύει key-value pairs αλλά και γράφους. Η γλώσσα ερωτημάτων είναι η AQL (από το Arango) που η σύνταξή της μοιάζει με JavaScript.

  Παράδειγμα δημιουργίας σχέσεων σε γράφους:

  ```
  var rel = graph_module._relation("isCustomer", ["shop"],
  ["customer"]);
  ```

  Το ενδιαφέρον είναι ότι οι functions της ArangoDB που επεκτείνουν τη γλώσσα γράφονται και αυτές σε JavaScript η οποία εκτελείται μέσα στη βάση χρησιμοποιώντας τη μηχανή V8, ακριβώς όπως κάνει και το Node.

- **Elasticsearch**: άλλη μια βάση δεδομένων γραμμένη σε Java που χρησιμοποιεί το JSON για αποθήκευση δεδομένων, η Elasticeseach χρησιμοποιείται κυρίως για εταιρικές μηχανές αναζήτησης. Σχετικά πρόσφατη (2010) η Elasticsearch είναι η βασική μηχανή αναζήτησης της Wikipedia και υπόσχεται αποτελέσματα και analytics σε πραγματικό χρόνο.



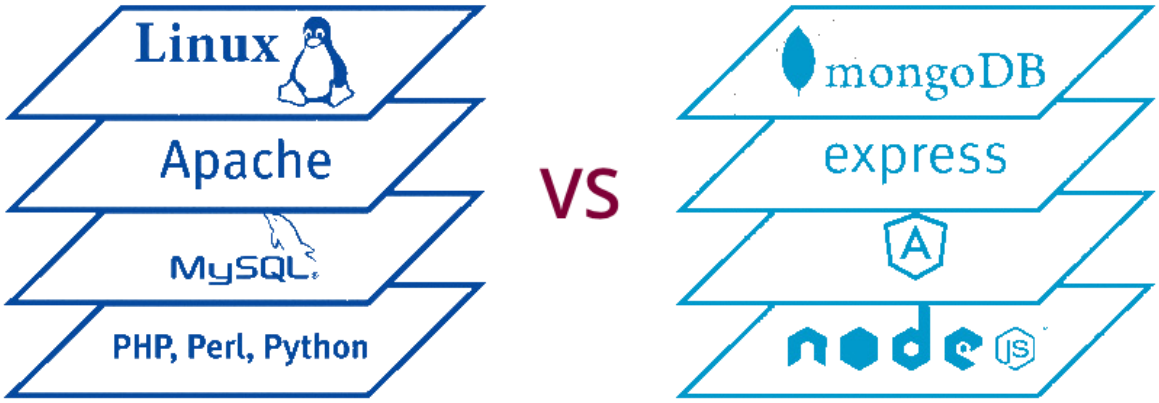

- **GraphQL**: αν και δεν πρόκειται για ενδιάμεση βάση δεδομένων αλλά γλώσσα ερωτημάτων, η GraphQL δημιουργήθηκε και χρησιμοποιείται από το Facebook από το 2012. Πρόσφατα το Facebook αποφάσισε να κάνει open source το project *(από τα τέλη του 2015 είναι σε technical preview)* ενώ παρατηρείται μεγάλη κινητικότητα στη σελίδα του στο github.com/ graphql/graphql-js Η GraphQL αρχίζει και χρησιμοποιείται και σε άλλα projects για τον ενδιάμεσο σταθμό των data που επιστρέφονται από μία βάση (Mongo, SQL, REST API's) σε μορφή JSON. Το GraphQL χρησιμοποιείται ήδη αρκετά σε συνδυασμό με τις υπόλοιπες τεχνολογίες του Facebook (το React για το front end, το Flux ως pattern, και το Relay ως framework για την απόκτηση των δεδομένων από τις πηγές τους)

# ΚΕΦ.5: Full Stack JavaScript – MEAN και 3REE

## 5. MEAN – το νέο στάνταρ στα stack

Συμπληρώνοντας τα παραπάνω κεφάλαια θα σταθούμε λίγο στο full stack που ήταν στο απόγειό του το 2014, το MEAN. Όπως βλέπουμε και από το σχήμα αφορά το δωρεάν (open source) σύνολο των εργαλείων MongoDB (για τη βάση δεδομένων), Express (το συνεκτικό web application framework), την Angular (το δημοφιλέστερο framework από την Google) και το Node.

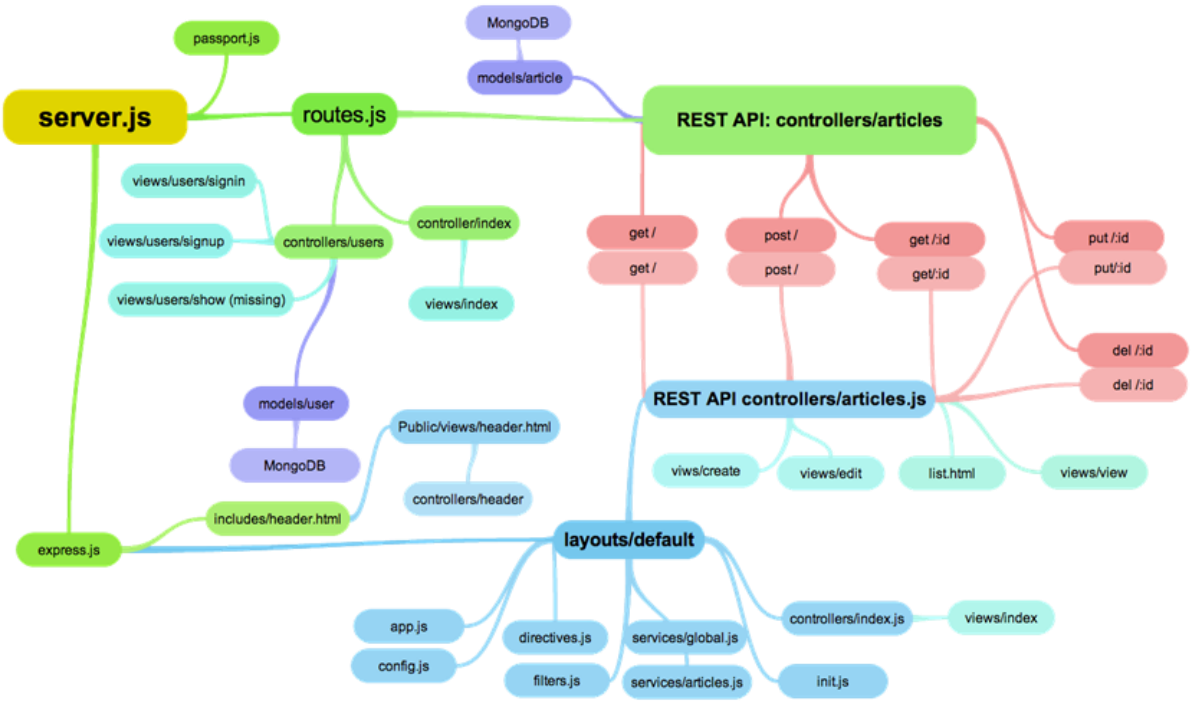
Για να καταλάβουμε καλύτερα το ρόλο του MEAN στις μέρες μας θα μπορούσαμε να το συγκρίνουμε με το πιο πετυχημένο LAMP (παραπάνω από τα μισά site παγκοσμίως βασίζονται σε αυτό, κυρίως λόγω εφαρμογών όπως Facebook, WordPress και Joomla):




Οι developers του LAMP λοιπόν χρησιμοποιούν (συνήθως) τον Apache σαν web server που τρέχει στο Linux, μία (ή περισσότερες) MySQL βάσεις και ως γλώσσα προγραμματισμού συνήθως την PHP. Αυτό είναι και το stack στο οποίο βασίζονται οι πιο πετυχημένες εφαρμογές κατασκευής sites ή blogs όπως Joomla, WordPress αλλά και παλαιότερα το Facebook.

Παρότι τα τέσσερα αρχικά δηλώνουν διαφορετικά πράγματα, είναι αρκετά δημοφιλή αν σκεφτεί κανείς ότι μια τυχαία αναζήτηση στον όρο MEAN vs LAMP θα φέρει πάνω από 10 εκατομμύρια αποτελέσματα.

Το MEAN δεν αντικαθιστά στην πραγματικότητα ούτε το λειτουργικό, το οποίο μπορεί να παραμείνει το Linux ούτε αντικαθιστά τον Apache αφού το node μπορεί και πολλές φορές είναι καλό να συνυπάρχει και με κάποιον web server. Για το 2015 φαίνεται ότι το React του facebook αποκτά τη μεγαλύτερη προσοχή ως διάδοχος της Angular. Αυτό που μπορούμε να παρατηρήσουμε είναι ότι στην περίπτωση του MEAN υπάρχει μια προτροπή στον προγραμματιστή να χρησιμοποιήσει την τεχνική προγραμματισμού MVVM και γενικά τύπου MVC, ειδικά λόγω της Angular. Παρακάτω βλέπουμε ένα καλό σχήμα για το διαχωρισμό αρμοδιοτήτων στο MEAN stack (από το site 100percentjs.com):




Φυσικά όσο διαδίδονται νεότερα frameworks που έχουν περισσότερη επιτυχία το MEAN μπορεί να αλλάξει όπως για παράδειγμα το MERN που αντί για την ANGULAR έχει το REACT


Προχωρώντας ακόμα περισσότερο (2015), αλλάζοντας δηλαδή κομμάτια στο MEAN stack βρίσκουμε πολλές ακόμα υλοποιήσεις όπως το 3REE και το SWARM

## 5.1. 3REE και νεότερα stack

Στο 3REE (github.com/GordyD/3ree) αλλάζει η βάση δεδομένων σε RethinkDB που όπως περιγράψαμε σε προηγούμενο κεφάλαιο είναι από μόνη της αρκετά reactive βάση δεδομένων, αφού στέλνει push notifications όταν γίνεται αλλαγή στη βάση, ενώ εστιάζει σε πολλές από τις τεχνολογίες του Facebook, καταρχάς το React για την εμφάνιση και το Redux για το state (μεθοδολογία Flux αντί για MVC):

| Τεχνολογία | Αφορά | Εκδοση |
|---|---|---|
| React | View layer | 15.0.2 |
| React Router | Universal routing | 2.4.0 |
| Redux | State management | 3.5.0 |
| RethinkDB | Persistance layer | 2.3.1 |
| Express | Node.js server framework | 4.13.0 |
| Socket.io | Used for realtime communication between clients and server | 1.4.0 |
| Webpack | Module bundling + build for client | 1.13.0 |
| Superagent | Universal http requests | 1.8.0 |
| Stylus | Expressive, dynamic, robust CSS | 0.54.0 |

Αντίστοιχα πιστεύουμε θα παρουσιαστούν πολλά περισσότερα στο μέλλον, όπως πχ το MERN (mern.io) [Mongo, Express, React, Node, Redux, Webpack] που όλα βασίζονται γενικά σε βιβλιοθήκες του Node (npm).

Η τάση αυτή για πιο δυναμικά site που όλα τα δεδομένα είναι συνεχώς ανανεωμένα είναι εμφανής και στο SWARM, μια τεχνολογία που ασχολείται πιο πολύ με το μοντέλο (το M στο MVC).

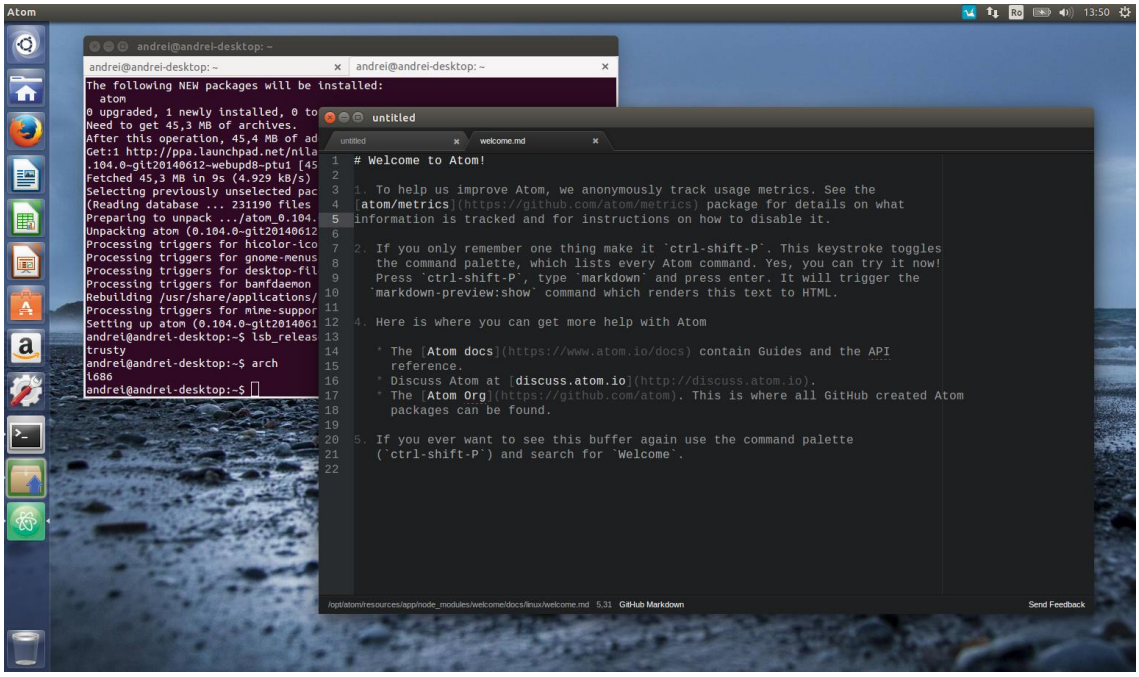
Βασισμένο στο React και στο Node, διασυνδέει κλασικές βάσεις (MySQL, Oracle, MongoDB) και συγχρονίζει συνεχώς τα δεδομένα τους μεταξύ των συνδεμένων sessions των browsers στη λογική του CRDT (conflict-free replicated data type). Το SWARM όπως και η GraphQL δεν αποτελούν frameworks αλλά ενδιάμεσες τεχνολογίες για τη μεταφορά των δεδομένων των βάσεων. Όπως θα δούμε και παρακάτω με το Meteor framework, η λογική είναι ότι κρατούνται οι αλλαγές (τα logs) και αυτά συγχρονίζονται όταν ο χρήστης είναι online (κλασική περίπτωση κινητού τηλεφώνου που ο χρήστης είναι σε κάποιο site, το ενημερώνει ενώ είναι off line και μόλις συνδεθεί στο δίκτυο της κινητής το site στέλνει τις αλλαγές).

Τέλος θα θέλαμε να αναφερθούμε και στο JavaScript backend project Horizon.io στο οποίο συνεισφέρει η ομάδα που ανέπτυξε και την RethinkDB πάνω στην οποία βασίζεται.

# ΚΕΦ.6: JavaScript στο Desktop

## 6. Electron – το παιδί του GitHub

Παίρνοντας τον αντίθετο δρόμο από το Node.js το Electron είναι ένα project του GitHub που μεταφέρει το Chromium στο desktop δίνοντας τη δυνατότητα να κατασκευαστούν “universal” desktop εφαρμογές με JavaScript, στα περιβάλλοντα για τα οποία φτιάχνεται το Chromium (Linux, Windows, Mac).




Φωτογραφία: Atom editor

### 6.1. Ξεκινώντας από το Atom

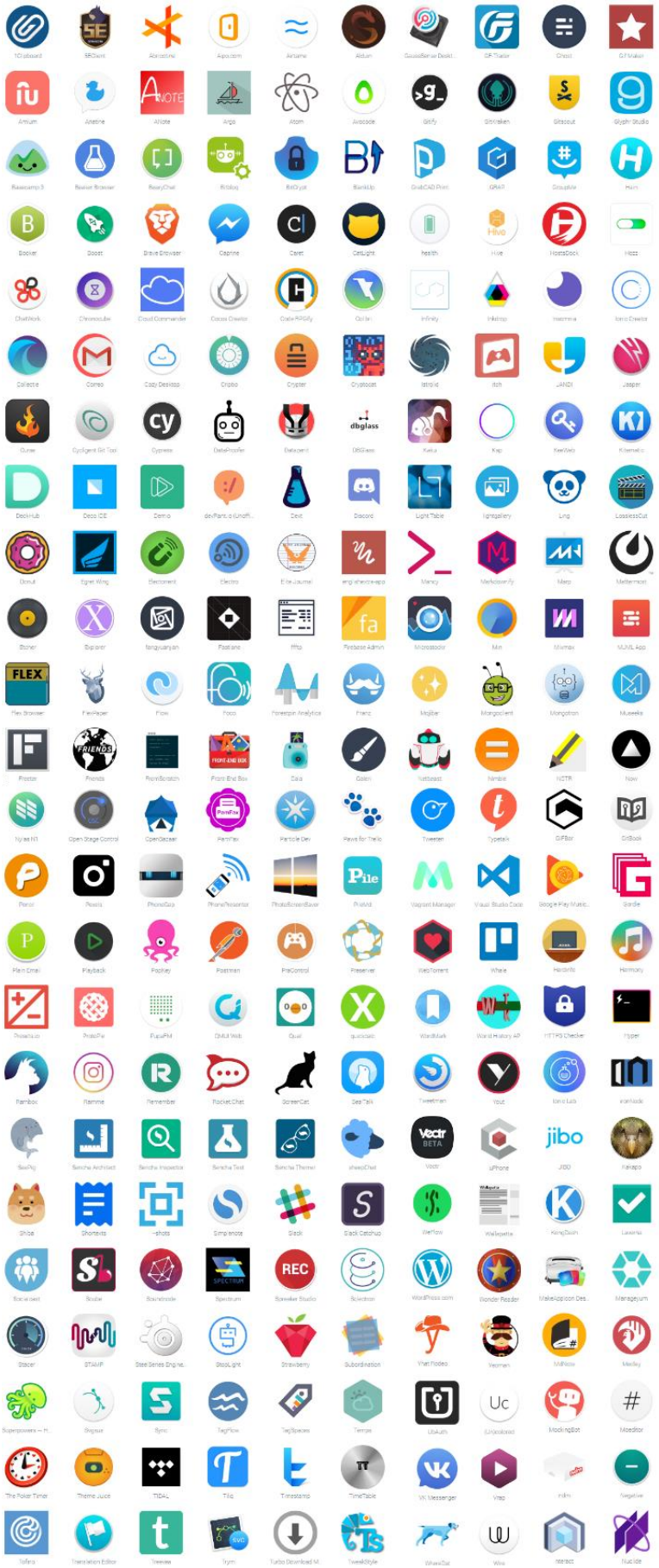
Πολύ δημοφιλές project αποτελεί ο text editor Atom από το οποίο ουσιαστικά γεννήθηκε το Electron (παλιά ονομασία Atom Shell). Η ίδια η ιδέα για τη δημιουργία του Electron ήταν να δημιουργηθεί ένας text editor χρησιμοποιώντας web τεχνολογίες (youtu.be/RaPmi-33rfc). Σημαντικό είναι το γεγονός ότι ο προγραμματιστής δεν χρειάζεται να ξέρει τι browser έχει ο χρήστης καθώς η συμβατότητα με το Chrome είναι

δεδομένη ενώ ο χρήστης αποκτάει αυτόματα και το Node οπότε είναι δεδομένες και όλες οι αντίστοιχες δυνατότητες και φυσικά η τεράστια παρακαταθήκη του npm. Μια καλή πηγή για περαιτέρω μελέτη πάνω στο οικοσύστημα του Electron είναι η σελίδα github.com/sindresorhus/awesome-electron στο GitHub.

Το Atom με τη σειρά του έδωσε το έναυσμα για τη δημιουργία του Visual Studio Code, του text editor που έφτιαξε η Microsoft σε JavaScript αλλά και TypeScript. Η πιο πετυχημένη εφαρμογή αυτή τη στιγμή που τρέχει σε JavaScript στο desktop είναι το Spotify[78] ενώ πολύ ενδιαφέροντα project είναι το Slack και το WordPress Desktop.

Παλαιότερα πετυχημένο ήταν και το αντίστοιχο NW.js (node-webkit) με το οποίο τρέχουμε node εφαρμογές από την πλευρά του DOM.

Μερικές Εφαρμογές φτιαγμένες με το stack του Electron (electron.atom.io/apps/)



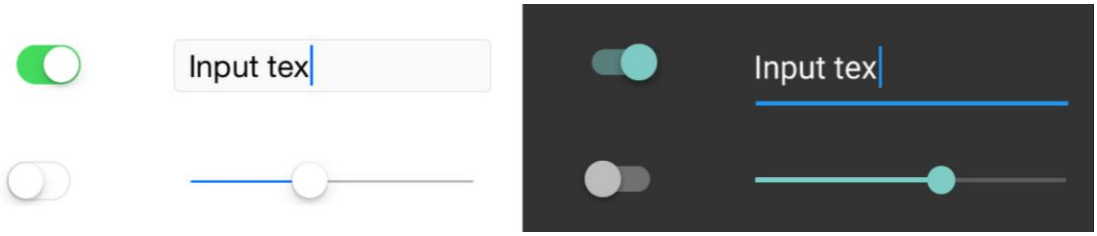
# ΚΕΦ.7: JavaScript στο smartphone

## 7. PhoneGap – το παιδί της Adobe

Ανάλογα με το Electron, το PhoneGap είναι η δημοφιλέστερη βιβλιοθήκη ανάπτυξης εφαρμογών σε JavaScript για κινητά τηλέφωνα. Το PhoneGap ξεκίνησε το 2009 ως εργαλείο ανάπτυξης εφαρμογών για iPhone σε JavaScript, μιας που οι πρώτες εκδόσεις του δημοφιλούς κινητού δεν επέτρεπαν τον προγραμματισμό σε native περιβάλλον (Objective C) αλλά μόνο μέσω του browser. Καθώς όλα τα κινητά περιλαμβάνουν την JavaScript στον browser τους, το Cordova (η καρδιά του PhoneGap) ουσιαστικά επιτρέπει να χρησιμοποιήσουμε εξαρτήματα του κινητού τηλεφώνου όπως το GPS, την κάμερα, την πυξίδα κ.ο.κ. αλλά και δεδομένα όπως τον τηλεφωνικό κατάλογο, φωτογραφίες κλπ μέσα από την JavaScript που εκτελείται στον browser. Μετά το 2011 η εταιρεία (Nitobi) που ανέπτυξε το PhoneGap αγοράστηκε από την Adobe ενώ προστέθηκαν πολύ περισσότερες συσκευές κινητών εκτός από Apple στις υποστηριζόμενες, όπως BlackBerry, Bada, Symbian, Windows και φυσικά Android καθιστώντας τη βιβλιοθήκη σχεδόν μονόδρομο για μια εταιρεία που θέλει να αναπτύξει εφαρμογές για όλα τα κινητά χρησιμοποιώντας ενιαίο κώδικα. Αυτή τη στιγμή πάνω από το 15% των εφαρμογών (buildwith statistics) για κινητά είναι φτιαγμένες με Gordova.

Αντίστοιχο project με το PhoneGap αλλά και βασισμένο σε αυτό, είναι το XDK της Intel και το CocoonJS το οποίο είναι πιο πολύ προσανατολισμένο σε WebGL εφαρμογές, αλλά και το Ionic το οποίο προτιμούν οι προγραμματιστές της Angular καθώς βασίζεται σε αυτό.

Μια άλλη πρόσφατη προσέγγιση αποτελεί το manifoldjs το οποίο αντιγράφει τα περιεχόμενα ενός site και το μετατρέπει σε “native” εφαρμογή Android, iOS ή Windows χρησιμοποιώντας και αυτό την τεχνολογία του Cordova.




Φωτογραφία: React Native: τα ίδια controls αριστερά σε Apple και δεξιά σε Android

## 7.1. React Native

Αντίθετα οι οπαδοί του React προτιμούν το React Native του Facebook, το οποίο παρεμπιπτόντως δεν χρησιμοποιεί το webview αλλά δικά του native views σε Android και iOS (facebook.github.io/react-native).

Το React Native αυτή τη στιγμή (2016) αποτελεί την πιο δημοφιλή τεχνολογία ανάπτυξης νέων εφαρμογών σε κινητά καθώς το Facebook υπόσχεται μέχρι και 60FPS ταχύτητα στην οθόνη, παρόλ' αυτά απαιτεί την εκμάθηση του React (jsx) και νέων τρόπων (pattern) προγραμματισμού.

Στο React Native «χτίζουμε» το view κομμάτι της εφαρμογής σε JavaScript, με τον ίδιο τρόπο που χτίζουμε το HTML κομμάτι σε web εφαρμογές φτιαγμένες με το framework React:

```
import React, { Component } from 'react';
import { Text, View } from 'react-native';

class WhyReactNativeIsSoGreat extends Component {
  render() {
    return (
      <View>
        <Text>
          If you like React on the web, you'll like React Native.
        </Text>
        <Text>
          You just use native components like 'View' and 'Text',
          instead of web components like 'div' and 'span'.
        </Text>
      </View>
    );
```

```
  }
}
```

Με το React Native όπως αναφέρει το Facebook «φτιάχνουμε πραγματικές mobile εφαρμογές»

## 7.2. NativeScript – μια διαφορετική προσέγγιση

Αντίθετα οι οπαδοί του React προτιμούν το React Native του Facebook, το οποίο παρεμπιπτόντως δεν χρησιμοποιεί το webview αλλά δικά του native views σε Android και iOS (facebook.github.io/react-native).

Μια αντίθετη πορεία σε σχέση με το React Native πήρε η εταιρεία Telerik ανακοινώνοντας τον Ιούνιο του 2014 τη NativeScript, μια εναλλακτική πρόταση ανάπτυξης εφαρμογών για κινητά. Το project που είναι ακόμα σε εξέλιξη εμφανίστηκε τον Μάρτιο του 2015, βασίζεται στην JavaScript μηχανή που βρίσκεται σε όλα τα κινητά ως μέρος του browser τους αλλά αντί να εμφανίζει την εφαρμογή στο DOM (το webview) το view κομμάτι υλοποιείται με κλήσεις σε native controls του κινητού. Ετσι το interface χτίζεται σε XML όπως παρόμοια γίνεται στο Android με την Java ενώ ο κώδικας γράφεται σε JavaScript και τρέχει στο αντίστοιχο engine (V8 στα Android, Chakra σε κινητά με Windows και JavaScriptCore στο iOS). Παράδειγμα του Hello World σε iOS:

```
var alert = new UIAlertView();
alert.message = "Hello world!";
alert.addButtonWithTitle( "OK" );
alert.show();
```

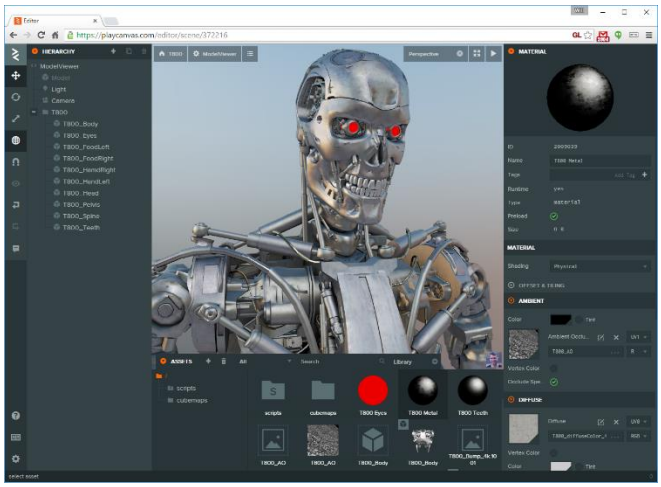
Παρότι και η JavaScript τρέχει στο δικό της VM όπως και η Java, όπως περιεγράφηκε στα προηγούμενα κεφάλαια (asm.js, web assembly), πλέον γίνεται compilation σε κώδικα μηχανής και έτσι η Telerik υπόσχεται μέχρι και 60fps απόδοση στην οθόνη.

Εκτός από τα controls του interface, με τον ίδιο τρόπο η NativeScript έχει access σε μεγάλο μέρος του native API του κινητού, συμπεριλαμβανομένων των κλάσεων, αντικειμένων, δομών όπως αριθμοί (το native java.lang.Math), το GPS, την κάμερα κλπ.

Επίσης υποστηρίζεται η TypeScript και η Angular (στην έκδοση 2 λόγω των components). Στα αρνητικά η NativeScript δεν έχει ακόμα το community ή την ωριμότητα των άλλων λύσεων ενώ έχει ακόμα περιορισμένο CSS.

# ΚΕΦ.8: Γραφικά στη JavaScript

## 8. Λύσεις 3D γραφικών και WebGL



Φωτογραφία: ο editor του PlayCanvas σε μόλις 650KB κώδικα JavaScript

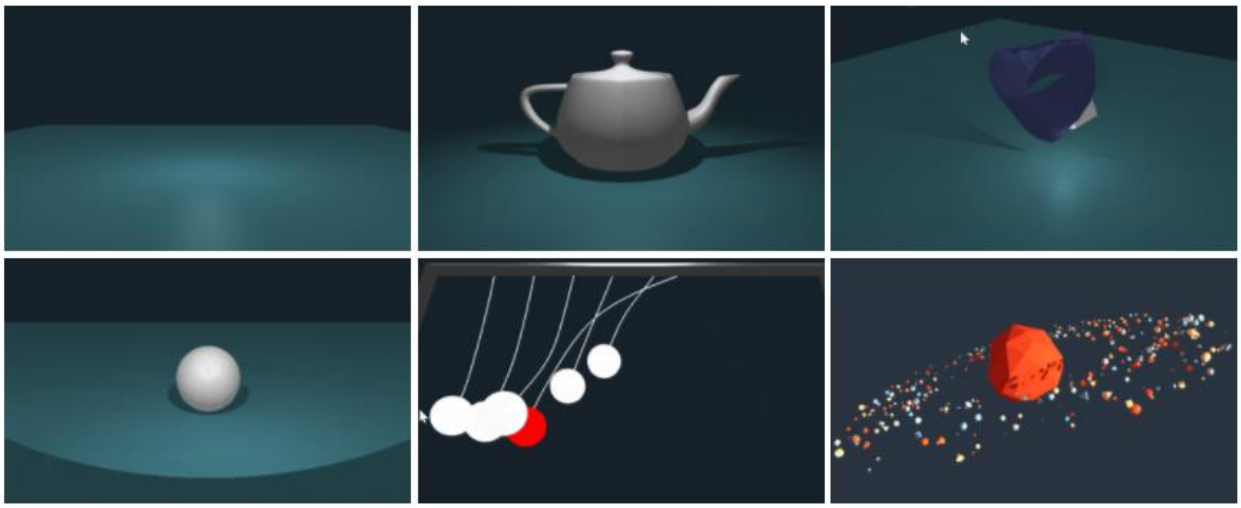
### 8.1. WebGL και PlayCanvas για γραφικά σε JavaScript

Αρχικά η βιβλιοθήκη WebGL σχεδιάστηκε για να επεκτείνει τις δυνατότητες της JavaScript ώστε να παράγει τρισδιάστατα γραφικά στον browser που θα μπορούσαν να τρέχουν στη GPU του υπολογιστή χωρίς plug-ins. Ξεκίνησε το 2006 ως πειραματικό project του Mozilla με την ονομασία Canvas 3D από τον Vladimir Vukicevic ο οποίος πρότεινε το API στην Khronos Group ώστε να γίνει ανοιχτό πρότυπο. Ετσι αρχικά ενσωματώθηκε στον Chrome και το Firefox τον Φεβρουάριο και τον Μάρτιο του 2011 αντίστοιχα. Αν και τα πρώτα demo αφορούσαν παιχνίδια (Quake, Team Fortress 2) αρκετά επιστημονικά projects υλοποιήθηκαν με αυτό, αλλά και η NASA ανέπτυξε μια προσομοίωση του Mars Curiosity.

Ένα μήνα μετά την ανακοίνωση του WebGL (Απρίλιο 2011) ο Will Eastcott, πρώην προγραμματιστής της Electronic Arts, της SONY αλλά και της Blizzard ξεκίνησε την PlayCanvas η οποία επιτρέπει την ανάπτυξη 3D εφαρμογών σε JavaScript στον browser αλλά και σε κινητό μέσω του CocoonJS. Στο project προστέθηκε το 2013 και ο Ελληνας Καλπιάς Ηλίας Βάιος και ένα χρόνο αργότερα, όταν η Apple ανακοίνωσε την επίσημη

υποστήριξη της WebGL στο Safari του OSX και του iOS 8, το PlayCanvas έγινε Open Source (github.com/playcanvas).

Το PlayCanvas είναι μια πλήρης μηχανή παιχνιδιών με τη μηχανή φυσικής μεταφερμένη σε asm.js από την μηχανή Bullet (project Ammo.js)[80].



Φωτογραφία: 3D γραφικά χρησιμοποιώντας την WhitestormJS και την Three.js

## 8.2. Three.js – η δημοφιλέστερη βιβλιοθήκη γραφικών

Εκτός από το PlayCanvas μια άλλη πολύ ενδιαφέρουσα βιβλιοθήκη είναι η threejs (github.com/mrdoob/three.js) με πολλά παραδείγματα στον επίσημο ιστοτόπο της threejs.org

Η Three.js είναι η δημοφιλέστερη βιβλιοθήκη γραφικών κυρίως επειδή είναι η ίδια μέρος άλλων βιβλιοθηκών όπως η WhitestormJS. Η Whitestorm αξίζει ιδιαίτερης αναφοράς γιατί εκμεταλλεύεται τους WebWorkers εντός των browsers για να πετύχει καλύτερα frame rates και φυσική κίνηση, ως αποτέλεσμα του multithreading

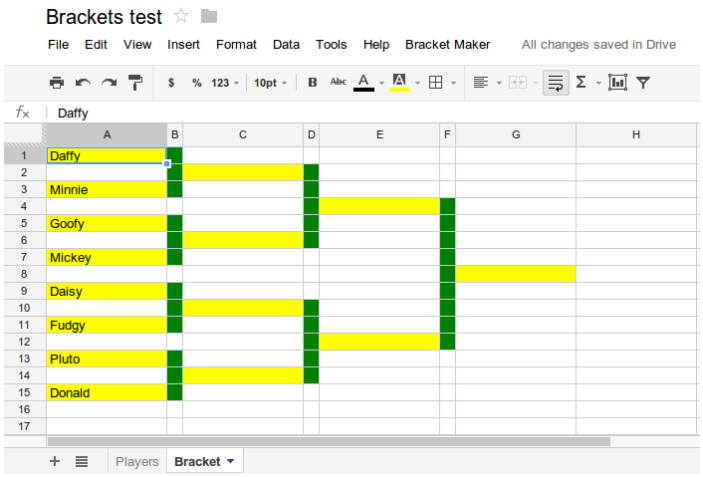
Βέβαια υπάρχουν και άλλες βιβλιοθήκες για ανάπτυξη παιχνιδιών σε JavaScript με πολύ διαδεδομένη την δωρεάν Phaser.io (πλήρης υποστήριξη για sprites και particles, με physics engine και animation), και τις συνδρομητικές ImpactJS, Construct

(www.scirra.com/store/construct-2) και GameMaker (www.yoyogames.com/ gamemaker)

Ένα πολύ ωραίο παράδειγμα για όποιον θέλει να ξεκινήσει με καθαρή JavaScript υπάρχει και στον επίσημο ιστοτόπο της Mozilla (https://developer.mozilla.org/ en-US/docs/Games/Tutorials/ 2D_Breakout_game_pure_JavaScript)

# ΚΕΦ.9: η JavaScript ως ενσωματωμένη γλώσσα

## 9. Προγραμματίζοντας JavaScript μέσα σε άλλες εφαρμογές



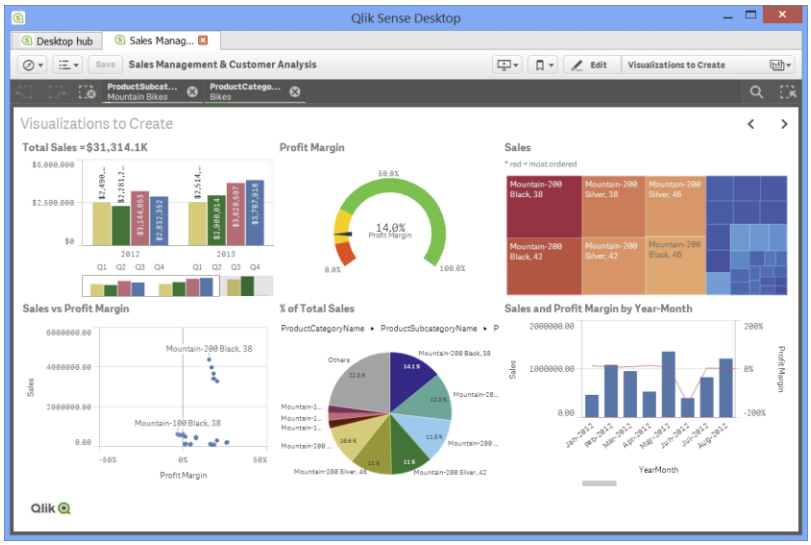

Φωτογραφία: Εφαρμογή JavaScript που τρέχει μέσα στο Google Sheets

### 9.1. Από την ActionScript στην Apps Script

Το οικοσύστημα Javascript ή καλύτερα της ECMAScript από πολύ παλιά περιλάμβανε αρκετές εφαρμογές. Ηδη από την έκδοση 4 (πριν το 2000) είχε ενσωματωθεί στα προϊόντα της Adobe με την παραλλαγή ActionScript ενώ έχει ενσωματωθεί και στον Acrobat Reader για τις ανάγκες κατασκευής «ζωντανών» PDF αρχείων από το 2005 (Acrobat 7.0). Η Adobe έχει ενσωματώσει παλαιότερη έκδοσή της (3) ως Adobe ExtendScript και στα προϊόντα της InDesign, Illustrator, Photoshop, Premiere Pro και άλλα. Η JavaScript έχει ενσωματωθεί και σε πολλά άλλα συστήματα όπως πχ το εργαλείο μουσικής τεχνολογίας MAX/MSP, αλλά και σε άλλες γλώσσες (πχ project Nashorn στην Java 8), QtScript για το περιβάλλον Qt (που την προηγούμενη δεκαετία ήταν το βασικό εργαλείο ανάπτυξης εφαρμογών σε NOKIA Symbian περιβάλλοντα).

Εκδόσεις της γλώσσας υπάρχουν με πολύ μικρό αποτύπωμα (πχ το Duktape τρέχει την Ecmascript E5 σε μόλις 385Kb μνήμης) ενώ σε 256Kb τρέχει η JerryScript της Samsung (github.com/Samsung/jerryscript) που έχει σχεδιαστεί για Internet Of Things

Μια άλλη προσέγγιση έχει δοθεί από την Google της οποίας ο ίδιος ο Chrome browser περιέχει κατάστημα αγοράς apps φτιαγμένων σε JavaScript ενώ προσφάτως ενσωμάτωσε την Google Apps Script στα προϊόντα της Docs, Sheets και Forms (developers.google.com/apps-script) δίνοντας τη δυνατότητα να επεκτείνουμε τις εφαρμογές με νέα μενού και λειτουργικότητα όπως είχε κάνει παλαιότερα η Microsoft με την VB script στο Office.



Φωτογραφία: το περιβάλλον visualisation του QLIKSense

## 9.2. JavaScript σε εμπορικές εφαρμογές

Ένα άλλο ενδιαφέρον παράδειγμα βλέπουμε στην εφαρμογή Business Intelligence QLIK Sense. Βασισμένο στο node και το ενσωματωμένο Chromium project, το Qlik Sense εκμεταλλεύεται πλήρως τις τεχνολογίες των WebSockets, του Canvas και είναι βασισμένο στην AngularJS, το Bootstrap και το RequireJS. Όλα τα visualization extensions γράφονται σε JavaScript (github.com/ stefanwalther/qliksense-extension-tutorial)

Εκτός των ξένων εταιρειών, η Ελληνική NUBIS από το 2012 είχε κατασκευάσει CloudERP που έδινε τη δυνατότητα να γράφεται εμπορικός κώδικας στο front end με JavaScript

# ΚΕΦ.10: Το Cloud ως εφαρμογή της JavaScript

## 10. JAWS: JavaScript + AWS Stack

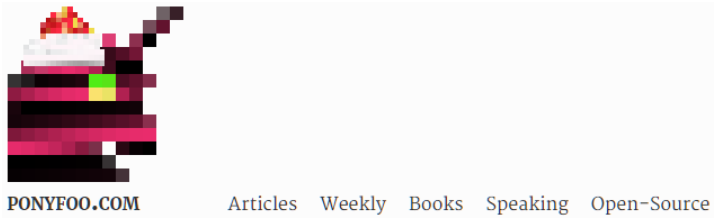

Φωτογραφία: Το stack JAWS* της AMAZON

Ισως το πιο εντυπωσιακό σε λύσεις cloud και πρόσφατο trend είναι οι serverless cloud εφαρμογές. Στην ουσία δεν πληρώνουμε καθόλου για servers, δίσκους, μνήμες όπως γινόταν (και γίνεται) μέχρι σήμερα αλλά η κυρίως εφαρμογή μας είναι ένα μικρό κομμάτι του συνόλου των cloud υπηρεσιών που χρησιμοποιούμε. Στη συγκεκριμένη περίπτωση της AMAZON στο serverless framework (github.com/serverless/serverless)* κεντρικό σημείο αποτελεί το λεγόμενο AWS Lambda, τα οποία είναι functions που εκτελούνται όταν έχουμε κάποιο event. Στο JAWS όλη η εφαρμογή και το framework είναι σε NodeJS. Η κάθε function που όπως προτείνουν (separation of concern) αφορά ένα service ή λειτουργία κάθε φορά, χρησιμοποιεί και διάφορα resources όπως βάσεις, υπολογιστική ισχύ, και άλλα microservices. Οι χρεώσεις γίνονται μόνο όταν εκτελούνται τα συγκεκριμένα services (που καταναλώνουν ισχύ ή εκτελούν άλλα services) ή όταν υπάρχει εξερχόμενη κίνηση στο δίκτυο ενώ το scalability είναι θεωρητικά άπειρο. Ένα παράδειγμα της εταιρείας για 16000 requests/μέρα με μέση χρήση 200ms έχει κόστος 3 δολάρια τη μέρα με δύο κλασικά Elastic Cloud (EC2) ενώ με τη serverless λύση το κόστος πέφτει στα 0,05 δολάρια τη μέρα. Το μήνα δηλαδή η συγκεκριμένη εφαρμογή πέφτει από κόστος 90 ευρώ σε 1,5 ευρώ.

**το framework (τέλη 2016) είναι ακόμα σε beta και απαιτεί εγγραφή early access*

# ΚΕΦ.11: Ερωτηματολόγιο χρήσης JavaScript

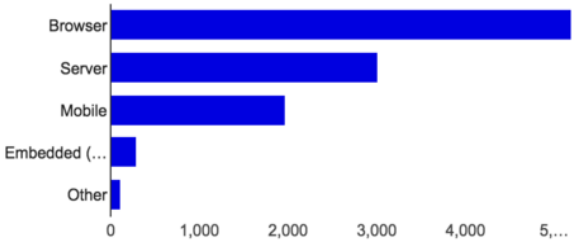
### 11. Ερωτηματολόγιο χρήσης JavaScript από το ponyfool




Φωτογραφία: ο ιστότοπος ponyfoo που έκανε την έρευνα

Κλείνοντας το 2015 και με την ευκαιρία συμπλήρωσης 20 ετών από τη δημιουργία της γλώσσας, δημοσιεύθηκε μια έρευνα σε 5350 προγραμματιστές JavaScript στον ιστότοπο ponyfoo[79] η οποία είναι ιδιαίτερα διαφωτιστική για την παρούσα κατάσταση στην κοινότητα της JavaScript τα αποτελέσματα της οποίας αναπαράγονται ως έχουν:

### 11.1. Προγραμματισμός JavaScript στο front end, back end ή κινητό

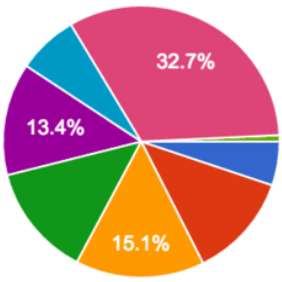



| | | |
|---|---|---|
| Browser | **5196** | 97.4% |
| Server | **3021** | 56.6% |
| Mobile | **1975** | 37% |
| Embedded (Tessel, Raspberry, etc.) | **296** | 5.5% |
| Other | **115** | 2.2% |

### 11.2. Προγραμματισμός στο εργασιακό περιβάλλον ή σε παράλληλα projects

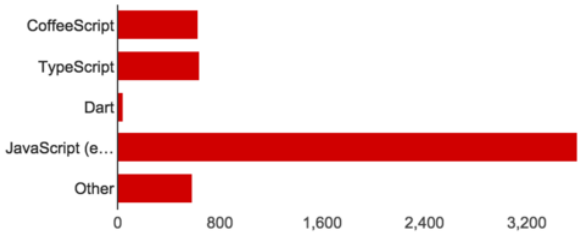

| | | |
|---|---|---|
| At work | **5060** | 94.9% |
| For side projects | **4385** | 82.3% |
| Other | **70** | 1.3% |

### 11.3. Ετη προγραμματισμού σε JavaScript



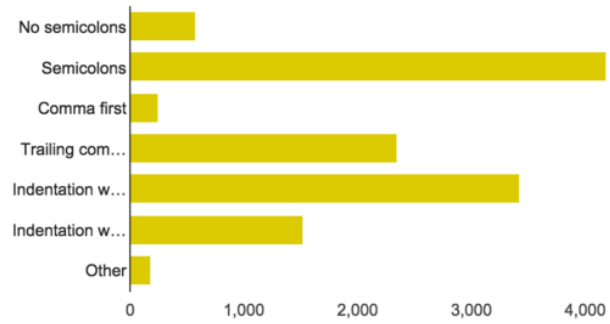

| | | |
|---|---|---|
| One year or less | **278** | 5.2% |
| Two years or less | **662** | 12.4% |
| Three years or less | **803** | 15.1% |
| Four years or less | **711** | 13.3% |
| Five years or less | **713** | 13.4% |
| Six years or less | **379** | 7.1% |
| Over six years | **1746** | 32.7% |
| I'm Brendan Eich | **41** | 0.8% |

### 11.4. Ποιες γλώσσες που κάνουν compile σε Javascript χρησιμοποιείτε ;



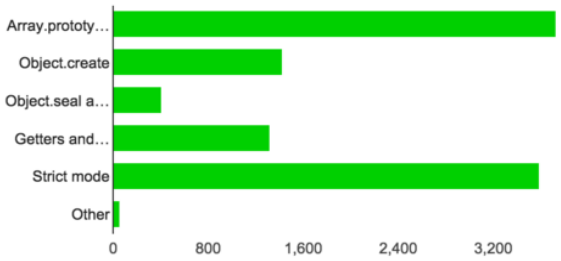

| | | |
|---|---|---|
| CoffeeScript | **634** | 15% |
| TypeScript | **642** | 15.2% |
| Dart | **45** | 1.1% |
| JavaScript (e.g ES6 to ES5) | **3598** | 85% |
| Other | **584** | 13.8% |

### 11.5. Στυλιστικός τρόπος προγραμματισμού



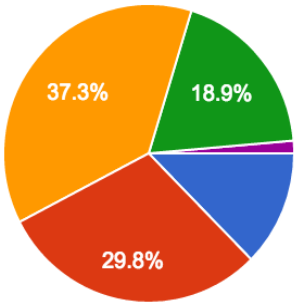

| | | |
|---|---|---|
| No semicolons | **576** | 11% |
| Semicolons | **4186** | 79.9% |
| Comma first | **254** | 4.9% |
| Trailing commas | **2353** | 44.9% |
| Indentation with spaces | **3430** | 65.5% |
| Indentation with tabs | **1526** | 29.1% |
| Other | **188** | 3.6% |

### 11.6. Χρησιμοποιείτε εντολές της EcmaScript 5 ;



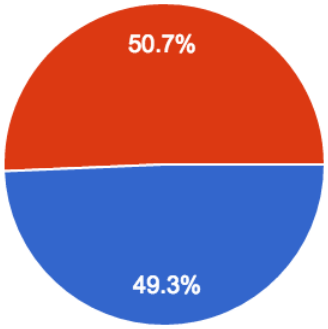

| | | |
|---|---|---|
| Array.prototype.forEach and friends | **3730** | 79.2% |
| Object.create | **1425** | 30.2% |
| Object.seal and Object.freeze | **406** | 8.6% |
| Getters and setters | **1320** | 28% |
| Strict mode | **3595** | 76.3% |
| Other | **56** | 1.2% |

**11.7. Εχετε χρησιμοποιήσει την ES6 ;**



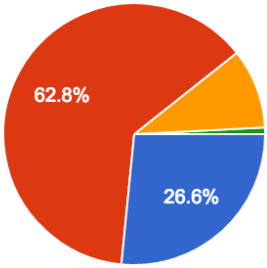

| | | |
|---|---|---|
| Exclusively | **651** | 12.6% |
| Extensively | **1547** | 29.8% |
| Occasionally | **1936** | 37.3% |
| Never | **979** | 18.9% |
| Other | **71** | 1.4% |

**11.8. Ξέρετε τι ετοιμάζεται για την JavaScript 2016 ;**



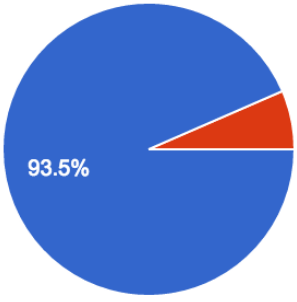

| | | |
|---|---|---|
| Yes | **2588** | 49.3% |
| No | **2658** | 50.7% |

**11.9. Κατανοείτε την ES6 ;**



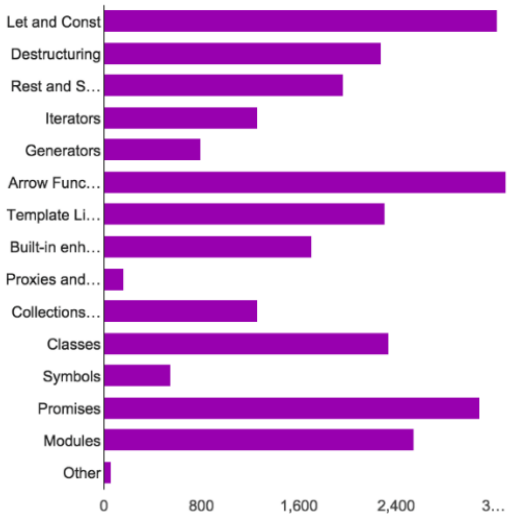

| | | |
|---|---|---|
| Really well | **1399** | 26.6% |
| I understand the basics | **3306** | 62.8% |
| Not at all | **519** | 9.9% |
| Other | **43** | 0.8% |

**11.10. Θεωρείτε την ES6 καλύτερη ;**



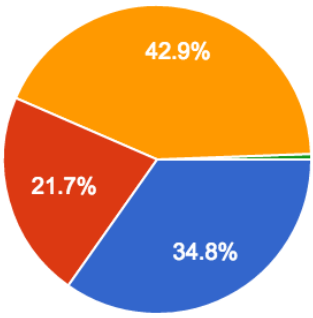

| | | |
|---|---|---|
| Yes | **4716** | 93.5% |
| No | **328** | 6.5% |

### 11.11. Χρησιμοποιείτε εντολές της ES6 ;



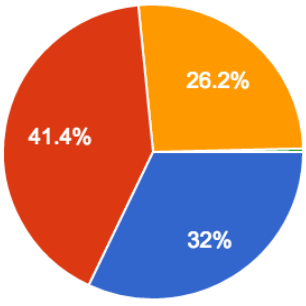

| | | |
|---|---|---|
| Let and Const | **3229** | 77.8% |
| Destructuring | **2279** | 54.9% |
| Rest and Spread | **1964** | 47.3% |
| Iterators | **1259** | 30.3% |
| Generators | **794** | 19.1% |
| Arrow Functions | **3304** | 79.6% |
| Template Literals | **2307** | 55.6% |
| Built-in enhancements (new syntax, methods) | **1707** | 41.1% |
| Proxies and/or Reflection | **165** | 4% |
| Collections (Map, Set, WeakMap, WeakSet) | **1261** | 30.4% |
| Classes | **2343** | 56.5% |
| Symbols | **545** | 13.1% |
| Promises | **3088** | 74.4% |
| Modules | **2552** | 61.5% |
| Other | **62** | 1.5% |

### 11.12. Γράφετε unit tests ;



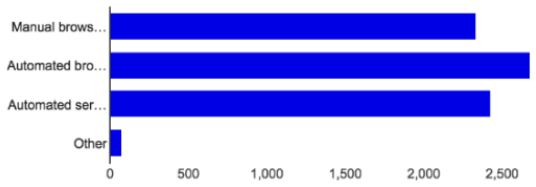

| | | |
|---|---|---|
| Yes | **1848** | 34.8% |
| No | **1154** | 21.7% |
| Sometimes | **2279** | 42.9% |
| Other | **31** | 0.6% |

### 11.13. Γράφετε continuous Integration tests ;



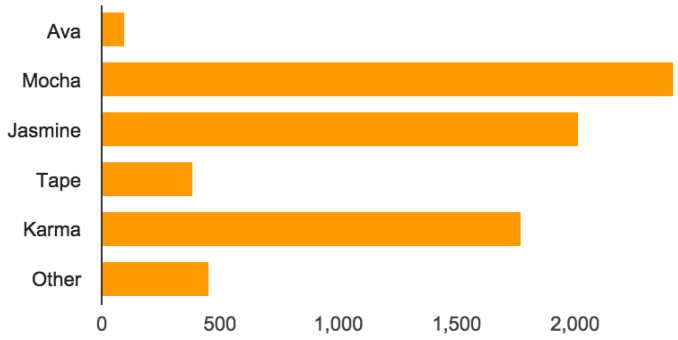

| | | |
|---|---|---|
| Yes | **1698** | 32% |
| No | **2193** | 41.4% |
| Sometimes | **1388** | 26.2% |
| Other | **20** | 0.4% |

### 11.14. Πως τρέχετε τα test σας ;



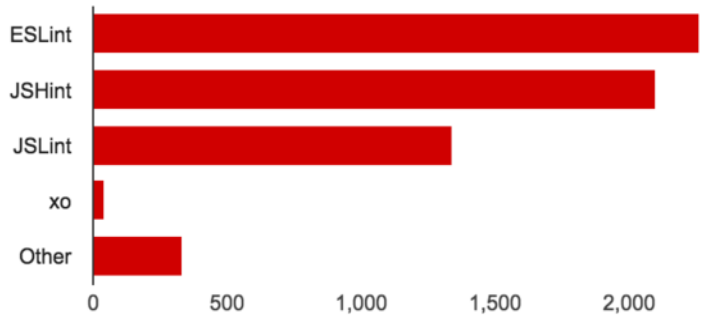

| | | |
|---|---|---|
| Manual browser testing (e.g using DevTools) | **2332** | 51.3% |
| Automated browser testing (PhantomJS, others) | **2680** | 59% |
| Automated server JavaScript tests | **2432** | 53.5% |
| Other | **78** | 1.7% |

### 11.15. Ποιες βιβλιοθήκες χρησιμοποιείτε για τα test ;



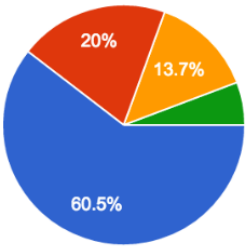

| | | |
|---|---|---|
| Ava | **96** | 2.4% |
| Mocha | **2418** | 61.5% |
| Jasmine | **2014** | 51.3% |
| Tape | **386** | 9.8% |
| Karma | **1774** | 45.2% |
| Other | **453** | 11.5% |

### 11.16. Χρησιμοποιείτε εργαλεία ποιότητας κώδικα ;



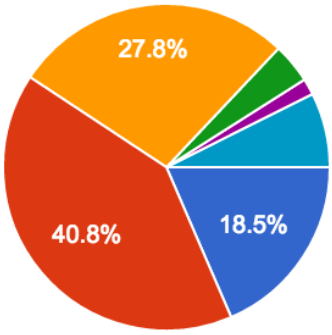

| | | |
|---|---|---|
| ESLint | **2263** | 50.5% |
| JSHint | **2097** | 46.8% |
| JSLint | **1338** | 29.9% |
| xo | **46** | 1% |
| Other | **334** | 7.5% |

### 11.17. Τι εργαλείο χρησιμοποιείτε για τα dependencies στο front end ;



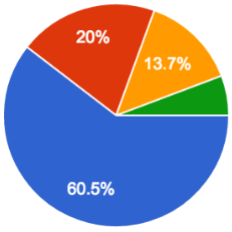

| | | |
|---|---|---|
| npm | **3117** | 60.5% |
| Bower | **1031** | 20% |
| By the grace of bespoke <script> tags | **707** | 13.7% |
| Other | **298** | 5.8% |

### 11.18. Τι εργαλείο χρησιμοποιείτε για τα builds ;



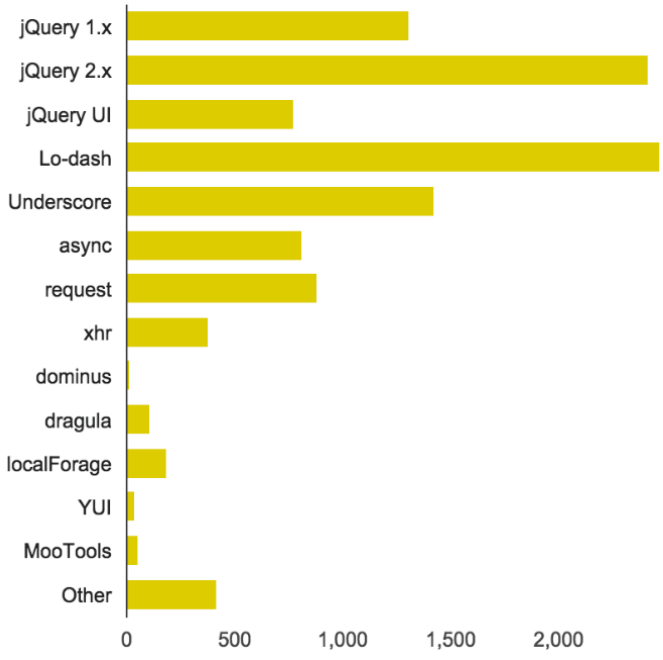

| | | |
|---|---|---|
| Grunt | **911** | 18.5% |
| Gulp | **2007** | 40.8% |
| npm run | **1366** | 27.8% |
| Make | **195** | 4% |
| Broccoli | **79** | 1.6% |
| Other | **364** | 7.4% |

### 11.19. Ποιο εργαλείο χρησιμοποιείτε για να φορτώσετε JavaScript στο font end ;



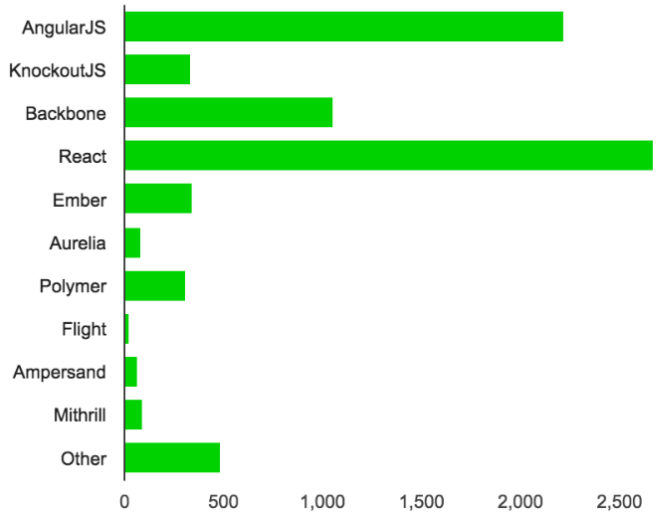

| | | |
|---|---|---|
| npm | **3117** | 60.5% |
| Bower | **1031** | 20% |
| By the grace of bespoke <script> tags | **707** | 13.7% |
| Other | **298** | 5.8% |

### 11.20. Ποιες βιβλιοθήκες χρησιμοποιείτε ;



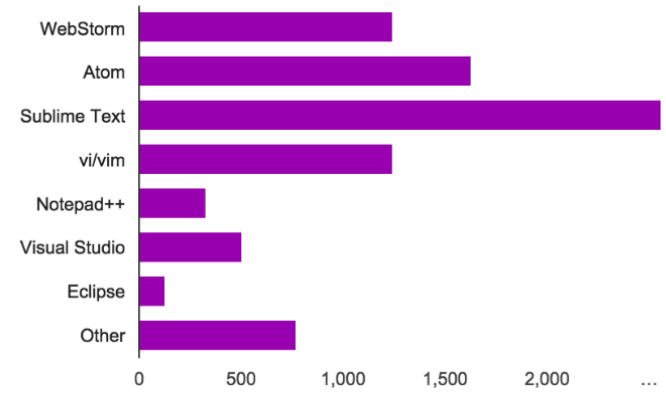

| | | |
|---|---|---|
| jQuery 1.x | **1306** | 27.7% |
| jQuery 2.x | **2419** | 51.3% |
| jQuery UI | **778** | 16.5% |
| Lo-dash | **2470** | 52.3% |
| Underscore | **1427** | 30.2% |
| async | **813** | 17.2% |
| request | **884** | 18.7% |
| xhr | **377** | 8% |
| dominus | **11** | 0.2% |
| dragula | **107** | 2.3% |
| localForage | **189** | 4% |
| YUI | **40** | 0.8% |
| MooTools | **52** | 1.1% |
| Other | **415** | 8.8% |

### 11.21. Ποια frameworks χρησιμοποιείτε ;



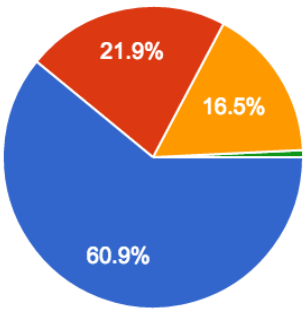

| | | |
|---|---|---|
| AngularJS | **2218** | 47.8% |
| KnockoutJS | **333** | 7.2% |
| Backbone | **1059** | 22.8% |
| React | **2672** | 57.6% |
| Ember | **346** | 7.5% |
| Aurelia | **82** | 1.8% |
| Polymer | **309** | 6.7% |
| Flight | **27** | 0.6% |
| Ampersand | **64** | 1.4% |
| Mithrill | **91** | 2% |
| Other | **490** | 10.6% |

### 11.22. Ποιόν editor προτιμάτε ;



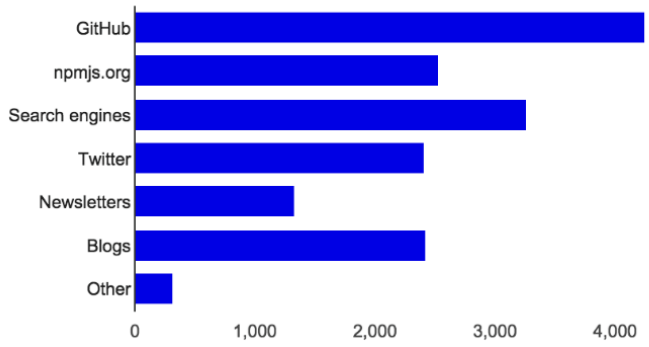

| | | |
|---|---|---|
| WebStorm | **1242** | 25.6% |
| Atom | **1633** | 33.6% |
| Sublime Text | **2560** | 52.7% |
| vi/vim | **1247** | 25.7% |
| Notepad++ | **327** | 6.7% |
| Visual Studio | **508** | 10.5% |
| Eclipse | **130** | 2.7% |
| Other | **773** | 15.9% |

### 11.23. Ποιο λειτουργικό σύστημα προτιμάτε ;




| | | |
|---|---|---|
| Mac | **3209** | 60.9% |
| Linux | **1154** | 21.9% |
| Windows | **867** | 16.5% |
| Other | **40** | 0.8% |

### 11.24. Που βρίσκετε έτοιμο κώδικα ή βιβλιοθήκες και εργαλεία ;

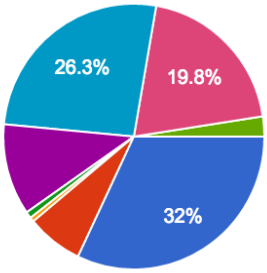



| | | |
|---|---|---|
| GitHub | **4251** | 81.1% |
| npmjs.org | **2534** | 48.4% |
| Search engines | **3266** | 62.3% |
| Twitter | **2409** | 46% |
| Newsletters | **1334** | 25.5% |
| Blogs | **2427** | 46.3% |
| Other | **322** | 6.1% |

### 11.25. Συμμετέχετε σε κοινωνικές εκδηλώσεις σχετικές με την JavaScript ;

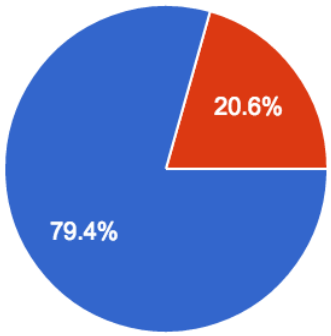

| | | |
|---|---|---|
| Yes, I've been to Meetups | **2376** | 74% |
| Yes, I've been to Conferences | **1902** | 59.2% |
| Other | **281** | 8.8% |

### 11.26. Ποιους browsers υποστηρίζετε στις JavaScript εφαρμογές ;



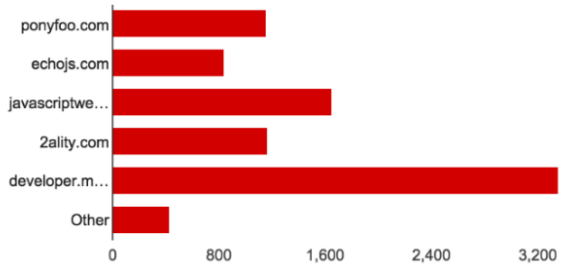

| | | |
|---|---|---|
| Only evergreen browsers | **1665** | 32% |
| Safari is the new IE | **361** | 6.9% |
| IE6+ | **28** | 0.5% |
| IE7+ | **42** | 0.8% |
| IE8+ | **579** | 11.1% |
| IE9+ | **1365** | 26.3% |
| IE10+ | **1029** | 19.8% |
| Other | **127** | 2.4% |

### 11.27. Ενημερώνεστε για νέες λειτουργίες στην JavaScript ;



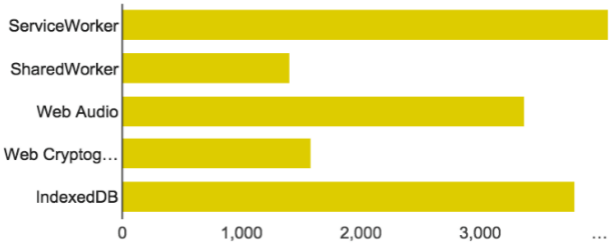

| | | |
|---|---|---|
| Yes | **4155** | 79.4% |
| No | **1077** | 20.6% |

### 11.28. Από πού ενημερώνεστε για νέες λειτουργίες της JavaScript ;



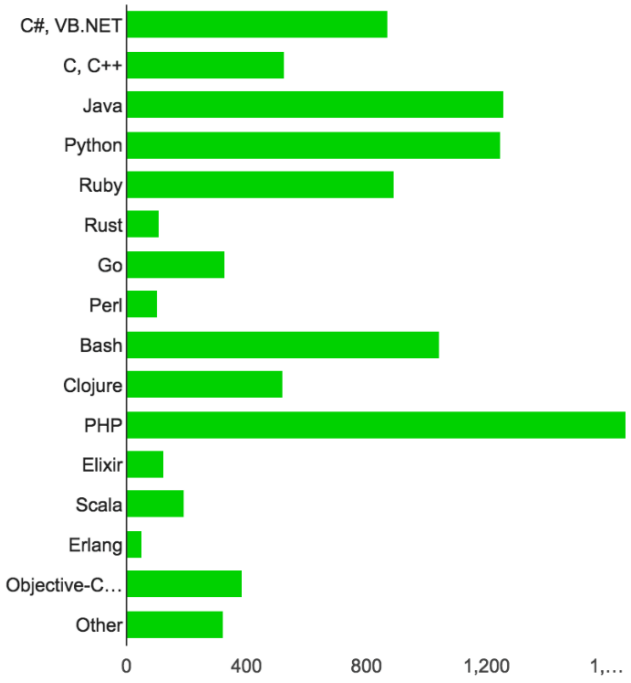

| | | |
|---|---|---|
| ponyfoo.com | **1159** | 26.2% |
| echojs.com | **846** | 19.1% |
| javascriptweekly.com | **1651** | 37.3% |
| 2ality.com | **1167** | 26.3% |
| developer.mozilla.org (MDN) | **3357** | 75.8% |
| Other | **431** | 9.7% |

### 11.29. Ποια από τις παρακάτω λειτουργίες της JavaScript γνωρίζετε ;




| | | |
|---|---|---|
| ServiceWorker | **4076** | 85.1% |
| SharedWorker | **1405** | 29.3% |
| Web Audio | **3377** | 70.5% |
| Web Cryptography | **1582** | 33% |
| IndexedDB | **3795** | 79.3% |

### 11.30. Ποια άλλη γλώσσα προγραμματισμού εκτός JavaScript χρησιμοποιείτε ;



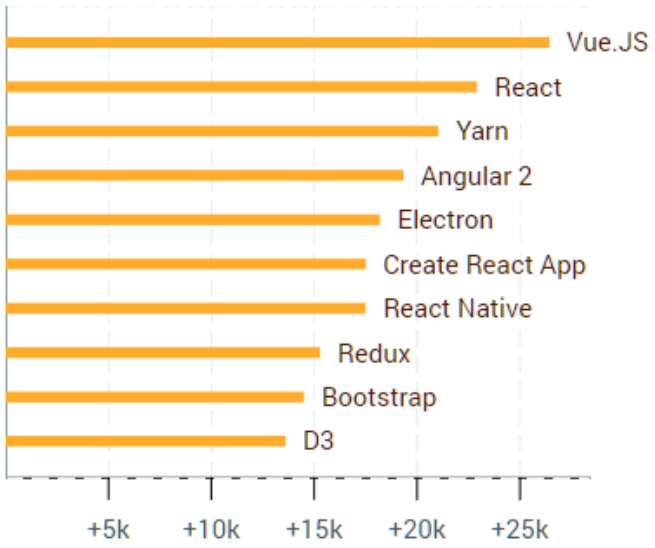

| | | |
|---|---|---|
| C#, VB.NET | **870** | 18.7% |
| C, C++ | **525** | 11.3% |
| Java | **1258** | 27.1% |
| Python | **1246** | 26.8% |
| Ruby | **894** | 19.3% |
| Rust | **112** | 2.4% |
| Go | **327** | 7% |
| Perl | **105** | 2.3% |
| Bash | **1044** | 22.5% |
| Clojure | **522** | 11.2% |
| PHP | **1666** | 35.9% |
| Elixir | **125** | 2.7% |
| Scala | **193** | 4.2% |
| Erlang | **50** | 1.1% |
| Objective-C, Swift | **389** | 8.4% |
| Other | **324** | 7% |

# ΚΕΦ.12: Η εργαλειοθήκη της JavaScript το 2016

## 12. Ανάλυση των καλύτερων εργαλείων του 2016

Μια σημαντική ανασκόπηση έγινε με το κλείσιμο του 2016 για τα πιο πετυχημένα JavaScript projects του 2016, τα οποία συνθέτουν και την εργαλειοθήκη ενός καλού JavaScript developer. Θα δούμε όλα τα εργαλεία όπως παρουσιάστηκαν στον ιστότοπο risingstars2016.js.org

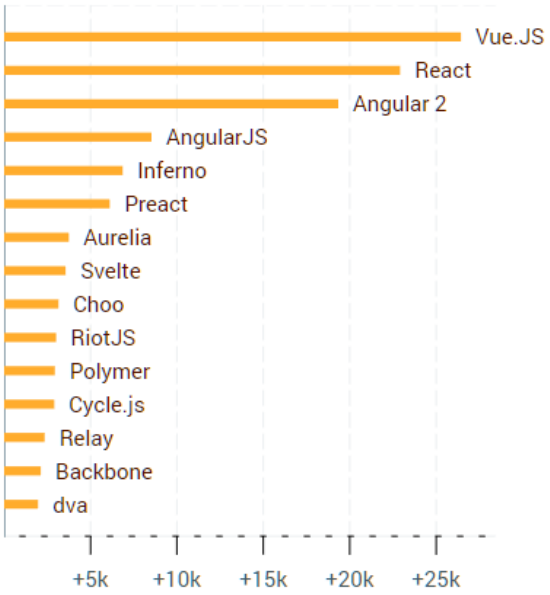
### 12.1. Ποιο πετυχημένα Projects του 2016




Το Vue.JS και δύο projects του Facebook, το React και ο νέος package manager Yarn ήταν τα 3 πιο πετυχημένα projects στο github το 2016

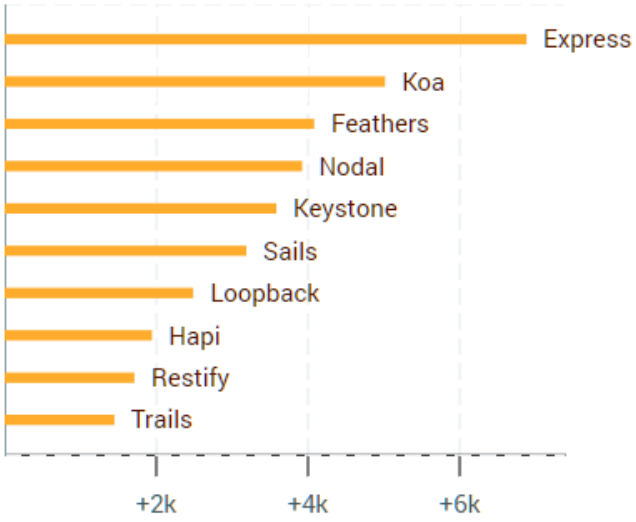
Το Vue.js ήδη χρησιμοποιείται από το Alibaba, το μεγαλύτερο εμπορικό site της Κίνας

### 12.2. Front-end Frameworks



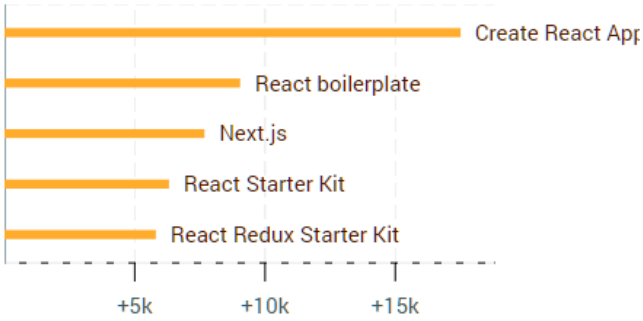

Για το 2016 ο newcomer είναι το Vue το οποίο εκτόπισε στην 3η θέση την Angular που βέβαια παραμένει στην κορυφή παγκοσμίως ως front-end εργαλείο. Αντίθετα το React παραμένει σε ισχύ, κυρίως για το οικοσύστημα που παρουσιάζει (flux, graphQL, native κλπ)

### 12.3. Node.js Frameworks



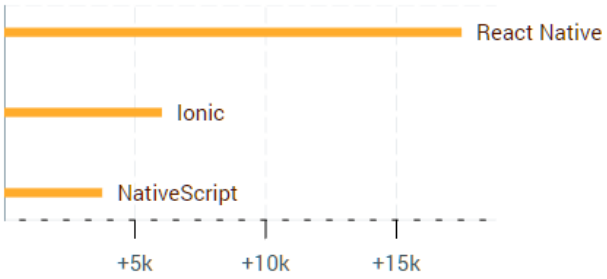

Ολοι οι full stack JavaScript developers γνωρίζουν το Express. Πολύ ενδιαφέρον project το Feathers για όσους θέλουν να φτιάξουν microservices χρησιμοποιώντας το node. Στην 6η θέση το Sails είναι το πιο γνωστό MVC framework για node

### 12.4. React Boilerplates



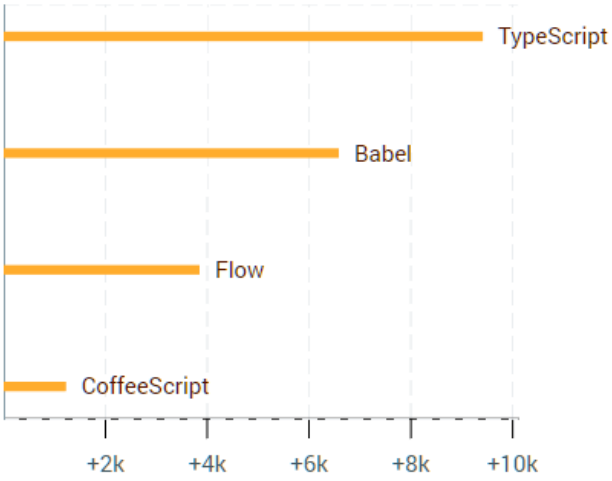

Τα boilerplates είναι ο «τυφλοσούρτης» για κάθε framework. Ετοιμάζουν μια αρχική εφαρμογή μαζί με όλο το directory tree. Δεν είναι τυχαίο που το πιο χρησιμοποιημένο boilerplate για React είναι του ίδιου του δημιουργού του React, Dan Abramov.

### 12.5. Mobile



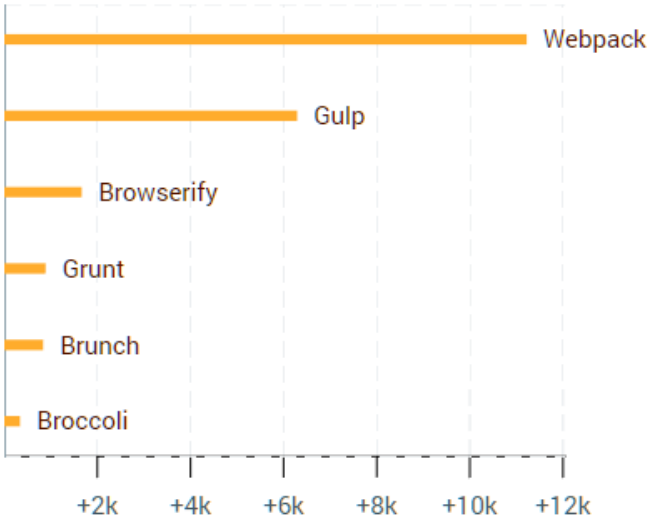

Είναι πρώτη φορά που εκτοπίζεται η χρήση του webview (phonegap, ionic) από το σετ εργαλείων και τη θέση του παίρνουν νέες λύσεις που εμφανίστηκαν μόλις πριν 2 χρόνια, το React Native και η NativeScript.

### 12.6. Compilers



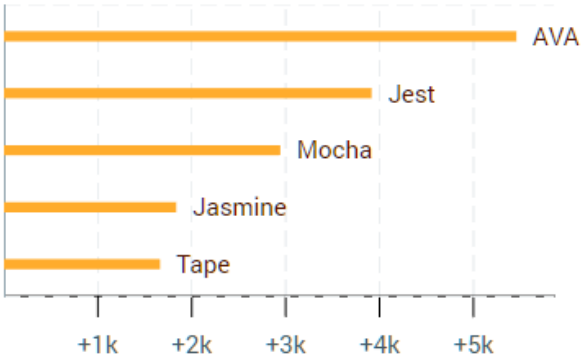

Το 2016 υπήρχε τρομακτική άνοδος της TypeScript εξαιτίας της χρήσης της στην Angular 2, η οποία εκτόπισε την CoffeeScript. Στη λίστα υπάρχει φυσικά η Babel, ο μόνος τρόπος χρήσης νέων ES6 και ES7 χαρακτηριστικών σε παλιούς browsers.

## 12.7. Build Tools



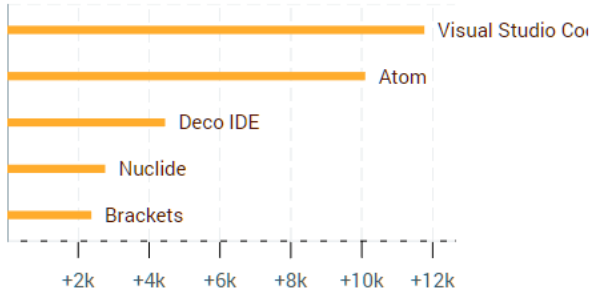

Αν και το Gulp ξεκίνησε ως το εργαλείο για να γράφονται batch αρχεία με το node, κατέληξε ως το πιο γνωστό build tool, τη δουλειά του οποίου κάνει πολύ καλύτερα το webpack. Στη λίστα βλέπουμε και το browserify, το βασικό εργαλείο για να μεταφέρουμε βιβλιοθήκες του node στο front-end.

## 12.8. Testing Frameworks



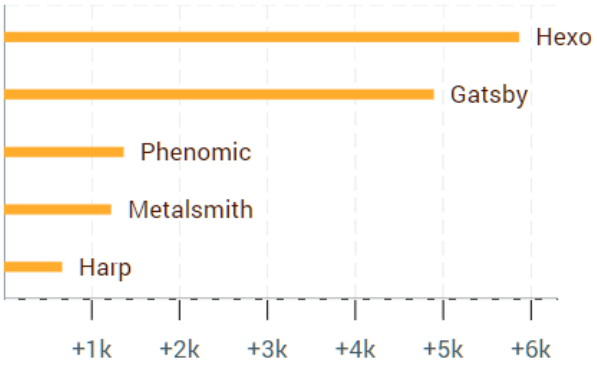

Σε σχέση με το ερωτηματολόγιο της JavaScript, εδώ βλέπουμε να εμφανίζεται το Jest, εξαιτίας της πολύ καλής συνεργασίας με το React.

## 12.9. IDE




Πολύ ενδιαφέρουσα εξέλιξη, είναι η ότι στην πρώτη θέση βλέπουμε για το 2016 το Visual Studio Core, κι αυτό λόγω του IntelliSense που δίνει για την TypeScript. Και τα δύο projects είναι βασισμένα στο Electron, με το ένα να είναι γραμμένο σε TypeScript και το άλλο σε CoffeeScript, ενώ το Brackets της Adobe είναι γραμμένο σε JavaScript

## 12.10. Static Site Generators

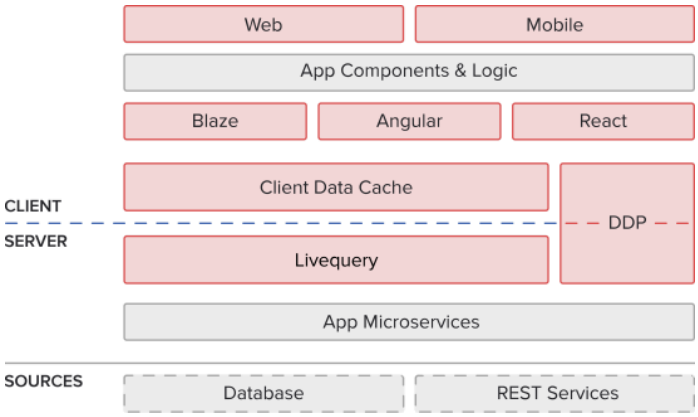



Τα sites του 2016 δεν χτίζονται πια σε WordPress αλλά σε εργαλεία όπως το Hexo. Το Gatsby και το Phenomic ανεβαίνει λόγω της εκτενής χρήσης του React.

# ΚΕΦ.13: Παρουσιάζοντας ένα full stack framework

## 13. Meteor – ένα ισομορφικό framework

Το Meteor ανήκει στην κατηγορία των ισομορφικών frameworks με την έννοια ότι γράφουμε κώδικα JavaScript στην πλευρά του client και του server, ενώ χρησιμοποιείται και η JavaScript για την προσπέλαση των δεδομένων. Η λογική στην ανάπτυξη των εφαρμογών με το Meteor πλησιάζει πιο πολύ το MVVM pattern[86] με τη διαφορά ότι ένα μικρό αντίγραφο των δεδομένων κρατιούνται και στην μεριά του ViewModel (στον client) σε μια μίνι in-memory έκδοση της MongoDB, την minimongo (github.com/meteor/meteor/tree/devel/packages/minimongo). Για το View κομμάτι το meteor χρησιμοποιεί templates στη μορφή του handlebars.js (στη δική του διάλεκτο που ονομάζονται “Spacebars”). Ενώ στην πρόσφατη έκδοση προστέθηκαν τα δύο πιο δημοφιλή Angular και React.



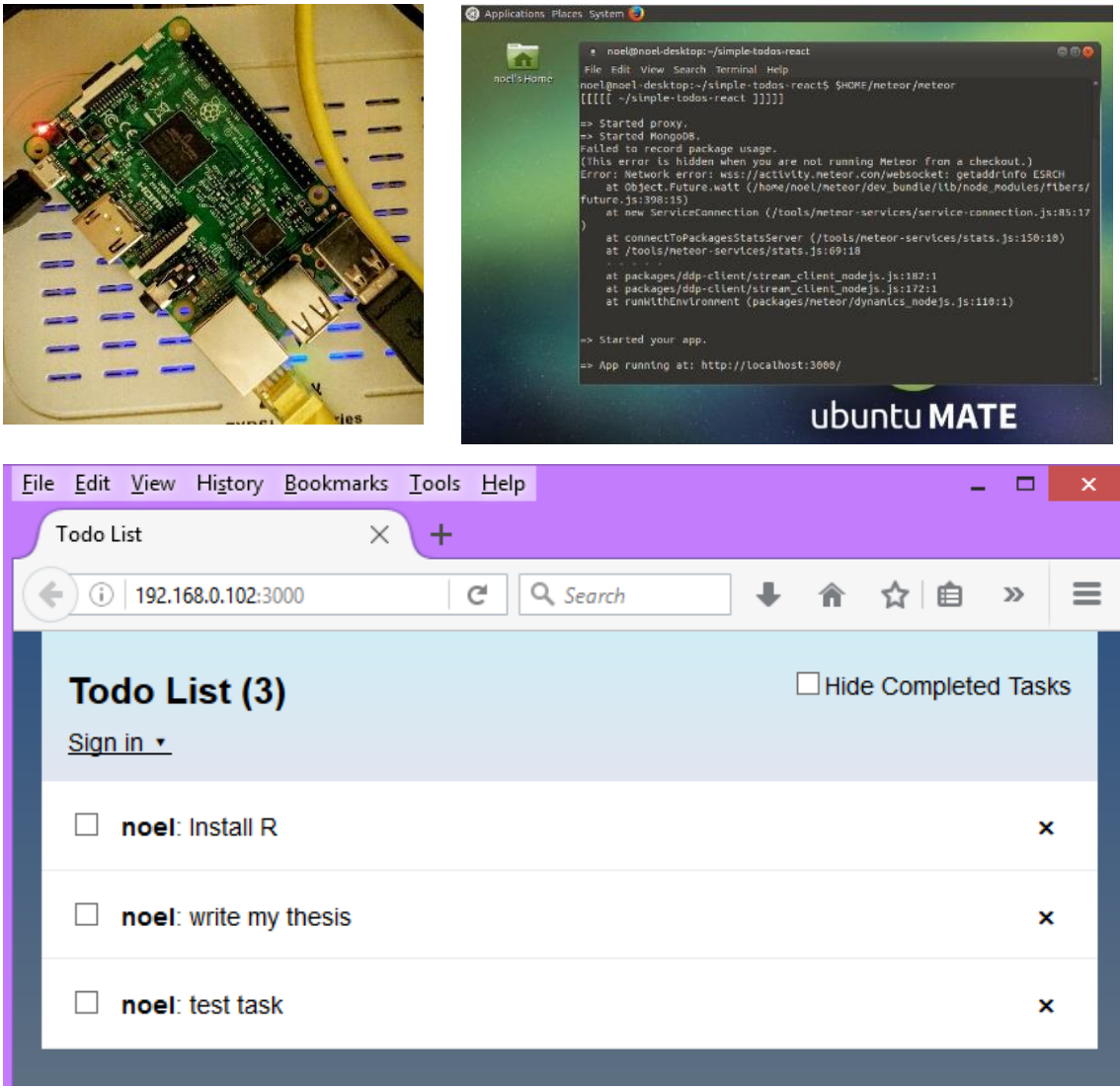

Φωτογραφία: η αρχιτεκτονική του meteor

### 13.1. Η αρχιτεκτονική του meteor

Στη λογική του Meteor και δεδομένου ότι η τάση βρίσκεται αυτή τη στιγμή σε εργαλεία τύπου boilerplates όπου στήνεται η αρχική δομή των αρχείων και των βιβλιοθηκών “don’t reinvent the wheel”, το Meteor έχει ήδη έτοιμες λύσεις για login, registration, authentication και setup της βάσης, πράγματα που θεωρούνται πλέον αυτονόητα σε κάθε web εφαρμογή.

Αυτό που κάνει πολύ ιδιαίτερο το Meteor είναι ότι οι αλλαγές στους clients και στη βάση συγχρονίζονται συνεχώς μέσω μηχανισμού publish/subscribe και αυτές με τη σειρά τους αντικατοπτρίζονται real time[100] στο view με έναν τρόπο που στις μέρες μας ονομάζεται reactive (διαδραστικό σε ελεύθερη μετάφραση). Μάλιστα θα λέγαμε ότι σε ένα μοντέρνο περιβάλλον που χιλιάδες χρήστες είναι συνδεδεμένοι σε μία web εφαρμογή είτε από το κινητό τους είτε από τον browser, αναμένουν ότι αλλαγές γίνονται (ανακοινώνονται) από τη βάση, να τις βλέπουν άμεσα, χωρίς δηλαδή τη χρήση του refresh. Αυτός ο συγχρονισμός των δεδομένων γίνεται μέσω του πρωτοκόλου DDP (Distributed Data Protocol) το οποίο με τη σειρά του είναι βασισμένο στα websockets (sockjs). Στην πλευρά του server υπάρχει φυσικά το NodeJS ενώ το framework συνοδεύεται από δικό του packaging σύστημα παρόμοιο του npm που ονομάζεται atmosphere.

Η εταιρεία που το αναπτύσει αναφέρει ότι τα έσοδά της προέρχονται από τις cloud υπηρεσίες τις οποίες παρέχει για όποιους θέλουν να κάνουν hosting των εφαρμογών της στο galaxy όπως το ονομάζει. Το ίδιο το framework είναι open source και βασίζεται σε μεγάλο βαθμό σε άλλα πετυχημένα open source projects όπως θα δούμε παρακάτω, με αποτέλεσμα να είναι εφικτή η μεταφορά του και σε άλλες πλατφόρμες.




Φωτογραφία: δοκιμαστική εφαρμογή to-do list που φτιάχτηκε για της ανάγκες της εργασίας. Πάνω αριστερά: το Meteor τρέχει σε Ubuntu server σε Rasbery Pi3 με λιγότερο από 3 Watt κατανάλωσης

## 13.2. Υλοποιώντας μια To-Do List

Η παρουσίαση της εφαρμογής που βλέπουμε στη φωτογραφία παραπάνω γίνεται με το Blaze που είναι η UI βιβλιοθήκη του Meteor χωρίς να είναι απαραίτητα η μόνη. Ηδη από το 2015 το Meteor μπορεί να χρησιμοποιήσει την Angular ενώ το τελευταίο διάστημα γίνεται πιο δημοφιλές το React. Η ίδια η εφαρμογή πακετάρεται εύκολα με το Cordova ώστε να τρέξει σαν να ήταν native (hybrid) σε Android ή iOS αρκεί να έχει υιοθετηθεί κατάλληλη responsive σχεδίαση. Η εφαρμογή To-Do list είναι το βασικό παράδειγμα στο tutorial του Meteor. Για το σκοπό της εργασίας έγινε μια δοκιμή της

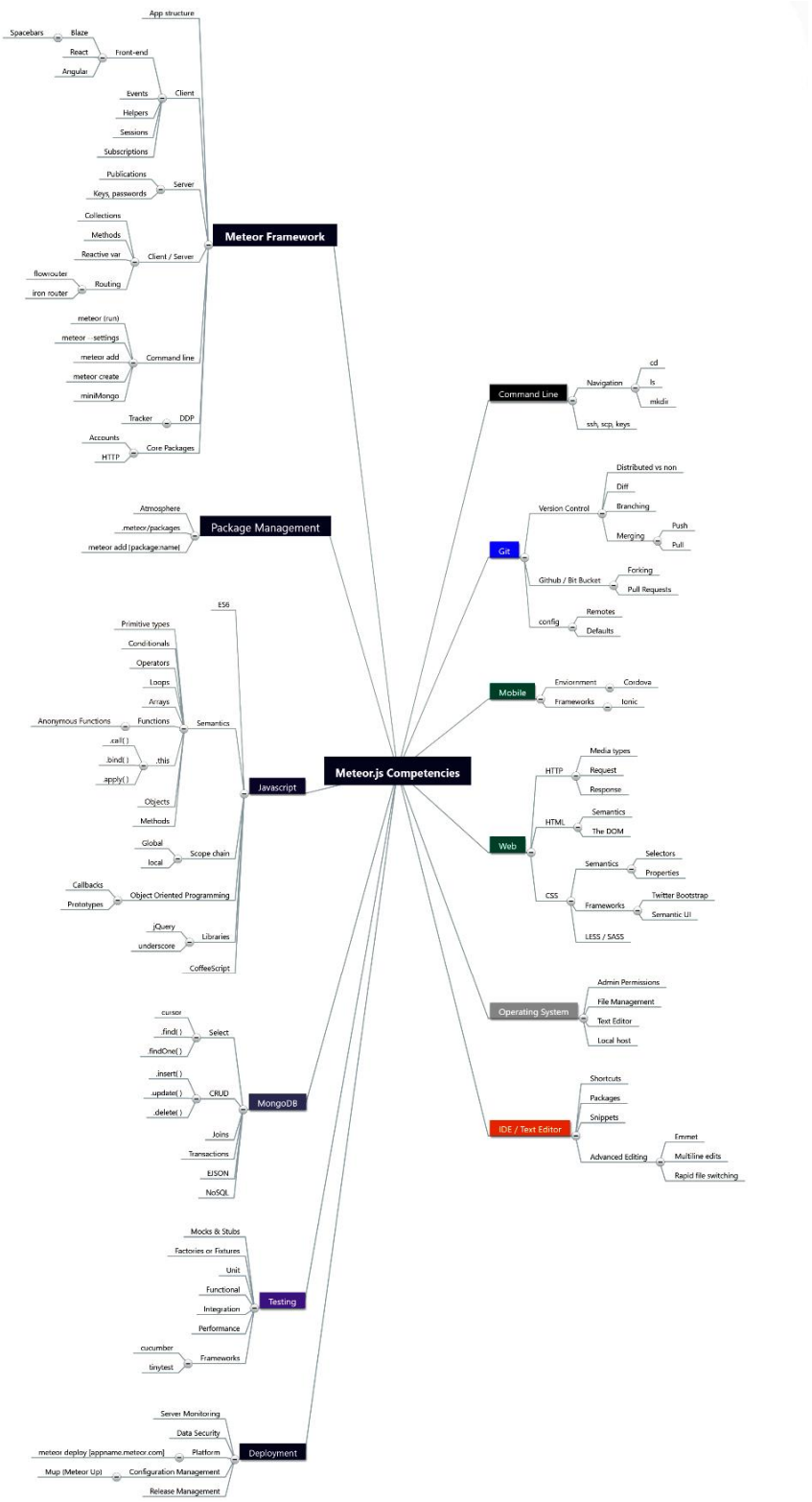
εφαρμογής to-do σε meteor πάνω σε πλατφόρμα Rasberry PI χρησιμοποιώντας το universal fork (github.com/4commerce-technologies-AG/meteor)

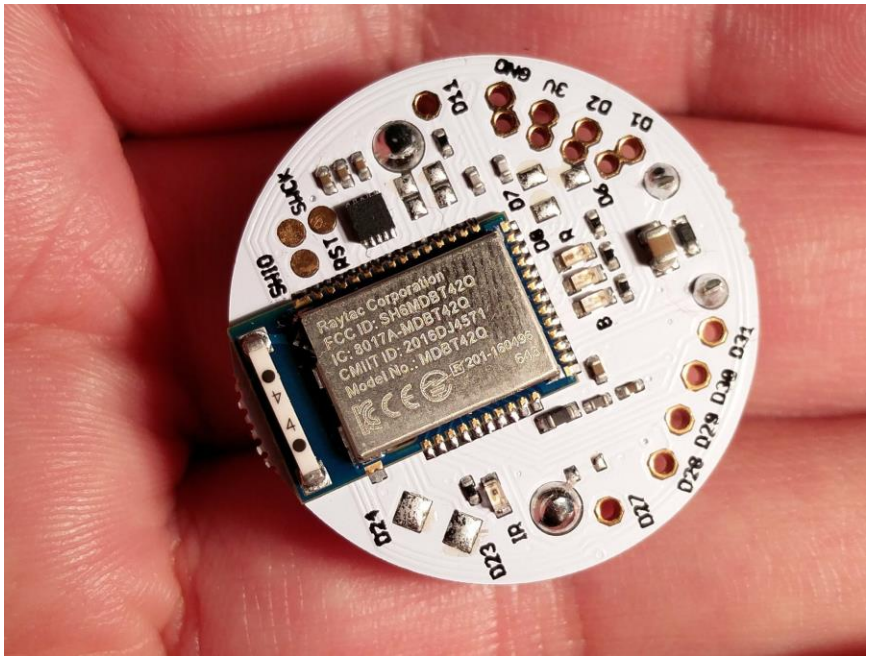
Το Blaze με τη σειρά του χρησιμοποιεί μια βιβλιοθήκη διαδραστικού προγραμματισμού (reactive programming) που λέγεται Tracker. Το reactive programming είναι μια τεχνοτροπία (RX) που συναντάμε σε πολλές γλώσσες προγραμματισμού (παράδειγμα API: ReactiveX ή Bacon.js στη JavaScript) όπου οι ίδιες οι μεταβλητές στον κώδικα αλλάζουν ασχέτως της ροής του. Ειδικά στην περίπτωσή μας, και εφόσον οι μεταβλητές είναι δεμένες με κάποιο πεδίο στη βάση και αλλάζουν συνέχεια τιμές, με το tracker παρακολουθούμε τις αλλαγές αυτές και δένουμε functions οι οποίες εκτελούνται όταν οι μεταβλητές αλλάζουν. Οι αλλαγές στις τιμές μεταφέρονται όπως αναφέραμε με το DDP στο φορμάτ EJSON. Η διαφορά του EJSON με το JSON είναι ότι υπάρχει πάντα ημερομηνία και ώρα στο αντικείμενο ώστε να κρατηθεί το πιο πρόσφατο στη βάση. Το ίδιο το πρωτόκολλο DDP ήδη έχει μεταφερθεί και σε άλλες γλώσσες (C#, Go, Haskell, Java, PHP, Ruby, Swift κλπ) ώστε να χρησιμοποιείται και σε εκτός Meteor εφαρμογές.

Το ίδιο συμβαίνει και με την MongoDB. Σε περίπτωση που η κεντρική MongoDB που χρησιμοποιεί η εφαρμογή μας αλλάζει και από άλλες εφαρμογές που τρέχουν παράλληλα, οι αλλαγές στη βάση συγχρονίζονται και αυτές στο μοντέλο μας και εμφανίζονται στο view model, χωρίς να απαιτείται επιπλέον προγραμματισμός.

## 13.3. Δομή μιας τυπικής εφαρμογής σε meteor

Μια τυπική δομή μιας εφαρμογής σε meteor παρουσιάζεται στο Mindmap της Shavonnah Tièra από goo.gl/BnzEaw :






Φωτογραφία: puckjs, ο μικρότερος αυτόνομος υπολογιστής JavaScript

## 13.4. Υλοποιώντας μια εφαρμογή IoT

Κατά τη διάρκεια συγγραφής της εργασίας αναφέραμε (κεφάλαιο 2.5) την περίπτωση του puck.js Πράγματι, συμμετείχα στην εκστρατεία του Kickstarter και όταν παρέλαβα δύο puckjs τα Χριστούγεννα του 2016 εξερεύνησα τη δυνατότητα χρήσης της συσκευής που παρέχει θερμόμετρο, φωτόμετρο, μαγνητόμετρο, τρία έγχρωμα led και ένα κουμπί. Η συσκευή για τη διασύνδεση με τον έξω κόσμο χρησιμοποιεί Bluetooth Low Energy με *εμβέλεια έως 50 μέτρα* και NFC, ενώ η μπαταρία της ανάλογα με τη χρήση μπορεί να φτάσει *ένα χρόνο*, παρότι με πειράματα που έκανα εάν η συσκευή είναι συνδεμένη καταναλώνει αρκετή ενέργεια με τυπική αναμενόμενη διάρκεια ενός μήνα.

Δοκιμάζοντας τη συσκευή μου ήρθε η ιδέα της κατασκευής ενός κουμπιού SOS «Emergency Center» για ηλικιωμένους που θα φορούσαν το puckjs ως μενταγιόν. Κατά το πάτημα του κουμπιού (όλη η συσκευή είναι σαν ένα μεγάλο κουμπί) θα μπορούσα να ανάψω το κόκκινο led και ταυτόχρονα να δώσω σήμα σε ένα κοντινό κινητό τηλέφωνο που θα τρέχει μια εφαρμογή άμεσης κλήσης, το οποίο θα καλούσε το άτομο που προσέχει τον ηλικιωμένο. Μάλιστα η εφαρμογή θα μπορούσε να επεκταθεί αργότερα (προς πώληση) ελέγχοντας με βάση το μαγνησιόμετρο εάν ο ηλικιωμένος έχει πέσει κάτω (απότομη μεταβολή των μετρήσεων) ή αν υπάρχει πρόβλημα

θερμοκρασίας στο χώρο (αύξηση εκτός ορίων ή απότομη αλλαγή-φωτιά-φωτισμός κλπ).

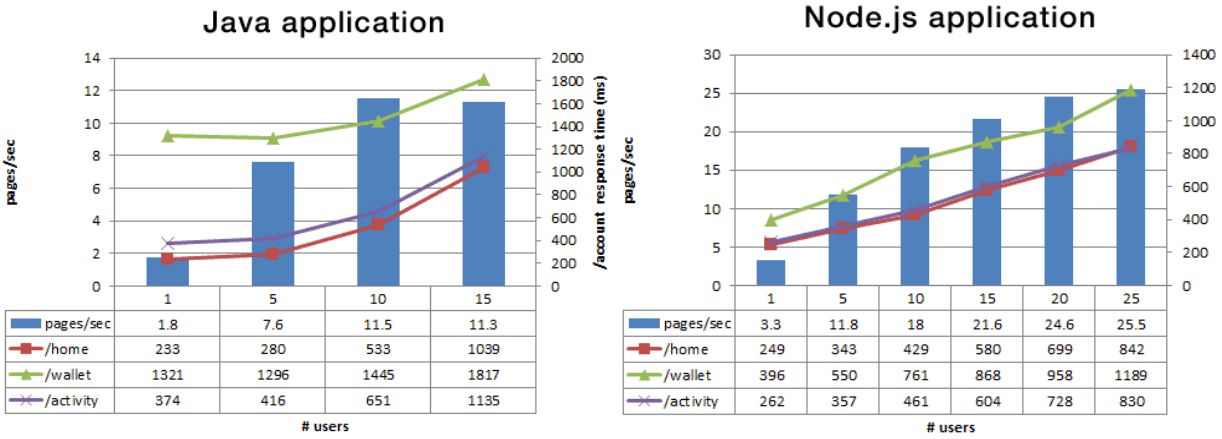
Στις πρώτες δοκιμές υπήρχε μεγάλη αποτυχία δεδομένου ότι το πρωτόκολλο BLE (Bluetooth 4) δεν υποστηρίζεται ακόμα καλά από τους επιτραπέζιους υπολογιστές, ενώ η τάση για physical web (επικοινωνία μέσω του browser απευθείας σε συσκευή BLE) είναι υπό δοκιμή ακόμα (υποστήριξη μόνο σε android και σε desktop chrome από έκδοση 56 και μετά, και μόνο εφόσον ο υπολογιστής έχει δυνατότητα Bluetooth 4 με BLE)

Μετά από προσωπική επικοινωνία με τον κατασκευαστή Gordon Williams προτάθηκαν οι εξής δυνατότητες: χρήση του πρωτόκολλου BPAP (Bluetooth phonebook access profile) ή χρήση της NativeScript που παρουσιάστηκε στο κεφάλαιο 7.2 για χρήση μέσω native API του κινητού. Στην αναζήτηση λύσης μας διευκόλυνε ο David Smart, δημιουργός του IDE DroidScript, ο οποίος είχε νωρίτερα κατασκευάσει ένα plugin για το puckjs.

```
app.LoadPlugin( "PuckJS" );
function OnStart()
{
  lay = app.CreateLayout( "Linear", "VCenter,FillXY" );
  txt = app.CreateText( "Press your puck" );
  lay.AddChild( txt );
  app.AddLayout( lay );
  puck = app.CreatePuckJS();
  puck.SetOnConnect( OnConnect );
  puck.SetOnButton( OnButton );
  puck.Scan( "Puck" );
}
function OnConnect()
{
  puck.WatchButton( true );
}
function OnButton( state )
{
  app.Call("+306984497993" );
}
```

Όπως ανέφερε, το DroidScript μπορεί να παράγει και κώδικα σε service να τρέχει συνέχεια στο background της συσκευής.

# ΚΕΦ.14: Μελέτες περιπτώσεων




Φωτογραφία: Διπλάσια request το δευτερόλεπτο πέτυχε το Paypal με περίπου 35% μικρότερο χρόνο αναμονής

### 14. Μεγάλες εγκαταστάσεις JavaScript

Η εργασία δε θα ήταν ολοκληρωμένη αν δεν παρουσιάζαμε μελέτες περιπτώσεων σε μεγάλες εγκαταστάσεις JavaScript. Όπως έχει δηλώσει ο Quincy Larson, ιδρυτής του freeCodeCamp: «ότι μπορεί να γραφτεί σε JavaScript, γράφεται σε JavaScript», «το Netflix χρησιμοποιεί το Node.js για να διαχειριστεί το streaming 2,7 δισεκατομμυρίων ωρών βίντεο κάθε μήνα». Σύμφωνα με τον οργανισμό Node.js μεγάλες εγκαταστάσεις είναι η Walmart (20000 hits/δευτερόλεπτο), η NASA για τα data transfers, το Groupon, το GoDaddy και η Capital One.

### 14.1. PayPal

Η πιο ενδιαφέρουσα μεταστροφή εγκατάστασης σε full stack JavaScript αποτελεί η περίπτωση του PayPal[91] (180 εκατομμύρια ενεργοί χρήστες). Παρότι ο τρόπος υλοποίησης και οι τεχνολογίες που χρησιμοποιούνται γενικότερα παίζουν τεράστιο ρόλο στη ταχύτητα μιας εφαρμογής, οι μηχανικοί του Paypal κατάφεραν να απαντούν σε διπλάσια server requests με σελίδες που στέλνονταν 200ms πιο γρήγορα μάλιστα χρησιμοποιώντας 1 πυρήνα σε σχέση με 5 πυρήνες στην Java. Πιο ενδιαφέρον είναι το

γεγονός ότι αυτό επιτεύχθηκε με λιγότερους προγραμματιστές που έγραψαν 40% λιγότερο κώδικα.

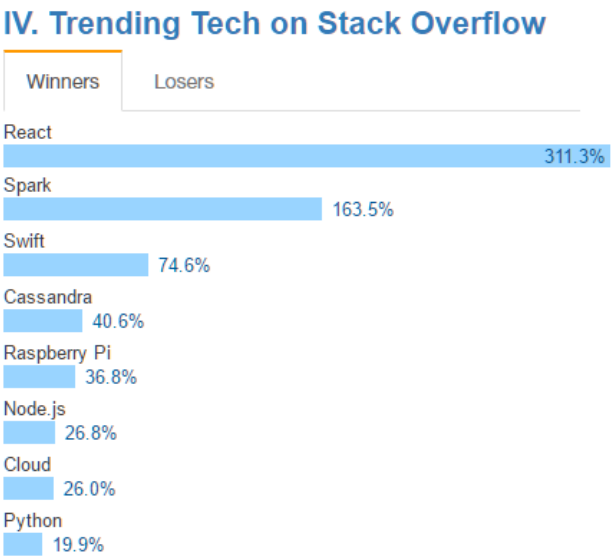
### 14.2. LinkedIN

Εκτός από το PayPal, η εταιρεία LinkedIN χρησιμοποίησε Node και HTML5 για να μετατρέψει τις mobile (web) και native (iOS, Android) εφαρμογές της από Ruby on Rails σε JavaScript πετυχαίνοντας δεκαπλάσια ταχύτητα εκτέλεσης όπως ανέφεραν οι μηχανικοί της[92]. Όπως αναφέρουν οι μηχανικοί του δικτύου των 400 εκ ενεργών χρηστών, χρειάστηκαν να αναλύσουν διάφορες τεχνολογίες (Ruby, Node, Java, Scala) και το Node ήταν η καλύτερη τεχνολογία δεδομένου ότι η εφαρμογή είναι πρακτικά απλά ένας καταναλωτής των διαφόρων web services της εταιρείας. Η εταιρεία προσέλαβε μηχανικούς της μηχανής V8 για το project χάρη στο οποίο μπόρεσε να απαντήσει διπλάσια requests με το 1/3 των servers.

### 14.3. UBER και Spotify

Ένα καλό παράδειγμα πετυχημένης mobile εφαρμογής σε JavaScript αποτελεί το UBER με 150 χιλιάδες οδηγούς και συνολικά 1 δις εκτελεσμένων διαδρομών. Παρότι το UBER χρησιμοποιεί αρκετά το Node[93] αρκετές εφαρμογές της είναι γραμμένες στη γλώσσα Go της Google καθώς και σε Python.

Αντίθετα με το UBER, το Spotify χρησιμοποιεί τη JavaScript για την desktop[95] εφαρμογή της (100 εκατομμύρια χρήστες, Ιούνιος 2016).




Φωτογραφία: Το React είναι η πρώτη σε trend τεχνολογία στο Stack Overflow

### 14.4. Facebook

Τέλος θα ήταν σοφό να αναφερθούμε στην περίπτωση του Facebook, της εφαρμογής με τη μεγαλύτερη εγκατάσταση χρηστών παγκοσμίως (τη στιγμή που γράφεται αυτή η εργασία οι ενεργοί χρήστες το μήνα ξεπερνάνε το 1,7 δισεκατομμύριο) που τρέχει σε πάνω από 180 χιλιάδες servers τους οποίους μάλιστα η ίδια η εταιρεία σχεδιάζει. Παρόλο το μέγεθος της (αξία 360 δις) η εταιρεία έχει μόνο 15 χιλιάδες υπαλλήλους (η Amazon για παράδειγμα έχει 270 χιλιάδες).

Το Facebook γράφτηκε τα Χριστούγεννα του 2003 σε PHP ως μια οπτική αναπαράσταση μιας βάσης δεδομένων MySQL. Η PHP είχε μια παράλληλη ιστορία με την JavaScript καθώς μάλιστα δημιουργήθηκε και αυτή το 1995 ως ένα μικρό σπιτικό project. Αντίστοιχα η δημιουργία της MySQL είχε ξεκινήσει λίγους μήνες νωρίτερα. Μάλιστα ο 20χρονος τότε Zuckerberg είχε δημιουργήσει το site μέσα σε μία βδομάδα.

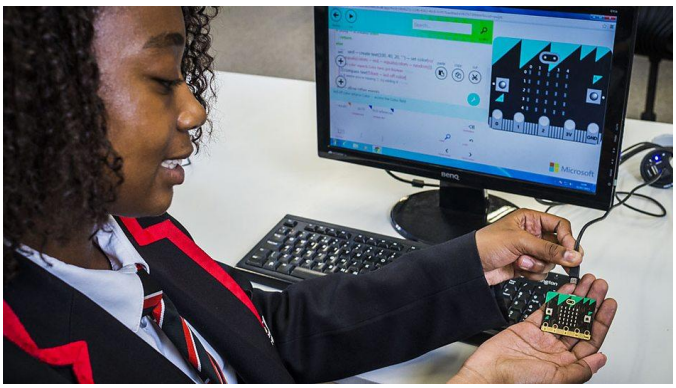
Το 2012 ο Zuckerberg είχε κάνει μια ιστορική δήλωση: «το μεγαλύτερό μας λάθος ήταν ότι επενδύσαμε στην HTML5»[102] αντιλαμβανόμενος ότι η αγορά μετά την έλευση του iPhone το 2007 έγινε «mobile first» τονίζοντας ότι δεν υπήρχε τότε δυνατή ομάδα να γράφει native mobile εφαρμογές. Έπρεπε λοιπόν η εταιρεία να πάρει μια στρατηγική κατεύθυνση για την ανάπτυξη εφαρμογών, μιας που κάθε αλλαγή όπως

είπαμε επηρεάζει πάνω από 1,5 δις χρηστών. Σε αυτό το σημείο πρέπει να τονίσουμε ότι το επιστημονικό δυναμικό της εταιρείας δημιουργεί δικές του γλώσσες προγραμματισμού (βλέπε Hack, HipHop VM). Ετσι το διάστημα αυτό (2012) ξεκίνησε τη δημιουργία ενός δικού της framework για ταχύτερη ανάπτυξη εφαρμογών σε πολλές πλατφόρμες με κομβικό σημείο το React και τεχνολογίες δορυφόρους όπως το Flux, το Relay το GraphQL που περιγράφονται στην εργασία.

Το React εμφανίστηκε στην αγορά το 2013 και όταν ξεκίνησε η συγγραφή αυτής της εργασίας άρχισε να γίνεται παγκοσμίως γνωστό μέχρι που έφτασε No 6 project στο GitHub (github-ranking.com/repositories). Το React όπως είπαμε επιτρέπει πολύ γρήγορη εναλλαγή περιεχομένου σε μια δυναμική σελίδα γράφοντας στο (virtual) DOM απευθείας από την JavaScript (jsx).

Το 2015 η εταιρεία παρουσίασε το React Native με το οποίο ο προγραμματιστής γράφει JavaScript αλλά περιγράφει στοιχεία (components) του Android και του iOS αντί του DOM, δίνοντας έτσι τη δυνατότητα να γραφτούν native εφαρμογές σε κινητά χρησιμοποιώντας JavaScript. Μάλιστα οι ταχύτητες που πετυχαίνει σε σύγχρονες συσκευές είναι της τάξης των 60FPS με αίσθηση που πλησιάζει . Ετσι, το React Native έγινε η πλατφόρμα κοινής ανάπτυξης που ο Zuckerberg τόσο ήθελε.
Το 2016 το Twitter (της οποίας το Bootstrap είναι το #1 repository στο GitHub) ξεκίνησε μια αντίστροφη πορεία, δημιουργώντας το React Native for Web (github.com/necolas/react-native-web), το οποίο με τη σειρά του, παρουσιάζει στο web τα components του React Native, δίνοντας τη δυνατότητα κάποιον που γράφει μια native εφαρμογή για κινητό σε JavaScript, να την εμφανίσει ως ιστοσελίδα.

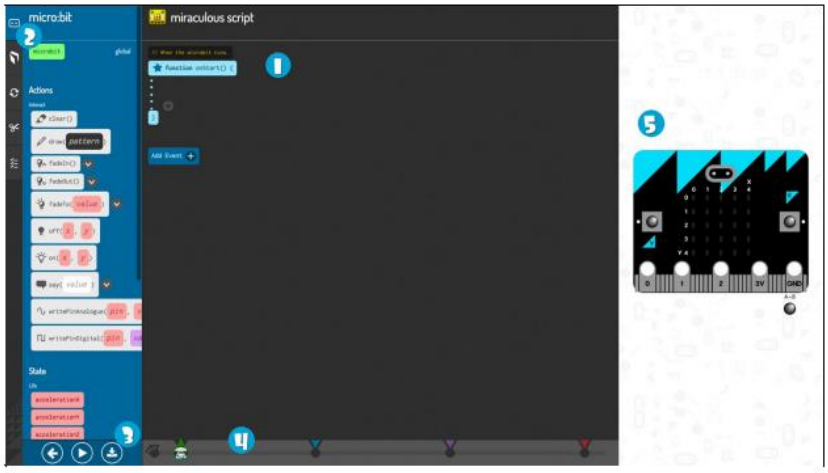
## ΚΕΦ.15: JavaScript στο εκπαιδευτικό σύστημα



Φωτογραφία: Η μικρή πλακέτα-υπολογιστής BBC MicroBit θα μοιραστεί το 2016 σε 1 εκατομμύριο μαθητές

### 15. JavaScript στα σχολεία

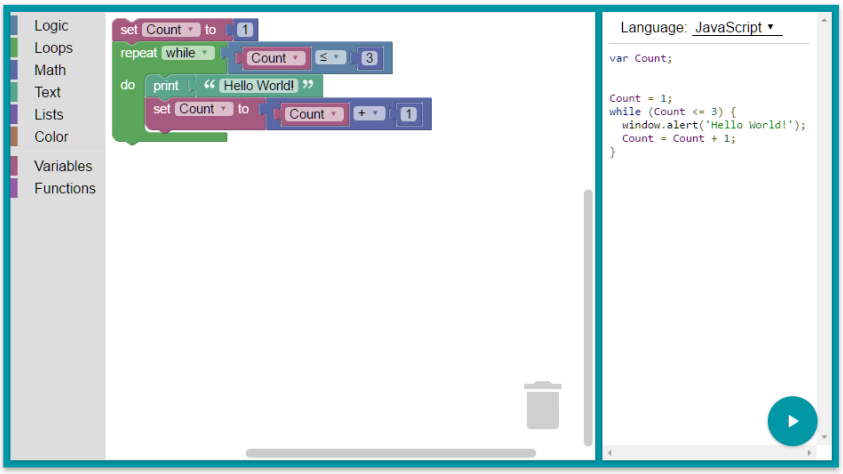
Όπως είχε δηλώσει ο δημιουργός της JavaScript Brendan Eich[68] η JavaScript προοριζόταν ως η αναλογία της VisualBasic ως προς την C++ στο web, δηλαδή η γλώσσα που θα βρισκόταν στο front end όταν το back end θα ήταν η Java. Στις μέρες μας αναβιώνει μετά από 33 χρόνια ένα project της Βρετανικής κυβέρνησης που αποτέλεσε σταθμό στην ιστορία της πληροφορικής, το λεγόμενο BBC Micro. Το project εισήγαγε τον μικροϋπολογιστή στα σχολεία της Αγγλίας το 1981 και αργότερα αποτέλεσε το έναυσμα για τη δημιουργία του επεξεργαστή ARM[74]. Στις μέρες μας το νέο BBC MicroBit θα εισαχθεί στη δημοτική εκπαίδευση της Αγγλίας ώστε τα παιδιά από 7 χρονών να έρθουν σε επαφή με προγραμματιστικά περιβάλλοντα, κυρίως JavaScript, Blockly, Python, and C++ ξεκινώντας από την απλή γλώσσα TouchDevelop της Microsoft με σχήματα[75]. Ο μικροσκοπικός υπολογιστής θα μοιραστεί δωρεάν τον Οκτώβριο του 2015 σε όλα τα βρετανικά σχολεία[76].




Φωτογραφία: Code Kingdoms JavaScript editor για τον micro:bit στα Αγγλικά σχολεία

### 15.1. Code Kingdoms JavaScript

Πρόκειται για το εργαλείο της εταιρείας Code Kingdoms (codekingdoms.com) για εκμάθηση JavaScript στα Αγγλικά σχολεία που έχει διανεμηθεί η πλακέτα micro:bit Η εταιρεία κατασκεύασε το εργαλείο αρχικά για την εκμάθηση της Java.

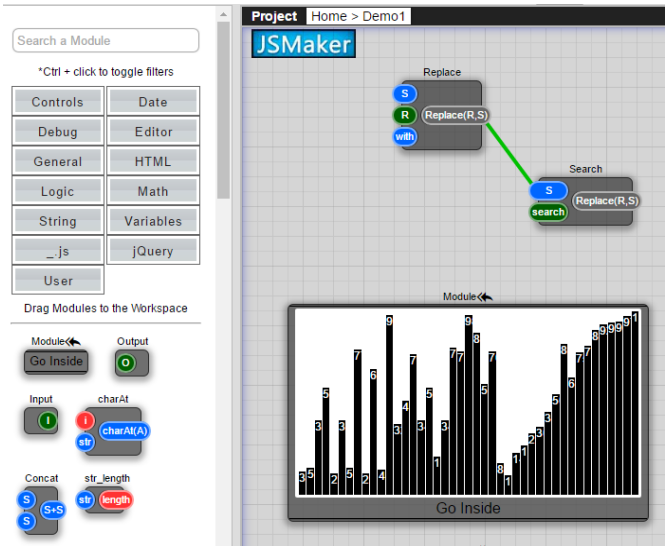



Φωτογραφία: Blockly, ο γραφικός editor της *Google for Education*

### 15.2. Blockly

Πρόκειται για JavaScript βιβλιοθήκη της Google (2012) του Neil Fraser (developers.google.com/blockly), παρόμοια με την Scratch για οπτική ανάπτυξη εφαρμογών για το web αλλά και για android (App Inventor του MIT). Παρότι μπορεί να χρησιμοποιηθεί και επεκτείνεται και σε άλλες γλώσσες (PHP, Python, Lua, Google Dart) η χρήση του γίνεται ιδιαίτερα για την εκμάθηση της JavaScript στα σχολεία. Στα

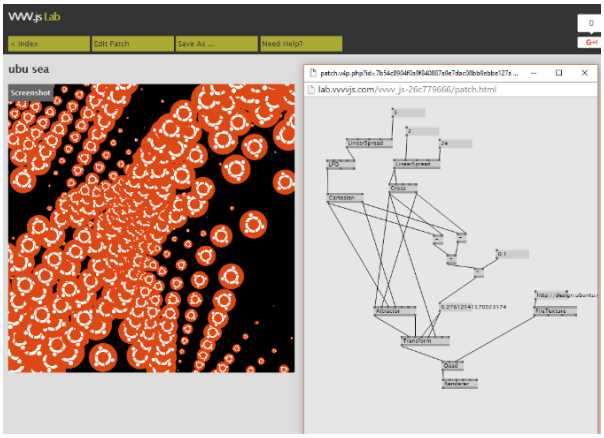
Αγγλικά σχολεία διατίθεται από την Microsoft ως Microsoft Block Editor για χρήση με το micro:bit από παιδιά ηλικίας 11-12 χρονών




Φωτογραφία: JSMaker

## 15.3. JSMaker

Κατασκευασμένο από τον ισραηλινό Barak Igal, πρόκειται για ένα περιβάλλον οπτικού προγραμματισμού JavaScript




Φωτογραφία: vvvv.js

## 15.4. VVVV.js

Οπως αναφέρεται στον ιστότοπο VVVVjs.com είναι ένα εργαλείο για prototyping και ανάπτυξη εφαρμογών συνδέοντας κόμβους. Το vvvvjs επιτρέπει τη δημιουργία 2D και 3D γραφικών χωρίς να χρειάζεται να γραφτεί κώδικας.

# ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Dongarra, J., & Sullivan, F. (2000, January 15). Guest Editors' Introduction: The Top 10 Algorithms. Retrieved January 2, 2015, from http://www.computer.org/csdl/mags/cs/2000/01/c1022.html

[2] Turing, A. (1950). COMPUTING MACHINERY AND INTELLIGENCE. In A QUARTERLY REVIEW OF PSYCHOLOGY AND PHILOSOPHY (236th ed., Vol. LIX, p. 433–460). MIND.

[3] Holland, J. (1975). Adaptation in Natural and Artificial Systems (p. 211). University of Michigan Press.

[4] Ingo Rechenberg (1971). Evolutionsstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (PhD thesis, http://tocs.ulb.tu-darmstadt.de/18420222.pdf)

[5] David E. Goldberg (1987). Computer-aided pipeline operation using genetic algorithms and rule learning. PART II: Rule learning control of a pipeline under normal and abnormal conditions. Engineering with Computers
Volume 3, Issue 1 , pp 47-58

[6] J.-L. Deneubourg, S. Aron, S. Goss, and J. M. Pasteels.1990. The self-organizing exploratory pattern of the Argentine ant. Journal of Insect Behavior, 3:159–168

[7] M. Dorigo, Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italy, 1992.

[8] G. Di Caro and M. Dorigo. AntNet: Distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research, 9:317–365, 1998.

[9] Nothegger, C., Mayer, A., Chwatal, A., & Raidl, G. (2010). Solving the post enrolment course timetabling problem by ant colony optimization. Annals of Operations Research, 325-339.

[10] Vatroslav Dino Matijaš, Goran Molnar, Marko Čupić, Domagoj Jakobović, Bojana Dalbelo Bašić. (2010). University Course Timetabling Using ACO: A Case Study on Laboratory Exercises. Knowledge-Based and Intelligent Information and Engineering Systems. Lecture Notes in Computer Science Volume 6276, pp 100-110

[11] Marinakis, Y., Marinaki, M., & Zopounidis, C. (2008). Application Of Ant Colony Optimization To Credit Risk Assessment. New Mathematics and Natural Computation, 107-107.

[12] Martens, D., Gestel, T., Backer, M., Haesen, R., Vanthienen, J., & Baesens, B. (2009). Credit rating prediction using Ant Colony Optimization. Journal of the Operational Research Society, 561-573.

[13] James Kennedy, Russell Eberhart (1995). Particle Swarm Optimization. Proceedings of IEEE International Conference on Neural Networks IV. pp. 1942–1948

[14] Beni, G., Wang, J. (June 26–30 1989). Swarm Intelligence in Cellular Robotic Systems, Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy,

[15] Ana Madureira, Nelson Sousa, Ivo Pereira (2011). SWARM INTELLIGENCE FOR SCHEDULING: A REVIEW. Business Sustainability II, SWARM INTELLIGENCE

[16] Ioannis X. Tassopoulos, Grigorios N. Beligiannis, (2012). Using particle swarm optimization to solve effectively the school timetabling problem, Soft Computing, 16(7), pp. 1229-1252

[17] Ioannis X. Tassopoulos, Grigorios N. Beligiannis. (2012),A hybrid particle swarm optimization based algorithm for high school timetabling problems, Applied Soft Computing, 12(11), pp. 3472-3489

[18] Ioannis X. Tassopoulos, Grigorios N. Beligiannis (2012), "Solving effectively the school timetabling problem using particle swarm optimization", Expert Systems with Applications, 39(5), pp. 6029-6040

[19] Lei Zhang, Yuehui Chen, Runyuan Sun, Shan Jing, Bo Yang (2008). A Task Scheduling Algorithm Based on PSO for
Grid Computing. International Journal of Computational Intelligence Research. ISSN 0973-1873 Vol.4, No.1 (2008), pp. 37–43

[20] Shu-Chuan Chu, Yi-Tin Chen, Jiun-Huei Ho. (2006) Timetable Scheduling Using Particle Swarm Optimization. Proceedings of the First International Conference on Innovative Computing, Information and Control (ICICIC'06)

[21] Pisut Pongchairerks(2009) Particle swarm optimization algorithm applied to scheduling problems. Thai Science 10.2306/scienceasia1513-1874.2009.35.089

[22] Jun Tang, (August 2010), Solving RFID Networks Scheduling Problems Using Hybird Binary Particle Swarm Optimization Algorithm. Applied Mechanics and Materials, 29-32, 966

[23] Yannis Marinakisa, Magdalene Marinaki, Georgios Dounias (2008). Particle swarm optimization for pap-smear diagnosis. Expert Systems with Applications Volume 35, Issue 4, November 2008, Pages 1645–1656

[24] Yang, Xin-She (2008). Nature-Inspired Metaheuristic Algorithms. Frome: Luniver Press. ISBN 1-905986-10-6.

[25] Iztok Fister, Xin-She Yang, Iztok Fister, Janez Brest, Dusan Fister (July 2013). A Brief Review of Nature-Inspired Algorithms for Optimization. ELEKTROTEHNISKI VESTNIK ˇ 80(3): 116–122, 2013

[26] Chris P. Duif (2004), A review of conventional explanations of anomalous observations during solar eclipses. arXiv:
grqc/0408023 v5

[27] Pejman Sanaei, Reza Akbari, Vahid Zeighami, Sheida Shams (December 2012). Using Firefly Algorithm to Solve Resource Constrained Project Scheduling Problem. Proceedings of Seventh International Conference on Bio-Inspired Computing: Theories and Applications (BIC-TA 2012) Advances in Intelligent Systems and Computing Volume 201, 2013, pp 417-428

[28] Xin-She Yang, Suash Deb, Simon Fong (2014) Bat Algorithm is Better Than Intermittent Search Strategy. arXiv:1408.5348 [math.OC]

[29] ABBASS HA (27–30 May 2001) Marriage in honey bees optimisation: A haplometrosis polygynous swarming approach. Proc. Congress on Evolutionary Computation, CEC2001, Seoul, Korea. 207–214.

[30] ABBASS HA (7–11 July 2001) A single queen single worker honey–bees approach to 3-SAT. Proc. Genetic and Evolutionary Computation Conference (GECCO2001), San Francisco. 807–814.

[31] ABBASS HA (9–11 July 2001) A pleometrosis MBO approach to satisfiability. Proc. International Conference on Computational Intelligence for Modeling, Control and Automation, CIMCA2001, Las Vegas, USA.

[32] The Nobel Prize in Physiology or Medicine 1973. (1973, January 1). Retrieved February 3, 2015, from http://www.nobelprize.org/nobel_prizes/medicine/laureates/1973/index.html

[33] D.T. Pham, A. Ghanbarzadeh, E. Koç, S. Otri , S. Rahim , M. Zaidi (2006) The Bees Algorithm – A Novel Tool for Complex Optimisation Problems

[34] A. Mirzakhani Nafchi, A. Moradi, A. Ghanbarzadeh, A. Rezazadeh, E.Soodmand (2011) Solving engineering optimization problems using the Bees Algorithm. Humanities, Science and Engineering (CHUSER), 2011 IEEE Colloquium on P 162 – 166

[35] Unknown author. Retrieved Jan 1, 2015. http://goo.gl/2R3z3B

[36] Yannis Marinakis, Magdalene Marinaki, Georgios Dounias (2008) Honey Bees Mating Optimization Algorithm for the Vehicle Routing Problem. Nature Inspired Cooperative Strategies for Optimization (NICSO 2007) Studies in Computational Intelligence Volume 129, pp 139-148

[37] Magdalene Marinakia, , Yannis Marinakisb, , Constantin Zopounidis (2010) Honey Bees Mating Optimization algorithm for financial classification problems. Applied Soft Computing, Volume 10, Issue 3, June 2010, Pages 806–812

[38] Babak Amiri, Mohammad Fathian (2007) INTEGRATION OF SELF ORGANIZING FEATURE MAPS AND HONEY BEE MATING OPTIMIZATION ALGORITHM FOR MARKET SEGMENTATION. Journal of Theoretical and Applied Information Technology

[39] Nasser R. Sabar, Masri Ayob, Graham Kendall (August 2009) Solving Examination Timetabling Problems using Honey-bee Mating Optimization (ETP-HBMO). Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009) Dublin, Ireland

[40] Nasser R. Sabara, Masri Ayoba, Graham Kendallb, Rong Qub (February 2012) A honey-bee mating optimization algorithm for educational timetabling problems. European Journal of Operational Research Volume 216, Issue 3, Pages 533–543

[41] OMID BOZORG HADDAD1, ABBAS AFSHAR, MIGUEL A. MARINO (2006) Honey-Bees Mating Optimization (HBMO) Algorithm: A New Heuristic Approach for Water Resources Optimization. Water Resources Management 20: 661–680 DOI: 10.1007/s11269-005-9001-3

[42] Yannis Marinakis, Magdalene Marinaki, Georgios Dounias (October 2011). Honey bees mating optimization algorithm for the Euclidean traveling salesman problem. Information Sciences: Volume 181, Issue 20, Pages 4684–4698. Special Issue on Interpretable Fuzzy Systems

[43] Muhammad Rozi Malim, Ahamad Tajudin Khader, Adli Mustafa (October 2006), Artificial immune algorithms for university timetabling. Proceedings of the 6th international conference on practice and theory of automated timetabling, Brno, Czech Republic, pp 234-245

[44] Muhammad Rozi Malim (March 2010), Adapting immune system based algorithms for class timetabling, Information Retrieval & Knowledge Management, (CAMP), 2010 International Conference, IEEE, pp 215-222

[45] MUHAMMAD ROZI MALIM, AHAMAD TAJUDIN KHADER, ADLI MUSTAFA (2006), An Immune-Based Approach to University Course Timetabling: Negative Selection Algorithm. the Proc. of the 2nd IMTGT Regional Conf. on Mathematics, Statistics and Applications, Univ. of Sains Malaysia, Penang pp 13-15

[46] Carlos A. Coello Coello, Daniel Cortés Rivera, Nareli Cruz Cortés (2003), Use of an Artificial Immune System for Job Shop Scheduling. Lecture Notes in Computer Science Volume 2787, pp 1-10

[47] M Afshari, H Sajedi (June 2012), A novel artificial immune algorithm for solving the job shop scheduling problem. International Journal of Computer Applications, International Journal of Computer Applications (0975 – 888). Volume 48– No.14

[48] Zohreh Davarzani, Mohammad-R Akbarzadeh-T, Nima Khairdoost (July 2012), Multiobjective Artificial Immune Algorithm for Flexible Job Shop Scheduling Problem, International Journal of Hybrid Information Technology Vol. 5, No. 3

[49] Carlos Coello-Coello and Nareli Cruz-Cortes, "Use of emulations of the immune system to handle constraints in

evolutionary algorithms", Intelligent Engineering Systems through Artificial Neural Networks (ANNIE'2001), St. Louis

Missouri, USA, November 2001.

[50] Gary Bernhardt (April 2014), The Birth & Death of JavaScript, PyCON 2014 Conference presentation video on https://www.destroyallsoftware.com/talks/the-birth-and-death-of-javascript

[51] TIOBE Software Index (February 2015), “February Headline: JavaScript at highest position ever”, retrieved from: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html

[52] Stephen O'Grady (2016, July 20). The RedMonk Programming Language Rankings: June 2016. Retrieved July 25, 2016, from http://redmonk.com/sograd y/2016/07/20/language-rankings-6-16/

[53] Alyson La (2015, Aug 19). Language Trends on GitHub. Retrieved June 16, 2016, from https://github.com/blog/2047-language-trends-on-github

[54] Garrett, J. (2005, February 18). Ajax: A New Approach to Web Applications | Adaptive Path. Retrieved March 6, 2015, from http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/

[55] DANGOOR, K. (2009, January 29). What Server Side JavaScript needs. Retrieved March 6, 2015, from http://www.blueskyonmars.com/2009/01/29/what-server-side-javascript-needs/

[56] Handy, A. (2011, June 23). Node.js pushes JavaScript to the server-side. Retrieved March 6, 2015, from http://sdtimes.com/node-js-pushes-javascript-to-the-server-side/

[57] Peacekeeper - free universal browser test for HTML5. (n.d.). Retrieved March 6, 2015, from http://peacekeeper.futuremark.com/

[58] Minto, Rob (27 March 2009). "The genius behind Google’s web browser". Financial Times. Retrieved March 6, 2015, from http://www.ft.com/cms/s/0/03775904-177c-11de-8c9d-0000779fd2ac.html

[59] Schenker, J. (2008, November 12). Google's Chrome: The Danish Magic Inside. Retrieved March 6, 2015, from http://www.bloomberg.com/bw/stories/2008-11-12/googles-chrome-the-danish-magic-insidebusinessweek-business-news-stock-market-and-financial-advice

[60] Brunner, G. (2013, March 28). Unreal Engine 3 ported to JavaScript and WebGL, works in any modern browser | ExtremeTech. Retrieved March 6, 2015, from http://www.extremetech.com/gaming/151900-unreal-engine-3-ported-to-javascript-and-webgl-works-in-any-modern-browser

[61] Cambus, F. (2014, June 12). Emulators written in JavaScript. Retrieved March 6, 2015, from http://www.cambus.net/emulators-written-in-javascript/

[62] List of languages that compile to JS. (2015, March 3). Retrieved March 11, 2015, from https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS

[63] Serdar, Y. (2014, June 26). 9 programming languages that make coding JavaScript a joy. Retrieved March 11, 2015, from http://www.infoworld.com/article/2606392/javascript/156855-X-languages-that-compile-to-JavaScript.html

[64] Turner, J. (2015, March 5). Angular 2: Built on TypeScript. Retrieved March 11, 2015, from http://blogs.msdn.com/b/typescript/archive/2015/03/05/angular-2-0-built-on-typescript.aspx

[65] Chesters J. (2015, March 2). JavaScript Frameworks in the Real World Retrieved March 11, 2015, from http://www.infoq.com/research/javascript-frameworks-2015

[66] Brendan Eich (2008, April 3). Popularity. Article from his personal blog retrieved from Google Cache https://brendaneich.com/2008/04/popularity in August 12, 2015

[67] Press Release (1995, December 5). NETSCAPE AND SUN ANNOUNCE JAVASCRIPT, THE OPEN, CROSS-PLATFORM OBJECT SCRIPTING LANGUAGE FOR ENTERPRISE NETWORKS AND THE INTERNET. Retrieved August 13, 2015 from https://web.archive.org/web/20070916144913/http://wp.netscape.com/newsref/pr/newsrelease67.html

[68] Hamilton, Naomi. The A-Z of Programming Languages: JavaScript. Computerworld. Computerworld, 31 July 2008. Web. 12 Aug. 2015. <http://www.computerworld.com.au/article/255293/a-z_programming_languages_javascript/>.

[69] Jesse James Garrett (2005, February 18) Ajax: A New Approach to Web Applications, from http://www.adaptivepath.com/ideas/ajax-new-approach-web-applications/ retrieved August 13, 2015

[70] Brendan Eich (2015, June 17). From ASM.JS to WebAssembly. Article from his personal blog retrieved in August 12, 2015

[71] ECMA International (October 2013) ECMA-404 Standard: The JSON Data Interchange Format. http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf retrieved August 13, 2015

[72] NETSCAPE INTRODUCES NETSCAPE ENTERPRISE SERVER(TM) 2.0. (n.d.) >The Free Library. (2014). Retrieved Aug 13 2015 from http://www.thefreelibrary.com/NETSCAPE+INTRODUCES+NETSCAPE+ENTERPRISE+SERVER(TM)+2.0-a018056425

[73] Kevin McCarthy (January 2011) Node.js Interview: 4 Questions with Creator Ryan Dahl. Retrieved Aug 13 2015 from http://bostinno.streetwise.co/2011/01/31/node-js-interview-4-questions-with-creator-ryan-dahl/

[74] Γιώργος Τσεκούρας, Νόελ Κουτλής (Μάρτιος 2014) ACORN BBC, Μια μοναδική ιστορία επιτυχίας που ξεκίνησε από το σχολείο και εξελίχθηκε σε έναν από τους πιο επιτυχημένους μικροεπεξεργαστές. Πρακτικά συνεδρίου ΠΕΚΑΠ 2014 http://pekap.tsopokis.gr/synedrio/praktika/2014/ergasies/2Tsekouras2-full_Corrected.pdf[75] Matt Brian (July 7, 2015). How the BBC's Micro:bit came to be. Engadget, retrieved Aug 30 2015 from http://www.engadget.com/2015/07/07/bbc-micro-bit-explained/

[76] BBC Make it Digital (2015). Introducing the BBC micro:bit. Retrieved Aug 30 2015 from BBC MicroBit official site http://www.bbc.co.uk/programmes/articles/4hVG2Br1W1LKCmw8nSm9WnQ/introducing-the-bbc-micro-bit

[77] Dingman, H. (2015, January 6). Internet Archive brings over 2000 free classic MS-DOS games to your browser. Retrieved November 19, 2015, from http://www.pcworld.com/article/2865916/internet-archive-brings-over-2000-free-classic-ms-dos-games-to-your-browser.html

[78] Johansson, M. (2015, June 1). How is Javascript used within the Spotify desktop application? Retrieved November 19, 2015, from https://www.quora.com/How-is-Javascript-used-within-the-Spotify-desktop-application

[79] Bevacqua, N. (2015, November 29). JavaScript Developer Survey Results. Retrieved December 16, 2015, from https://ponyfoo.com/articles/javascript-developer-survey-results

[80] Eastcott, W., & Robert Nyman, R. (2014, June 4). PlayCanvas Goes Open Source. Retrieved January 10, 2016, from https://hacks.mozilla.org/2014/06/playcanvas-goes-open-source/

[81] Granja, D. I., & Ruiz, J. J. (1986). An Interview with George B. Dantzig: The Father of Linear Programming - The College Mathematical Journal. Retrieved January 3, 2016, from http://www.phpsimplex.com/en/Dantzig_interview.htm

[82] George B. Dantzig (1963). . Linear programming and extensions. Princeton University Press and the RAND Corporation.

[83] Dunning, I. (2013, September 8). SimplexJS. Retrieved January 3, 2016, from http://iaindunning.com/ blog/simplexjs.html

[84] Seth Thompson (2016, March 15) Experimental support for WebAssembly in V8, Retrieved March 19, 2016, from http://v8project.blogspot.gr/2016/03/experimental-support-for-webassembly.html

[85]StackOverflow (2016, March 17) Developer Survey Results 2016, Retrieved March 19, 2016, from http://stackoverflow.com/research/developer-survey-2016

[86] John Gossman (2005, October 8) Introduction to Model/View/ViewModel pattern for building WPF apps, Retrieved March 19, 2016, from https://blogs.msdn.microsoft.com/johngossman/2005/10/08/introduction-to-modelviewviewmodel-pattern-for-building-wpf-apps/

[87] Abel, A. (2015, August 5). JerryScript & IoT.js: JavaScript for IoT from Samsung. Retrieved February 23, 2016, from http://www.infoq.com/news/2015/08/iotjs-jerryscript-samsung

[88] Krill, P. (2015, July 29). Samsung banks on JavaScript, Node.js for IoT. Retrieved February 23, 2016, from http://www.infoworld.com/article/2953719/javascript/samsung-banks-on-javascript-node-js-for-iot.html

[89] Klint, F. (2015, November 23). WordPress.com Gets a New Face and Joins the JavaScript Age. Retrieved February 23, 2016, from http://www.wired.com/2015/11/wordpress-com-gets-a-new-face-and-joins-the-javascript-age/

[90] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, Martin Riedmiller.(2013, December 19). Playing Atari with Deep Reinforcement Learning" Retrieved from Cornell University Library May 24, 2016 http://arxiv.org/pdf/1312.5602v1.pdf

[91] Harrell, J. (2013, November 22). Node.js at PayPal. Retrieved May 26, 2016, from https://www.paypal-engineering.com/2013/11/22/node-js-at-paypal/

[92] DELL, J. (2011, August 16). Exclusive: How LinkedIn used Node.js and HTML5 to build a better, faster app. Retrieved May 26, 2016, from http://venturebeat.com/2011/08/16/linkedin-node/

[93] Krill, P. (2015, December 11). Node.js, Google Go drive Uber. Retrieved May 26, 2016, from http://www.infoworld.com/article/3014207/javascript/nodejs-google-go-drive-uber.html

[94] Ecma International (June 14, 2016). ECMAScript® 2016 Language Specification. Retrieved June 25, 2016, from https://tc39.github.io/ecma262/

[95] Mattias Petter Johansson (2015, April 12). How is Javascript used within the Spotify desktop application? Retrieved June 26, 2016, from https://www.quora.com/How-is-Javascript-used-within-the-Spotify-desktop-application

[96] Nick Larsen (2016, June 24). Build Your First Thing With WebAssembly. Retrieved June 26, 2016, from http://cultureofdevelopment.com/blog/build-your-first-thing-with-web-assembly/

[97] Rob Eisenberg (2015, January 26). Introducing Aurelia. Retrieved July 26, 2016, from http://blog.durandal.io/2015/01/26/introducing-aurelia/

[98] Marcello Fera, Fabio Fruggiero, Alfredo Lambiase, Giada Martino and Maria Elena Nenni (2013, March). Production Scheduling Approaches for Operations Management, Operations Management, Marcello Fera, Fabio Fruggiero, Alfredo Lambiase, Giada Martino and Maria Elena Nenni (2013). Production Scheduling Approaches for Operations Management, Operations Management, Prof. Massimiliano Schiraldi (Ed.), InTech, DOI: 10.5772/55431. Available from: http://www.intechopen.com/books/operations-management/production-scheduling-approaches-for-operations-management, InTech, DOI: 10.5772/55431. Retrieved July 26, 2016, from: http://www.intechopen.com/books/operations-management/production-scheduling-approaches-for-operations-management

[99] Yudong Zhang,Shuihua Wang,, and Genlin Ji (2015, February 12) A Comprehensive Survey on Particle Swarm Optimization Algorithm and Its Applications. Mathematical Problems in Engineering Volume 2015, Article ID 931256 Retrieved July 26, 2016 from http://www.hindawi.com/journals/mpe/2015/931256/

[100] Rust, S., Schelling, J., & Schipper, D. (2015, Jun 8) Building Real-Time Web Applications with Meteor. Universiteit Leiden, Retrieved July 26, 2016 from http://mediatechnology.leiden.edu/images/uploads/ docs/wt2015_meteor.pdf

[101] Evan Schwartz. 2016. A Payment Protocol of the Web, for the Web: Or, Finally Enabling Web Micropayments with the Interledger Protocol. In Proceedings of the 25th International Conference Companion on World Wide Web (WWW '16 Companion). International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 279-280. DOI: http://dx.doi.org/10.1145/2872518.2889305

[102] Olanoff, D. (2012). Mark Zuckerberg: Our Biggest Mistake Was Betting Too Much On HTML5. Retrieved October 18, 2016, from https://techcrunch.com/2012/09/11/mark-zuckerberg-our-biggest-mistake-with-mobile-was-betting-too-much-on-html5/

[103] Gary Wolf (June 1995). The Curse of Xanadu, Wired Magazine. Retrieved November 23, 2016 from https://www.wired.com/1995/06/xanadu/

[104] James Gillies; R. Cailliau (2000). How the Web was born: the story of the World Wide Web. Oxford University Press. pp. 213–217. ISBN 978-0-19-286207-5

[105] Dharma Shukla, Shireesh Thota, Karthik Raman, Madhan Gajendran, Ankur Shah, Sergii Ziuzin, Krishnan Sundaram, Miguel Gonzalez Guajardo, Anna Wawrzyniak, Samer Boshra, Renato Ferreira, Mohamed Nassar, Michael Koltachev, Ji Huang, Sudipta Sengupta, Justin Levandoski, and David Lomet. 2015. Schema-agnostic indexing with Azure DocumentDB. Proc. VLDB Endow. 8, 12 (August 2015), 1668-1679.

[106] Katz, D. (2012, January 4). The Future of CouchDB. Retrieved December 16, 2016, from http://damienkatz.net/ 2012/01/the_future_of_couchdb.htmlv