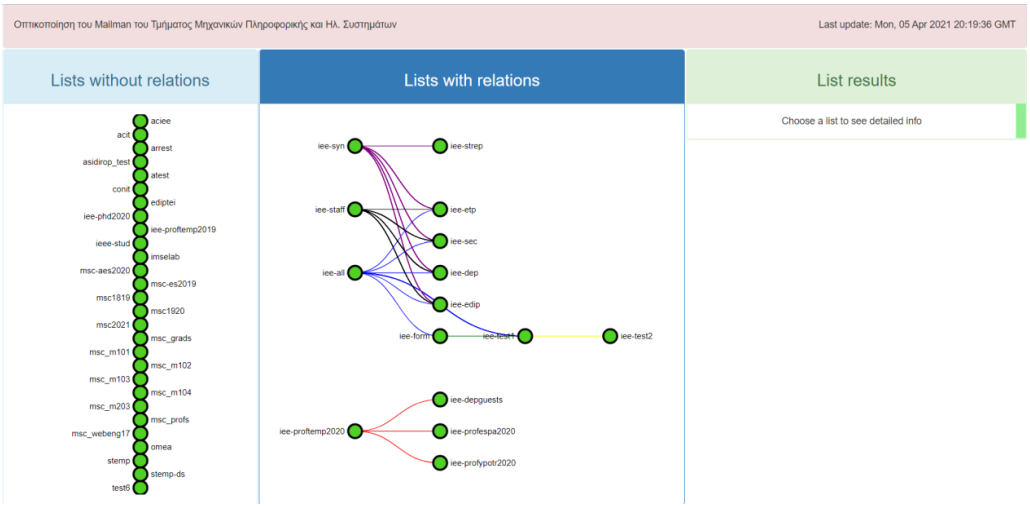
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Οπτικοποίηση ιεραρχικής δομής λιστών ηλεκτρονικού ταχυδρομείου»



Του φοιτητή
Παντελή Διαμαντίδη
Αρ. Μητρώου: 174934

Επιβλέπων
Σιδηρόπουλος Αντώνης
Επίκουρος Καθηγητής

Ημερομηνία 15/06/2021

Τίτλος Δ.Ε. Οπτικοποίηση ιεραρχικής δομής λιστών ηλεκτρονικού ταχυδρομείου
Κωδικός Δ.Ε. 20193
Ονοματεπώνυμο φοιτητή Παντελής Διαμαντίδης
Ονοματεπώνυμο εισηγητή Σιδηρόπουλος Αντώνης
Ημερομηνία ανάληψης Δ.Ε. 16/10/2020
Ημερομηνία περάτωσης Δ.Ε. 15/06/2021

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Πρόλογος

Η πτυχιακή αυτή εργασία επιλέχθηκε για το ενδιαφέρον του σκοπού της. Οι λίστες ηλ. ταχυδρομείου αποτελούν ένα πολύ ενδιαφέρον αντικείμενο, ειδικότερα όταν πρόκειται για την οπτικοποίηση τους, η οποία ήταν μία μοναδική εμπειρία για εμένα ως φοιτητή.

Περίληψη

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η δημιουργία μίας web εφαρμογής για τη γραφική απεικόνιση των mailing lists ενός mailman server, χρησιμοποιώντας τις δυνατότητες visualization της HTML5. Συγκεκριμένα, να ανακτηθούν οι mailing lists του mailman της σχολής μας, μέσω ενός API, να κατηγοριοποιηθούν σε 2 ομάδες (λίστες με ιεραρχικές σχέσεις – λίστες χωρίς ιεραρχικές σχέσεις) και να παρουσιαστούν γραφικά οι ομάδες που δημιουργήθηκαν, καθώς και οι ιεραρχικές σχέσεις της πρώτης, σε μορφή δέντρου. Επίσης, μέσω της εφαρμογής, ο χρήστης θα μπορεί να ανακτήσει τα mails των μελών της κάθε λίστας, επιλέγοντας τη. Στο 1ο κεφάλαιο, θα αναλυθεί η HTML5 και οι νέες δυνατότητες που προσφέρει, δίνοντας έμφαση σε αυτές που έχουν να κάνουν με το visualization. Στο 2ο κεφάλαιο, θα αναλυθεί η βιβλιοθήκη D3.js, που χρησιμοποιήθηκε για τη γραφική απεικόνηση των λιστών και θα επεξηγηθεί το πως αυτή εκμεταλλεύται το εργαλείο SVG της HTML5 για τη δημιουργία διαδραστικών δικτύων απο στοιχεία, με δενδρική και άλλες δομές. Στο 3ο κεφάλαιο θα γίνει μία εκτενής αναφορά στο GNU mailman και στο πως αυτό λειτουργεί, καθώς είναι απαραίτητη η κατανόηση του εργαλείου αυτού για την υλοποίηση της πτυχιακής εργασίας. Τελικά, στο 4ο κεφάλαιο θα πραγματοποιηθεί η περιγραφή του πρακτικού μέρους της εργασίας.

«Hierarchical structure visualization of mailing lists»

«Pantelis Diamantidis»

Abstract

The goal of this thesis is the creation of a web application for the graphic illustration of the mailing lists from a mailman server, using the visualization features of HTML5. Specifically, to retrieve the mailing lists from the mailman server of our faculty, through an API, to divide them into 2 groups (lists with hierarchical relations – lists without hierarchical relations) and to create a graphic illustration of the groups and the relations of the first, in a tree form. Also, the user will be able to retrieve the mails of the members of each list, by clicking on it. In the 1st chapter, there will be a review of HTML5 and its new possibilities, focusing on the ones that have to do with visualization. The 2nd chapter will make an analyzation of the D3 library, which is used for the graphic illustration of the lists, by using the SVG tool of HTML5 for the creation of interactive networks of elements, with tree or other forms. In the 3rd chapter there will be an extensive description of GNU mailman and on how does it work, because the understanding of that tool is necessary for the implementation of this thesis. Finally, the 4th chapter will contain in detail the practical implementation, which is the goal of this thesis.

Περιεχόμενα

Κατάλογος Σχημάτων

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
Π.Ε.	Πτυχιακή Εργασία

Κεφάλαιο 1ο: Η HTML5

1.1 Εισαγωγή

Αυτό το κεφάλαιο θα ασχοληθεί με την HTML5. Αρχικά, θα γίνει μία παρουσίαση της. Στη συνέχεια, θα γίνει ανάλυση των πλεονεκτημάτων της σε σχέση με τις προηγούμενες εκδόσεις της HTML. Στο τέλος θα αναλυθούν κάποια αξιοσημείωτα νέα χαρακτηριστικά και δυνατότητες που έφερε η HTML5.

1.2 Η δημιουργία της HTML5

Η δημιουργία της HTML5 αποτελεί ένα τεράστιο σημείο καμπής στην ανάπτυξη των web προτύπων. Η προέλευση αυτής της καινοτομίας πηγάζει απο μία ομάδα επαγγελματιών ιδιωτικών εταιρειών που δεν ήταν ικανοποιημένοι με τη πορεία του οργανισμού ο οποίος εποπτεύει την ανάπτυξη αυτών των προτύπων, τη W3C. Τον Ιούνιο του 2004, σε ένα σεμινάριο της W3C για τις web εφαρμογές και τα σύνθετα έγγραφα, αρκετοί προγραμματιστές της Opera και της Mozilla (και στη συνέχεια και κάποιοι απο την Apple), σχημάτισαν την ομάδα εργασίας «Web Hypertext Application Technology Working Group (WHATWG)», η οποία ήταν ανεξάρτητη απο την W3C. Στην εκδήλωση, προγραμματιστές της Opera και της Mozilla παρουσίασαν το δικό τους όραμα για το μέλλον του παγκόσμιου ιστού, το οποίο μπορεί να συνοψιστεί ως εξής: “Να μετασχηματίσουμε την HTML4 σε ένα πρότυπο ικανό να υποστηρίξει νέα χαρακτηριστικά για τους σύγχρονους προγραμματιστές web εφαρμογών”. Τόνισαν επίσης, τη σημαντικότητα 7 αρχών που περιείχε η πρόταση τους. Αυτές ήταν οι παρακάτω [1]:

- **Συμβατότητα προς τα πίσω (Backwards compatibility):**
  Οι τεχνολογίες web εφαρμογών πρέπει να βασίζονται σε τεχνολογίες με τις οποίες οι συγγραφείς είναι οικείοι, συμπεριλαμβανομένων των HTML, CSS, DOM και JavaScript. Οι βασικές δυνατότητες των web εφαμοργών πρέπει να είναι εφαρμόσιμες χρησιμοποιώντας τις ήδη υπάρχον συμπεριφορές και τις ήδη υπάρχον δυνατότητες scripting και style sheets. Οποιαδήποτε λύση δε μπορεί να ανταποκριθεί στο μεγαλύτερο ποσοστό των browsers στην αγορά χωρίς τη χρήση δυαδικών plugin, πιθανότατα θα αποτύχει.

- **Καλά καθορισμένος χειρισμός σφαλμάτων:**
  Ο χειρισμός σφαλμάτων πρέπει να οριστεί σε ένα επίπεδο λεπτομέρειας όπου οι browsers δε θα χρειάζεται να κατασκευάζουν τους δικούς τους μηχανισμούς διαχείρισης σφαλμάτων.

- **Οι χρήστες δε πρέπει να εκτίθενται σε σφάλματα συγγραφής:**
  Οι προδιαγραφές πρέπει να καθορίζουν ακριβείς συμπεριφορές αποκατάστασης για κάθε πιθανό σενάριο σφάλματος. Ο χειρισμός σφαλμάτων πρέπει στο μεγαλύτερο μέρος του να αποτελείται απο πιο διακριτικές μεθόδους αποκατάστασης (όπως στη CSS), παρά με προφανείς και καταστροφικές αποτυχίες (όπως στη XML).

- **Πρακτική χρήση:**
  Κάθε χαρακτηριστικό που προστίθεται στις προδιαγραφές των web εφαρμογών θα πρέπει να αιτιολογείται απο μία πρακτική περίπτωση χρήσης. Το αντίστροφο δεν είναι απαραίτητα αλήθεια: Κάθε περίπτωση χρήσης δεν εγγυάται ένα νέο χαρακτηριστικό. Οι περιπτώσεις χρήσης θα πρέπει κατά προτίμηση να βασίζονται σε πραγματικές ιστοσελίδες όπου οι συγγραφείς προηγουμένως χρησιμοποιούσαν μία φτωχή λύση για να επιλύσουν τον περιορισμό.

- **Το scripting είναι εδώ για να μείνει:**
  Ωστόσο, θα πρέπει να αποφεύγεται όταν μπορεί να χρησιμοποιηθεί μία πιο βολική δηλωτική σήμανση. Το scripting θα πρέπει να παραμένει ίδιο σε κάθε συσκευή και εμφάνιση, εκτός αν ο σκοπός είναι μία στοχευμένη λύση για μία συγκεκριμένη συσκευή.

- **Η δημιουργία διαφορετικών προφίλ ανάλογα με τη συσκευή πρέπει να αποφεύγεται:**
  Οι συγγραφείς θα πρέπει να μπορούν να συμπεριλάβουν και να βασιστούν στα ίδια χαρακτηριστικά, είτε σε έκδοση desktop είτε σε mobile, για την ίδια εφαρμογή χρήστη.

- **Ανοιχτή διαδικασία:**
  Ο παγκόσμιος ιστός έχει επωφεληθεί απο την ανάπτυξη του σε ένα ανοιχτό περιβάλλον. Οι web εφαμοργές θα είναι ο πυρήνας του παγκόσμιου ιστού, και η ανάπτυξη του θα πρέπει να γίνεται ανοιχτά. Λίστες ηλ. ταχυδρομείου, αρχεία και προσχέδια προδιαγραφών, θα πρέπει να είναι συνεχώς ορατά και προσβάσιμα στο κοινό.

Η W3C, ωστόσο, έκανε μία επίσημη δήλωση με την οποία διαβεβαίωνε ότι δε θα προσφερθεί βοήθεια ή πόροι σε οποιαδήποτε πρόταση που δε προερχόταν απο τις ήδη υπάρχουσες ομάδες εργασίας. Λόγω αυτής της ηχηρής απόρριψης, η ομάδα αποφάσισε να συνεχίσει τη δουλειά της εκτός της W3C. Καταχώρησαν το domain με όνομα ‘whatwg.org’, στο οποίο ξεκίνησαν να δουλεύουν πάνω στις ιδέες τους. Αυτή η ανεπίσημη ομάδα εργασίας, ανοιχτή σε τρίτους, αποτελούνταν απο προμηθευτές browser και διάφορα ενδιαφερόμενα μέλη που ήθελαν να γυρίσουν στις ρίζες της HTML, παρά να αναπτύξουν άλλες γλώσσες, όπως η XHTML. Αυτή η προσέγγιση βασιζόταν στην εξασφάλιση backwards compatibility στο νέο πρότυπο, ένα χαρακτηριστικό που εκτιμήθηκε ιδιαίτερα απο τους web developers, και το οποίο είναι ένα προσόν που αναμφίβολα συνέβαλε στη συνοχή των μελλοντικών τεχνολογικών αναπτύξεων.

Αντιθέτως, η W3C ήθελε να συνεχίσει την ανάπτυξη της XHTML2. Η γλώσσα αυτή δεν προσέφερε backwards compatibility με την HTML και απαιτούσε ένα διαφορετικό «MIME type» (ο τύπος αρχείου που διευκρινίζεται στην αρχή κάθε ιστοσελίδας). Αυτή η επιλογή συνεπαγόταν την απόρριψη όλων των προηγούμενων επιτευγμάτων στην HTML, επειδή οι browsers πάντα παρέβλεπαν τα συντακτικά λάθη στην HTML, και μέχρι τότε κανείς δεν είχε σκεφτεί το πως να τα καθορίσει. Μπορούμε να πούμε ότι μέχρι εκείνο το σημείο, το web development βασίστηκε σε μεγάλο βαθμό σε διάφορους προγραμματιστές browser οι οποίοι έκαναν τα προϊόντα τους συμβατά με αυτά των ανταγωνιστών τους, αγνοόντας προδιαγραφές και πρότυπα. Οι περισσότεροι browser επικεντρώθηκαν στο να είναι σε θέση να παρουσιάσουν τις ομάδες των ετικετών που παράγονταν, με τον καλύτερο τρόπο. Η ομάδα WHATWG αφιέρωσε 5 χρόνια δουλειάς για να τεκμηριώσει το πως αναλύεται και τεμαχίζεται η HTML κατάλληλα με ένα τρόπο συνεπή με τις υπάρχουσες ιστοσελίδες.

Ένας απο τους ειδικούς, ο οποίος ήταν ιδρυτικό μέλος της ομάδας WHATWG, εξήγησε σε συνέντευξη του τα αίτια αυτής της κατάστασης. Συγκεκριμένα, είπε: “Ανησυχούσαμε για την XHTML2 και δε πιστεύαμε ότι η XHTML ήταν το μέλλον, επομένως, τα χαρακτηριστικά το οποία μας ενδιέφεραν ήταν όσα θα επέκτειναν τον Παγκόσμιο Ιστό χωρίς να ‘σπάσουν’ το backwards compatibility. Όλα τα υπόλοιπα χαρακτηριστικά, όπως τα web sockets, ήταν πρόσθετα. Αλλά ο κύριος στόχος ήταν δύο πράγματα: η επέκταση του Παγκόσμιου Ιστού χωρίς το ‘σπάσιμο’ του backwards compatibility, και η διαλειτουργικότητα. Οπότε, ένα απο τα κύρια χαρακτηριστικά της HTML5 το οποίο δε μιλάει τόσο ο κόσμος, επειδή δεν είναι ελκυστικό, είναι αυτό που αποκαλέσαμε ‘αλγόριθμος parsing’. Ο αλγόριθμος parsing στην HTML5, φροντίζει ότι κάθε ιστοσελίδα θα φτιάξει το ίδιο DOM σε κάθε browser.” (Bruce Lawson, Opera).

Πέρα απο αυτό το εκπληκτικό έργο, η ομάδα δούλεψε επίσης στην εγγενής υποστήριξη (χωρίς plugins) ήχου και βίντεο, στο στοιχείο ‘canvas’ (το οποίο χρησιμοποιείται για τη σχεδίαση γραφικών) και άλλες προδιαγραφές web εφαρμογών. Ο λόγος για την εγκαθίδρυση της ομάδας ήταν η διαφωνία με το ακαδημαϊκό όραμα της W3C για τα πρότυπα και ο στόχος να προσφερθεί μία πολύ πιο πρακτική εστίαση στο νέο πρότυπο.

Ωστόσο, η W3C φαινόταν να αναζητά αντικαταστάτη για την HTML, ανάμεσα σε πολλές τεχνολογίες και ιδιαίτερα την XHTML2. Η αλήθεια είναι ότι, 2 χρόνια μετά το συνέδριο, η XHTML2 αδυνατούσε να σημειώσει πρόοδο, ενώ τα νέα χαρακτηριστικά της HTML έθεσαν υψηλές προσδοκίες. Αυτό το γεγονός, σε συνδυασμό με τη επίκριση προς τη W3C για την αργή της πρόοδο και την απώλεια συγκεκριμένων αποτελεσμάτων, ήταν κάποιοι απο τους λόγους που επισπεύσαν την ανακοίνωση του Tim Berners-Lee και της W3C τον Οκτώβριο του 2006. Απο εκείνη τη μέρα θα συνεργάζονταν μαζί με την ομάδα WHATWG για να προσθέσουν νέα χαρακτηριστικά και να προωθήσουν την ανάπτυξη της HTML.

Ακολούθως, τον Οκτώβριο του 2009, η W3C διέλυσε την ομάδα εργασίας της XHTML2 και εγκατέλειψε όλες τις δραστηριότητες της στη συγκεκριμένη γλώσσα, με σκοπό να επικεντρωθεί πλήρως στην ανάπτυξη της HTML5. Επίσης, το 2008, αυτή η κοινοπραξία δημοσίευσε ένα προσχέδιο του προτύπου, χάρη στις κοινές προσπάθειες των δύο οργανισμών. Στα επόμενα χρόνια οι browsers άρχισαν να υποστηρίζουν την HTML5, με τον Mozilla Firefox να είναι ο πρωτοπόρος browser σε αυτό, και ξεκίνησε μία φάση αύξησης της επίγνωσης σχετικά με τη συγκεκριμένη τεχνολογία.

Ωστόσο, αυτό που πέτυχε το μεγαλύτερο αντίκτυπο όσον αφορά την κοινωνική διάδοση, ήταν μία επιστολή του Steve Jobs με όνομα “Thoughts on Flash”. Η δημοσίευση του επισήμανε τη προτεραιότητα που θα έπρεπε να δοθεί στις κινητές συσκευές κατά την ανάπτυξη προτύπων, και τα προβλήματα που δημιουργούνται απο ιδιόκτητα λογισμικά, όπως το Flash. Επίσης, παίνεψε τις HTML5, CSS και JavaScript για τα χαρακτηριστικά τους και την ανοιχτού προτύπου φύση τους. Πολλοί ειδικοί που παρείχαν συνέντευξη, αναγνώρισαν τη σημαντικότητα των δηλώσεων αυτών απο τον διάσημο, πρώην διευθύνων σύμβουλο της Apple.

Άλλες ισχυρές εταιρείες και πλατφόρμες, όπως οι YouTube, SlideShare και Scribd, επίσης δήλωσαν τη δημόσια υποστήριξη τους προς την HTML5, καθώς και τη πρόθεση τους να εφαρμόσουν αυτή τη τεχνολογία στις εφαρμογές τους όσο το δυνατόν περισσότερο. Απο τότε, η ανάπτυξη της HTML5 συνεχίσηκε σταδιακά, πρώτα ως υποψήφια σύσταση της W3C (το 2011), και αργότερα ως επίσημη σύσταση της, στις 28 Οκτωβρίου του 2014 [1].

1.3 Τα βασικά της HTML5

Η HTML είχε πάντα να κάνει με τη ενδοσύνδεση. Τη δεκαετία του ’90, ο παγκόσμιος ιστός ήταν γεμάτος με αρχεία που περιείχαν πολυσέλιδα έγγραφα. Διάβαζες αυτά τα τεράστια έγγραφα κάνοντας ‘scroll’, όπως θα έκανες με μία εγκυκλοπαίδεια. Πολλές από τις πρώτες εκδόσεις της HTML είχαν σκοπό να αντιμετωπίσουν αυτό το πρόβλημα, το οποίο είχε ευρέως θεωρηθεί ως επιζήμιο για την αναγνωσιμότητα και τη χρηστικότητα του διαδικτύου. Θεμελιώδης, λοιπόν, για την HTML ήταν η ετικέτα «a», η οποία έδινε στο έγγραφο τη δυνατότητα για τη σύνδεση με άλλο έγγραφο. Έτσι, κάθε πολυσέλιδο έγγραφο μετατράπηκε σε πολλά μονοσέλιδα έγγραφα, διασυνδεδεμένα μεταξύ τους.

Επιστρέφοντας στο παρόν, αυτή η δυνατότητα διασύνδεσης πραγμάτων στο διαδίκτυο, παραμένει η κύρια δυνατότητα της HTML5. Απλώς τώρα, αρχίζουμε να συνειδητοποιούμε το πρωτότυπο όραμα της HTML, και να διασυνδέουμε πολλά περισσότερα από αρχεία υπερκειμένου με στατικές εικόνες. Οπότε, νέες προσθήκες στοιχείων, όπως το «audio» και το «video», δεν είναι τίποτα παραπάνω από λογικές

επεκτάσεις παλαιών στοιχείων, όπως το «a». Πλέον, μπορεί να γίνει εισαγωγή εικόνων, ήχου και βίντεο, κατευθείαν στο έγγραφο. Πιο σημαντικό γεγονός είναι ότι μπορεί να γίνει εύκολη διαχείρηση τους απο τη JavaScript. Αυτό είναι που πρέπει να κρατήσουμε ως το πιο κρίσιμο πλεονέκτημα της HTML5 επι των προηγούμενων εκδόσεων της: τη δυνατότητα εισαγωγής 'πραγμάτων' κατευθείαν στο έγγραφο, και της εύκολης χειραγώγησης τους από τη JavaScript. Δηλαδή, τη δυνατότητα σύνδεσης κομματιών απο διαφορετικά μέρη, στο ίδιο έγγραφο, με ένα ουσιαστικό τρόπο, με σημασιολογικά στοιχεία που κάνουν τα κομμάτια αυτά προσβάσιμα στη JavaScript [2].

Κάποιες βασικές πληροφορίες που πρέπει να ξέρουμε για την HTML5, θα αναλυθούν στα επόμενα υποκεφάλαια.

1.3.1 Δεν είναι ένα ενιαίο πράγμα

Μπορεί, δικαίως, να αναρωτιέστε, εάν οι παλαιότεροι browser υποστηρίζουν την HTML5, ωστέ να ξέρετε εάν τα web pages που θα δημιουργήσετε με αυτή θα είναι προσβάσιμα στον καθένα. Αυτή η ερώτηση από μόνη της είναι παραπλανητική, καθώς η HTML5 δεν είναι ένα ενιαίο πράγμα, όπως μία βιβλιοθήκη JavaScript. Αντιθέτως, είναι μία συλλογή από ξεχωριστές δυνατότητες. Οπότε, δε μπορείτε να εντοπίσετε εάν ένας browser υποστηρίζει την HTML5. Ωστόσο, μπορείτε να εντοπίσετε εάν αυτός υποστηρίζει συγκεκριμένες δυνατότητες της, όπως τις «canvas» ή «video».

Ίσως φαντάζεστε την HTML ως ετικέτες μέσα σε αγκύλες, αλλά δεν είναι μόνο αυτό. Η HTML5 ορίζει επίσης πως αυτές οι ετικέτες αλληλοεπιδρούνε με τη JavaScript, μέσω του Document Object Model (DOM). Για παράδειγμα, η HTML5 δεν ορίζει απλά ένα στοιχείο <video>, αλλά υπάρχει ένα αντίστοιχο API στο DOM για τα στοιχεία <video>. Μπορεί να γίνει χρήση αυτού του API για να εντοπιστούν οι υποστηριζόμενες μορφές βίντεο, για να γίνει αναπαραγωγή ενός βίντεο, για να γίνει παύση ή σίγαση του, για να ενημερωθούμε για το τι ποσοστό του βίντεο έχει κατέβει, και οτιδήποτε άλλο χρειάζεται για να δημιουργηθεί μία πλούσια εμπειρία χρήστη γύρω από την ετικέτα <video> [3].

1.3.2 Δε χρειάζεται να 'πετάξετε' τίποτα

Η HTML4 είναι η πιο επιτυχημένη γλώσσα σήμανσης και η HTML5 έχει χτιστεί επάνω σε αυτή την επιτυχία. Δε χρειάζεται να αλλάξετε τις ετικέτες ή οτιδήποτε άλλο στις web εφαρμογές σας χτισμένες σε HTML4. Ούτε χρειάζεται να μάθετε ξανά τα όσα ξέρετε για την HTML. Εάν η εφαρμογή σας δούλευε χθες σε HTML4, θα δουλεύει και σήμερα σε HTML5.

Βέβαια, αν θέλετε να βελτιώσετε τις web εφαρμογές που χτίσατε σε HTML4, τότε η HTML5 είναι απολύτως κατάλληλη για αυτό. Για παράδειγμα, η HTML5 υποστηρίζει πλήρως τα στοιχεία <form> της HTML4, αλλά περιλαμβάνει επίσης νέες επιλογές για την εισαγωγή δεδομένων. Κάποιες πιο προχωρημένες, όπως «date picker» και «slider», και άλλες πιο διακριτικές, όπως την επιλογή «email», η οποία απλώς αλλάζει το πληκτρολόγιο του χρήστη (στις κινητές συσκευές) σε ένα πιο βολικό για την εισαγωγή email πληκτρολόγιο. Οι παλαιότεροι browser που δεν υποστηρίζουν αυτή τη δυνατότητα, θα εμφανίσουν απλά το κανονικό πληκτρολόγιο, με τη λειτουργικότητα να παραμένει σωστή. Αυτό σημαίνει ότι μπορείτε να αρχίσετε να αναβαθμίζετε τις HTML4 εφαρμογές σας σε HTML5 άφοβα, ακόμη κι αν κάποιοι χρήστες χρησιμοποιούν παλαιότερους browser [3].

1.3.3 Είναι πολύ εύκολο να ξεκινήσετε.

Η 'αναβάθμιση' σε HTML5 είναι απλώς μία αλλαγή στο «doctype». Το doctype πρέπει να είναι η πρώτη γραμμή κώδικα σε κάθε HTML σελίδα. Οι προηγούμενες εκδόσεις της HTML ορίζονταν με πολλά διαφορετικά doctypes και η επιλογή του σωστού μπορούσε να γίνει δύσκολη. Στην HTML5, υπάρχει

μόνο ένα doctype, το οποίο είναι αυτό: "<!DOCTYPE html>". Η 'αναβάθμιση' σε HTML5 δε θα επηρεάσει τον κώδικα που έχετε γράψει ήδη, επειδή η λειτουργικότητα και οι ετικέτες της HTML4 υποστηρίζονται από την HTML5. Αλλά, θα σας δώσει την επιλογή να χρησιμοποιήσετε όλες τις νέες δυνατότητες και ετικέτες που προσφέρει η HTML5, όπως τα στοιχεία <article>, <section>, <header>, <footer> και άλλα [3].

1.3.4 Λειτουργεί ήδη

Είτε θέλετε να 'ζωγραφίσετε' σε ένα στοιχείο «canvas», είτε θέλετε να αναπαράγετε βίντεο, είτε να δημιουργήσετε καλύτερα «forms», ή ακόμη κι αν θέλετε να προβείτε στην υλοποίηση web applications που λειτουργούν εκτός σύνδεσης, θα ανακαλύψετε ότι η HTML5 είναι καλά υποστηριζόμενη. Οι Firefox, Chrome, Safari, Opera αλλά και οι περισσότεροι browser στις κινητές συσκευές, ήδη υποστηρίζουν τις δυνατότητες «canvas», «video», «geolocation», «local storage» και πολλές ακόμα. Επίσης, η Google υποστηρίζει ήδη τα «microdata», τα οποία είναι μεταδεδομένα εμφωλευμένα σε ήδη υπάρχοντα στοιχεία σε ιστοσελίδες [3].

Στη συνέχεια, θα προχωρήσουμε σε μία ανάλυση των σημαντικότερων στοιχείων που προστέθηκαν στην HTML5.

1.4 Οι νέες δυνατότητες

Στο υποκεφάλαιο αυτό θα γίνει μία παρουσίαση κάποιων σημαντικών νέων δυνατοτήτων της HTML5. Πρόκειται για τις 'Canvas', 'Video', 'Audio', 'Web Workers' και 'SVG'.

1.4.1 Canvas

Ο canvas είναι μία άμεσης λειτουργίας χαρτογραφημένη περιοχή της οθόνης, που μπορεί να χειραγωγηθεί από τη JavaScript. Ο όρος 'άμεσης λειτουργίας' αναφέρεται στον τρόπο με τον οποίο ο canvas αποδίδει τα pixel στην οθόνη. Ο canvas επανασχεδιάζει πλήρως τη χαρτογραφημένη οθόνη σε κάθε frame, χρησιμοποιώντας κλήσεις του Canvas API μέσω της JavaScript. Αυτό καθιστά τον canvas πολύ διαφορετικό από τα Flash, Silverlight ή και το SVG, τα οποία λειτουργούν σε λειτουργία διατήρησης. Σε αυτή τη λειτουργία, μία λίστα εμφάνισης αντικειμένων διατηρείται από τον αποδοτέα γραφικών, και τα αντικείμενα εμφανίζονται στην οθόνη σύμφωνα με τα χαρακτηριστικά που ορίζονται στον κώδικα (π.χ. θέση x, θέση y, κ.τ.λ.). Αυτό κρατάει τον προγραμματιστή μακριά από χαμηλού επιπέδου λειτουργίες, αλλά του δίνει λιγότερο έλεγχο στη τελική απόδοση των γραφικών στην οθόνη.

Το βασικό API του canvas περιλαμβάνει ένα περιβάλλον 2 διαστάσεων, που επιτρέπει στον προγραμματιστή να ζωγραφίζει διάφορα σχήματα, να αποδώσει κείμενο και να παρουσιάσει εικόνες, απευθείας σε μία καθορισμένη περιοχή του παραθύρου του browser. Προσφέρει δυνατότητες όπως εφαρμογή χρωμάτων, περιστροφή (σχημάτων, κειμένων, κτλ.), καθορισμός διαφάνειας αντικειμένων, χειρισμός κάθε pixel, καθώς και διάφορους τύπους γραμμών, καμπύλων, κουτιών και γεμισμάτων, για την επαύξηση των σχημάτων, κειμένων και εικόνων που τοποθετούνται επάνω στον canvas. Από μόνο του όμως, το βασικό API του canvas προσφέρει πολύ λίγα για τη δημιουργία εφαρμογών χρησιμοποιώντας αυτή τη τεχνολογία. Ωστόσο, προσθέτοντας λειτουργικότητα της JavaScript, για είσοδο δεδομένων, χρονομετρητές, κλάσεις, αντικείμενα, ήχο, προσθήκη events κτλ., είναι δυνατή η δημιουργία εντυπωσιακών εφαρμογών, απεικονίσεων, ακόμη και παιχνιδιών, χρησιμοποιώντας τον canvas.

Το DOM αντιπροσωπεύει όλα τα στοιχεία σε μία HTML σελίδα. Στη περίπτωση του στοιχείου canvas, το ίδιο το στοιχείο είναι προσβάσιμο μέσω του DOM, όμως τα γραφικά αντικείμενα που δημιουργούνται και τοποθετούνται επάνω του, δεν είναι. Αυτό συμβαίνει επειδή, όπως αναφέρθηκε, ο canvas λειτουργεί σε 'άμεση λειτουργία', και ουσιαστικά δεν έχει αντικείμενα, αλλά μόνο οδηγίες για το τι να ζωγραφίσει σε κάθε frame [4].

Στη συνέχεια θα παρουσιαστούν παραδείγματα κώδικα για τη δημιουργία και τη χειραγώγηση του στοιχείου canvas, στην HTML και στη JavaScript.

1.4.1.1 Το στοιχείο στην HTML

Ένα στοιχείο canvas στην HTML έχει τη παρακάτω μορφή:

```
<canvas id="canvasOne" width="500" height="300">
    Your browser doesn't support canvas.
</canvas>
```

Παρατηρούμε ότι ο canvas έχει 3 κύρια χαρακτηριστικά: Το 'id', το 'width' και το 'height'.

Το 'id' χρησιμοποιείται για να δώσουμε μία μοναδική ταυτότητα στο στοιχείο αυτό, ώστε να είναι εύκολος ο εντοπισμός του από τη JavaScript, με την οποία θα γίνει ο χειρισμός του.

Το 'width' και το 'height' προσδιορίζουν, αντίστοιχα, το πλάτος και το ύψος του στοιχείου που δημιουργείται, σε pixel.

Μεταξύ των tag ανοίγματος (<canvas>) και κλεισίματος (</canvas>) του canvas, υπάρχει η δυνατότητα να προστεθεί κείμενο, το οποίο θα εμφανίζεται σε περίπτωση που το στοιχείο canvas δεν υποστηρίζεται από τον τρέχον browser [4].

1.4.1.2 Χρήση από τη JavaScript

Για να εντοπίσουμε στη JavaScript το αντικείμενο που δημιουργήσαμε στο προηγούμενο παράδειγμα, κάνουμε χρήση του DOM. Χρειαζόμαστε μία αναφορά στο canvas αντικείμενο, για να γνωρίζουμε που θα εμφανίσουμε τις κλήσεις του Canvas API που θα γίνουν από τη JavaScript. Για να πάρουμε την αναφορά, χρησιμοποιούμε τη παρακάτω εντολή:

```
var theCanvas = document.getElementById("canvasOne");
```

Με την εντολή αυτή, τοποθετούμε την αναφορά του αντικειμένου με όνομα 'canvasOne', στη μεταβλητή 'theCanvas'. Μία ακόμη χρήσιμη ενέργεια, είναι ο έλεγχος του αν υποστηρίζεται ο canvas στον συγκεκριμένο browser. Ένας απλός τρόπος να ελεγχθεί αυτό, είναι ελέγχοντας το αν υπάρχει το πλαίσιο του canvas, με τον παρακάτω κώδικα:

```
if (!theCanvas || !theCanvas.getContext) {
    return;
}
```

Ο κώδικας αυτός ελέγχει δύο πράγματα. Αρχικά ελέγχει εάν το αντικείμενο 'theCanvas' έχει κάποια τιμή, άρα ελέγχει εάν το 'id' που δόθηκε στη παραπάνω εντολή, αντιστοιχεί όντως σε στοιχείο. Στη συνέχεια ελέγχει εάν αυτό το αντικείμενο, επιστρέφει το πλαίσιο του canvas (το οποίο ζητείται με το getContext). Εάν δε το επιστρέφει, τότε σημαίνει ότι ο canvas δεν υποστηρίζεται. Η εντολή return, σταματά την εκτέλεση του κώδικα JavaScript σε περίπτωση που η δοκιμή υποστήριξης αποτύχει.

Όταν επιβεβαιώσουμε την υποστήριξη του canvas, μπορούμε να πάρουμε μία αναφορά στο πλαίσιο του canvas, ώστε να το χειριστούμε. Αυτό γίνεται με τη παρακάτω εντολή:

```
var context = theCanvas.getContext("2d");
```

Το '2d', που παίρνει η συνάρτηση 'getContext()' ως είσοδο, καθορίζει ότι το πλαίσιο το οποίο ζητάμε είναι δύο διαστάσεων (2D). Ο canvas υποστηρίζει και άλλα πλαίσια, όπως το τριών διαστάσεων (3D). Στη συνέχεια, αφού έχουμε τη μεταβλητή 'context' με αναφορά στο 2D πλαίσιο, μπορούμε να ξεκινήσουμε να σχεδιάζουμε πάνω σε αυτό:

```
context.fillStyle = "#ffffaa";

context.fillRect(0, 0, 500, 300);
```

Ο παραπάνω κώδικας θα δημιουργήσει ένα ορθογώνιο. Η εντολή 'fillStyle' καθορίζει το χρώμα του αντικειμένου που θα σχεδιαστεί, ενώ η εντολή 'fillRect' θα ζωγραφίσει ένα ορθογώνιο. Στην εντολή 'fillRect', οι 4 παράμετροι που ζητούνται είναι τα x, y, width και height αντίστοιχα. Οι παράμετροι x και y ορίζουν τις συντεταγμένες του canvas απο τις οποίες θα ξεκινήσει η σχεδίαση του αντικειμένου, ενώ οι παράμετροι width και height ορίζουν τις διαστάσεις του. Για την σχεδίαση κειμένου επάνω στο canvas, ακολουθείται παρόμοια διαδικασία:

```
context.fillStyle = "#000000";

context.font = "20px _sans";

context.fillText ("Hello World!", 195, 80);
```

Ο κώδικας αυτός, θα ζωγραφίσει το κείμενο 'Hello World!' στην οθόνη, με γραμματοσειρά '20px _sans' και με χρώμα κειμένου το μαύρο. Οι παράμετροι της εντολής 'fillText' είναι το string που θα εμφανιστεί, το x και το y αντίστοιχα. Στον canvas υπάρχει επίσης δυνατότητα εμφάνισης εικόνων. Ένα παράδειγμα εισαγωγής εικόνας:

```
var helloWorldImage = new Image();

helloWorldImage.src = "helloworld.gif";

helloWorldImage.onload = function () {

        context.drawImage(helloWorldImage, 160, 130);

}
```

Στο παράδειγμα αυτό, εμφανίζεται η εικόνα με path 'helloworld.gif' στον canvas. Παρατηρείται η προσθήκη της εντολής 'onload', σε αντίθεση με τα προηγούμενα αντικείμενα που προστέθηκαν στον canvas. Αυτή η εντολή εξασφαλίζει ότι η εικόνα θα έχει φορτωθεί επιτυχώς, προτού γίνει προσπάθεια εμφάνισης της στον canvas. Σε περίπτωση που δε προστεθεί αυτή η εντολή 'ασφαλείας', ο κώδικας δε θα περιμένει τη φόρτωση του αντικειμένου και η εισαγωγή της εικόνας θα αποτύχει [4].

Το αποτέλεσμα όλων των εντολών που εκτελέστηκαν παραπάνω, και οδηγούν στην εισαγωγή ενός κίτρινου ορθογωνίου, ενός 'Hello World!' κειμένου και ενός 'Hello World!' gif επάνω στον canvas, παρουσιάζονται στη Εικόνα 1.1.

Εικόνα 1.1: Hello world σε canvas

1.4.1.3 Εξαγωγή του canvas σε εικόνα

Η εξαγωγή του canvas ως εικόνα, μπορεί να γίνει με χρήση της μεθόδου 'toDataURL()'. Η μέθοδος αυτή επιστρέφει ένα string απο δεδομένα, τα οποία αντιπροσωπεύουν τη χαρτογραφημένη εικόνα του canvas, όπως είναι ζωγραφισμένη εκείνη τη στιγμή. Θα μπορούσε να παρομοιαστεί με ένα στιγμιότυπο οθόνης. Δίνοντας το κατάλληλο όνομα τύπου σαν παράμετρο, μπορούμε να πάρουμε αυτό το string δεδομένων σε διαφορετικούς τύπους. Ο προκαθορισμένος τύπος είναι ο 'image/png' και άλλοι τύποι, όπως ο 'image/jpeg', είναι διαθέσιμοι. Ο παρακάτω κώδικας θα ανοίξει ένα νέο παράθυρο, το οποίο θα περιέχει τον canvas ως εικόνα τύπου 'png':

```
window.open(theCanvas.toDataURL(),"canvasImage","left=0,top=0,width="
+ theCanvas.width + ",height=" + theCanvas.height +",toolbar=0,resizable=0");
```

Πιο συγκεκριμένα, δημιουργείται ένα νέο παράθυρο με URL την εικόνα του canvas, με όνομα 'canvasImage' και διαστάσεις ίσες με αυτές του canvas. Η συγκεκριμένη λειτουργεία θα μπορούσε να χρησιμοποιηθεί σε web παιχνίδια ή εφαρμογές, ως στιγμιότυπο οθόνης, δίνοντας τη δυνατότητα αποθήκευσης της τρέχουσας εικόνας του canvas οποιαδήποτε στιγμή [4].

1.4.2 Video

Τα βίντεο έχουν αρχίσει να γίνονται η κύρια πηγή πληροφορίας στο διαδίκτυο, με γοργούς ρυθμούς. Σύμφωνα με μελέτη της CISCO, το 2014 τα βίντεο αποτελούσαν το 57% της παγκόσμιας κίνησης του διαδικτύου, σε σύγκριση με το 2010, όπου αποτελούσαν το 40% [5].

Το στοιχείο <video> είναι ένα νέο στοιχείο το οποίο προστέθηκε με την HTML5 και μας επιτρέπει να εισάγουμε βίντεο στην ιστοσελίδα μας, το οποίο μέχρι πρότινος ήταν αδύνατο χωρίς plugin τρίτων, όπως το πολύ γνωστό σε όλους μας Adobe Flash, ή το QuickTime της Apple. Το <video> είναι σχεδιασμένο να λειτουργεί χωρίς κάποιο script εντοπισμού υποστήριξης. Μπορούμε να δηλώσουμε πολλαπλά αρχεία βίντεο, με διαφορετικούς τύπους, στο ίδιο στοιχείο, και οι browsers οι οποίοι υποστηρίζουν το στοιχείο <video>, θα επιλέξουν ένα απ' τα διαθέσιμα αρχεία βίντεο, ανάλογα με το

ποιους τύπους βίντεο υποστηρίζουν. Απο την άλλη, οι browser που δεν υποστηρίζουν το στοιχείο <video>, θα το αγνοήσουν εντελώς, κι αυτό είναι ένα πλεονέκτημα που μπορεί να το χρησιμοποιήσει ο σχεδιαστής υπέρ του, δίνοντας εντολή να παίξουν τα βίντεο απο ένα plugin αντί αυτού [3].

Πριν απο τη προσθήκη του <video>, η αναπαραγωγή βίντεο γινόταν με τη χρήση των plugin που προαναφέρθηκαν, μέσω του tag <object>. Αυτό επέφερε κάποια σημαντικά μειονεκτήματα, επειδή το βίντεο έπαιζε μέσω ενός 'μαύρου κουτιού', απο την οπτική γωνία του browser. Άρα, η CSS δε μπορούσε να χρησιμοποιηθεί για να διαμορφώσει την εμφάνιση του βίντεο, ή να επιβάλλει αλλαγές σε αυτή. Επίσης, ούτε στη JavaScript υπήρχε κάποιος συγκεκριμένος τρόπος για να ελέγξει αυτό το 'μαύρο κουτί'. Ουσιαστικά, δεν υπήρχε δυνατότητα να γίνουν αλλαγές στη γραφική διεπαφή αναπαραγωγής βίντεο. Μεγάλο μειονέκτημα αποτελούσε και η απώλεια υποστήριξης των plugin αυτών σε κινητές συσκευές [5].

Στα επόμενα υποκεφάλαια, όπως και με το στοιχείο canvas, θα παρουσιαστούν παραδείγματα και δυνατότητες του στοιχείου <video>.

1.4.2.1 Το στοιχείο και οι ιδιότητες του

Ένα απλό στοιχείο <video> στην HTML έχει τη παρακάτω μορφή:

```
<video id="thevideo" width="320" height="240">
  <source src="muirbeach.mp4" type="video/mp4;">
  <source src="muirbeach.webm" type="video/webm;">
  <source src="muirbeach.ogg" type="video/ogg;">
</video>
```

Όπως και με το στοιχείο <canvas>, καθορίζονται αρχικά ένα id για τον εντοπισμό του αντικειμένου στη JavaScript, και οι διαστάσεις του, με τα width και height. Στη συνέχεια υπάρχει δυνατότητα δήλωσης πολλών διαφορετικών πηγών βίντεο, για να καλύψουμε τις περιπτώσεις υποστήριξης των browser. Στο στοιχείο <source>, εκτός από το 'src' που δηλώνει τη τοποθεσία του βίντεο, δηλώνεται επίσης και ο τύπος του βίντεο, με την ιδιότητα 'type' [4].

Ωστόσο, υπάρχουν πολλές ακόμα ιδιότητες που μπορούν να προστεθούν στο στοιχείο <video>. Κάποιες απο αυτές είναι οι [4]:

- **autoplay**
  Παίρνει τη τιμή true ή false.
  Οδηγεί σε αναγκαστική αναπαραγωγή του βίντεο, όταν έχει φορτώσει.

- **loop**
  Παίρνει τη τιμή true ή false.
  Επαναλαμβάνει την αναπαραγωγή του βίντεο, μόλις αυτό τελειώσει.

- **volume**
  Παίρνει μία τιμή μεταξύ του 0 και του 1.
  Ορίζει το επίπεδο έντασης του βίντεο που παίζει.

- **poster**
  Ως τιμή παίρνει ένα URL.
  Ορίζει μία εικόνα που θα εμφανίζεται όσο το βίντεο φορτώνει.

Επίσης, υπάρχουν και αρκετές ιδιότητες του <video> προσβάσιμες απο την JavaScript, αλλά και μέθοδοι για τη χειραγώγηση του [4]:

- **duration**
  Η διάρκεια του βίντεο σε δευτερόλεπτα.

- **currentTime**
  Ο τρέχων χρόνος αναπαραγωγής του βίντεο σε δευτερόλεπτα.

- **ended**
  Επιστρέφει true ή false, ανάλογα με το εάν η αναπαραγωγή του βίντεο έχει τελειώσει ή όχι.

- **muted**
  Επιστρέφει true ή false, ανάλογα με το εάν το βίντεο είναι σε σίγαση ή όχι.

- **paused**
  Επιστρέφει true ή false, ανάλογα με το εάν το βίντεο είναι σε παύση ή όχι.

- **play()**
  Μέθοδος που καλείται για την έναρξη αναπαραγωγής του βίντεο.

- **pause()**
  Μέθοδος που καλείται για τη παύση του βίντεο που αναπαράγεται.

1.4.2.2 Χρήση του στοιχείου σε συνδυασμό με τον canvas

Το στοιχείο <video> ήδη έχει την ιδιότητα ‘poster’ για να προβάλλει μία εικόνα μέχρι να ξεκινήσει η αναπαραγωγή του βίντεο, και αρκετές λειτουργίες όπως το ‘autoplay’ και το ‘loop’. Άρα είναι λογικό να έχει κανείς την απορία, γιατί χρειάζεται ο canvas. Η απάντηση είναι ότι οι δυνατότητες του <video> έχουν να κάνουν με την αναπαραγωγή του αρχείου, ενώ με τον canvas μπορούμε να χειραγωγήσουμε το ίδιο το στοιχείο. Θα παρουσιαστούν κάποιες συνοπτικές οδηγίες για τον συνδυασμό τους.

Καταρχήν, παρόλο που η εμφάνιση του βίντεο θα γίνει στον canvas, χρειαζόμαστε και πάλι ένα στοιχείο <video> στην HTML, για να φορτώσουμε και να αναπαράγουμε το βίντεο. Το βίντεο θα παιχτεί ως εικόνα στον canvas, με τη μέθοδο ‘drawImage()’, που παρουσιάστηκε νωρίτερα:

```
canvasElement.drawImage(videoElement , 85, 30);
```

Η παραπάνω μέθοδος θα εμφανίσει το <video> στοιχείο μέσα στον canvas, ωστόσο θα εμφανιστεί μόνο το πρώτο frame του βίντεο, καθώς το προσθέσαμε ως εικόνα. Δυστυχώς δεν υπάρχει κάποια μέθοδος για εισαγωγή βίντεο στον canvas, οπότε η λύση που απαιτείται σε αυτή τη περίπτωση, είναι ένα interval ανανέωσης της εικόνας που παρουσιάζεται:

```
videoElement.play();

setInterval(drawScreen, 33);
```

Με τις παραπάνω εντολές ξεκινάει η αναπαραγωγή του βίντεο, και καλείται η μέθοδος ‘drawScreen()’ κάθε 33ms, ώστε το βίντεο να αναπαράγεται με ρυθμό 30 FPS (Frames per second). Μπορούμε φυσικά να αλλάξουμε αυτή τη μεταβλητή για να επιτύχουμε μεγαλύτερο ή μικρότερο ρυθμό ανανέωσης. Η μέθοδος ‘drawScreen()’ παρουσιάζεται παρακάτω:

```
function drawScreen () {
        context.drawImage(videoElement , 85, 30);
}
```

Με τις οδηγίες που παρουσιάστηκαν, μπορούμε να αναπαράγουμε βίντεο μέσω του canvas. Με αυτό, κερδίζουμε τρομερά πλεονεκτήματα, όπως τη δυνατότητα εμφάνισης μηνυμάτων σε οποιοδήποτε σημείο της εικόνας του βίντεο, αλλά και δυνατότητες όπως περιστροφή του βίντεο μέσα στον canvas, ή προσθήκη event και κουμπιών για τη χειραγώγηση του βίντεο. Γενικά, η δυνατότητα χειραγώγησης όλων των νέων στοιχείων απο τη JavaScript, είναι το κλειδί που δημιούργησε το πλήθος των συνδυασμών και των δυνατοτήτων αυτών. Το αποτέλεσμα των εντολών που εκτελέστηκαν σε αυτό το υποκεφάλαιο, παρουσιάζεται στην Εικόνα 1.2 [4].

Εικόνα 1.2: Αναπαραγωγή βίντεο μέσα στον canvas

1.4.3 Audio

Πριν απο την HTML5, για να ενσωματώσει κάποιος ένα αρχείο ήχου σε μία ιστοσελίδα, έπρεπε (όπως και στη περίπτωση του βίντεο) να χρησιμοποιήσει ένα plugin τρίτων [6]. Η βασική χρήση του στοιχείου <audio> είναι πολύ παρόμοια με αυτή του <video>. Η μόνη απαραίτητη ιδιότητα είναι το ‘src’, η οποία καθορίζει τη διεύθυνση προς κάποιο αρχείο, το οποίο θα παίξει στον browser [4]. Ένα παράδειγμα δήλωσης του στοιχείου στην HTML είναι το παρακάτω:

```
<audio controls>
        <source src="johann_sebastian_bach_air.ogg">
        <source src="johann_sebastian_bach_air.mp3">
        An audio clip from Johann Sebastian Bach.
```

</audio>

Με τον κώδικα αυτό δηλώνονται δύο μορφές του τραγουδιού ‘johann_sebastian_bach_air’ προς αναπαραγωγή. Όπως και με το στοιχείο <video>, ο browser θα αναπαράγει το πρώτο αρχείο που υποστηρίζεται [7]. Η εμφάνιση ενός <audio> στοιχείου (με controls, διότι χωρίς αυτά δεν υπάρχει κάποιο γραφικό), παρουσιάζεται στην Εικόνα 1.3.

Εικόνα 1.3: Γραφική διεπαφή του στοιχείου <audio>

Στη συνέχεια αναφέρονται κάποιες επιπλέον ιδιότητες που μπορούν να προστεθούν σε ένα στοιχείο <audio> [6]:

- **autoplay**
  Παίρνει τιμή true ή false.
  Διευκρινίζει εάν το αρχείο θα αναπαραχθεί αυτόματα με τη φόρτωση του.

- **controls**
  Παίρνει τιμή true ή false.
  Διευκρινίζει εάν το στοιχείο θα εμφανιστεί μαζί με κουμπιά ελέγχου.

- **muted**
  Ορίζει την αρχική κατάσταση του στοιχείου σε σίγαση.

- **loop**
  Οδηγεί σε αναπαραγωγή του αρχείου σε βρόγχο.

Οι παραπάνω ιδιότητες είναι διαθέσιμες για προσθήκη στο στοιχείο, στην HTML. Στη JavaScript, κάποιες ιδιότητες του στοιχείου, τις οποίες μπορούμε να ανακτήσουμε, είναι οι παρακάτω [7]:

- **duration**
  Η διάρκεια του αρχείου ήχου σε δευτερόλεπτα.

- **paused**
  Επιστρέφει true εάν το αρχείο ήχου είναι σε παύση, ή δεν έχει ξεκινήσει ακόμα η αναπαραγωγή του.

- **ended**
  Επιστρέφει true εάν η αναπαραγωγή του αρχείου ήχου έχει τελειώσει.

- **startTime**
  Επιστρέφει τον νωρίτερο χρόνο στον οποίο μπορεί να επιστρέψει η αναπαραγωγή του αρχείου ήχου. Η συγκεκριμένη τίμη, στις περισσότερες περιπτώσεις είναι 0. Σε περίπτωση live streaming όμως, μπορεί να είναι μεγαλύτερη, εάν παλαιότερα μέρη του περιεχομένου έχουν απελευθερωθεί απο τη σελίδα.

- **currentSrc**
  Επιστρέφει ένα string που αντιπροσωπεύει τη διεύθυνση του αρχείου που έχει φορτωθεί ή αναπαράγεται. Δηλαδή, το ποιο απο τα πολλαπλά sources επέλεξε τελικά ο browser.

Επίσης, οι μέθοδοι 'play()' και 'pause()', που επεξηγήθηκαν στο υποκεφάλαιο του <video>, δουλεύουν με τον ίδιο τρόπο στο στοιχείο <audio>, δηλαδή για έναρξη και παύση αναπαραγωγής αντίστοιχα. Φυσικά, όπως και στη περίπτωση του <video>, τα στοιχεία <canvas> και <audio> μπορούν να συνδυαστούν για τη δημιουργία πρωτότυπων εμφανίσεων στον αναπαραγωγέα ήχου [4].

1.4.4 Web Workers

Οι 'Web Workers' είναι μία συλλογή απο μεθόδους που χρησιμοποιείται για την εκτέλεση πολλών script στο παρασκήνιο. Δηλαδή, ενεργοποιεί το 'multi-threading', δίνοντας τη δυνατότητα να εκτελούνται ταυτόχρονα όλες οι εργασίες παρασκηνίου [8]. Το API των 'Web Workers' επιτρέπει στους προγραμματιστές να τρέχουν script που εκτελούν εργασίες με μεγάλο υπολογιστικό κόστος, χωρίς να παρεμποδιστεί η λειτουργία της διεπαφής χρήστη [9].

Όσο ισχυροί κι αν φαίνονται οι 'Web Workers', υπάρχουν ορισμένα πράγματα που δε μπορούν να κάνουν. Για παράδειγμα, όταν ένα script εκτελείται μέσα σε ένα Web Worker, δε μπορεί να έχει πρόσβαση στο αντικείμενο 'window', το οποίο σημαίνει ότι οι 'Web Workers' δεν έχουν άμεση πρόσβαση στην ιστοσελίδα και το DOM. Επίσης, παρόλο που δε μπορούν να εμποδίσουν τη γραφική διεπαφή στον browser, μπορούν να χρησιμοποιούν πολλούς κύκλους CPU και να μειώσουν την απόκριση του συστήματος.

Ένα απλό παράδειγμα χρήσης των 'Web Workers', για να γίνουν πιο εύκολα κατανοητοί, είναι η περίπτωση μίας απλής web εφαρμογής, η οποία χρειάζεται να πραγματοποιεί κάποιες πράξεις ή εντολές στο παρασκήνιο, οι οποίες δεν είναι επιθυμητό να παρεμβαίνουν στη διαδραστικότητα της εφαρμογής. Δημιουργώντας έναν Web Worker, μπορούμε να εκτελούμε τις εντολές σε αυτόν, και προσθέτοντας έναν event listener, να 'ακούμε' τα μηνύματα που αυτός στέλνει.

Ένας Web Worker αρχικοποιείται δίνοντας το URL ενός JavaScript αρχείου, το οποίο περιέχει τον κώδικα που θα εκτελεί ο Web Worker [10]. Στη συνέχεια, θα παρουσιαστούν παραδείγματα απλής χρήσης των 'Web Workers', ξεχωριστά για τον κώδικα στην ιστοσελίδα και για τον κώδικα στο JavaScript αρχείο του Web Worker.

1.4.4.1 Κώδικας στην ιστοσελίδα

Ένα παράδειγμα αρχικοποίησης ενός Web Worker είναι το παρακάτω:

```
worker = new Worker("echoWorker.js");
```

Για να αποστείλλουμε ένα μήνυμα απο την ιστοσελίδα στον Web Worker, χρησιμοποιούμε την εντολή 'postMessage'. Επίσης, για να 'ακούμε' τα μηνύματα που στέλνει αυτός, προσθέτουμε και έναν event listener:

```
document.getElementById("helloButton").onclick = function() {
        worker.postMessage("Here's a message for you");
}
worker.addEventListener("message", messageHandler, true);
function messageHandler(mes) {
```

```
        console.log(mes);
    }
```

Με τις παραπάνω εντολές, αρχικά στέλνουμε ένα μήνυμα στον Web Worker με κάθε κλικ ενός κουμπιού, και στη συνέχεια προσθέτουμε έναν event listener που καλεί τη συνάρτηση ‘messageHandler’, η οποία εμφανίζει τα μηνύματα που στέλνει ο Web Worker. Για να τερματίσουμε τη λειτουργία του Web Worker, χρησιμοποιείται η εντολή ‘terminate’:

```
worker.terminate();
```

Ένας τερματισμένος Web Worker, δεν ανταποκρίνεται στα μηνύματα, ούτε εκτελεί εντολές πλέον. Επίσης, δεν είναι δυνατό να γίνει επανεκκίνηση του. Αντί αυτού, μπορεί να δημιουργηθεί ένας νέος με το ίδιο URL [10].

1.4.4.2 Κώδικας στον Web Worker

Στο αρχείο του Web Worker, η διαδικασία που ακολουθούμε για την αποστολή και λήψη μηνυμάτων απο την ιστοσελίδα, είναι παρόμοια:

```
addEventListener("message", messageHandler, true);
function messageHandler(mes) {
    postMessage("worker says: " + mes.data + " too");
}
```

Ο κώδικας αυτός, προσθέτει έναν event listener για τα μηνύματα που έρχονται απο την ιστοσελίδα, ο οποίος καλεί τη συνάρτηση ‘messageHandler’, η οποία στέλνει ένα μήνυμα πίσω στην ιστοσελίδα. Μέσω του πεδίου ‘data’ (‘mes.data’ στον κώδικα), μπορούμε να ανακτήσουμε το μήνυμα που έστειλε η ιστοσελίδα. Στο αρχείο του Web Worker, καθώς η πρόσβαση στο DOM δεν είναι διαθέσιμη, για την εισαγωγή επιπλέον script αρχείων, μπορεί να χρησιμοποιηθεί η εντολή ‘importScripts’:

```
importScripts(“helper.js”, “anotherHelper.js”);
```

Η εντολή αυτή θα εισάγει τα αρχεία JavaScript, με τη σειρά που αναφέρθηκαν. Μία άλλη λειτουργικότητα που είναι διαθέσιμη στον Web Worker, είναι αυτή του API της JavaScript για τον χρόνο. Η χρήση χρονομετρητών με το setTimeout και επαναλήψεων με το setInterval προσφέρουν ακόμη περισσότερες δυνατότητες στους Web Workers:

```
var t = setTimeout(postMessage, 2000, "delayed message");
```

Τέλος, αξίζει να αναφερθεί ότι ένας Web Worker, έχει τη δυνατότητα να δημιουργεί νέους Web Workers:

```
var subWorker = new Worker("subWorker.js");
```

Αυτή η δυνατότητα όμως θα πρέπει να χρησιμοποιείται πολύ προσεκτικά. Για παράδειγμα, ένας Web Worker που δημιουργεί αναδρομικά νέους Web Workers δίνοντας ως URL τον εαυτό του, θα μπορούσε να προκαλέσει υπερφόρτωση του επεξεργαστή [10].

1.4.5 SVG

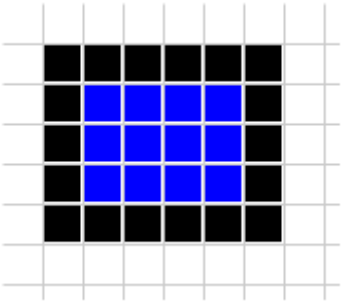
Η SVG (Scalable Vector Graphics) είναι μία γλώσσα που χρησιμοποιείται για τη περιγραφή δισδιάστατων αντικειμένων. Πριν απο την HTML5, χρησιμοποιούνταν ως plugin. Με την HTML5 όμως, έχει ενσωματωθεί πλήρως σε αυτή, μέσω του στοιχείου <svg>. Έτσι, μπορούμε πλέον να

δημιουργήσουμε ιστοσελίδες με γραφικά SVG, με αυτά να είναι εντελώς συμβατά με την υπόλοιπη ιστοσελίδα. Δηλαδή, ακολουθούν τις διαμορφώσεις απο τη CSS και η JavaScript έχει τη δυνατότητα αλληλεπίδρασης με τα αντικείμενα SVG, για τη σχεδίαση γραφικών, ή τη προσθήκη εφέ σε ενέργειες του χρήση πάνω στα αντικείμενα.

Η SVG αντιπροσωπεύει ένα μέσο οπτικοποίησης που είναι υψηλότερου επιπέδου σε σχέση με τον Canvas. Η SVG διαθέτει ένα DOM και έχει ένα μοντέλο ενεργειών, που δεν είναι διαθέσιμο στον Canvas. Άρα, για εφαρμογές που απαιτούν γραφικά με διαδραστικότητα, η SVG είναι η καλύτερη πλατφόρμα [11]. Για τους λόγους που αναφέρθηκαν, η SVG είναι αυτή που επιλέχθηκε έναντι του Canvas, για την οπτικοποίηση λιστών ηλεκτρονικού ταχυδρομείου, που είναι και το θέμα της πτυχιακής αυτής εργασίας. Καθώς το ζητούμενο ήταν εκτός απο την οπτικοποίηση των λιστών, η προσθήκη αλληλεπιδράσεων σε γεγονότα όπως ‘mouse over’, ή ‘click’, αλλά και η εκμετάλλευση της πιο ιεραρχικής φύσης της SVG, για τη καλύτερη διαμόρφωση της τελικής εφαρμογής.

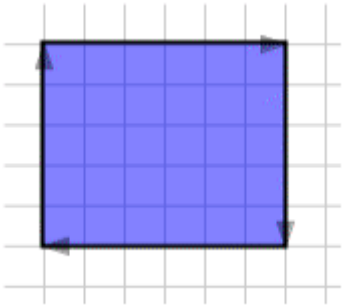
Πριν δοθούν περισσότερες λεπτομέρειες σχετικά με την SVG, θα ήταν χρήσιμο να επεξηγηθούν τα 2 σημαντικότερα συστήματα απόδοσης γραφικών σε υπολογιστές, δηλαδή το σύστημα ράστερ και το σύστημα διανυσμάτων.

Στα συστήματα ράστερ, μία εικόνα αποδίδεται ως ένα ορθογώνιος πίνακας αποτελούμενος απο pixel. Κάθε pixel αντιπροσωπεύεται είτε απο τις RGB τιμές του, είτε ως ένας δείκτης σε μία λίστα χρωμάτων. Αυτή η ακολουθία απο pixel, συνήθως αποθηκεύεται σε συμπιεσμένη μορφή. Και επειδή οι περισσότερες σύγχρονες συσκευές απεικόνισης είναι επίσης συσκευές ράστερ, για να παρουσιάσουν μία εικόνα χρειάζονται απλώς ένα πρόγραμμα που να αποσυμπιέσει την ακολουθία και να τη μεταφέρει στην οθόνη. Ένα παράδειγμα ενός συστήματος ράστερ παρουσιάζεται στο Σχήμα 1.1.



Σχήμα 1.1: Παράδειγμα ενός συστήματος ράστερ

Απο την άλλη, στα συστήματα διανυσμάτων, μία εικόνα αποδίδεται σαν μία ακολουθία απο γεωμετρικά σχήματα. Αντί να λαμβάνει ένα σύνολο απο pixel, το πρόγραμμα απεικόνισης λαμβάνει εντολές για τον σχεδιασμό σχημάτων σε συγκεκριμένες συντεταγμένες. Ένα παράδειγμα ενός συστήματος διανυσμάτων παρουσιάζεται στο Σχήμα 1.2.



Σχήμα 1.2: Παράδειγμα ενός συστήματος διανυσμάτων

Κάποιοι περιγράφουν το σύστημα διανυσμάτων ως ένα σετ οδηγιών για μία απεικόνηση, ενώ το σύστημα ράστερ ως χρωματισμένα σημεία σε συγκεκριμένα μέρη. Σε αντίθεση με το σύστημα ράστερ, τα γραφικά στο σύστημα διανυσμάτων 'αντιλαμβάνονται' τι είναι, επειδή σχεδιάζονται ως αντικείμενα, και όχι ως μία ακολουθία απο pixel. Έτσι, τα αντικείμενα αυτά μπορούν να αλλάξουν το σχήμα και το χρώμα τους [12]. Η SVG λειτουργεί με το σύστημα διανυσμάτων, το οποίο προσφέρει μεγάλα πλεονεκτήματα για τους σκοπούς αυτής της Πτυχιακής Εργασίας.

Επιστρέφοντας στη περιγραφή της SVG, τα γραφικά που παρουσιάζει μπορεί να είναι είτε στατικά, είτε δυναμικά. Η συμμόρφωση της SVG με το συντακτικό της XML, αλλά και ο περιγραφική μορφή που χρησιμοποιεί για τη παρουσίαση των γραφικών, τη καθιστούν μία γλώσσα παρουσίασης γραφικών μη σχετική με την ανάλυση της εικόνας [13].

Κάποια από τα στοιχεία που μπορούν να παρουσιαστούν στην SVG παρουσιάζονται παρακάτω [14]:

- **Κατασκευαστικά:**
  <svg>, <defs>, <desc>, <title>, <metadata>, <symbol>, <use>, <g>, <switch>, <a>, <view>

- **Σχήματα:**
  <circle>, <ellipse>, <rect>, <line>, <polyline>, <polygon>, <path>, <cursor>

- **Κείμενο:**
  <text>, <tspan>, <tref>, <textPath>, <altGlyph>, <altGlyphDef>, <altGlyphItem>, <glyphRef>, <font>

- **Στυλ:**
  <style>, <marker>, <linearGradient>, <radialGradient>, <stop>, <pattern>

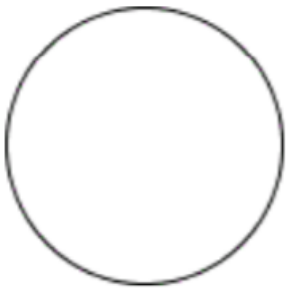
- **Εφέ:**
  <clipPath>, <mask>, <filter>

- **Animations:**
  <animate>, <set>, <animateMotion>, <animateTransform>, <animateColor>, <mpath>

Για την ευκολότερη κατανόηση του τρόπου λειτουργίας της SVG, θα παρουσιαστεί με παραδείγματα κώδικα, και τα αντίστοιχα αποτελέσματα τους σε εικόνες. Ένα απλό παράδειγμα ενός στοιχεία <svg> είναι το παρακάτω:

```
<svg width="140px" height="170px">
    <title>Simple SVG element</title>
    <circle cx="70" cy="95" r="50" style="stroke: black; fill: none" />
</svg>
```

Ο παραπάνω κώδικας δημιουργεί ένα στοιχείο <svg> με τίτλο ‘Simple SVG element’, με πλάτος 140px και ύψος 170px, στο οποίο περιέχεται ένα στοιχείο <circle>. Το στοιχείο <circle> θα δημιουργήσει ένα κύκλο εντός του στοιχείου <svg>. Τα ‘cx’ και ‘cy’ διευκρινίζει τις συντεταγμένες τοποθέτησης του και το ‘r’ την ακτίνα του. Το αποτέλεσμα που θα προκύψει παρουσιάζεται στο Σχήμα 1.3.



Σχήμα 1.3: Παράδειγμα κύκλου στην SVG

Τα υπόλοιπα διαθέσιμα σχήματα δημιουργούνται με παρόμοιο τρόπο. Η προεπιλεγμένη μονάδα μέτρησης είναι τα pixel. Οι παρακάτω δύο εντολές θα δημιουργήσουν και οι δύο ένα στοιχείο <svg> με πλάτος 200 pixel και ύψος 150 pixel:

```
<svg width="200" height="150">
<svg width="200px" height="200px">
```

Άλλες διαθέσιμες μονάδες μέτρησης είναι οι παρακάτω:

- **in** (ίντσες)
- **mm** (χιλιοστά του μέτρου)
- **cm** (εκατοστά του μέτρου)
- **pc** (picas, το 1/6 της ίντσας)
- **pt** (points, το 1/72 της ίντσας)
- **ex** (το ύψος του χαρακτήρα ‘x’)

Εάν ένα <svg> στοιχείο βρίσκεται εντός ενός άλλου <svg> στοιχείου, είναι εφικτό το μέγεθος του να καθοριστεί με ποσοστά, τα οποία μετρούνται με βάση το μέγεθος του εξωτερικού στοιχείου. Επίσης, είναι εφικτό να γίνει συνδυασμός δύο μονάδων μέτρησης για τον ορισμό ενός στοιχείου <svg>:

```
<svg width="2cm" height="36pt">
```

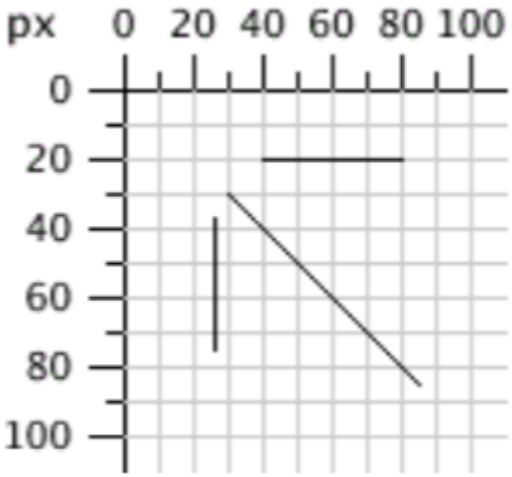
Εκτός από τα κλασσικά σχήματα, όπως ο κύκλος και το ορθογώνιο, άξια περιγραφής είναι τα στοιχεία <line>, <polyline> και <path>. Ο κώδικας για τη δημιουργία ενός στοιχείου <line> παρουσιάζεται παρακάτω:

```
<line x1="start-x" y1="start-y" x2="end-x" y2="end-y">
```

Τα ‘x1’-‘y1’ και ‘x2’-‘y2’ καθορίζουν τις συντεταγμένες αρχής και τέλους της γραμμής, αντίστοιχα. Κάποια παραδείγματα γραμμών παρουσιάζονται στον κώδικα που ακολουθεί.

```
<svg width="200px" height="200px" viewBox="0 0 200 200">
    <!-- horizontal line -->
    <line x1="40" y1="20" x2="80" y2="20" style="stroke: black;" />
    <!-- vertical line -->
    <line x1="0.7cm" y1="1cm" x2="0.7cm" y2="2.0cm" style="stroke: black;" />
    <!-- diagonal line -->
    <line x1="30" y1="30" x2="85" y2="85" style="stroke: black;" />
</svg>
```

Ο κώδικας αυτός δημιουργεί 3 γραμμές, μία οριζόντια, μία κάθετη και μία διαγώνια. Το γραφικό αποτέλεσμα παρουσιάζεται στο Σχήμα 1.4.



Σχήμα 1.4: Παραδείγματα γραμμών στην SVG

Από την άλλη, το στοιχείο <polyline>, είναι ουσιαστικά μία σειρά γραμμών. Συνήθως χρησιμοποιείται όταν υπάρχει ανάγκη από ένα σχήμα πολλαπλών γραμμών, που όμως δεν ΄κλείνει’, όπως το πολύγωνο. Ένα <polyline> παρουσιάζεται στον κώδικα παρακάτω:

```
<svg width="200px" height="200px" viewBox="0 0 200 200">
    <polyline
    points="5 20, 20 20, 25 10, 35 30, 45 10, 55 30, 65 10, 75 30, 80 20, 95 20"
    style="stroke: black; stroke-width: 3; fill: none;" />
</svg>
```

Το γραφικό αποτέλεσμα του παραπάνω <polyline> παρουσιάζεται στο Σχήμα 1.5.

Σχήμα 1.5: Παράδειγμα polyline στην SVG

Η ιδιότητα ‘fill’ της CSS, όπως παρατηρούμε στον παραπάνω κώδικα, είναι καθορισμένη στη τιμή ‘none’. Αυτό ίσως φαίνεται περιττό, αλλά είναι αναγκαίο, καθώς η SVG ίσως προσπαθήσει να γεμίσει τα κενά μεταξύ των πολλαπλών γραμμών και να οδηγήσει σε μη επιθυμητό αποτέλεσμα. Ένα παράδειγμα του ίδιου κώδικα χωρίς το ‘fill : none’ παρουσιάζεται στο Σχήμα 1.6.

Σχήμα 1.6: Παράδειγμα ανεπιθύμητου γεμίσματος των κενών ενός polyline από την SVG

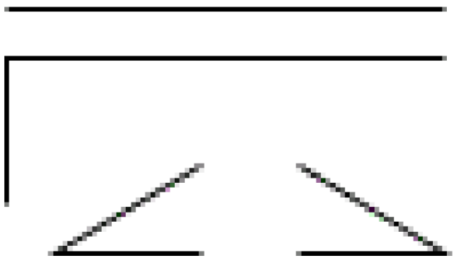
Το τελευταίο στοιχείο της SVG που θα αναλυθεί, είναι το <path>. Όλα τα βασικά σχήματα που περιεγράφηκαν δεν είναι τίποτα παραπάνω από σύντομες μορφές του πιο γενικού στοιχείου <path>. Είναι καλή τεχνική να χρησιμοποιούμε αυτές τις συντομεύσεις. Κάνουν τον SVG κώδικα μας πιο αναγνώσιμο και καλύτερα δομημένο. Το <path> είναι πιο γενικό από αυτές. Σχεδιάζει το περίγραμμα οποιουδήποτε αυθαίρετου σχήματος, καθορίζοντας μία σειρά συνδεδεμένων γραμμών και καμπυλών. Όλα τα δεδομένα που καθορίζουν το περίγραμμα που θα σχεδιαστεί βρίσκονται στην ιδιότητα ‘d’ του στοιχείου <path>. Τα δεδομένα αυτά αποτελούνται από εντολές ενός χαρακτήρα, όπως το ‘M’ που σημαίνει ‘move to’ και το ‘L’ που σημαίνει ‘line to’, ακολουθούμενα φυσικά από τις επιθυμητές συντεταγμένες για την εκάστοτε εντολή.

Κάθε <path> ξεκινάει πάντα με μία εντολή ‘move to’. Αυτή η εντολή ορίζει τη τοποθεσία έναρξης της σχεδίασης του περιγράμματος. Η εντολή δίνεται με τον χαρακτήρα ‘M’ και τις συντεταγμένες να ακολουθούν, χωρισμένες είτε με κόμμα είτε με κενό. Στη συνέχεια ακολουθούν μία ή περισσότερες εντολές ‘line to’, με τον χαρακτήρα ‘L’ και συντεταγμένες. Στο παρακάτω παράδειγμα δημιουργούνται κάποια σχήματα με τη χρήση στοιχείων <path>:

```
<g style="stroke: black; fill: none;">
    <!-- single line -->
    <path d="M 10 10 L 100 10" />
    <!-- a right angle -->
    <path d="M 10, 20 L 100, 20 L 100,50" />
    <!-- two thirty-degree angles -->
    <path d="M 40 60, L 10 60, L 40 42.68, M 60 60, L 90 60, L 60 42.68" />
```

</g>

Το πρώτο στοιχείο <path> σχεδιάζει μία απλή γραμμή, ξεκινώντας από τις συντεταγμένες ‘10,10’ και καταλήγοντας στις ‘100,10’. Το δεύτερο στοιχείο, σχεδιάζει μία 2 ευθείες σε γωνία 90 μοιρών. Το τρίτο <path>, από την άλλη, σχεδιάζει 2 σχήματα, χρησιμοποιώντας 2 φορές την εντολή ‘move to’ για να μετακινήσει την ‘πένα’ σχεδίασης σε άλλο σημείο. Τα 2 αυτά σχήματα είναι γωνίες 30 μοιρών. Το αποτέλεσμα των τριών <path> παρουσιάζεται στο Σχήμα 1.7.



Σχήμα 1.7: Παραδείγματα στοιχείων <path> στην SVG

Μία ακόμη χρήσιμη εντολή, στο σχεδιασμό ενός σχήματος με το <path>, είναι η ‘close path’. Αυτή η εντολή σχεδιάζει μία ευθεία γραμμή προς το αρχικό σημείο του τρέχοντος σχήματος. Δηλαδή, μία ευθεία γραμμή προς τις συντεταγμένες της τελευταίας εντολής ‘M’. Για τη συγκεκριμένη εντολή χρησιμοποιείται το γράμμα ‘Z’. Ένα παράδειγμα χρήσης του παρουσιάζεται παρακάτω:

<g style="stroke: black; fill: none;">

<!-- rectangle with closepath -->

<path d="M 60 10, L 90 10, L 90 30, L 60 30, Z" />

</g>

Με την εντολή ‘Z’ στο τέλος, η ‘πένα’ σχεδίασης επιστρέφει στις συντεταγμένες ’60,10’, οι οποίες είχαν καθοριστεί με την εντολή ‘M’ ως αρχικό σημείο. Μία γραφική απεικόνιση του ορθογωνίου που δημιουργείται παρουσιάζεται στο Σχήμα 1.8 [12].

Σχήμα 1.8: Παραδείγματα χρήσης της εντολής ‘close path’ στην SVG

Με την εντολή ‘Z’ στο τέλος, η ‘πένα’ σχεδίασης επιστρέφει στις συντεταγμένες ’60,10’, οι οποίες είχαν καθοριστεί με την εντολή ‘M’ ως αρχικό σημείο. Μία γραφική απεικόνιση του ορθογωνίου που δημιουργείται παρουσιάζεται στο Σχήμα 1.8.

Η SVG είναι η τεχνολογία που χρησιμοποιείται για τη Πτυχιακή Εργασία αυτή, ωστόσο δε χρησιμοποιείται άμεσα, όπως παρουσιάστηκε παραπάνω, αλλά έμμεσα, μέσω της βιβλιοθήκης ‘d3.js’, η οποία θα παρουσιαστεί στο επόμενο κεφάλαιο.

1.5 Επίλογος

Σε αυτό το κεφάλαιο έγινε μία παρουσίαση της HTML5. Αρχικά, έγινε μία ιστορική αναδρομή για τη πορεία της δημιουργίας της, από το όραμα μίας ομάδας προγραμματιστών, έως και τη τελική ενσωμάτωση της στον παγκόσμιο ιστό, η οποία άργησε πολύ και αντιμετώπισε μεγάλα εμπόδια. Στη συνέχεια, παρουσιάστηκαν τα κύρια χαρακτηριστικά της, και το ποια προβλήματα επιλύουν. Τέλος, έγινε μία εκτενής ανάλυση των κυριότερων νέων δυνατοτήτων που προστέθηκαν σε αυτή, με αρκετά παραδείγματα για τη κατανόηση της λειτουργίας τους.

Κεφάλαιο 2ο: Η βιβλιοθήκη d3.js

2.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθεί η βιβλιοθήκη ‘d3.js’, η οποία χρησιμοποιήθηκε για τους σκοπούς αυτής της Πτυχιακής Εργασίας. Αρχικά, θα γίνει μία σύντομη θεωρητική εισαγωγή στη βιβλιοθήκη. Στα επόμενα υποκεφάλαια θα παρουσιαστούν πρακτικά παραδείγματα χρήσης της, για τη καλύτερη κατανόηση της λειτουργικότητας της.

2.2 Εισαγωγή στη d3.js

Η ‘d3.js’ είναι μία βιβλιοθήκη JavaScript ανοιχτού κώδικα που προσφέρει τη δυνατότητα εύκολης διαχείρισης HTML εγγράφων που βασίζονται σε δεδομένα. Χρησιμοποιεί τη JavaScript για να εφαρμόσει μία χαρτογράφηση των δεδομένων στα έγγραφα αυτά. Το ‘d3’ είναι συντομογραφία του ‘Data Driven Documents’ και θεωρείται από πολλούς ως μία βιβλιοθήκη οπτικοποίησης δεδομένων. Αυτό είναι ορθό εν μέρει, καθώς έχει αυτή τη δυνατότητα, ωστόσο προσφέρει πολλά ακόμα, όπως:

- Επιλογή των στοιχείων στο HTML DOM με αποτελεσματικό τρόπο.
- Δυνατότητα διαμόρφωσης των στοιχείων του DOM δυναμικά.
- Σύνδεση των δεδομένων με τα οπτικά αντικείμενα που δημιουργεί.
- Προδιαγραφές για το χειρισμό προσθηκών ή αφαιρέσεων στοιχείων δεδομένων.
- Ορισμός ενός μοντέλου αλληλεπίδρασης μεταξύ χρήστη και δεδομένων.
- Δυνατότητα προσδιορισμού τροποποιήσεων στα οπτικοποιημένα δεδομένα, βάση δυναμικών αλλαγών στα δεδομένα.

Γενικά, η ‘d3.js’ μας βοηθά να ‘ζωντανέψουμε’ τα δεδομένα μας, χρησιμοποιώντας τις HTML, CSS και SVG. Επικεντρώνεται στον τρόπο με τον οποίο παρουσιάζονται, στις δυναμικές αλλαγές στην οπτικοποίηση ανταποκρινόμενη σε τροποποιήσεις αυτών, αλλά και στην αλληλεπίδραση του χρήστη με τα δεδομένα, μέσω της οπτικοποιημένης μορφής τους. Χρησιμοποιώντας τη βιβλιοθήκη αυτή, μπορούν να δημιουργηθούν πλούσιες εφαρμογές οπτικοποίησης δεδομένων [15].

Ένας λόγος που η βιβλιοθήκη αυτή έγινε δημοφιλής, είναι η προσέγγιση της στα δεδομένα. Αντί να διαβάζει τα στοιχεία ένα-ένα, επαναλαμβανόμενα, και να τα σχεδιάζει στην οθόνη, η d3 επιτρέπει μία σιωπηρή (ή αλλιώς υπονοούμενη) δηλωτική αναπαράσταση. Δηλαδή, η λογική της d3 σκέφτεται το ‘πως απαρτίζεται η απεικόνιση’ παρά το ‘πως κάθε αντικείμενο διατάσσεται στη σκηνή’. Ένας ακόμη λόγος δημοτικότητας της, που προαναφέρθηκε και παραπάνω, είναι η σαφής εστίαση στα κατώτατα web standards, δηλαδή τις HTML, CSS και SVG. Αυτό που δεν αναφέρθηκε είναι τα τρία μεγάλα πλεονεκτήματα αυτής της εστίασης, τα οποία είναι τα παρακάτω:

- **Συμβατότητα:** Η d3 εκμεταλλεύεται πλήρως τις HTML, CSS και SVG. Επομένως, οι προγραμματιστές έχουν τη δυνατότητα χρήσης όλων των βασικών χαρακτηριστικών των τεχνολογιών αυτών, κατά τη σύνθεση και τη σχεδίαση των οπτικοποιήσεων τους.

- **Debugging:** Η d3 θα προσαρτήσει στο DOM όλα τα HTML στοιχεία και τα style τους. Θα προσαρτήσει επίσης και όλα τα στοιχεία SVG, μαζί με τις CSS ιδιότητες τους. Αυτό κάνει εφικτό να βλέπουμε όλα τα στοιχεία που δημιουργούνται από τη d3 απλά ανοίγοντας τα ‘developer tools’ από τον browser μας. Αυτό επιτρέπει τους προγραμματιστές να χρησιμοποιούν τα εργαλεία για debugging που χρησιμοποιούσαν και πριν, και με τα οποία είναι οικείοι.

- **Απόδοση:** Η d3 βασίζεται στην SVG και ως εκ τούτου διευκολύνει τη βελτιστοποίηση της απόδοσης των αλληλεπιδράσεων και των animations, δίνοντας πλήρη πρόσβαση σε όλες τις δυνατότητες της SVG. Στις περισσότερες άλλες βιβλιοθήκες γραφικών, συνήθως οι δυνατότητες περιορίζονται σε αυτές που προσφέρονται από το API της βιβλιοθήκης.

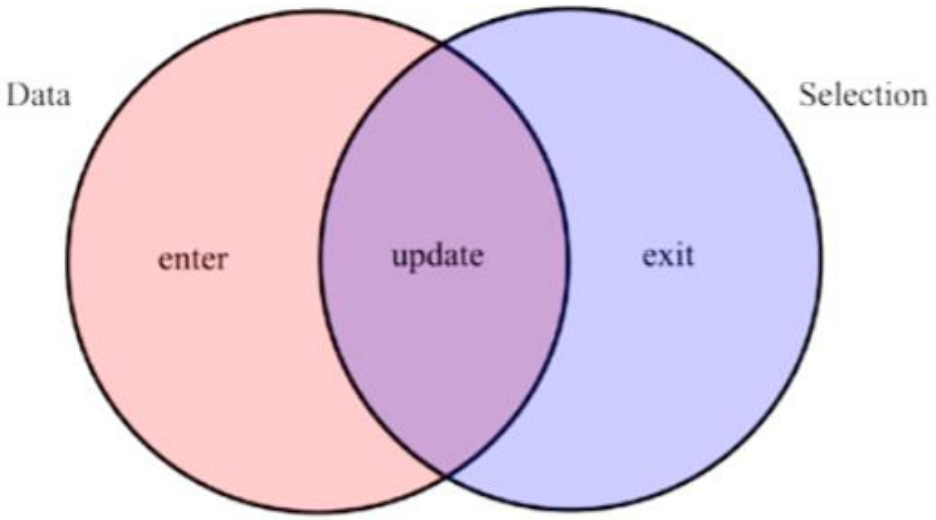
Ωστόσο, το χαρακτηριστικό που διακρίνει τη d3 από τις υπόλοιπες βιβλιοθήκες, είναι ένα: η έννοια των 'data joins'. Όταν δεσμεύουμε ένα πίνακα από δεδομένα, η d3 αυτόματα τέμνει το παλιό σύνολο δεδομένων, με το νέο. Δημιουργεί τελικά τρία νέα σύνολα δεδομένων:

1) Το σύνολο εισόδου (**enter set**), το οποίο αποθηκεύει όλα τα στοιχεία από το νέο σύνολο δεδομένων, τα οποία δεν ανήκουν στο παλιό σύνολο δεδομένων, και άρα πρέπει να προστεθούν.

2) Το σύνολο ενημέρωσης (**update set**), το οποίο αποθηκεύει όλα τα στοιχεία από το νέο σύνολο δεδομένων, τα οποία όμως ανήκουν και στο παλιό σύνολο δεδομένων, και πρέπει να ενημερωθούν.

3) Το σύνολο εξόδου (**exit set**), το οποίο αποθηκεύει όλα τα στοιχεία από το παλιό σύνολο δεδομένων, τα οποία δεν ανήκουν στο νέο σύνολο δεδομένων, και άρα πρέπει να αφαιρεθούν.

Το Σχήμα 2.1 παρουσιάζει τα παραπάνω τρία σύνολα, δίνοντας στο παλιό σύνολο το όνομα 'Selection' και στο νέο το όνομα 'Data'.



Σχήμα 2.1: Τα σύνολα δεδομένων που παράγει η d3

Η λειτουργικότητα αυτή, με μία πρώτη ματιά ίσως δε φαίνεται ιδιαίτερα χρήσιμη. Ωστόσο, αυτή η δυνατότητα της d3 μας δίνει πρακτικά ένα δυναμικό τρόπο ανανέωσης δεδομένων, με την επιλογή να σβήσουμε όσα στοιχεία δε θέλουμε πλέον, τοποθετώντας τα στο σύνολο εξόδου, ενώ ταυτόχρονα προσθέτουμε νέα στοιχεία, τοποθετώντας τα στο σύνολο εισόδου. Επίσης, προσφέρεται η δυνατότητα να εκμεταλλευτούμε τα στοιχεία προς διαγραφή, πιθανώς αποθηκεύοντας κάποιες τιμές τους. Αυτές φυσικά είναι ελάχιστες από τις πολλές δυνατότητες χρήσης της παραπάνω λειτουργικότητας [16].

Η d3 υποστηρίζεται από όλους τους μοντέρνους browser, αλλά κάποιες λειτουργίες αποτυγχάνουν σε παλαιότερους browser. Κάποιες πληροφορίες που δεν αναφέρθηκαν, είναι αρχικά ότι η απόκρυψη δεδομένων, με σκοπό να γίνεται περιορισμένη/διαφορετική πρόσβαση μεταξύ των διάφορων χρηστών, είναι δύσκολο να επιτευχθεί στη d3 [17]. Επίσης, το γεγονός ότι η d3 βασίζεται στην SVG για την οπτικοποίηση, είναι πολύ θετικό, διότι η SVG είναι η καθορισμένη προδιαγραφή του 'World Wide Web Consortium' για διανυσματικά γραφικά. Η χρήση της SVG για οπτικοποίηση έχει αρχίσει να γίνεται τάση στις διαδικτυακές εφαρμογές [18].

Μία γενική ιδέα της βιβλιοθήκης 'd3.js' παρουσιάστηκε σε αυτό το υποκεφάλαιο, σε θεωρητικό επίπεδο. στο επόμενο υποκεφάλαιο θα παρουσιαστούν πρακτικά παραδείγματα για τη καλύτερη κατανόηση της λειτουργίας της.

2.3 Πρακτικά παραδείγματα της d3

Ξεκινώντας, θα αναφερθούν οι πιο σημαντικές συναρτήσεις που χρησιμοποιούνται στη d3, ανά τύπους. Σε παρένθεση αναφέρονται οι τιμές/μεταβλητές εισόδου της κάθε συνάρτησης [16].

- **Επιλογή στοιχείων:**
  - **.select(selector|node)**
  - **.selectAll(selector|nodes)**
  - **.filter(selector)**
  - **.sort(comparator)**

- **Τροποποίηση περιεχομένου:**
  - **.attr(name[, value])**
  - **.style(name[, value[, priority]])**
  - **.property(name[, value])**
  - **.text([value])**
  - **.append(name)**
  - **.insert(name[, before])**
  - **.remove()**

- **Δέσμευση δεδομένων:**
  - **.data([values[, key]])**
  - **.enter()**
  - **.exit()**

- **Animations και αλληλεπίδραση:**
  - **.on(type[, listener[, capture]])**
  - **.transition()**

- **Έλεγχος ροής:**
  - **.each(function)**
  - **.call(function[, arguments…])**

Όπως μπορεί κανείς εύκολα να καταλάβει από τη πληθώρα των διαφορετικών τύπων συναρτήσεων, αλλά και το πλήθος αυτών, οι δυνατότητες της d3 είναι πάρα πολλές. Στη συνέχεια θα παρουσιαστούν κάποιες από τις παραπάνω εντολές στη πράξη, με επεξηγήσεις για τον τρόπο λειτουργίας τους.

Η πρώτη εντολή που θα παρουσιαστεί, είναι η αυτή της επιλογής στοιχείων της ιστοσελίδας μέσω της d3, δηλαδή η εντολή ‘select’. Μία απλή χρήση της φαίνεται παρακάτω:

```
d3.select('body')
    .append('h1')
    .text('Hello World!');
```

Η συγκεκριμένη εντολή, επιλέγει το στοιχείο ‘body’ της ιστοσελίδας, και προσάπτει σε αυτό ένα νέο στοιχείο ‘h1’, το οποίο περιέχει το κείμενο ‘Hello World’ [15]. Η εντολή ‘select’ όμως, επιλέγει ένα μόνο στοιχείο, και το πρώτο σε περίπτωση πολλαπλών αντιστοιχίσεων. Για να επιλέξουμε περισσότερα στοιχεία, χρειαζόμαστε την εντολή ‘selectAll’:

```
d3.selectAll('p')
    .attr('align', 'center');
```

Χρησιμοποιώντας τη ‘selectAll’, επιλέγουμε όλα τα στοιχεία ‘p’ της ιστοσελίδας μας. Επίσης, με την εντολή ‘attr’, δίνουμε ιδιότητες στα επιλεγμένα στοιχεία. Σε αυτή τη περίπτωση, δόθηκε η εντολή η στοίχιση να είναι στο κέντρο. Θα παρουσιαστούν και άλλες εντολές με τις ‘selectAll’ και ‘select’ και άλλων εντολών σε συνδυασμό με αυτές, για τη καλύτερη κατανόηση της χρήσης τους.

```
d3.selectAll('.item')
    .style('background-color', 'red');
```

Με το ‘.item’ ως είσοδο στην εντολή ‘selectAll’, επιλέγονται όλα τα στοιχεία που ανήκουν στη κλάση ‘item’. Στη συνέχεια, με την εντολή ‘style’ προσθέτουμε ή αλλάζουμε ήδη υπάρχουσες ιδιότητες της CSS στα συγκεκριμένα αντικείμενα.

Εδώ αξίζει να αναφερθεί ότι οι εντολές όπως η ‘attr’ και η ‘style’, έχουν 2 τρόπους λειτουργίας. Ένας τρόπος είναι όπως παρουσιάστηκαν, δηλαδή με δύο τιμές ως εισόδου, όπου η πρώτη αναφέρεται στην ιδιότητα που θα μεταβληθεί, και η δεύτερη στη τιμή που θα πάρει. Ο άλλος τρόπος λειτουργίας τους, είναι με είσοδο μία μόνο τιμή. Η τιμή αυτή αναφέρεται σε μία ήδη υπάρχουσα ιδιότητα, της οποίας η τιμή επιστρέφεται ως έξοδος: Για παράδειγμα, στον παραπάνω κώδικα, μετά την ανάθεση της τιμής ‘background-color’, είναι εφικτή η ανάκτηση της τιμής της, με τον παρακάτω κώδικα:

```
d3.select('.item').style('background-color');
```

Παρακάτω, επιλέγεται το στοιχείο με id ‘first-item’, και με την εντολή ‘remove’ γίνεται διαγραφή του [16].

```
d3.select('#first-item')
    .remove();
```

Αξιοσημείωτη λεπτομέρεια είναι, ότι δεν υπάρχει όριο στο πόσες εντολές θα εκτελεστούν μαζί σε ένα ‘selection’ της d3, αρκεί να χωρίζονται με τελεία μεταξύ τους. Φυσικά μπορούν να εκτελεστούν και δύο ή παραπάνω ίδιες εντολές (π.χ. ‘attr’) [19]:

```
d3.select("body")
    .append("svg")
    .attr("width", 400)
    .attr("height", 200)
```

```
        .style("background-color", "purple");
```

Υπάρχει και η δυνατότητα χρήσης μίας συνάρτησης για τη παραγωγή τιμών σε εντολές όπως η ‘attr’ και η ‘style’. Ας υποθέσουμε μία σελίδα με τα παρακάτω στοιχεία:

```
<div id='div1'>A</div>
<div id='div2'>B</div>
<div id='div3'>C</div>
<div id='div4'>D</div>
```

Εάν θέλουμε να κάνουμε μεταβολές στην εμφάνιση ή τις ιδιότητες των στοιχειών αυτών, είναι εφικτό με το ‘selectAll’ της d3. Εάν όμως θέλουμε να τους δώσουμε διάφορες τιμές στις ιδιότητες και όχι τις ίδιες; Τότε θα χρησιμοποιήσουμε μία συνάρτηση (‘function’):

```
d3.selectAll("div")
        .style('background-color', function (d, i) {
            return (i % 2 === 0) ? "lightblue" : "lightgray";
        });
```

Με τη χρήση της παραπάνω εντολής, τα ‘div’ παίρνουν τις τιμές ‘lightblue’ και ‘lightgray’ εναλλάξ, ανάλογα με τη θέση τους στον πίνακα που δημιουργείται από τη ‘selectAll’ (την οποία ανακτάμε από τη μεταβλητή ‘i’ στη συνάρτηση). Το οπτικό αποτέλεσμα της εντολής παρουσιάζεται στην Εικόνα 2.1.

Εικόνα 2.1: Χρήση συνάρτησης για απόδοση ιδιοτήτων στη d3

Οι εντολές ‘select’ και ‘selectAll’, μπορούν να χρησιμοποιηθούν και μαζί, στην ίδια εντολή. Αυτό προσφέρει πολλές δυνατότητες επίσης, καθώς κάνει την επιλογή στοιχείων που βρίσκονται πιο χαμηλά στην ιεραρχία, αρκετά εύκολη:

```
d3.select('body')
        .selectAll('div');
```

Αν και πρόκειται για ένα αρκετά απλό παράδειγμα, όπου επιλέγουμε όλα τα ‘div’ από το στοιχείο ‘body’, οι δυνατότητες αυτού του συνδυασμού μας δίνει ουσιαστικά τον τρόπο να επιλέξουμε οποιοδήποτε στοιχείο θέλουμε, ακόμα κι αν δεν έχει ‘id’ ή κλάση [15].

Μία πολύ χρήσιμη εντολή είναι η ‘.each()’. Η συγκεκριμένη εντολή, εκτελεί μία συνάρτηση για κάθε στοιχείο που έχει επιλεγεί. Στο παρακάτω κώδικα, επιλέγονται όλα τα στοιχεία ‘p’, και εμφανίζεται η ιδιότητα ‘text’, για το καθένα απ’ αυτά [16].

```
d3.selectAll('p').each(function(d, i){
        var self = d3.select(this);
        console.log(self.text());
});
```

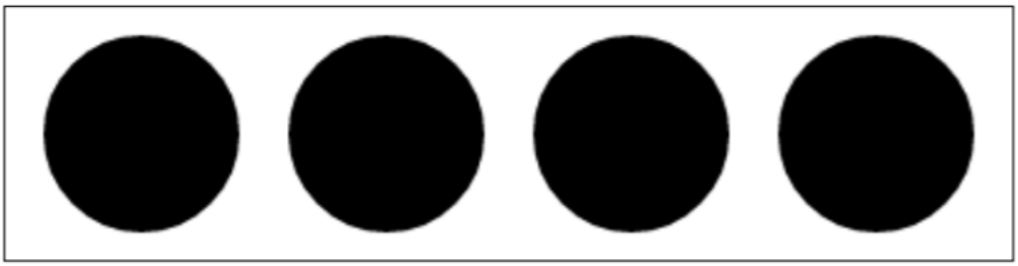
Με αυτές τις εντολές θα κλείσει αυτό το υποκεφάλαιο, που είχε ως στόχο να εισάγει τον αναγνώστη στη λογική της λειτουργίας της d3, παρουσιάζοντας τις κυριότερες συναρτήσεις που χρησιμοποιούνται σε αυτή. Στο επόμενο υποκεφάλαιο, θα παρουσιαστούν τα σύνολα δεδομένων της d3 (δηλαδή τα 'enter set', 'exit set' και 'update set'), αυτή τη φορά στη πράξη.

2.4 Τα σύνολα δεδομένων στη πράξη

Η d3 παρέχει τη συνάρτηση 'data([values[, key]])', για τη δέσμευση ενός πίνακα αυθαίρετων δεδομένων σε ένα 'selection'. Τρέχοντας αυτή τη συνάρτηση, θα επιστραφεί ένα νέο 'selection', που έχει αποθηκευμένα εσωτερικά τα δεδομένα δέσμευσης, και έχει δεσμεύσει κάθε στοιχείο των δεδομένων δέσμευσης με ένα στοιχείο του 'selection'. Η χρήση της παραμέτρου 'key' είναι προαιρετική και χρησιμοποιείται όταν θέλουμε να προσθέσουμε ένα αναγνωριστικό σε κάθε στοιχείο του πίνακα αυθαίρετων δεδομένων. Παρακάτω παρουσιάζεται ένα απλό παράδειγμα χρήσης της συνάρτησης.

```
<svg width="400" height="200">
        <circle cx="50" cy="50" r="40" />
        <circle cx="150" cy="50" r="40" />
        <circle cx="250" cy="50" r="40" />
        <circle cx="350" cy="50" r="40" />
</svg>
<script type="text/javascript">
        // Create a data array
        var data = [10, 20, 30, 40];
        // Bind data array to the Selection
        var circles = d3.selectAll('circle').data(data);
        console.log(circles.data());
        // [10, 20, 30, 40]
</script>
```

Η λογική της χρήσης της συνάρτησης 'data' είναι αρκετά απλή, αλλά μπορεί να προσφέρει πολύτιμες δυνατότητες. Θα παρουσιαστεί ένα παράδειγμα που επεκτείνει το παραπάνω, αλλά πρώτα θα παρουσιαστούν γραφικά οι παραπάνω κύκλοι, για τη κατανόηση του τελικού αποτελέσματος. Οι κύκλοι του παραπάνω παραδείγματος, παρουσιάζονται γραφικά στην Εικόνα 2.2.

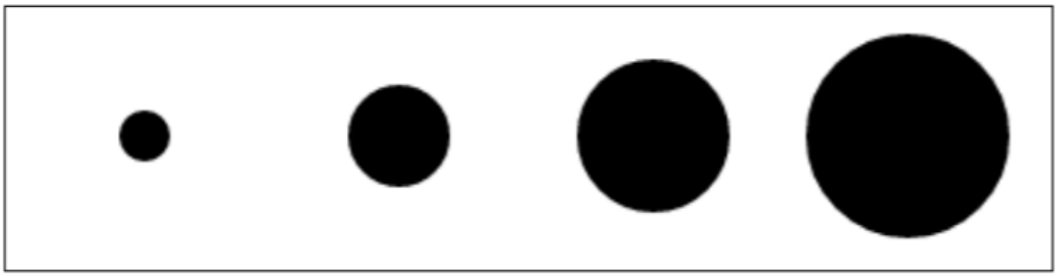


Εικόνα 2.2: Παράδειγμα των κύκλων στο παραπάνω παράδειγμα

Η προσθήκη μίας μόνο εντολής στον παραπάνω κώδικα, η οποία αλλάζει την ιδιότητα ‘r’ για τον κάθε κύκλο, ανάλογα με την δεσμευμένη τιμή στο κάθε στοιχείο, από την εντολή ‘data’, μπορεί να αλλάξει εντελώς το τελικό αποτέλεσμα. Η εντολή παρουσιάζεται παρακάτω.

```
circles.attr('r', function(d, i) { return d; });
```

Η γραφική αναπαράσταση των κύκλων μετά τη προσθήκη της εντολής αυτής, παρουσιάζεται στην Εικόνα 2.3.



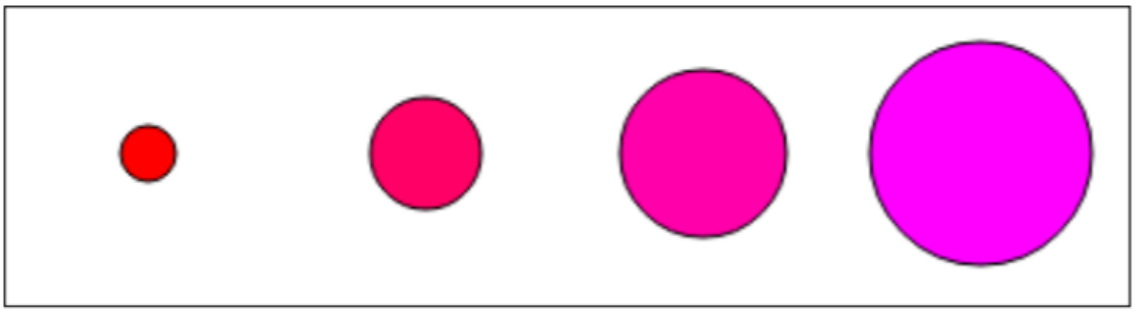
Εικόνα 2.3: Παράδειγμα των κύκλων μετά τη αλλαγή της ιδιότητας ‘r’

Και η λειτουργικότητα της ‘data’ δε σταματά εδώ. Υπάρχει δυνατότητα ο πίνακας ‘data’ να περιέχει πίνακες ιδιοτήτων ως στοιχεία. Αυτές οι ιδιότητες θα είναι προσβάσιμες από το τρέχον ‘selection’, μετά τη δέσμευση τους, απλά καλώντας τες με το όνομα τους. Ένα παράδειγμα παρουσιάζεται παρακάτω.

```
<svg width="400" height="200">
        <circle /><circle /><circle /><circle />
</svg>
<script type="text/javascript">
        // Create a data array containing objects
        var data = [ {cx:50, cy:50, r: 10, color: '#ff0000'},
        {cx:150, cy:50, r: 20, color: '#ff0066'},
        {cx:250, cy:50, r: 30, color: '#ff00aa'},
        {cx:350, cy:50, r: 40, color: '#ff00ff'} ];
```

```
        // Bind data array to the Selection
        var circles = d3.selectAll('circle').data(data);
        // Use dynamic properties for the radius
        circles.attr('r', function(d, i) { return d.r; });
        circles
        // Set the stroke color to black
        .attr('stroke', 'black')
        // Set the fill color depending of the bound object
        .attr('fill', function(d, i) { return d.color; })
        // Set the x coordinate of the center depending of the bound object
        .attr('cx', function(d, i) { return d.cx; })
        // Set the y coordinate of the center depending of the bound object
        .attr('cy', function(d, i) { return d.cy; });
    </script>
```

Ουσιαστικά, όλο το ‘χτίσιμο’ των κύκλων έγινε μέσα στον κώδικα JavaScript, μέσω της d3, χάρη στην ιδιότητα της δέσμευσης των ‘data’ στα επιλεγόμενα στοιχεία. Η γραφική αναπαράσταση των κύκλων μετά το παραπάνω παράδειγμα, παρουσιάζεται στην Εικόνα 2.4.



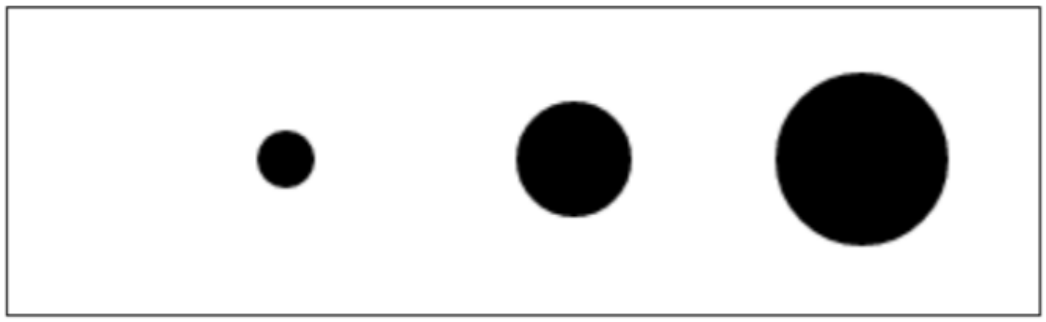
Εικόνα 2.4: Παράδειγμα των κύκλων που ΄χτίστηκαν’ μέσω των ΄data’

Όταν δεσμεύουμε δεδομένα στο ‘selection’ με τη συνάρτηση ‘data’, η d3 παρέχει τις συναρτήσεις ‘.enter()’ και ‘.exit()’, οι οποίες επιστρέφουν τα ‘enter set’ και ‘exit set’ αντίστοιχα. Το ‘update set’, από την άλλη, επιστρέφεται ουσιαστικά με τη κλήση του ίδιου του ‘selection’.

```
    <svg width="800" height="500"></svg>
    <script type="text/javascript">
        var svg = d3.select('svg');
        // Get an empty Selection of all circle elements
        // and bind a data array to the selection
```

```
var circles = svg.selectAll('circles').data([10, 20, 30]);
// Get the enter Selection and append circle elements
circles.enter().append('circle');
// The circle elements have been merged to the Selection
circles.attr('cx', function(d, i) { return (i+1)*100; }).attr('cy', 50)
.attr('r', function(d) { return d; });
</script>
```

Το παραπάνω παράδειγμα, αρχικά κάνουμε ένα ‘selection’ το οποίο δε περιέχει κύκλους. Στη συνέχεια, δεσμεύουμε τον πίνακα δεδομένων ‘[10, 20, 30]’ στο κενό από κύκλους ‘selection’. Στη συνέχεια, η εντολή ‘enter()’ επιστρέφει όλο το σύνολο δεδομένων, καθώς κανένα από αυτά δεν ανήκει στο τρέχον ‘selection’. Οπότε, στη συνέχεια, η εντολή ‘append’ οδηγεί στη δημιουργία ενός κύκλου για κάθε στοιχείο δεδομένων, κάνοντας τους κύκλους αυτούς μέρος, πλέον, του ‘selection’. Το γραφικό αποτέλεσμα παρουσιάζεται στην Εικόνα 2.5.

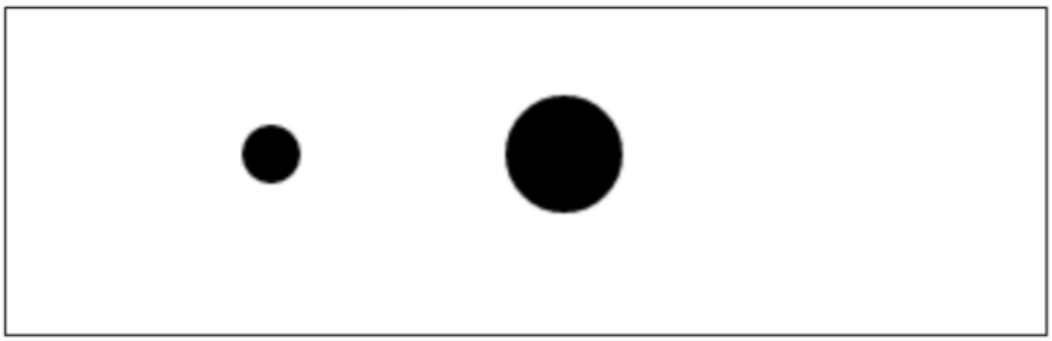


Εικόνα 2.5: Παράδειγμα των κύκλων που δημιουργήθηκαν μέσω του ‘enter set’

Μετά τη χρήση της εντολής ‘enter()’, τα στοιχεία θα συγχωνευθούν με το ‘update set’, και το ‘enter set’ θα εκκαθαριστεί αυτόματα. Ο επόμενος συνδυασμός εντολών που θα παρουσιαστεί, μετά τον ‘enter() - append()’, είναι ο ‘exit() – remove()’.

```
// Remove element 30 from the bound data
circles.data([10, 20])
        .exit().remove();
```

Ο παραπάνω κώδικας, δίνει ως δεδομένα, ένα πίνακα χωρίς το στοιχείο ‘30’, που χρησιμοποιήθηκε για τη δημιουργία κύκλου που πλέον υπάρχει στο DOM. Οπότε, η εντολή ‘exit()’ επιστρέφει ακριβώς αυτό το στοιχείο, και έτσι μπορούμε με την εντολή ‘remove()’ να το αφαιρέσουμε από το DOM. Το αποτέλεσμα του παραδείγματος παρουσιάζεται στην Εικόνα 2.6 [16].



Εικόνα 2.6: Οι κύκλοι του προηγούμενου παραδείγματος μετά την αφαίρεση ενός μέσω του ‘exit set’

Με αυτά τα παραδείγματα θα κλείσει και το κεφάλαιο της βιβλιοθήκης d3, καθώς θα γίνει περαιτέρω επεξήγηση εννοιών και εντολών στο τελευταίο κεφάλαιο, όπου θα παρουσιαστεί η πρακτική υλοποίηση.

2.5 Επίλογος

Σε αυτό το κεφάλαιο παρουσιάστηκε η βιβλιοθήκη ‘d3.js’. Αρχικά, έγινε μία εισαγωγή σε αυτή. Αναφέρθηκαν οι δυνατότητες, τα πλεονεκτήματα, καθώς και ο τρόπος λειτουργίας της, που περιλαμβάνει τα πολύ χρήσιμα ‘set’ δεδομένων. Στη συνέχεια, για την εμπέδωση των παρουσιαζόμενων εννοιών, παρουσιάστηκαν αρκετές πρακτικές περιπτώσεις χρήσης της βιβλιοθήκης, για τις κυριότερες συναρτήσεις που χρησιμοποιούνται, αλλά και για την χειραγώγηση των ‘set’ που δημιουργεί η d3.

Κεφάλαιο 3ο: Το GNU Mailman

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλυθεί το λογισμικό ‘GNU Mailman’, στο οποίο λειτουργούν και τα mailing lists της σχολής Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων, και με το οποίο ήταν απαραίτητη η εξοικείωση για την επίτευξη του στόχου αυτής της Πτυχιακής Εργασίας. Αρχικά, θα γίνει μία επεξήγηση του όρου ‘mailing list’. Τα δύο επόμενα υποκεφάλαια θα ασχοληθούν με το να γίνει μία καλή εισαγωγή στο Mailman και στη συνέχεια με τη περιγραφή της αρχιτεκτονικής του. Τελικά, στο τελευταίο υποκεφάλαιο θα παρουσιαστεί το Mailman του τμήματος μας.

3.2 Οι mailing lists

Σε αυτό το κεφάλαιο θα γίνει μία περιγραφή του όρου ‘mailing lists’, η οποία είναι απαραίτητη για τη περιγραφή του λογισμικού ‘GNU Mailman’, η οποία θα ακολουθήσει.

Οι ‘mailing lists’ χρησιμοποιούνται για τη διάδοση πληροφοριών εντός συγκεκριμένων ομάδων, όπως μία εργαστηριακή ομάδα, μία ομάδα που ασχολείται με ένα έργο, αλλά και απλώς για ομάδες χρηστών με κοινά ενδιαφέροντα. Τα μέλη μίας ‘mailing list’ μοιράζονται μία κοινή διεύθυνση αλληλογραφίας. Όταν κάποιος στέλνει ένα mail στη συγκεκριμένη διεύθυνση, το mail αυτό προωθείται αυτομάτως σε όλα τα μέλη της ‘mailing list’ [20].

Δύο κοινοί τύποι ‘mailing lists’ είναι οι λίστες ανακοινώσεων και οι λίστες συζήτησης. Οι λίστες ανακοινώσεων χρησιμοποιούνται με σκοπό ένα άτομο ή μία ομάδα να αποστέλλει ανακοινώσεις σε μία ομάδα ατόμων. Δύο παραδείγματα χρήσης αυτού του τύπου, είναι ένας εκδότης περιοδικού που θέλει να αποστείλει το περιοδικό στο κοινό, καθώς και μία μπάντα που θέλει να ανακοινώσει στο κοινό της τις επόμενες συναυλίες της.

Από την άλλη, οι λίστες συζήτησης επιτρέπουν μία ομάδα ανθρώπων να συζητήσουν θέματα μεταξύ τους, με τον καθένα να είναι διαθέσιμος να αποστείλει ένα mail στη λίστα και παραδοθεί σε όλα τα μέλη της ομάδας. Αυτός ο τύπος είναι επίσης διαχειρίσιμος. Δηλαδή, μπορεί να επιλεγεί να αποστέλλονται μόνο συγκεκριμένα μηνύματα στην ομάδα, ή ακόμα και να επιτρέπεται μόνο σε συγκεκριμένα άτομα να αποστέλλουν μηνύματα σε αυτή. Ένα παράδειγμα τέτοιας λίστας μπορεί να είναι μία ομάδα καθηγητών μίας σχολής, οι οποίοι συζητάνε μέσω ενός ‘mailing list’ [21].

Κάποιοι χρήσιμοι όροι σχετικά με τις ‘mailing lists’, είναι οι παρακάτω [21]:

1. Τα άτομα που είναι μέρος μίας λίστας συχνά αποκαλούνται ‘μέλη’ ή ‘συνδρομητές’.

2. Ως ‘διαχειριστές λίστας’ αναφερόμαστε στα μέλη που είναι υπεύθυνα για τη συντήρηση της λίστας. Μία λίστα μπορεί να έχει έναν ή και περισσότερους διαχειριστές.

3. Κάποιες λίστες ίσως έχουν υπεύθυνους που διαβάζουν τα μηνύματα και αποφασίζουν εάν θα διαβιβαστούν ή όχι σε όλους τους συνδρομητές. Αυτοί ονομάζονται ‘συντονιστές λίστας’.

3.3 Εισαγωγή στο GNU Mailman

Το Mailman είναι ένα σύστημα διαχείρισης mailing lists, που χρησιμοποιείται για τον χειρισμό λιστών αναδιανομής email. Το Mailman προσφέρει μία ιστοσελίδα για κάθε mailing list, και επιτρέπει στους χρήστες να εγγραφούν, διαγραφούν κλπ. σε αυτή, μέσω του παγκόσμιου ιστού. Επίσης, και οι

διαχειριστές των λιστών, μπορούν να διαχειρίζονται τις λίστες τους εξ' ολοκλήρου μέσω του παγκόσμιου ιστού. Το Mailman παρέχει ενσωματωμένες όλες τις λειτουργίες που πραγματοποιούν συνήθως οι χρήστες, όπως αρχειοθέτηση και άλλα. Το Mailman αναπτύχθηκε αρχικά από τον 'John Vega'. Στη συνέχεια ανέλαβε ο 'Ken Manheimer', ο οποίος έφερε το Mailman στην έκδοση '1.0b3'. Πλέον, το Mailman είναι ένα ομαδικό έργο ανοιχτού λογισμικού, με ηγέτες τους 'John Viega', 'Ken Manheimer' και 'Barry Warsaw' [22]. Το λογότυπο του GNU Mailman παρουσιάζεται στην Εικόνα 3.1.

Εικόνα 3.1: Το λογότυπο του GNU Mailman

Το Mailman υλοποιείται κυρίως σε Python, μία αντικειμενοστραφή, πολύ υψηλού επιπέδου, ανοιχτού κώδικα γλώσσα προγραμματισμού. Οι διαχειριστές των ιστοσελίδων Mailman, μπορούν να αλληλεπιδρούν με αυτό μέσω μίας σειράς σεναρίων γραμμής εντολών ή ακόμα και μέσω μίας διαδραστικής προτροπής Python. Όταν κυκλοφόρησε για πρώτη φορά το Mailman, το 'python.org' το υιοθέτησε γρήγορα και έκτοτε το χρησιμοποιεί.

Το Mailman 2.0 σημείωσε ορόσημο στην ανάπτυξή του, καθώς η έκδοση 2.0.13 είναι αρκετά σταθερή και χρησιμοποιείται σε χιλιάδες ιστοσελίδες. Εκτελείται παντού, από μικρές λίστες ομάδων ειδικών ενδιαφερόντων έως τεράστιες λίστες ανακοινώσεων, από εμπορικές εφαρμογές (RedHat, SourceForge, Apple, Dell, SAP), κοινότητες hacker (Samba, Gnome, KDE, Exim, Python), έως και σε πολλά εκπαιδευτικά ιδρύματα και μη κερδοσκοπικούς οργανισμούς. Υπάρχουν πολλές εγκαταστάσεις φιλοξενίας που παρέχουν υπηρεσίες Mailman και ολοένα και περισσότεροι διεθνείς οργανισμοί [23].

Από τις πρώτα βήματα με το ARPAnet μέχρι και σήμερα, τα email και τα συστήματα διαχείρισης mailing lists, έχουν κεντρικό ρόλο στο σχηματισμό και στη συμπεριφορά των κοινοτήτων στο διαδίκτυο. Με τη πάροδο του χρόνου, η ταχείας κλίμακας άνοδος του διαδικτύου, καθώς και η έλευση νέων και βελτιωμένων στρατηγικών για την οργάνωση των κοινοτήτων, απαιτούν και συνεχή ανάπτυξη των μηχανισμών που τα υποστηρίζουν. Ένα καλό σύστημα διαχείρισης mailing lists θα συμβάλει στη προώθηση της εξέλιξης των κοινοτήτων του Διαδικτύου, με το να αναπτύσσεται μαζί τους.

Η επεκτασιμότητα προσφέρει επίσης μία εξαιρετική ευκαιρία για επαφή με τον πυρήνα των χρηστών των mailing lists, δηλαδή τους διαχειριστές λιστών. Αυτοί οι διαχειριστές τυπικά είναι αρκετά κοντά με τους τελικούς χρήστες, ώστε να έχουν καθαρή εικόνα των αναγκών τους. Επιπροσθέτως, συχνά είναι αρκετά οικείοι με τη τεχνολογία, ώστε να είναι ικανοί να εφαρμόζουν βελτιώσεις για να ικανοποιήσουν αυτές τις ανάγκες. Οπότε, αυτή είναι μία μοναδική ευκαιρία να αξιοποιηθεί η ανάπτυξη ανοιχτού λογισμικού, επιτρέποντας στους ίδιους τους διαχειριστές να καθοδηγήσουν την ανάπτυξή του. Αυτό επιτρέπει τη πιο γρήγορη ανάπτυξη του λογισμικού, η οποία είναι και στενά προσαρμοσμένη στις ανάγκες της κοινότητας των χρηστών [22].

Το Mailman επιδιώκει την τήρηση των προτύπων, και ως εκ τούτου είναι διαλειτουργικό με ένα ευρύ φάσμα web servers και browsers, καθώς και με mail servers και clients. Από τους web servers, απαιτεί

τη δυνατότητα εκτέλεσης CGI scripts, ενώ από τους mail servers απαιτεί τη δυνατότητα φιλτραρίσματος μηνυμάτων μέσω προγραμμάτων. Η HTML που έχει ως έξοδο το Mailman είναι εξαιρετικά πεζή, οπότε σχεδόν κάθε browser μπορεί να λειτουργεί με αυτό, αρκεί να υποστηρίζει τα cookies. Επίσης, το Mailman λειτουργεί σε οποιοδήποτε λειτουργικό σύστημα τύπου Unix, όπως το GNU / Linux [23].

Το Mailman υποστηρίζει ένα μεγάλο πλήθος χαρακτηριστικών, όπως [23]:

- **Τρόποι παράδοσης καθορισμένοι από τον χρήστη:**
  Τα μέλη μπορούν να επιλέξουν να λαμβάνουν τα μηνύματα άμεσα ή σε παρτίδες που ονομάζονται ‘σύνοψη’. Τα μηνύματα που δεν ανήκουν σε μία ‘σύνοψη’, μπορούν να εξατομικευτούν ειδικά για τον παραλήπτη του εκάστοτε μηνύματος. Αυτό σημαίνει ότι διάφορες πτυχές του μηνύματος, όπως η κεφαλίδα ή το υποσέλιδο μπορεί να περιέχουν συγκεκριμένες πληροφορίες για το μέλος που λαμβάνει το μήνυμα.

- **Εκτεταμένες επιλογές απορρήτου:**
  Οι επιλογές αυτές επιτρέπουν στους διαχειριστές λίστας να χρησιμοποιούν πολιτικές για εγγραφή και κατάργηση εγγραφής (π.χ. ανοιχτή, μετά από επιβεβαίωση, ή μετά από έγκριση), αλλά και για τη δημοσίευση μηνυμάτων στη λίστα (π.χ. ανοιχτή, εποπτευόμενη, μόνο από μέλη, ή μόνο από εγκεκριμένα μέλη). Επίσης, παρέχει ορισμένες περιορισμένες άμυνες για την αποφυγή των spam.

- **Αυτόματη ανίχνευση αναπήδησης:**
  Οι αναπηδήσεις διευθύνσεων είναι μία καταστροφή για κάθε mailing list και το Mailman παρέχει δύο μηχανισμούς για την αυτόματη ανίχνευση αναπήδησης (αναπήδηση είναι ουσιαστικά η μη παράδοση ενός mail στον παραλήπτη).

- **Αρχειοθέτηση:**
  Το Mailman έρχεται συνοδευόμενο από μία αρχειοθήκη που ονομάζεται ‘Pipermail’. Κύρια πλεονεκτήματα του ‘Pipermail’ είναι ότι έρχεται ομαδοποιημένη με το Mailman, ότι ‘τρέχει’ σε Python, και ότι στο Mailman 2.1 είναι διεθνοποιημένη, επιτρέποντας την εμφάνιση των μηνυμάτων σε εναλλακτικές γλώσσες και κωδικοποιήσεις χαρακτήρων. Τα κύρια μειονεκτήματά του είναι ότι δεν υποστηρίζει την αναζήτηση και δεν είναι πολύ προσαρμόσιμο. Πάντως, το Mailman ενσωματώνει εύκολα εξωτερικές αρχειοθήκες.

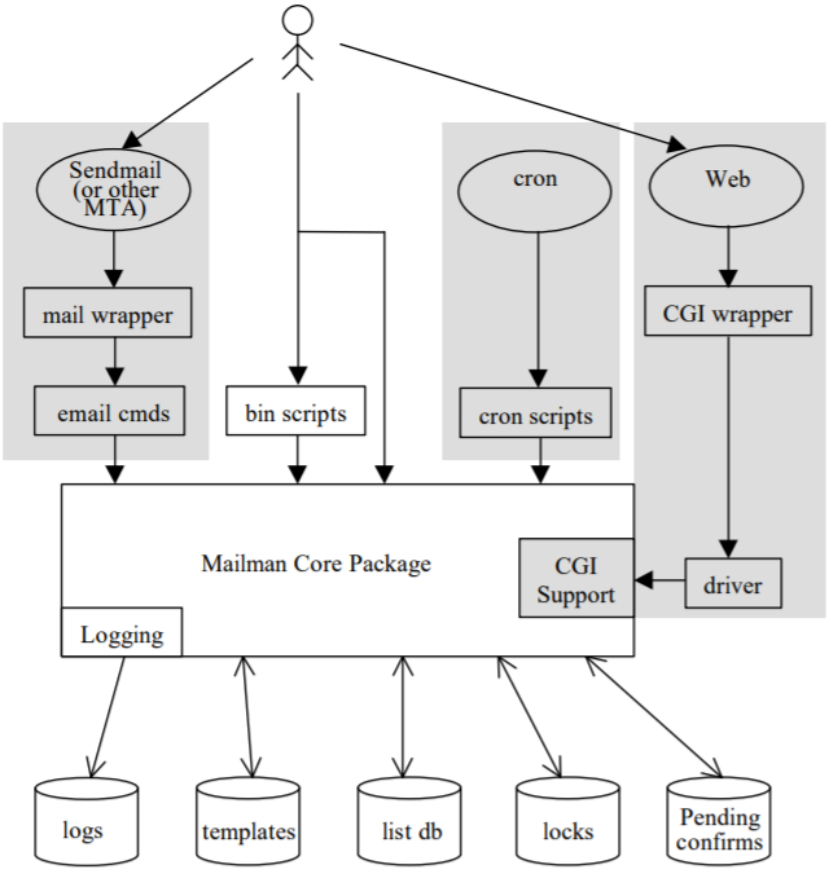
- **Αυτόματος ανταποκριτής, φιλτράρισμα περιεχομένου και θέματα:**
  Ο αυτόματος ανταποκριτής μπορεί να ρυθμιστεί για να στέλνει ένα μήνυμα κάθε φορά που κάποιος δημοσιεύει στη λίστα, ή στέλνει email στον ιδιοκτήτη της λίστας. Το φιλτράρισμα περιεχομένου επιτρέπει στον ιδιοκτήτη της λίστας να φιλτράρει ρητά ή να μεταβιβάζει συγκεκριμένους τύπους περιεχομένου. Τα θέματα επιτρέπουν στον ιδιοκτήτη της λίστας να εκχωρεί τα εισερχόμενα μηνύματα σε ομάδες. Ο αριθμός των ομάδων είναι διαμορφώσιμος, όπως και οι ίδιες. Τα μέλη μπορούν να ‘εγγραφούν’ σε ένα συγκεκριμένο θέμα, λαμβάνοντας μόνο το υποσύνολο της κίνησης της λίστας, που αντιστοιχεί στα επιθυμητά θέματα.

- **Εικονικά domains:**
  Το Mailman μπορεί να χρησιμοποιηθεί σε mail servers που υποστηρίζουν πολλά εικονικά domains. Για παράδειγμα, τα domain αλληλογραφίας των ‘python.org’ και ‘zope.org’ εκτελούνται στον ίδιο υπολογιστή, από την ίδια εγκατάσταση Mailman. Ο μόνος περιορισμός στο Mailman 2.1 είναι ότι μία λίστα με το ίδιο όνομα δε μπορεί να εμφανίζεται σε περισσότερα από ένα domain. Αυτός ο περιορισμός αναμένεται να αρθεί σε μελλοντικές εκδόσεις.

Γιατί όμως υλοποιήθηκε σε Python; Η Python είναι ιδιαίτερα κατάλληλη για την εφαρμογή ενός εκτεταμένου και μεταβαλλόμενου συστήματος. Ο συνδυασμός της καθαρής σύνταξης και σημασιολογίας βοηθά τον προγραμματιστή, πόσο μάλλον στη διαδικασία αλλαγής του υπάρχοντος κώδικα. Είναι κατ' εξοχήν δυναμική, επιτρέποντας την αλληλεπίδραση και τον προγραμματιστικό χειρισμό για σχεδόν τα πάντα στη συγκεκριμένη γλώσσα. Με την ικανοποίηση πρωτοτύπων και τις ανάγκες ταχείας ανάπτυξης, καθώς και εκείνες του γενικού προγραμματισμού, μπορεί να φανεί ότι ενισχύει τη συνεχή ανάπτυξη, όπου ένα σύστημα συνεχίζει να εξελίσσεται για να φιλοξενήσει έναν μεταβαλλόμενο κόσμο [22].

3.4 Η αρχιτεκτονική του Mailman

Το βασικό στοιχείο του Mailman είναι το αντικείμενο τύπου 'MailList', ένα στιγμιότυπο κλάσης που αντιπροσωπεύει μία μεμονωμένη λίστα αλληλογραφίας. Τα αντικείμενα MailList χρησιμοποιούνται σε scripts για να ληφθούν στοιχεία από τον web, τον cron (εντολή του Unix), και από άλλες πηγές, όπως την απευθείας εισαγωγή από τον χειριστή, ή έναν MTA (Mail Transfer Agent – Πράκτορας Μεταβιβάσεων mail). Λαμβάνοντας αυτές τις εισόδους μπορούν να μεταδίδουν μηνύματα μέσω του MTA, π.χ. δημοσιεύσεις σε λίστα, μηνύματα από τους διαχειριστές, αλλαγές σε βάσεις δεδομένων, ή απλά αλλαγές σε ρυθμίσεις χαρακτηριστικών των mailing lists. Το Σχήμα 3.1 παρουσιάζει τις διασυνδέσεις που αναφέρθηκαν. Τα στιγμιότυπα MailList ταυτίζονται με το 'Mailman Core Package' στο κέντρο.



Σχήμα 3.1: Οι διασυνδέσεις του Mailman

Το υποκεφάλαιο θα ασχοληθεί μόνο με τα αντικείμενα MailList, και όχι με τη γενική οργάνωση, η οποία παρουσιάζεται οπτικά για την επίγνωση της γενικής λειτουργίας του Mailman. Η τάξη MailList αποτελείται από πολλαπλές κληρονομικότητες από μια σειρά από κλάσεις προσανατολισμένες σε συγκεκριμένες εργασίες. Οι κλάσεις αυτές περιέχουν τις μεθόδους, τις δηλώσεις μεταβλητών και τις αρχικοποιήσεις που σχετίζονται με τη λειτουργικότητα ενός συγκεκριμένου υποσυστήματος. Παραδείγματα τέτοιων υποσυστημάτων είναι ο μηχανισμός παράδοσης και ο χειριστής εντολών.

Ο κώδικας της κλάσης MailList είναι υπεύθυνος για τον συντονισμό της αρχικοποίησης των υπόλοιπων κλάσεων (από τις οποίες έχει κληρονομήσει κώδικα), τον κεντρικό προσδιορισμό της συγκεκριμένης λίστας αλληλογραφίας, τη δημιουργία νέων λιστών αλληλογραφίας και τη διαχείριση των μόνιμων δεδομένων και του κλειδώματος των λιστών αλληλογραφίας. Ο εσωτερικός κώδικας ενός αντικειμένου MailList χειρίζεται επίσης το ανώτατο επίπεδο συνδρομών και δημοσίευσης μηνυμάτων, αλλά οι βασικές κλάσεις που είναι προσανατολισμένες σε συγκεκριμένες εργασίες, είναι υπεύθυνες για τα θεμέλια αυτών και όλων των άλλων λειτουργιών του αντικειμένου MailList. Σε αυτό το σημείο θα αναφερθούν οι βασικές αυτές κλάσεις και η εργασία της κάθε μίας.

- **MailCommandHandler**
  Αυτή η κλάση εφαρμόζει την ανάλυση και την εκτέλεση εντολών τύπου Majordomo που είναι ενσωματωμένες στα email για να ζητούν διευθύνσεις. Το Majordomo είναι ένας διαχειριστής λιστών ηλεκτρονικού ταχυδρομείου, που χρησιμοποιεί το Mailman. Παρά το γεγονός ότι οι χρήστες πιο τυπικά αλληλεπιδρούν με τις λίστες απευθείας μέσω της Web διεπαφής, για λόγους συμβατότητας, οι εντολές χρήστη μπορούν να εκδοθούν και μέσω email, χρησιμοποιώντας αυτή τη κλάση.

- **HTMLFormatter**
  Αυτή η κλάση χρησιμοποιείται για τη δημιουργία HTML για τη συγκεκριμένη λίστα, για την παρουσίαση μέσω της Web διεπαφής. Κατά κύριο λόγο, αυτή η κλάση χρησιμοποιεί μία βιβλιοθήκη widget (η οποία περιλαμβάνεται επίσης στο Mailman) για να πετύχει το στόχο της.

- **Deliverer:**
  Η κλάση αυτή πραγματοποιεί την παράδοση οποιουδήποτε email σχετίζεται με μία λίστα. Αυτό περιλαμβάνει την παράδοση των δημοσιεύσεων στους συνδρομητές, τις βεβαιώσεις εγγραφής, τις ανακοινώσεις στον διαχειριστή της λίστας σχετικά με τη δημιουργία λιστών, τη λίστα των λιστών που εκκρεμούν για έγκριση, τις ειδοποιήσεις συνδρομητών σχετικά με τους κωδικούς πρόσβασης και πολλά άλλα πράγματα. Το ηλεκτρονικό ταχυδρομείο χρησιμοποιείται για πολλά πράγματα από ένα σύστημα λιστών αλληλογραφίας, ακόμη και με μία ολοκληρωμένη Web διεπαφή.

- **ListAdmin**
  Αυτή η κλάση διαχειρίζεται την ουρά των αιτήσεων λίστας αλληλογραφίας, δηλαδή των αιτήσεων για δημοσιεύσεις ή για εγγραφές συνδρομητών, οι οποίες απαιτούν μία απόφαση από διαχειριστή (έγκριση ή απόρριψη). Επίσης είναι υπεύθυνη για την ειδοποίηση των διαχειριστών για τις αιτήσεις. Για παράδειγμα, μια λίστα μπορεί να ρυθμιστεί ώστε να απαιτεί την έγκριση διαχειριστή για οποιαδήποτε δημοσίευση, ή μπορεί μία δημοσίευση να μπει σε αναμονή λόγω ενεργοποίησης ενός trigger που προορίζεται να εμποδίσει ανεπιθύμητα μηνύματα.

- **Archiver**
  Η κλάση αυτή χειρίζεται την αρχειοθέτηση των αναρτημένων μηνυμάτων. Οι λίστες του Mailman μπορούν να έχουν δημόσιες ή ιδιωτικές αρχειοθετήσεις, και αυτή η κλάση τοποθετεί το δημοσιευμένο μήνυμα στην κατάλληλη τοποθεσία. Διασυνδέεται επίσης με εξωτερικούς αρχειοθέτες Hypertext, όπως το Pipermail (που αναφέρθηκε νωρίτερα), το οποίο συνδυάζεται με το Mailman.

- **Digester**
  Τα μέλη μίας λίστας αλληλογραφίας μπορούν να λαμβάνουν άμεσα τις δημοσιεύσεις που αναρτώνται, ή μπορούν να επιλέγουν να αποστέλλονται περιοδικά αθροιστικές 'χωνεύσεις' της κίνησης της λίστας. Αυτή η κλάση διαχειρίζεται τη συσσώρευση των 'χωνευμάτων', τη διαμόρφωση της μορφής του μηνύματος (απλό κείμενο ή κάποιον άλλο τύπο αρχείου, ανάλογα με την επιλογή του συνδρομητή), καθώς και την αποστολή των 'χωνευμάτων' στους αντίστοιχους συνδρομητές.

- **SecurityManager**
  Αυτή η κλάση, κατά κύριο λόγο, επαληθεύει τους κωδικούς πρόσβασης-εξουσιοδότησης για τον διαχειριστή του ιστότοπου, τους διαχειριστές λιστών και τους συνδρομητές. Εκτελεί επίσης το έργο της 'απολύμανσης' των κωδικών έγκρισης τύπου Majordomo από τις επικεφαλίδες των εγκρίσεων διαχειριστή που υποβάλλονται μέσω email.

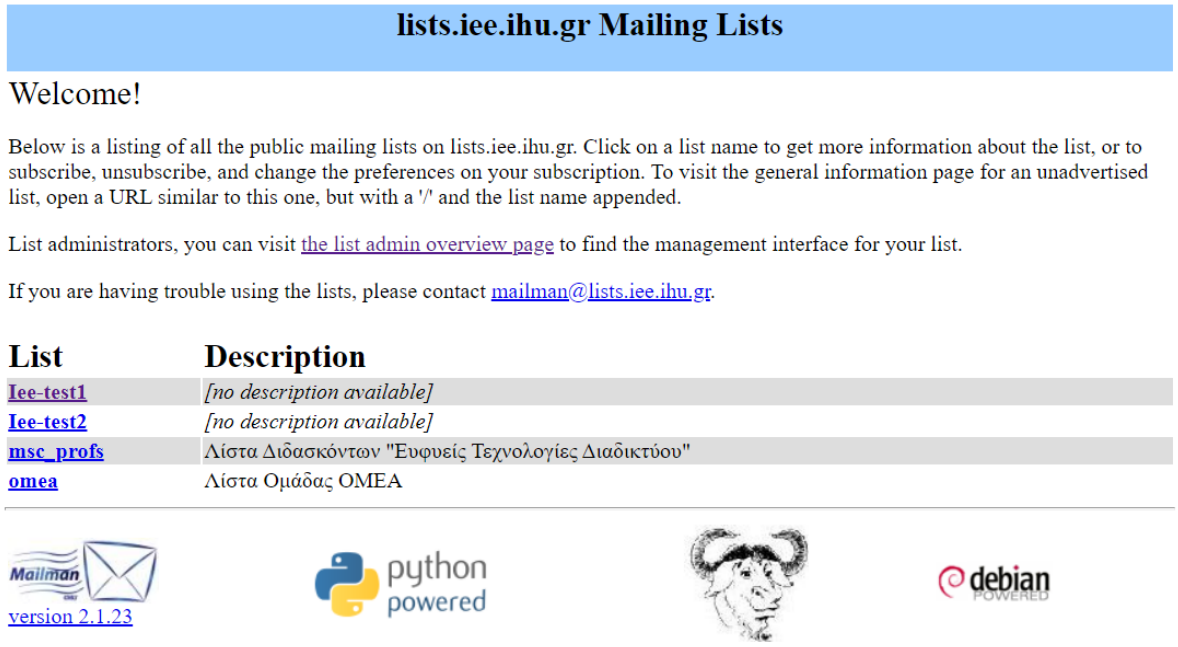
- **Bouncer**
  Το Mailman λαμβάνει ειδοποιήσεις αναπήδησης (δηλαδή αποτυχίας παράδοσης) για τις παραδόσεις των email και συγκεντρώνει καταμετρήσεις αναπήδησης, δίνοντας μία βαθμολογία σε όλα τα μέλη της λίστας αλληλογραφίας. Για σκορ που υπερβαίνουν κάποια καθορισμένα όρια εντός ενός καθορισμένου χρονικού ορίου, η κλάση αυτή πυροδοτεί τις ενέργειες που έχουν οριστεί στη συγκεκριμένη λίστα, συμπεριλαμβανομένης της απενεργοποίησης της παράδοσης αλληλογραφίας στο συγκεκριμένο μέλος ή, εάν έχει οριστεί από τον διαχειριστή της λίστας, ακόμη και κατάργηση της εγγραφής του μέλους από τη λίστα.

- **GatewayManager**
  Αυτή η κλάση χειρίζεται προαιρετικές πύλες email-to-Usenet για τις λίστες αλληλογραφίας. Το Usenet είναι παγκόσμιο κατανεμημένο σύστημα συζήτησης.

Οι παραπάνω κλάσεις αποτελούν τα θεμέλια του Mailman. Διαχειριζόμενες τις δικές τους κλειδαριές, τα στιγμιότυπα MailList μπορούν να χρησιμοποιηθούν ταυτόχρονα από πολλές διαδικασίες χωρίς συγκρούσεις. Προγράμματα που χειρίζονται στιγμιότυπα MailList εκτελούνται υπό CGI για Web διεπαφές, ή μέσω email προώθησης για εντολές, για διεπαφές email. Μια διαδραστική συνεδρία με στιγμιότυπα MailList παρέχει ένα εξαιρετικά χρήσιμο εργαλείο ανάπτυξης και εντοπισμού σφαλμάτων. Τα αντικείμενα MailList είναι εύκολο να αρχικοποιηθούν. Το μόνο που χρειάζεται είναι η συμπερίληψη του φακέλου του πακέτου Mailman στο PYTHONPATH και η γνώση της σωστής μονάδας για εισαγωγή. Με διαδραστική αρχικοποίηση, είμαστε σε θέση να δημιουργήσουμε και να δοκιμάσουμε μεμονωμένα υποσυστήματα, καθώς και τη συμπεριφορά της λίστας αλληλογραφίας στο σύνολό της, και μπορούμε επίσης να χρησιμοποιήσουμε διερευνητικά εργαλεία, όπως το πρόγραμμα εντοπισμού σφαλμάτων Python [22]. Στο επόμενο υποκεφάλαιο θα γίνει μία παρουσίαση της Web διεπαφής του Mailman του τμήματος μας.

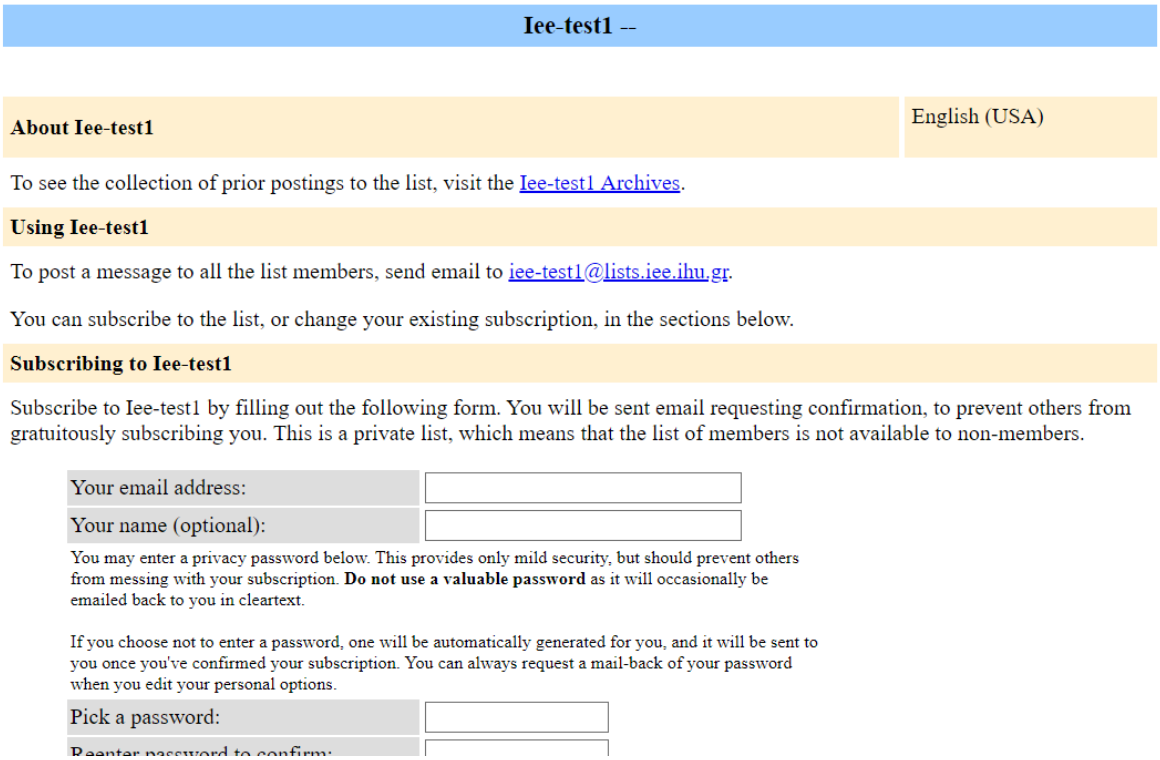
3.5 Το Mailman του τμήματος

Το Mailman του τμήματος μας τρέχει την έκδοση 2.1.23. Η αρχική σελίδα της Web διεπαφής παρουσιάζεται στην Εικόνα 3.2.

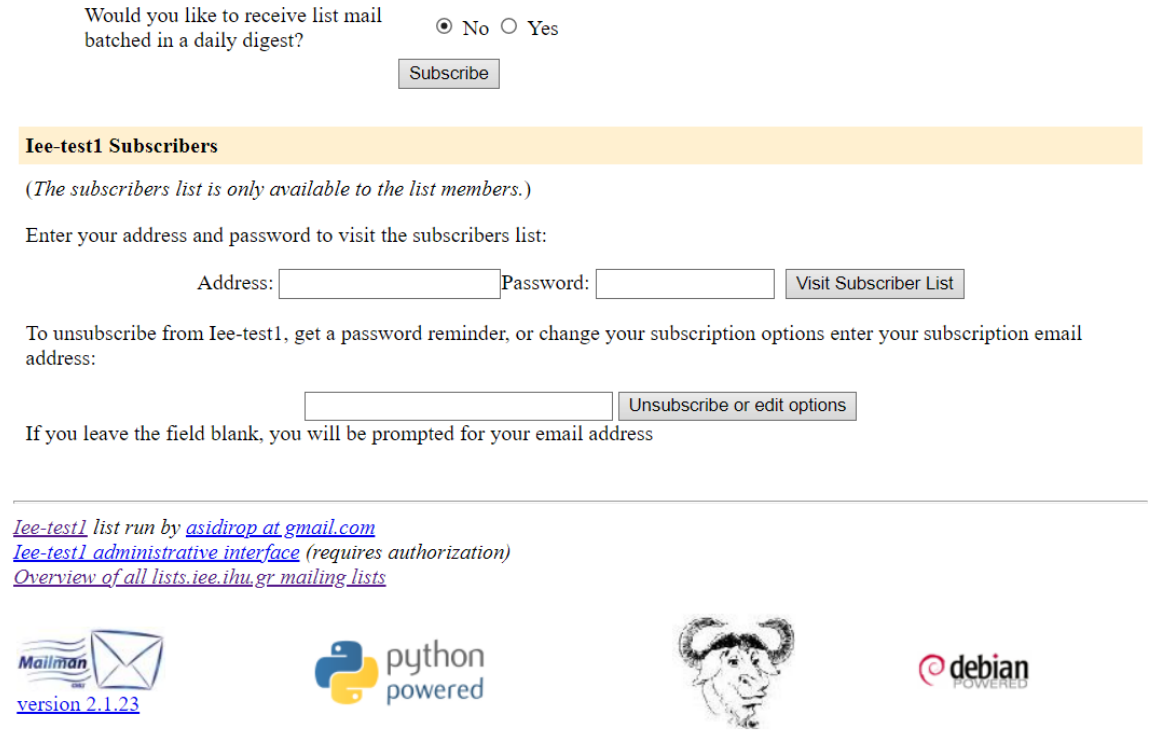


Εικόνα 3.2: Η αρχική σελίδα του Mailman του τμήματος

Όπως έχει αναφερθεί και παραπάνω, ένας λόγος της επιτυχίας του Mailman είναι η απλή του διεπαφή, η οποία μπορεί να ‘τρέξει’ ακόμη και στη πιο αδύναμη συσκευή. Είναι επίσης πολύ ξεκάθαρη όσο αφορά το περιεχόμενο της. Αρχικά, στο πάνω μέρος, μέσα στο μπλε πλαίσιο, αναφέρεται το γενικό link των λιστών (στη περίπτωση της σχολής μας είναι το ‘lists.iee.ihu.gr’). Ακολουθεί μία περιγραφή του περιεχομένου, το οποίο είναι ουσιαστικά οι δημόσιες λίστες του συγκεκριμένου Mailman, και φυσικά μία επεξήγηση των βασικών λειτουργιών και των link επικοινωνίας με την υποστήριξη. Ακολουθεί η λίστα των λιστών αλληλογραφίας της σχολής, με το όνομα και δίπλα μία περιγραφή για τα μέλη ή/και το περιεχόμενο της λίστας, εάν αυτή υπάρχει (παρατηρούμε πως μόνο 2 από τις 4 λίστες έχουν περιγραφή). Τέλος, στο κάτω μέρος της σελίδας, αναφέρεται η έκδοση του Mailman και εμφανίζονται τα λογότυπα κάποιων τεχνολογιών που χρησιμοποιήθηκαν, όπως η Python. Στη συνέχεια θα παρουσιαστεί η σελίδα μίας συγκεκριμένης λίστας αλληλογραφίας, που εμφανίζεται εάν αυτή επιλεχθεί από τη λίστα της αρχικής σελίδας με κλικ. Η σελίδα παρουσιάζεται στις Εικόνες 3.3 και 3.4.



Εικόνα 3.3: Η σελίδα μίας λίστας στο Mailman του τμήματος (1)



Εικόνα 3.4: Η σελίδα μίας λίστας στο Mailman του τμήματος (2)

Ξεκινώντας από την Εικόνα 3.3, αυτή το φορά στο μπλε πλαίσιο στο πάνω μέρος, εμφανίζεται το όνομα της συγκεκριμένης λίστας, η οποία σε αυτή τη περίπτωση είναι η 'iee-test1'. Στη συνέχεια υπάρχουν πολλές επιλογές για τη λίστα. Αρχικά, υπάρχει το πλαίσιο 'About iee-test1' από το οποίο μας δίνεται δυνατότητα πρόσβασης στα αρχειοθετημένα έγγραφα της λίστας. Στη συνέχεια, στο πλαίσιο 'Using iee-test1' εξηγείται πως για να σταλθεί ένα email στα μέλη της λίστας, αρκεί να το στείλουμε στη διεύθυνση 'iee-test1@lists.iee.ihu.gr'. Παρακάτω υπάρχει μία φόρμα εγγραφής στη λίστα, σε περίπτωση που θέλουμε να γίνουμε συνδρομητές. Επίσης αναφέρεται πως τα μέλη της λίστας δεν είναι εμφανή διότι η λίστα είναι ιδιωτική, και δεν επιτρέπεται η εμφάνιση των συνδρομητών, σε μη συνδρομητές. Περνώντας στην Εικόνα 3.4, υπάρχει μία προτροπή για εισαγωγή των στοιχείων για τα μέλη, τα οποία θέλουν να δουν τους συνδρομητές της λίστας. Επίσης υπάρχει επιλογή για κατάργηση εγγραφής ή επεξεργασία των ρυθμίσεων, για τα μέλη. Τέλος, πριν από τα λογότυπα των τεχνολογιών, αναφέρεται ο διαχειριστής της λίστας, δίνεται επιλογή για μετάβαση στη διεπαφή διαχειριστή (που προφανώς απαιτεί σύνδεση), και τελευταία επιλογή είναι η επιστροφή στην αρχική σελίδα του Mailman, που περιέχει την επισκόπηση όλων των λιστών.

Αυτή ήταν μία σύντομη περιγραφή της διεπαφής του Mailman. Η χρήση του είναι πολύ απλή και δεν υπάρχει λόγος να προβούμε σε περισσότερες λεπτομέρειες. Στο επόμενο υποκεφάλαιο ακολουθεί ο επίλογος και τα συμπεράσματα του κεφαλαίου αυτού.

3.6 Επίλογος

Η συνεχής δυναμική εξέλιξη των κοινοτήτων που εξυπηρετούνται από τα συστήματα διαχείρισης λίστας αλληλογραφίας υποδηλώνει ότι αυτά τα συστήματα είναι διαρκώς ημιτελή, με τουλάχιστον ορισμένες πτυχές να βρίσκονται συνεχώς υπό εξέλιξη. Η χρήση της Python προσφέρει τα πλεονεκτήματα της πρωτοτυπίας και της ταχείας εξέλιξης, τα οποία στη περίπτωση του Mailman είναι πολύτιμα. Το Mailman εκμεταλλεύεται πολλά από τα χαρακτηριστικά της Python, συμπεριλαμβανομένων των εγγενής προσανατολισμός αντικειμένου, πολλαπλή κληρονομικότητα, πολυμορφισμός, δομές ελέγχου υψηλού επιπέδου, όπως εξαιρέσεις, τα συμβατικά πρωτόκολλα, δυναμική πρόσβαση σε namespaces, και μία πληθώρα τυπικών βιβλιοθηκών [22].

Σε αυτό το κεφάλαιο αναλύθηκε το εργαλείο GNU Mailman. Αρχικά, έγινε μία επεξήγηση του όρου 'mailing lists', η οποία ήταν απαραίτητη για την εισαγωγή στο Mailman, η οποία ακολούθησε. Στην εισαγωγή, έγινε αναφορά στις τεχνολογίες που χρησιμοποιεί το Mailman, στους δημιουργούς και τις δυνατότητες του, αλλά και στα κυριότερα από τα πολλά χαρακτηριστικά που προσφέρει. Στη συνέχεια ακολούθησε μία ανάλυση της αρχιτεκτονικής του Mailman, όπου έγινε μία αναλυτική επεξήγηση του τρόπου λειτουργίας του στη Python, όπως το ποιες είναι οι κλάσεις θεμέλια, το τι εργασίες έχει αναλάβει η κάθε μία, και τελικά πως η κύρια κλάση MailList κληρονομεί όλες τις κλάσεις θεμέλια για την απλοποίηση της δημιουργίας των αντικειμένων και της διαχείρισης τους. Τελικά, έγινε μία σύντομη παρουσίαση της Web διεπαφής του Mailman του τμήματος μας, για τη καλύτερη κατανόηση όλων των ορισμών και λειτουργιών που αναφέρθηκαν.

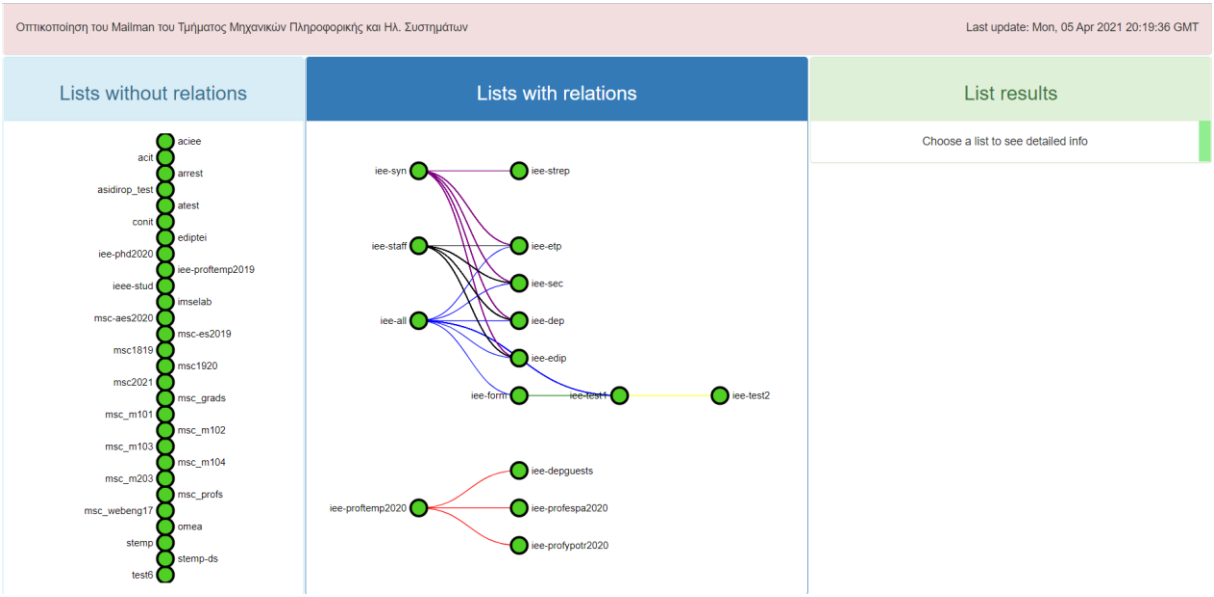
Κεφάλαιο 4ο: Η πρακτική εφαρμογή

4.1 Εισαγωγή

Αυτό το κεφάλαιο θα ασχοληθεί με τη πρακτική εφαρμογή η οποία υλοποιήθηκε στο πλαίσιο αυτής της Πτυχιακής Εργασίας. Η εφαρμογή αυτή αφορά την Οπτικοποίηση Λιστών Ηλεκτρονικού Ταχυδρομείου. Υλοποιήθηκε παίρνοντας τα δεδομένα από το Mailman του τμήματος μας, και χρησιμοποιώντας τις τεχνολογίες που παρουσιάστηκαν, δηλαδή την HTML5 σε συνδυασμό με τη JavaScript βιβλιοθήκη d3. Αρχικά, θα παρουσιαστεί οπτικά η τελική Web εφαρμογή. Στη συνέχεια θα αναλυθεί ο κώδικας του καθώς και όλα τα πρόσθετα που 'τρέχουν' πίσω από τη διεπαφή. Θα γίνει επεξήγηση σε πολλά κομμάτια του κώδικα, για τη καλύτερη κατανόηση του, αλλά και αναφορές σε εμπόδια που υπήρξαν και εν τέλει επιλύθηκαν, καταλήγοντας στο τελικό αποτέλεσμα.

4.2 Η διεπαφή της εφαρμογής

Σε αυτό το υποκεφάλαιο θα παρουσιαστεί η Web διεπαφή που δημιουργήθηκε και θα περιγραφούν αναλυτικά όλα τα μέρη της και οι δυνατότητες της. Η διεπαφή παρουσιάζεται στην Εικόνα 4.1.

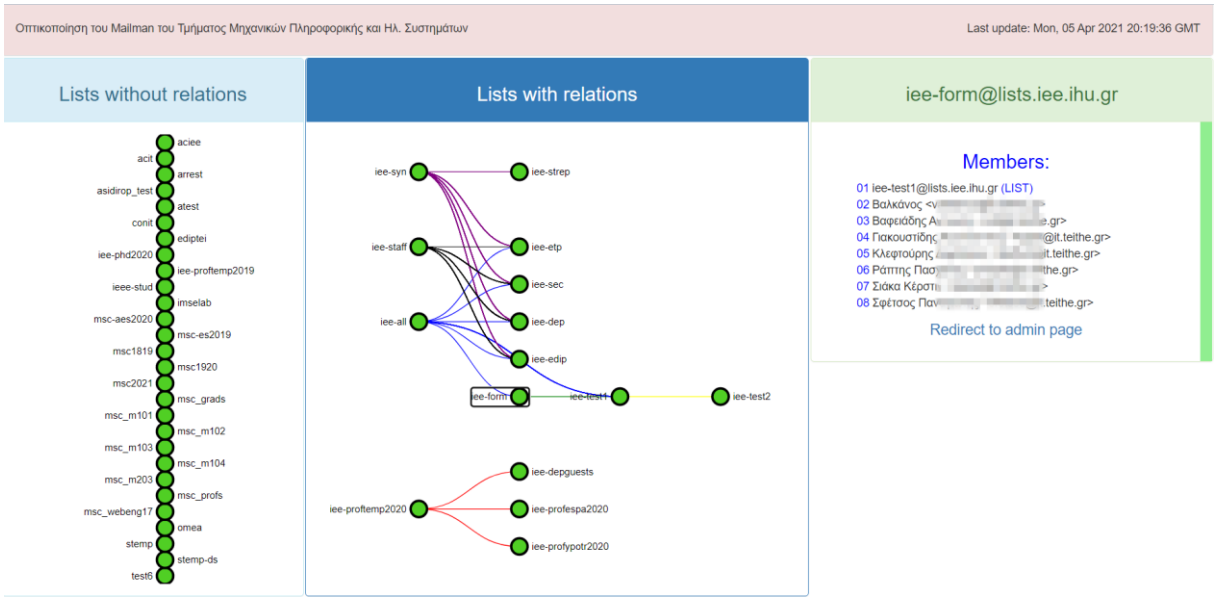


Εικόνα 4.1: Η αρχική σελίδα της διεπαφής

Στο επάνω μέρος της οθόνης, υπάρχει ένα πλαίσιο μέσα στο οποίο αναφέρεται το όνομα της εφαρμογής, στα αριστερά, καθώς και η ημερομηνία και ώρα της τελευταίας ενημέρωσης των δεδομένων από το Mailman του τμήματος, στα δεξιά. Παρακάτω, υπάρχουν τρία πλαίσια. Το πρώτο από αριστερά, περιέχει τις λίστες οι οποίες δεν έχουν συγγενικές σχέσεις με άλλες λίστες, άρα δε χρειάζονται κάποιο 'χτίσιμο' ιεραρχικής δομής και απλώς εμφανίζονται ως ανεξάρτητα στοιχεία, σε κάθετη διάταξη.

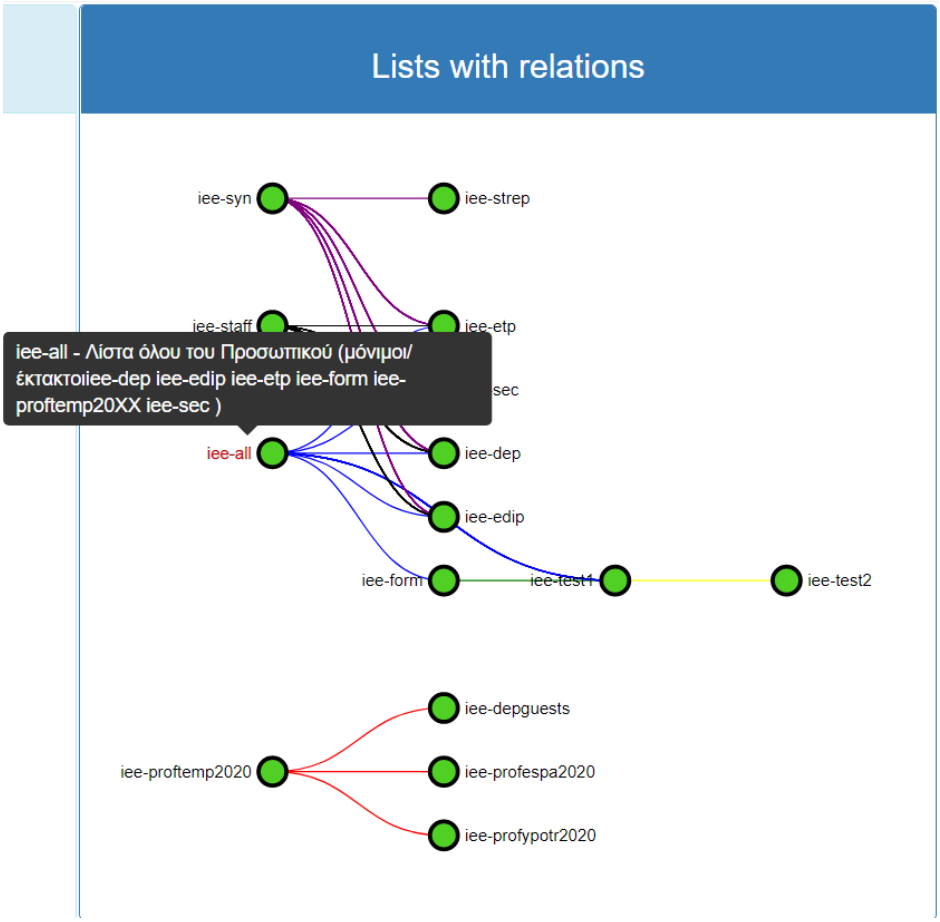
Το δεύτερο, περιέχει τις λίστες οι οποίες έχουν συγγενικές σχέσεις και τις αναπαριστά γραφικά, κατασκευάζοντας δέντρα ιεραρχίας, με διαφορετικά χρώματα στις γραμμές που ενώνουν τα στοιχεία, ανάλογα με το στοιχείο-γονέα. Όπως είναι προφανές, τα στοιχεία και οι σχέσεις αυτού του πλαισίου, όπως και τα στοιχεία του πρώτου, που δεν περιέχει σχέσεις, έχουν κατασκευαστεί με τη βοήθεια της βιβλιοθήκης d3.

Το τρίτο πλαίσιο, περιέχει τις λεπτομέρειες της επιλεγόμενης λίστας, δηλαδή τα μέλη της (εάν υπάρχουν), και ένα link ανακατεύθυνσης στη σελίδα διαχειριστή της λίστας. Σε αυτό το σημείο, το τρίτο πλαίσιο δεν περιέχει τίποτα από αυτά, καθώς καμία λίστα δεν είναι επιλεγμένη. Στην Εικόνα 4.2 παρουσιάζεται η ίδια σελίδα, με μόνη διαφορά ότι έχει επιλεχθεί η λίστα ‘iee-form’, για εμφάνιση αποτελεσμάτων.



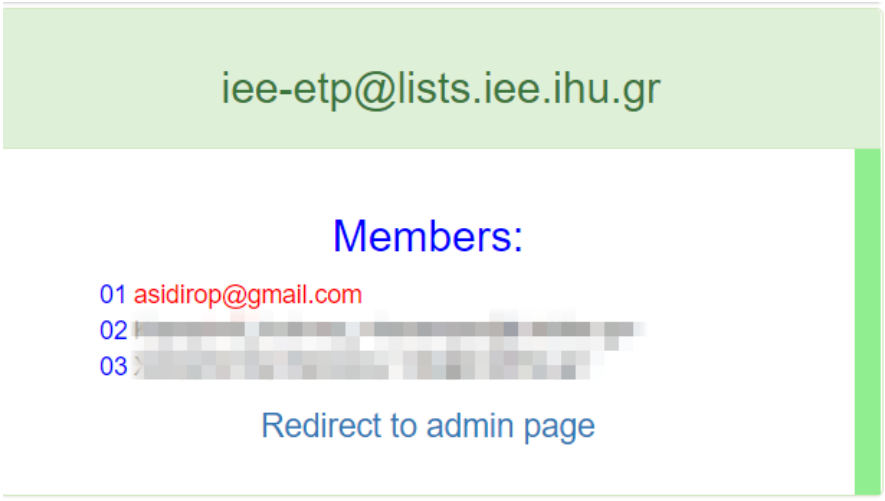
Εικόνα 4.2: Επιλογή της λίστας ‘iee-form’

Στη παραπάνω εικόνα, η επιλογή της λίστας ‘iee-form’ οδηγεί σε αλλαγή του κειμένου του τίτλου στο τρίτο πλαίσιο, με το όνομα της διεύθυνσης της λίστας (‘iee-form@lists.iee.ihu.gr’). Επίσης, εμφανίζονται τα μέλη της λίστας, η οποία επιλέχθηκε εσκεμμένα, διότι περιέχει ως μέλη και λίστα, αλλά και κανονικούς συνδρομητές. Όποιο από τα μέλη μίας λίστας είναι λίστα, έχει ένα μπλε κείμενο που αναγράφει ‘(LIST)’, για διευκρίνιση. Τα υπόλοιπα μέλη εμφανίζονται με το ονοματεπώνυμο τους (εάν υπάρχει) και το email τους στη συνέχεια. Στην εικόνα έχουν κρυφτεί εν μέρει τα αποτελέσματα των συνδρομητών, για λόγους ιδιωτικότητας. Τέλος, ακριβώς μετά τη λίστα μελών, υπάρχει ένα κείμενο ‘Redirect to admin page’, το οποίο ανακατευθύνει τον χρήστη στη σελίδα διαχειριστή της συγκεκριμένης λίστας. Στη συνέχεια θα παρουσιαστούν και θα σχολιαστούν και άλλες εικόνες, οι οποίες επιδεικνύουν τις υπόλοιπες δυνατότητες της διεπαφής.



Εικόνα 4.3: Εμφάνιση tooltip με τη περιγραφή της λίστας

Όπως φαίνεται στην Εικόνα 4.3, όταν ο κέρσορας βρίσκεται πάνω από το όνομα μίας λίστας, το όνομα της τονίζεται με κόκκινο χρώμα, και εμφανίζεται ένα tooltip με τη περιγραφή της λίστας, όπως αυτή υπάρχει στο Mailman του τμήματος. Εάν μία λίστα δεν έχει περιγραφή, εμφανίζει το αντίστοιχο μήνυμα.



Εικόνα 4.4: Τονισμός της επιλεγόμενης διεύθυνσης email

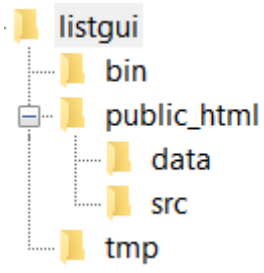
Στην Εικόνα 4.4, φαίνεται πως, όπως στη περίπτωση των λιστών, έτσι και στις διευθύνσεις email, υπάρχει ένας κόκκινος τονισμός, όταν ο κέρσορας βρίσκεται επάνω τους.

Η διάταξη της ιστοσελίδας έχει βασιστεί στη πολύ γνωστή CSS βιβλιοθήκη 'bootstrap'. Τα πλαίσια (divs) της ιστοσελίδας αλλάζουν δυναμικά μέγεθος, και θέση, εάν π.χ. δε χωράνε οριζόντια στην οθόνη. Γενικότερα, η συμπεριφορά της ιστοσελίδας είναι δυναμική και προσαρμόσιμη σε κάθε οθόνη και συσκευή, όπως είναι απαραίτητο για κάθε εφαρμογή σε HTML5. Αυτό αναπαρίσταται και στην Εικόνα 4.5.

Εικόνα 4.5: Παρουσίαση της προσαρμογής μετά από αλλαγή μεγέθους του παραθύρου

4.3 Η οργάνωση της εφαρμογής

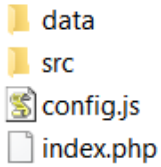
Η οργάνωση των φακέλων της εφαρμογής παρουσιάζεται στην Εικόνα 4.6.



Εικόνα 4.6: Οργάνωση των φακέλων της εφαρμογής

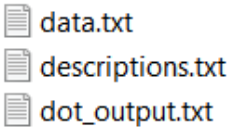
Στον φάκελο 'bin', περιέχεται το αρχείο ενημέρωσης των δεδομένων. Ο φάκελος 'tmp', είναι ένας φάκελος για την αποθήκευση προσωρινών αρχείων που δημιουργούνται κατά την ενημέρωση των

δεδομένων και διαγράφονται στη συνέχεια. Ο φάκελος ‘public_html’, που περιέχει τον κώδικα και τα δεδομένα της εφαρμογής, παρουσιάζεται στην Εικόνα 4.7.



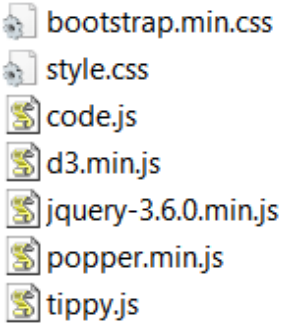
Εικόνα 4.7: Οργάνωση του φακέλου ‘public_html’

Ο φάκελος ‘data’, περιέχει τα δεδομένα που χρειάζονται για το χτίσιμο των δέντρων, καθώς και πληροφορίες για τις λίστες αλληλογραφίας. Ο φάκελος ‘src’ περιέχει όλες τις βιβλιοθήκες που χρειάζονται, καθώς και τα JavaScript και CSS αρχεία που δημιουργήθηκαν για τη χειραγώγηση και τον σχεδιασμό της ιστοσελίδας. Το αρχείο ‘config.js’ περιέχει τα ονόματα των μεταβλητών και των διαδρομών των αρχείων, που μπορεί να αλλάξουν. Το αρχείο ‘index.php’, περιέχει τη κυρίως εφαρμογή. Στην Εικόνα 4.8 παρουσιάζεται ο φάκελος ‘data’.



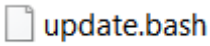
Εικόνα 4.8: Οργάνωση του φακέλου ‘data’

Το αρχείο ‘data.txt’, περιέχει τα δεδομένα των λιστών, δηλαδή τα ονόματα τους και τα μέλη τους. Το αρχείο ‘descriptions.txt’ περιέχει τις περιγραφές των λιστών, εάν αυτές υπάρχουν. Το αρχείο ‘dot_output.txt’ είναι ένα αρχείο το οποίο δημιουργείται από μία βιβλιοθήκη με όνομα ‘Graphviz’, και πρόκειται για ένα βοηθητικό εργαλείο στο χτίσιμο της ιεραρχικής δομής, διότι οι ζητούμενες δυνατότητες δε καλύπτονταν πλήρως από τη βιβλιοθήκη ‘d3’. Θα γίνει πιο λεπτομερής αναφορά σε αυτό το ζήτημα στο επόμενο υποκεφάλαιο, όπου θα παρουσιαστεί ο κώδικας της εφαρμογής. Στην Εικόνα 4.9 παρουσιάζεται ο επόμενος υποφάκελος του ‘public_html’, ο φάκελος ‘src’.



Εικόνα 4.9: Οργάνωση του φακέλου ‘src’

Ο φάκελος αυτός περιέχει τις βιβλιοθήκες JavaScript ‘d3’, ‘popper’, ‘tippy’ και ‘jQuery’. Η βιβλιοθήκη ‘d3’ αναλύθηκε εκτενώς και δε θα σχολιαστεί. Οι βιβλιοθήκες ‘popper’ και ‘tippy’ συνδυάζονται για την εμφάνιση των tooltips που παρουσιάστηκαν παραπάνω, στη Web διεπαφή. Η βιβλιοθήκη ‘jQuery’ χρησιμοποιείται για τη χρήση αιτημάτων ‘ajax’, για την ανάκτηση των αρχείων δεδομένων. Περιέχεται επίσης η CSS βιβλιοθήκη ‘bootstrap’. Τέλος, περιέχει τα αρχεία ‘code.js’ και ‘style.css’, είναι τα αρχεία του JavaScript κώδικα και της CSS για τη σχεδίαση της εφαρμογής αντίστοιχα. Το αρχείο ‘code.js’, περιέχει και το μεγαλύτερο μέρος του κώδικα που θα παρουσιαστεί, καθώς η εφαρμογή είναι εξ’ ολοκλήρου χτισμένη σε JavaScript. Ανεβαίνοντας δύο επίπεδα στην ιεραρχία των φακέλων, και φτάνοντας ξανά στον αρχικό φάκελο της εφαρμογής, στην Εικόνα 4.10 παρουσιάζεται ο υποφάκελος ‘bin’.



Εικόνα 4.10: Οργάνωση του φακέλου ‘bin’

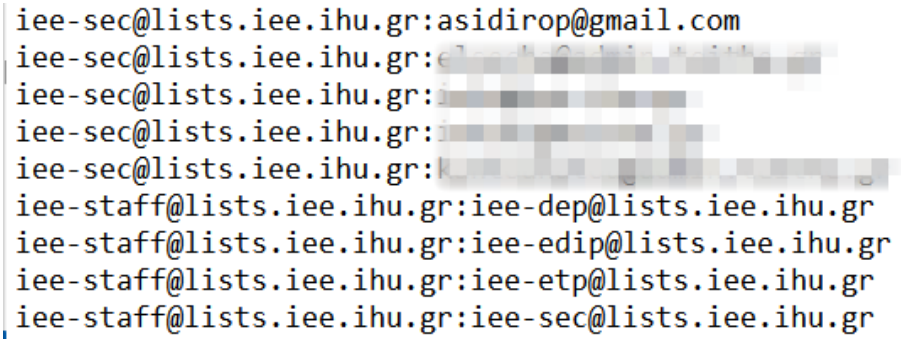
Ο φάκελος αυτός περιέχει μόνο ένα αρχείο, το ‘update.bash’. Το αρχείο αυτό είναι υπεύθυνο για την ενημέρωση των δεδομένων, καθώς και για τη χρήση της βιβλιοθήκης ‘Graphviz’ για τη δημιουργία του βοηθητικού αρχείου ‘dot_output.txt’. Ο κώδικας του θα αναλυθεί στο επόμενο υποκεφάλαιο.

4.4 Ο κώδικας της εφαρμογής

Σε αυτό το υποκεφάλαιο θα αναλυθεί ο κώδικας της εφαρμογής. Θα γίνει παρουσίαση όλων των αρχείων που δημιουργήθηκαν κατά την εκπόνηση αυτής της Πτυχιακής Εργασίας. Τα αρχεία αυτά είναι τα ακόλουθα:

- **update.bash**
- **config.js**
- **index.php**
- **style.css**
- **code.js**

Προτού όμως παρουσιαστούν τα αρχεία, είναι απαραίτητο να δοθεί μία εικόνα από τα αρχεία δεδομένων, ώστε να γίνει κατανοητός και ο κώδικας που τα δημιουργεί ή τα επεξεργάζεται. Στην Εικόνα 4.11 παρουσιάζεται ένα μέρος του αρχείου ‘data.txt’.



Εικόνα 4.11: Μέρος του αρχείου ‘data’

Τα δεδομένα σε αυτό το αρχείο αποτελούνται από ζευγάρια λίστας-μέλους που χωρίζονται μεταξύ τους με μία άνω κάτω τελεία (:). Το αριστερό μέρος είναι η διεύθυνση email της λίστας, ενώ το δεξιά η διεύθυνση email του μέλους. Το δεξιά μέρος φυσικά μπορεί να είναι η διεύθυνση email μίας λίστας, υποδηλώνοντας πως πρόκειται για υπολίστα. Στην εικόνα φαίνονται οι υπολίστες της 'iee-staff', οι οποίες είναι οι 'iee-dep', 'iee-edip', 'iee-etp' και 'iee-sec'. Όπως θα δούμε και αργότερα, ο κώδικας JavaScript ξεχωρίζει εάν πρόκειται για μέλος ή για υπολίστα, από το εάν η διεύθυνση περιέχει τη λέξη 'lists' στο όνομα της. Στη συνέχεια παρουσιάζεται ένα μέρος του αρχείου 'descriptions.txt', στην Εικόνα 4.12.

```
aciee:Λίστα Μόνιμου Εκπαιδευτικού Προσωπικού
acit:Λίστα Μόνιμου Εκπαιδευτικού Προσωπικού
arrest:
asidirop_test:
atest:
board1:
conit:Λίστα Έκτακτου Εκπαιδευτικού Προσωπικού
ediptei:Λίστα του συλλόγου ΕΔΙΠ των ΤΕΙ
```

Εικόνα 4.12: Μέρος του αρχείου 'descriptions'

Τα δεδομένα σε αυτό το αρχείο αποτελούνται από ζευγάρια λίστας-περιγραφής, και χωρίζονται και πάλι με μία άνω κάτω τελεία. Το αριστερό μέρος αποτελείται από τα ονόματα των λιστών, ενώ το δεξί μέρος από τις περιγραφές τους, εάν αυτές υπάρχουν (όπως φαίνεται και στην εικόνα, δεν έχουν όλες οι λίστες περιγραφή). Το τελευταίο αρχείο δεδομένων που θα παρουσιαστεί είναι το 'dot_output.txt' και ένα μέρος του φαίνεται στην Εικόνα 4.13.

```
iee-all 6.6527
iee-dep 5.1666
iee-edip 6.6944
iee-etp 2.2777
iee-form 8.8055
iee-sec 3.7083
iee-test1 8.2638
```

Εικόνα 4.13: Μέρος του αρχείου 'descriptions'

Τα δεδομένα σε αυτό το αρχείο αποτελούνται από ζευγάρια λίστας και τιμές του άξονα 'x'. Για να γίνει κατανοητό από που παράγονται αυτές οι τιμές του άξονα 'x', είναι απαραίτητη η περιγραφή του τρόπου λειτουργίας του 'Graphviz', που παράγει το αρχείο. Η εντολή 'dot' του 'Graphviz', ουσιαστικά χτίζει ένα δέντρο με τα δεδομένα που του δίνονται (σε αυτή τη περίπτωση δίνονται μόνο οι σχέσεις λιστών-υπολιστών, σε ένα αρχείο που δημιουργείται προσωρινά κατά την εκτέλεση του script ενημέρωσης). Ως έξοδο έχει τις συντεταγμένες 'x' και 'y' για κάθε στοιχείο, για το κατάλληλο χτίσιμο του δέντρου. Το χτίσιμο που κάνει το 'dot', έχει ένα πλεονέκτημα σε σχέση με αυτό της βιβλιοθήκης 'd3', το οποίο

είναι ότι χτίζει το δέντρο βάζοντας τους σχετικούς κόμβους όσο πιο κοντά είναι εφικτό (στον άξονα x). Από την άλλη, η ‘d3’, τοποθετεί τους κόμβους με τη σειρά εισόδου τους ως δεδομένα. Οπότε, στη συγκεκριμένη Πτυχιακή Εργασία, όπου η πολυπλοκότητα των δέντρων ήταν μεγάλη, με τη χρήση της ‘d3’ μόνο, το αποτέλεσμα δεν ήταν ικανοποιητικό. Υπήρχαν πολύ μεγάλες αποστάσεις από λίστες και υπολίστες, με αποτέλεσμα οι γραμμές που τις ενώνουν να τέμνονται πολύ και γενικότερα το αποτέλεσμα να είναι περιπλεγμένο και να προσφέρει κακή αισθητική στον χρήστη. Παίρνοντας τις τιμές του άξονα ‘x’ από την έξοδο του ‘dot’, και ταξινομώντας τους κόμβους στην είσοδο της ‘d3’ με αυτή τη σειρά, το πρόβλημα λύθηκε και πλέον οι λίστες διατάσσονται με τέτοιο τρόπο ώστε οι γραμμές που τέμνονται να είναι οι ελάχιστες δυνατές. Επιστέφουμε στο αρχείο ‘dot_output.txt’, αφού έγινε κατανοητό το γιατί χρειάζονται οι τιμές του άξονα ‘x’, και πως παράγονται. Τα ζευγάρια λίστας-τιμής ‘x’, χωρίζονται μεταξύ τους με ένα κενό χαρακτήρα (space). Αριστερά βρίσκονται οι λίστες, ενώ στο δεξί μέρος οι τιμές των ‘x’. Σε αυτό το σημείο θα προχωρήσουμε με την ανάλυση του κώδικα της εφαρμογής, αφού η δομή των αρχείων δεδομένων είναι πλέον γνωστή. Τα αρχεία θα παρουσιαστούν με τη σειρά που αναφέρθηκαν, με το αρχείο ‘update.bash’ να είναι πρώτο και να παρουσιάζεται στην Εικόνα 4.14.

update.bash

```
1  #!/bin/bash
2
3  curl http://lists.iee.ihu.gr/lists_vapi  -o /home/listgui/public_html/data/data.txt
4  curl http://lists.iee.ihu.gr/lists_vapi_info -o /home/listgui/public_html/data/descriptions.txt
5
6  filename='/home/listgui/public_html/data/data.txt'
7  dot_filename='/home/listgui/tmp/dot-file'
8
9  echo "digraph mygraph {" > $dot_filename
10
11 while read line; do
12
13 first=$(echo $line | cut -f1 -d":")
14 second=$(echo $line | cut -f2 -d":")
15
16 if [[ $second == *"lists"* ]]; then
17     output='"'$(echo $first | cut -f1 -d"@")'"'
18     output=$output'->"'$(echo $second | cut -f1 -d"@")'"'
19     echo $output >> $dot_filename
20 fi
21
22 done < $filename
23
24 echo "}" >> $dot_filename
25
26 dot $dot_filename -Tplain| grep '^node'| cut -f3,2 -d' ' | tr -d \" > /home/listgui/public_html/data/dot_output.txt
27 rm $dot_filename
```

Εικόνα 4.14: Ο κώδικας του αρχείου ‘update.bash’

Το συγκεκριμένο bash script αρχικά (στις γραμμές 3 και 4) ανακτά τα δεδομένα από το API, με την εντολή ‘curl’, και τα αποθηκεύει στα αρχεία που αντιστοιχούν. Στη συνέχεια, όλες οι επόμενες εντολές ασχολούνται με το ‘χτίσιμο’ του αρχείου ‘dot_output.txt’. Ξεκινώντας, γράφεται ένα προσωρινό αρχείο στον φάκελο ‘tmp’, με όνομα ‘dot-file’. Αυτό συμβαίνει γιατί η εντολή ‘dot’ χρειάζεται συγκεκριμένη διαμόρφωση στο αρχείο εισόδου της. Έτσι, στις γραμμές 9 έως 24, διαμορφώνεται αυτό το αρχείο. Οι γραμμές 9 και 24 τοποθετούν το άνοιγμα και το κλείσιμο του αρχείου, όπως αυτό χρειάζεται να είναι για να εισαχθεί στη ‘dot’, ενώ ενδιάμεσα, στις γραμμές 11 έως 22, διαβάζεται το αρχείο δεδομένων ‘data.txt’, για να ανακτηθούν και να εγγραφούν μέσα στο έγγραφο οι σχέσεις των λιστών. Αυτό φυσικά πρέπει και πάλι να γίνει με συγκεκριμένο τρόπο, οπότε υπάρχει η κατάλληλη επεξεργασία ώστε οι σχέσεις από τη μορφή ‘list:list’ να πάρουν τη μορφή ‘list → list’.

Αφότου τελειώσει η διαμόρφωση του αρχείου ‘dot-file’, ακολουθεί η εντολή ‘dot’, στη γραμμή 26, με είσοδο το αρχείο που μόλις δημιουργήθηκε. Ως έξοδο της ‘dot’ επιλέγεται απλό κείμενο (άλλες διαθέσιμες μορφές είναι οι ‘json’, ‘SVG’ και πολλές ακόμη), και από αυτό το κείμενο, χρησιμοποιώντας τις εντολές ‘grep’ και ‘cut’, παίρνουμε τελικά ως έξοδο μόνο τα ονόματα των λιστών και τη τιμή ‘x’ τους, τα οποία είναι τα μόνα που χρειαζόμαστε. Τέλος, κώδικας κλείνει με τη γραμμή 27, που απλώς σβήνει το αρχείο ‘dot-file’, το οποίο δε χρειάζεται πια. Το επόμενο αρχείο που θα αναλυθεί είναι το ‘config.js’, το οποίο παρουσιάζεται στην Εικόνα 4.15.

config.js

```
1  var admin_base = "http://lists.iee.ihu.gr/cgi-bin/mailman/admin/"
2
3  var data_file = "data/data.txt"
4
5  var description_file = "data/descriptions.txt"
6
7  var dot_file = "data/dot_output.txt"
```

Εικόνα 4.15: Ο κώδικας του αρχείου ‘config.js’

Ο κώδικας αυτού του αρχείου δε χρειάζεται ιδιαίτερες εξηγήσεις, καθώς πρόκειται για ένα πολύ απλό αρχείο διαμόρφωσης των μεταβλητών των διευθύνσεων που χρησιμοποιούνται στον κώδικα JavaScript. Πρόκειται για τέσσερις μεταβλητές. Η ‘admin_base’ περιέχει το γενικό link ανακατεύθυνσης στη σελίδα διαχειριστή (και αρκεί η προσθήκη του ονόματος της λίστας σε αυτό το link για τη σωστή ανακατεύθυνση). Η ‘data_file’ περιέχει τη διεύθυνση για το αρχείο δεδομένων ‘data.txt’, ενώ οι ‘description_file’ και ‘dot_file’ περιέχουν τις διευθύνσεις των ‘descriptions.txt’ και ‘dot_output.txt’ αντίστοιχα. Έτσι, στον κώδικα χρησιμοποιούνται τα ονόματα των μεταβλητών, για να αποφευχθούν προβλήματα σε μία περίπτωση αλλαγής διεύθυνσης αρχείου. Το επόμενο αρχείο που θα παρουσιαστεί είναι το ‘index.php’, το παρουσιάζεται σε δύο μέρη, στις Εικόνες 4.16 και 4.17.

index.php

```
1  <!DOCTYPE html>
2
3  <html lang="en">
4    <head>
5      <meta charset="utf-8">
6
7      <title>Mailman lists GUI</title>
8
9      <script src="./src/jquery-3.6.0.min.js"></script>
10     <script src="./src/d3.min.js"></script>
11     <script src="./src/popper.min.js"></script>
12     <script src="./src/tippy.js"></script>
13     <script src="config.js"></script>
14     <link href="./src/bootstrap.min.css" rel="stylesheet">
15     <link href="./src/style.css" rel="stylesheet">
16
17   </head>
18
19   <body>
20     <div class="container-fluid">
21     <div class="panel panel-default">
22         <div class="panel-body" style="background-color: #f2dede;">
23             <h5 style="float: left; cursor: default;">Οπτικοποίηση του Mailman του Τμήματος Μηχανικών Πληροφορικής και Ηλ. Συστημάτων</h5>
24             <h5 style="float: right; cursor: default" id="lastUpdate">Last update: </h5>
25         </div>
```

Εικόνα 4.16: Ο κώδικας του αρχείου ‘index.php’ (1)

```
26      </div>
27      <div class="row">
28          <div class="col-lg-3">
29              <div class="panel panel-info">
30                  <div class="panel-heading"><h3 style="text-align: center; cursor: default;">Lists without relations</h3></div>
31                  <div class="panel-body" id="lists" style="height:80vh;"></div>
32              </div>
33          </div>
34          <div class="col-lg-5">
35              <div class="panel panel-primary">
36                  <div class="panel-heading"><h3 style="text-align: center; cursor: default;">Lists with relations</h3></div>
37                  <div class="panel-body" id="tree" style="height:80vh;"></div>
38              </div>
39          </div>
40          <div class="col-lg-4">
41              <div class="panel panel-success">
42                  <div class="panel-heading" id="showhead"><h3 style="text-align: center; cursor: default;">List results</h3></div>
43                  <div class="panel-body" id="show" style="max-height: 80vh; overflow-y: scroll;"><center>Choose a list to see detailed info<center></div>
44              </div>
45          </div>
46      </div>
47      </div>
48    <script src="./src/code.js"></script>
49    </body>
50  </html>
```

Εικόνα 4.17: Ο κώδικας του αρχείου ‘index.php’ (2)

Όπως φαίνεται, ο κώδικας στο αρχείο index είναι ελάχιστος. Αυτό συμβαίνει διότι η CSS και η JavaScript (που χτίζει σχεδόν όλη την εφαρμογή), βρίσκονται σε ξεχωριστά αρχεία. Η σελίδα αυτή περιλαμβάνει αρχικά τις διασυνδέσεις με όλες τις βιβλιοθήκες και τα αρχεία κώδικα που χρειάζονται, με τη χρήση των <script> για τα JavaScript αρχεία και <link> για τα CSS αρχεία. Αυτό συμβαίνει στο <head>, στις γραμμές 9 έως 15. Στη συνέχεια, στο <body>, υπάρχουν πολλά τα οποία δηλώνονται για τις ανάγκες χρήσης της βιβλιοθήκης ‘Bootstrap’, για την καλύτερη σχεδίαση της εφαρμογής. Τα τέσσερα κύρια , δηλαδή η κεφαλίδα και τα τρία ‘πλαίσια’ που υπάρχουν κάτω από αυτήν στην ιστοσελίδα, δηλώνονται στις γραμμές 22, 28, 34 και 40.

Τα υπόλοιπα δηλώνονται για τη λογική της σελίδας, όπως το στη γραμμή 20, που δηλώνει ότι η σελίδα θα έχει μία διάταξη τύπου ‘container’, ή το στη γραμμή 27 που δηλώνει ότι όσα περιλαμβάνονται σε αυτό, θα είναι στην ίδια σειρά (για αυτό τα τρία κύρια πλαίσια είναι το ένα δίπλα στο άλλο). Μέσα στα τέσσερα κύρια , βρίσκονται στοιχεία <h5> και άλλα , τα οποία προορίζονται είτε για τίτλους, είτε για την εμφάνιση πληροφοριών, είτε φυσικά για τη παρουσίαση των λιστών μέσα σε αυτά. Τέλος, στη γραμμή 48 δηλώνεται ο κώδικας της εφαρμογής. Είναι η μόνη δήλωση που γίνεται στο τέλος και όχι στο <head>, διότι με την εισαγωγή αυτού του αρχείου, χειραγωγούνται άμεσα κάποια της ιστοσελίδας, τα οποία πρέπει προφανώς να έχουν δηλωθεί πρώτα. Το επόμενο αρχείο που θα αναλυθεί είναι το αρχείο ‘style.css’, το οποίο παρουσιάζεται στις Εικόνες 4.18 και 4.19.

style.css

```
1   .node circle {
2     fill: #50CF24;
3     stroke: black;
4     stroke-width: 3px;
5     cursor : pointer;
6   }
7
8   .node text {
9       font: 12px sans-serif;
10      cursor : pointer;
11  }
12
13  .node text:hover {
14      fill: red;
15      stroke: black;
16      stroke-width: 0.25px;
17  }
18
19  .link {
20    fill: none;
21    stroke: black;
22    stroke-width: 1px;
23  }
24
25  ol {
26      list-style-type: none;
27      counter-reset: li;
28  }
```

Εικόνα 4.18: Ο κώδικας του αρχείου ‘style.css’ (1)

style.css

```
29
30  li:before {
31      counter-increment: li;
32      content: counter(li, decimal-leading-zero);
33      color: blue;
34      margin-right: 0.25em;
35  }
36
37  .container-fluid,.row,.col-lg-3,.col-lg-4,.col-lg-5 {
38      padding: 0 !important;
39      margin: 0 !important;
40  }
41
42  .panel{
43      margin: 1px 1px 1px 1px !important;
44  }
45
46  #show::-webkit-scrollbar {
47      width: 15px;
48  }
49
50  #show::-webkit-scrollbar-track {
51      background: lightgreen;
52  }
53
54  #show::-webkit-scrollbar-thumb {
55      background-color: #11ad00;
56      border-radius: 30px;
57  }
```

Εικόνα 4.19: Ο κώδικας του αρχείου ‘style.css’ (2)

Οι γραμμές 1 έως 17 ασχολούνται με τα αντικείμενα της κλάσης ‘node’, δηλαδή με τα αντικείμενα των λιστών που χτίζονται στη ‘d3’. Αρχικά ορίζεται η διαμόρφωση του κύκλου που αντιπροσωπεύει κάθε στοιχείο. Στη συνέχεια, ορίζεται η διαμόρφωση του κείμενου σε κάθε αντικείμενο λίστας, όπως και η ‘hover’ ιδιότητα του, που ορίζει να αλλάζει χρώμα το όνομα της λίστας σε κόκκινο, όταν το ποντίκι βρίσκεται επάνω στο αντικείμενο, για να δοθεί έμφαση.

Στις γραμμές 25 έως 35, η CSS που έχει γραφεί ασχολείται με τη διαμόρφωση του στοιχείου ‘ol’, δηλαδή της διατεταγμένης λίστας. Η χρήση αυτής της διαμόρφωσης γίνεται στην εμφάνιση της λίστας μελών μίας λίστας αλληλογραφίας, και φαίνεται παραπάνω, στην Εικόνα 4.4. Οι διαφορές με τις προεπιλεγμένες ρυθμίσεις είναι μικρές, και έγιναν για αισθητικούς λόγους.

Στις γραμμές 37 έως 44 γίνονται προσαρμογές στις ρυθμίσεις της βιβλιοθήκης ‘Bootstrap’, σε πολλές κλάσεις της, με σκοπό τη μείωση των κενών μεταξύ των αντικειμένων. Η χρήση του τελεστή ‘!important’ ουσιαστικά οδηγεί στην αναγκαστική χρήση αυτής της ρύθμισης, ακόμη και αν ορίζεται διαφορετικά μέσα στη βιβλιοθήκη και γενικά σε άλλα σημεία του κώδικα CSS.

Το τελευταίο στοιχείο που επεξεργάζεται η CSS, είναι το στοιχείο με id ‘show’, δηλαδή το στο οποίο παρουσιάζονται τα μέλη της επιλεγμένης λίστας. Αυτό συμβαίνει στις γραμμές 46 έως 57. Οι ρυθμίσεις που διαμορφώνονται έχουν να κάνουν μόνο με τη γραμμή κύλισης (‘scrollbar’). Για αισθητικούς λόγους και πάλι, επιλέγεται αυτή να έχει σκούρο πράσινο χρώμα, με το φόντο από πίσω της να είναι ανοιχτό πράσινο (για να ταιριάζει στα χρώματα του).

Το τελευταίο αρχείο, το οποίο έχει και τον περισσότερο κώδικα, αλλά και τη λογική της εφαρμογής γενικότερα, είναι το ‘code.js’. Το αρχείο αυτό θα παρουσιαστεί σε αρκετές εικόνες, ακολουθούμενες από μία περιγραφή για τη καθεμία, καθώς είναι ιδιαίτερα μεγάλο για να παρουσιαστεί σε ένα μέρος.

code.js

```
1
2   //my variables
3   var lists = [];
4   var mails = [];
5   var disLists = [];
6   var data = [];
7   var listHasParent = [];
8   var listAlreadyIn = [];
9   var relations = [];
10  var desc = [];
11
12  //d3 tree variables
13  var dataMap;
14  var treeData = [];
15  var dataMap2;
16  var treeData2 = [];
17  var root;
18  var root2;
19
20      // Get last file update
21      var update = $.ajax({
22          async: false,
23          url : data_file
24      }).getResponseHeader("Last-Modified");
25      document.getElementById("lastUpdate").innerHTML = document.getElementById("lastUpdate").innerHTML + update;
26
27      // Reading the data file
28      var reader = $.ajax({
29          async: false,
30          url : data_file
31      }).responseText;
32      var i = 0;
33      var res,res2;
34
35      // oso to length tou result einai mikrotero apo ton metriti
36      while(i<reader.length){
37          res = "";
38          res2 = "";
39          // oso den einai ':' o xaraktiras, diavase to 1o meros tou line
40          while(!(reader[i]===":")){
41              res = res + reader[i];
42              i++;
```

Εικόνα 4.20: Ο κώδικας του αρχείου ‘code.js’ (1)

Ο κώδικας του αρχείου ξεκινά με τις δηλώσεις κάποιων απαραίτητων μεταβλητών που θα χρειαστούν άμεσα στη συνέχεια. Κάποιες από αυτές (οι ‘my variables’) απαιτούνται για δικές μου προσθήκες στον κώδικα, ενώ οι υπόλοιπες (οι ‘d3 tree variables’) απαιτούνται για το χτίσιμο των δέντρων στη d3.

Στη συνέχεια, στις γραμμές 21 έως 24, μέσω ajax, γίνεται ανάκτηση της ημερομηνίας και ώρας της τελευταίας επεξεργασίας του αρχείου δεδομένων ‘data.txt’, και στη γραμμή 25 το αποτέλεσμα της ανάκτησης αναγράφεται στη σελίδα (μπορούμε να δούμε το αποτέλεσμα στην Εικόνα 4.1, επάνω δεξιά). Ακολουθεί η ανάκτηση του ίδιου του αρχείου ‘data.txt’, και πάλι μέσω ajax, στις γραμμές 28 έως 31, και μία αρχικοποίηση κάποιων μεταβλητών που θα χρειαστούν για την αποσύνθεση αυτής της ανάκτησης. Στις γραμμές 36 έως 42, αρχίζει μία επανάληψη ‘while’, η οποία θα περιγραφεί μετά την Εικόνα 4.21, ώστε να υπάρχει μία πλήρης εικόνα της.

code.js

```
43          }
44          i++;
45          // oso den einai newline o xaraktiras, diavase to 2o meros tou line
46          while(!(reader[i]==="\n")){
47              res2 = res2 + reader[i];
48              i++;
49          }
50          lists.push(res);
51          if(!(disLists.includes(res))) disLists.push(res);
52          res2 = res2.replace(/(\r\n|\n|\r)/gm, "").replace(/</g, " ").replace(/>/g, " ");
53          mails.push(res2);
54          i++;
55      }
56
57      // Reading the description file
58      var reader2 = $.ajax({
59                        url: description_file,
60                        async: false
61                    }).responseText;
62      i = 0;
63
64      // oso to length tou result einai mikrotero apo ton metriti
65      while(i<reader2.length){
66          res = "";
67          res2 = "";
68          // oso den einai ':' o xaraktiras, diavase to 1o meros tou line
69          while(!(reader2[i]===":")){
70              res = res + reader2[i];
71              i++;
72          }
73          i++;
74          // oso den einai newline o xaraktiras, diavase to 2o meros tou line
75          while(!(reader2[i]==="\n")){
76              res2 = res2 + reader2[i];
77              i++;
78          }
79          desc.push({"list" : res, "desc" : res2});
80          i++;
81      }
82
83      // Diaxwrismos twn data, dimiourgia sxeswn parent-child
84      for(var i=0; i<lists.length; i++){
```

Εικόνα 4.21: Ο κώδικας του αρχείου ‘code.js’ (2)

Η εξωτερική εντολή ‘while’, που ξεκινά στη γραμμή 36, συνεχίζεται μέχρι τη γραμμή 55. Εντός της περιέχει δύο ακόμη εντολές ‘while’, οι οποίες έχουν ως εργασία να ξεχωρίζουν τα δύο μέρη κάθε σχέσης λιστών (γονέας – παιδί), και να τα αποθηκεύουν στις μεταβλητές ‘res’ και ‘res2’ αντίστοιχα. Μετά από αυτό, η λίστα αποθηκεύεται σε ένα πίνακα με λίστες, αλλά και σε ένα πίνακα με τις μοναδικές λίστες, ο οποίος δεν έχει διπλότυπα, στις γραμμές 50 και 51 αντίστοιχα. Η γραμμή 52 αφαιρεί τους χαρακτήρες ‘<’ και ‘>’ από τα email των μελών, διότι προκαλούν λογικά προβλήματα (στην HTML οι χαρακτήρες αυτοί δηλώνουν στοιχεία, όπως το , οπότε το email θεωρείται στοιχείο εάν οι χαρακτήρες μείνουν). Η γραμμή 53 αποθηκεύει το μέλος σε ένα πίνακα με μέλη. Τέλος, η γραμμή 54 αυξάνει τον μετρητή ‘i’, ο οποίος ουσιαστικά συγκρίνεται κάθε φορά με το μήκος των χαρακτήρων του reader, στην εξωτερική εντολή ‘while’, για να σταματήσει το διάβασμα όταν οι χαρακτήρες έχουν τελειώσει.

Παρακάτω, στις γραμμές 58 έως 81, ακολουθεί μία παρόμοια διαδικασία με αυτή για το αρχείο ‘data.txt’, μόνο που αυτή τη φορά ανακτάται το αρχείο ‘descriptions.txt’, και με τις εντολές ‘while’ αποθηκεύονται οι λίστες και οι περιγραφές τους, στον πίνακα ‘desc’ (γραμμή 79).

code.js

```
85          if(mails[i].includes("lists")){
86              if(listAlreadyIn.includes(mails[i])){
87                  data.forEach(function(d){
88                      if(d.name==mails[i]){
89                          var count = 2;
90                          while(d['parent'+count.toString()]) count++;
91                          d['parent'+count.toString()] = lists[i];
92                          relations.push({'name' : mails[i], 'parent' : lists[i]});
93                      }
94                  });
95              }
96              else{
97                  data.push({ "name" : mails[i], "parent" : lists[i]});
98                  if(!(listHasParent.includes(mails[i]))) listHasParent.push(mails[i]);
99                  listAlreadyIn.push(mails[i]);
100                 relations.push({'name' : mails[i], 'parent' : lists[i]});
101             }
102         }
103     }
104
105     // Prosthiki twn listwn pou den exoun parent sto 1o epipedo
106     disLists.forEach(function(list){
107         if(!(listHasParent.includes(list))) data.push({ "name" : list, "parent" : "Top Level"});
108     });
109     data.push({ "name" : "Top Level", "parent" : "null"});
110
111     // Svinw osa lists den exoun paidia k den einai paidia, gia na mpoun sto tree 'no relations'
112     var data2 = [];
113     data = data.filter(function(d){
114         var ret = 0;
115         if(d.parent=="Top Level"){
116             relations.forEach(function(e){
117                 if(d.name==e.parent) ret = 1;
118             });
119         }
120         else return 1;
121         if(!ret) data2.push(d);
122         return ret;
123     });
124     data2.push({ "name" : "Top Level", "parent" : "null"});
125
```

Εικόνα 4.22: Ο κώδικας του αρχείου ‘code.js’ (3)

Η επεξεργασία των δεδομένων που αποθηκεύτηκαν ξεκινάει με τις γραμμές 84 έως 103, στις οποίες γίνεται μία επανάληψη για να ελεγχθεί το κάθε ζευγάρι γονιού – παιδιού, για την αποθήκευση των σχέσεων λιστών – υπολιστών. Αρχικά ελέγχεται εάν το παιδί περιέχει τη λέξη ‘lists’ στο κείμενο του, γεγονός που υποδεικνύει ότι είναι λίστα, και άρα πρέπει να αποθηκευτεί αυτή η σχέση στον πίνακα ‘data’, που θα χρησιμοποιηθεί για τη κατασκευή του δέντρου αργότερα. Η σχέση αυτή αποθηκεύεται από τη μεριά του παιδιού, δηλαδή ως ‘ο κόμβος Β έχει γονέα τον κόμβο Α’. Αυτό δημιουργεί ένα πρόβλημα, καθώς στη d3 είναι δυνατή η δήλωση ενός μόνο γονέα, πράγμα που δεν αρκούσε στην εφαρμογή που κατασκευάστηκε. Αυτό επιλύθηκε με την αποθήκευση περισσότερων γονέων ως ιδιότητες του κόμβου (‘parent2’, ‘parent3’, κτλ.) και σχεδιάζοντας χειροκίνητα στη συνέχεια (χωρίς τη βοήθεια της d3) από ένα ‘path’ για κάθε ένα από τους υπόλοιπους γονείς, που το ενώνει με το παιδί.

Επόμενο βήμα είναι η προσθήκη των λιστών που δεν έχουν γονείς (δηλαδή είναι στο πρώτο επίπεδο του δέντρου) να προστεθούν στον πίνακα ‘data’. Αυτό συμβαίνει στις γραμμές 106 με 108. Οι λίστες αυτές φαίνεται να έχουν γονέα τον κόμβο ‘Top Level’, ο οποίος δηλώνεται στη γραμμή 109. Πρόκειται για ένα ‘ψεύτικο’ κόμβο, ο οποίος αργότερα κρύβεται, αλλά χρειάζεται αρχικά γιατί η d3 περιμένει μόνο ένα κόμβο στο πρώτο επίπεδο (άρα δημιουργούμε ένα ψευδό-επίπεδο για να το επιλύσουμε).

Οι γραμμές 112 με 124, έχουν σκοπό τη δημιουργία ενός δεύτερου πίνακα δεδομένων, του ‘data2’, ο οποίος θα γεμίσει με τιμές, από τους κόμβους του πίνακα ‘data’, οι οποίοι δεν έχουν παιδιά ΚΑΙ δεν είναι παιδιά, δηλαδή τους κόμβους που δεν έχουν σχέσεις με άλλους κόμβους. Αυτό γίνεται αρκετά απλά, με τον έλεγχο των λιστών στο πρώτο επίπεδο (που σίγουρα δεν είναι παιδιά), για το εάν είναι γονείς. Εάν δεν είναι ούτε γονείς, διαγράφονται από τον πίνακα ‘data’ και προστίθενται στον πίνακα ‘data2’. Τα δεδομένα του ‘data2’ θα χρησιμοποιηθούν για τη κατασκευή του ‘δέντρου’ λιστών χωρίς σχέσεις (δέντρου σε εισαγωγικά καθώς είναι δέντρο μόνο θεωρητικά, πρακτικά φαίνεται ως μία λίστα από λίστες).

code.js

```
125
126     // Sta nodes pou exoun pollous parent, orizw ws parent auton me to megalytero depth
127     for(var i=0; i<data.length; i++){
128         var par;
129         var counter = 0;
130         var count = 2;
131         var levels = [];
132
133         par = data[i].parent;
134         while(par!="null"){
135             for(var j=0; j<data.length; j++){
136                 if(data[j].name==par) {
137                     par = data[j].parent;
138                     break;
139                 }
140             }
141             counter++;
142         }
143         levels.push(counter);
144
145         while(data[i]['parent'+count.toString()]){
146             par = data[i]['parent'+count.toString()];
147             counter = 0;
148             while(par!="null"){
149                 for(var j=0; j<data.length; j++){
150                     if(data[j].name==par) {
151                         par = data[j].parent;
152                         break;
153                     }
154                 }
155                 counter++;
156             }
157             levels.push(counter);
158             count++;
159         }
160
161         //allagi 1ou parent me ton maxDepth parent
162         var max = Math.max.apply(Math, levels);
163         if(max!=levels[0]){
164             for(var j=1; j<levels.length; j++){
165                 if(max==levels[j]){
166                     var swap = data[i].parent;
```

Εικόνα 4.23: Ο κώδικας του αρχείου ‘code.js’ (4)

Στη συνέχεια χρειάζεται να ορίσουμε τον σωστό πρώτο γονέα στο κάθε παιδί, καθώς με βάση αυτόν το παιδί θα πάει στο ανάλογο επίπεδο του δέντρου (π.χ. αν ο πρώτος γονέας είναι στο επίπεδο 1, το παιδί θα είναι στο 2). Αυτό δημιουργούσε προβλήματα στην εφαρμογή, διότι υπήρχαν παιδιά που είχαν γονείς στο ίδιο επίπεδο με αυτά, ή ακόμα χειρότερα, επιτρεπόταν ακόμη και να είναι σε προηγούμενο επίπεδο από αυτά, και η σχέση να φαίνεται αντίθετη απ’ αυτή που ισχύει. Οπότε, ο κώδικας στις γραμμές 127 έως 173 (ο υπόλοιπος κώδικας είναι στην επόμενη εικόνα), βρίσκει το βάθος (επίπεδο) των γονέων ενός παιδιού, και ορίζει ως πρώτο γονέα, αυτόν με το πιο μεγάλο βάθος, για την αποφυγή αυτών των λαθών.

code.js

```
167                     data[i].parent = data[i]['parent'+(j+1).toString()];
168                     data[i]['parent'+(j+1).toString()] = swap;
169                     break;
170                 }
171             }
172         }
173     }
174
175     // Taxinomisi twn nodes ana depth symfwna me to x pou pairnw apo to dot
176     var reader3 = $.ajax({
177         async: false,
178         url : dot_file
179     }).responseText;
180
181     var dot = [];
182     i=0
183     while(i<reader3.length){
184         res = "";
185         res2 = "";
186         while(!(reader3[i]===" ")){
187             res = res + reader3[i];
188             i++;
189         }
190         i++;
191         while(!(reader3[i]==="\n")){
192             res2 = res2 + reader3[i];
193             i++;
194         }
195         dot.push({"name" : res, "x" : res2});
196         i++;
197     }
198
199     var depths = [];
200     var nodeDepths = [];
201     data.forEach(function(d){
202         var par = d.parent;
203         var counter = 0;
204
205         while(par!="null"){
206             for(var j=0; j<data.length; j++){
207                 if(data[j].name==par) {
208                     par = data[j].parent;
```

Εικόνα 4.24: Ο κώδικας του αρχείου ‘code.js’ (5)

Επόμενο βήμα είναι η ταξινόμηση των κόμβων σύμφωνα με τη κατασκευή του ‘dot’, διαδικασία που έχει αναφερθεί και νωρίτερα. Αρχικά, στις γραμμές 176 με 179, ανακτάται το αρχείο με τα δεδομένα του ‘dot_output.txt’. Στις γραμμές 183 με 197, αποθηκεύονται οι τιμές του αρχείου (λίστα – τιμή ‘x’) σε ένα πίνακα με όνομα ‘dot’. Τέλος, στις γραμμές 201 έως 249 (οι υπόλοιπες βρίσκονται στην επόμενη εικόνα), συγκρίνονται οι τιμές ‘x’ για τους κόμβους, ανά επίπεδο (depth), και γίνεται ταξινόμηση τους ανάλογα με τις τιμές ‘x’. Η διαδικασία τελειώνει με τη διαγραφή των κόμβων του τρέχοντος επιπέδου από τον πίνακα ‘data’, και την εισαγωγή τους ξανά, αυτή τη φορά με τη σωστή (ταξινομημένη) σειρά.

code.js

```
                    break;
                }
            }
            counter++;
        }
        if(counter){
            nodeDepths.push({'name' : d.name, 'depth' : counter});
            if(!(depths.includes(counter))) depths.push(counter);
        }
    });

    depths.forEach(function(d){
        var nodes = [];
        var nodeX = [];
        var group = [];

        for(var i=0; i<nodeDepths.length; i++)
            if(nodeDepths[i].depth==d)
                nodes.push(nodeDepths[i].name);
        if(nodes.length>1){
            nodes.forEach(function(e){
                dot.forEach(function(j){
                    if(j.name==e.substring(0, e.lastIndexOf("@")))
                        nodeX.push({'name' : e, 'x' : parseFloat(j.x)});
                });
            })
            nodeX.sort((a, b) => (a.x > b.x) ? 1 : -1);
            nodeX.forEach(function(node){
                data = data.filter(function(dat){
                    if(node.name==dat.name){
                        group.push(dat);
                        return 0;
                    }
                    else return 1;
                });
            });
            group.forEach(function(gr){
                data.push(gr);
            });
        }
    });

```

Εικόνα 4.25: Ο κώδικας του αρχείου ‘code.js’ (6)

code.js

```
// ********** Convert flat data into a nice tree ***************
    // create a name: node map
    dataMap = data.reduce(function(map, node) {
        map[node.name] = node;
        return map;
    }, {});

    // create the tree array
    data.forEach(function(node) {
        // add to parent
        var parent = dataMap[node.parent];
        if (parent) {
            // create child array if it doesn't exist
            (parent.children || (parent.children = []))
                // add node to child array
                .push(node);
        } else {
            // parent is null or missing
            treeData.push(node);
        }
    });

    // create a name: node map
    dataMap2 = data2.reduce(function(map, node) {
        map[node.name] = node;
        return map;
    }, {});

    // create the tree array
    data2.forEach(function(node) {
        // add to parent
        var parent = dataMap2[node.parent];
        if (parent) {
            // create child array if it doesn't exist
            (parent.children || (parent.children = []))
                // add node to child array
                .push(node);
        } else {
            // parent is null or missing
            treeData2.push(node);
        }
    });
```

Εικόνα 4.26: Ο κώδικας του αρχείου ‘code.js’ (7)

Το τελευταίο βήμα πριν το χτίσιμο των δέντρων, είναι η μετατροπή των ‘επίπεδων’ δεδομένων που υπάρχουν στους πίνακες ‘data’ και ‘data2’, σε μορφή δέντρου, καθώς αυτή είναι η μορφή στην οποία περιμένει η d3 τα δεδομένα. Ο κώδικας αυτός ουσιαστικά φέρνει τα δεδομένα στη παρακάτω μορφή:

[{

"name": "Top Level",

"parent": "null",

"children": [{

"name": "Level 2: A",

"parent": "Top Level",

"children": [{

Αυτή είναι η μορφή που περιμένει να δεχτεί η βιβλιοθήκη d3. Οι γραμμές 253 με 292 εκτελούν αυτή τη μετατροπή, πρώτα για τον πίνακα ‘data’ και στη συνέχεια για τον ‘data2’. Φυσικά η διαδικασία είναι ίδια και για τους δύο πίνακες.

code.js

```
293
294         // Klisi twn synartisewn gia xtisimo twn trees
295         root = treeData[0];
296         root2 = treeData2[0];
297         treeMake();
298         listsMake();
299
300 function treeMake() {
301
302   document.getElementById("tree").innerHTML = "";
303   var tree;
304   var diagonal;
305   var svg;
306
307   // ************** Generate the tree diagram    *****************
308     var width = document.getElementById("tree").offsetWidth;
309     var height = document.getElementById("tree").offsetHeight*0.95;
310
311     tree = d3.layout.tree()
312         .size([height, width]);
313
314     diagonal = d3.svg.diagonal()
315         .projection(function(d) { return [d.y, d.x]; });
316
317     svg = d3.select("#tree").append("svg")
318         .attr("width", width)
319         .attr("height", height)
320         .append("g");
321
322   // Compute the new tree layout.
323   var nodes = tree.nodes(root).reverse(),
324       links = tree.links(nodes);
325
326   // Svinw to pseutiko root pou eftiaksa kai ta links tou
327   nodes = nodes.filter(function(d){
328     return d.depth;
329   });
330
331   links = links.filter(function(d){
332     return d.source.name != "Top Level";
333   });
334
```

Εικόνα 4.27: Ο κώδικας του αρχείου ‘code.js’ (8)

Το τελευταίο βήμα είναι να οριστούν ως roots τα πρώτα στοιχεία των δύο πινάκων που δημιουργήθηκαν στη μορφή που παρουσιάστηκε, καθώς πρόκειται για τους κόμβους από τους οποίους θα ξεκινήσει το χτίσιμο του δέντρου. Στις γραμμές 295 έως 298, ορίζονται τα roots και καλούνται δύο συναρτήσεις.

Η συνάρτηση ‘treeMake()’, είναι αυτή που θα κατασκευάσει το δέντρο με σχέσεις (δηλαδή το μεσαίο πλαίσιο στην εικόνα της ιστοσελίδας που παρουσιάστηκε νωρίτερα. Από την άλλη, η συνάρτηση ‘listsMake()’, θα κατασκευάσει το δέντρο χωρίς σχέσεις. Οι διαδικασίες των κατασκευών είναι ίδιες, μέχρι το σημείο εισόδου των κόμβων στην SVG. Από εκεί και πέρα, η διαδικασία της ‘listsMake()’ έχει τελειώσει, ενώ η ‘treeMake()’ συνεχίζει με την εισαγωγή των links που ενώνουν τους κόμβους και δηλώνουν τις σχέσεις γονέα – παιδιού. Αυτό αναφέρεται διότι θα παρουσιαστεί μόνο η συνάρτηση ‘treeMake()’, η οποία περιέχει και τον κώδικα της ‘listsMake()’ (με κάποιες ελάχιστες διαφορές στις λεπτομέρειες και όχι στο αποτέλεσμα), για την αποφυγή επαναλήψεων.

Η συνάρτηση ‘treeMake()’ ξεκινά με την απαλοιφή του με όνομα ‘tree’, στο οποίο θα χτιστεί το δέντρο. Αυτό συμβαίνει διότι όπως θα δούμε αργότερα, η συνάρτηση ξανακαλείται για να το ξαναχτίσει, σε κάθε αλλαγή μεγέθους του παραθύρου, ώστε η ιστοσελίδα να είναι δυναμική. Επίσης δηλώνονται κάποιες απαραίτητες μεταβλητές, οι οποίες αρχικοποιούνται στη συνέχεια, μέσω της d3 ή του στοιχείου ‘tree’. Ουσιαστικά, στις γραμμές 308 έως 320, αρχικοποιούνται οι μεταβλητές ‘width’ και ‘height’, μέσω των οποίων στη συνέχεια δημιουργούνται στοιχεία μέσω της d3, στις αντίστοιχες μεταβλητές. Οι μεταβλητές αυτές χρειάζονται για να ενσωματωθούν όλα τα απαραίτητα στοιχεία, κάτω από το στοιχείο ‘tree’ (στην ιεραρχία). Μετά, μέσω αναφορών σε αυτές, προσθέτουμε κόμβους και χαρακτηριστικά σε αυτούς.

Στις γραμμές 323 και 324, αρχικοποιούνται οι κόμβοι και οι συνδέσεις τους, μέσω της μεταβλητής ‘root’ (πρόκειται για το ψεύτικο αρχικό κόμβο που προστέθηκε νωρίτερα) για τους κόμβους, και μέσω των ίδιων των κόμβων, για τις συνδέσεις (κάθε κόμβος έχει τουλάχιστον έναν ‘parent’ δηλωμένο). Αμέσως μετά, αφού το ψεύτικο root δε χρειάζεται πλέον, καθώς έχουμε τους κόμβους και τις συνδέσεις στη d3, τον σβήνουμε, μαζί με τις όποιες συνδέσεις του, στις γραμμές 327 με 333.

code.js

```
335  // Briskw to max depth
336  var maxDep = 0;
337  nodes.forEach(function(d) { if(d.depth>maxDep) maxDep=d.depth; });
338
339  // Normalize for fixed-depth
340  nodes.forEach(function(d) { d.y = d.depth * (width/(maxDep+1)); });
341
342  // Declare the nodes
343  var node = svg.selectAll("g.node")
344      .data(nodes, function(d) { return d.id || (d.id = ++i); });
345
346  // Enter the nodes
347  var nodeEnter = node.enter().append("g")
348      .attr("class", "node")
349      .attr("transform", function(d) {
350          return "translate(" + d.y + "," + d.x + ")"; })
351      .on("click", click)
352      .attr("data-tippy-content",function(d) {
353            var name = d.name.substring(0, d.name.lastIndexOf("@"));
354            var output;
355            for(var i=0; i<desc.length; i++)
356                if(desc[i].list==name)
357                    if(desc[i].desc=="") output = name + " - [no description available]";
358                    else output = name + " - " + desc[i].desc;
359            return output;
360      });
361
362  nodeEnter.append("circle")
363      .attr("r", 10);
364
365  nodeEnter.append("text")
366      .attr("x", function(d) {
367        for(var i=0; i<relations.length; i++)
368            if(relations[i]['parent']==d.name&&d.depth!=maxDep) return -15;
369        return 15;
370        })
371      .attr("dy", ".35em")
372      .attr("text-anchor", function(d) {
373        for(var i=0; i<relations.length; i++)
374            if(relations[i]['parent']==d.name&&d.depth!=maxDep) return "end";
375        return "start";
376        })
```

Εικόνα 4.28: Ο κώδικας του αρχείου ‘code.js’ (9)

Στις γραμμές 336 με 340, γίνεται διόρθωση στο ‘width’ των κόμβων, για τη προσαρμογή τους στην οθόνη, καθώς παρατηρήθηκαν προβλήματα με τις προεπιλεγμένες τιμές της d3, πιθανώς λόγω του περιορισμένου χώρου, καθώς το του δέντρου κατέχει μόνο το 40% περίπου της οθόνης, ενώ η d3 συνήθως χρησιμοποιείται για εφαρμογές που καταλαμβάνουν όλη την οθόνη. Η διόρθωση γίνεται βρίσκοντας το μέγιστο επίπεδο, και χρησιμοποιώντας τη σχέση ‘d.depth * (width / (maxDep+1))’, δηλαδή το επίπεδο του κόμβου επί του πηλίκου της διαίρεσης του συνολικού πλάτους του ‘tree’, με το μέγιστο πλάτος αυξημένο κατά ένα. Η σχέση αυτή προέκυψε μετά από προσωπικές δοκιμές σε διάφορες πιθανές σχέσεις, καθώς θεωρήθηκε η πιο ιδανική για τη διατήρηση μίας καλής εικόνας σε διάφορα μεγέθη παραθύρου.

Στη συνέχεια, στις γραμμές 342 έως 360, γίνεται η είσοδος των κόμβων στο στοιχείο <g>, το οποίο είναι ‘παιδί’ του <svg>. Τα στοιχεία αυτά υπάρχουν ήδη στην ιστοσελίδα. Όπως αναφέρθηκε στο κεφάλαιο της βιβλιοθήκης d3, η είσοδος γίνεται σε δύο βήματα. Αρχικά (γραμμές 343-344), εκτελείται η εντολή ‘selectAll’. Συνεχίζοντας, αφού τα δεδομένα βρίσκονται πλέον στο enter set, με την εντολή ‘enter()’, στη γραμμή 347, τα δεδομένα εισάγονται στην ιστοσελίδα. Όπως φαίνεται, τους προστίθενται και κάποιες ιδιότητες με την εισαγωγή τους (στις επόμενες γραμμές από τη 347, με τις εντολές ‘attr’ και ‘on’). Οι ιδιότητες αυτές, χρειάζονται για τη τοποθέτηση τους και για αλληλεπιδράσεις, όπως το κλικ στη συγκεκριμένη λίστα, με το οποίο πρέπει να εμφανιστούν τα μέλη της, ή το ‘data-tippy-content’, για την εμφάνιση του tooltip με τη περιγραφή της λίστας.

```
377         .text(function(d) { return d.name.substring(0, d.name.lastIndexOf("@")); })
378         .style("fill-opacity", 1);
379
380     // Declare the links…
381     var link = svg.selectAll("path.link")
382         .data(links, function(d) { return d.target.id; });
383
384     // Orismos xrwmatwn gia diaforetiko xrwma ana parent
385     var colors = ['red','yellow','green','blue','black','purple','hotpink','crimson','brown','coral','darkgreen','magenta','maroon','orangered'];
386     var deiktis = 0;
387     var linkColor = [];
388     nodes.forEach(function(d){
389       var used = false;
390       relations.forEach(function(e){
391           if((d.name==e.parent)&&!(used)){
392               if(colors[deiktis]){
393                   linkColor.push({'name' : d.name, 'color' : colors[deiktis++]});
394                   used = true;
395               }
396               else{
397                   deiktis = 0;
398                   linkColor.push({'name' : d.name, 'color' : colors[deiktis++]});
399                   used = true;
400               }}
401       });
402     });
403
404     // Enter the links.
405     link.enter().insert("path", "g")
406         .attr("class", "link")
407         .attr("d", diagonal)
408         .style("stroke", function(d){
409                             var ret;
410                             linkColor.forEach(function(e){
411                                 if(e.name==d.source.name) ret = e.color;
412                             });
413                             return ret;
414         });
415
416     // Prosthiki twn links gia tous ypoloipous parent (parent2,parent3,etc.)
417       nodes.forEach(function(couplingChild){
418           var count = 2;
```

Εικόνα 4.29: Ο κώδικας του αρχείου ‘code.js’ (10)

Στις γραμμές 362 με 378, γίνεται η προσθήκη του κύκλου και του κειμένου με το όνομα της λίστας, στο στοιχείο του κόμβου. Χωρίς τη προσθήκη αυτών, ο κόμβος υπάρχει στο DOM, αλλά δεν έχει κάποιο ‘σώμα’ για να αλληλεπιδράσουμε με αυτόν. Φυσικά, όπως και στον ίδιο τον κόμβο, έτσι και σε αυτά τα στοιχεία, δίνονται κάποιες ιδιότητες. Στον κύκλο δίνεται μόνο η ακτίνα του, ενώ στο κείμενο υπάρχουν διάφορες ιδιότητες, όπως το αν το κείμενο θα μπει δεξιά ή αριστερά απ’ τον κύκλο (ανάλογα με την ύπαρξη ή όχι παιδιών).

Αφού οι κόμβοι έχουν προστεθεί πλήρως (μαζί με τους κύκλους και το κείμενο τους), επόμενο βήμα είναι η προσθήκη των συνδέσεων (paths). Εδώ να αναφερθεί ότι η διαδικασία της συνάρτησης ‘listsMake()’ τελειώνει σε αυτό το σημείο, μέχρι το οποίο έχει ακολουθηθεί η ίδια ακριβώς διαδικασία, με διαφορές μόνο στις τιμές κάποιων ιδιοτήτων.

Η διαδικασία προσθήκης των path είναι παρόμοια με των κόμβων, με τις εντολές ‘selectAll’ και ‘enter()’ να απαιτούνται. Η μόνη προσθήκη είναι ένας επιλογέας χρώματος στα path για κάθε γονέα, ώστε να είναι ξεκάθαρες οι σχέσεις γονέων-παιδιών. Ο επιλογέας αυτός υλοποιείται στις γραμμές 385 με 402, και το επιλεγμένο χρώμα για κάθε γονέα καθορίζεται στη γραμμή 408 αλλάζοντας την ιδιότητα ‘stroke’ της CSS του συγκεκριμένου path.

code.js

```
419         while(couplingChild['parent'+count.toString()]){
420             var couplingParent = tree.nodes(root).find(function(d){
421                     return d['name'] === couplingChild['parent'+count.toString()];
422                 });
423
424             multiParents = [{
425                             parent: couplingParent,
426                             child: couplingChild
427                         }];
428
429             multiParents.forEach(function(multiPair) {
430                     link.enter().insert("path", "g")
431                     .attr("class", "link")
432                         .attr("d", function() {
433                             var oTarget = {
434                                 x: multiPair.child.x,
435                                 y: multiPair.child.depth*(width/(maxDep+1))
436                             };
437                             var oSource = {
438                                 x: multiPair.parent.x,
439                                 y: multiPair.parent.depth*(width/(maxDep+1))
440                             };
441                             return diagonal({
442                                 source: oSource,
443                                 target: oTarget
444                             });
445                         })
446                         .style("stroke", function(){
447                             var ret;
448                             linkColor.forEach(function(e){
449                                 if(e.name==multiPair.parent.name) ret = e.color;
450                             });
451                             return ret;
452                         });
453                 });
454             count++;
455     });
456 }
```

Εικόνα 4.30: Ο κώδικας του αρχείου ‘code.js’ (11)

Η τελευταία εργασία που εκτελείται στη συνάρτηση ‘treeMake()’, εμφανίζεται στις γραμμές 417 έως 455, και πρόκειται για τη προσθήκη των path για τους υπόλοιπους γονείς του κάθε κόμβου. Όπως αναφέρθηκε, η d3 μπορεί να συνδέσει μόνο ένα γονέα σε κάθε κόμβο, με την ιδιότητα ‘parent’. Για την ένωση των υπόλοιπων γονέων του κόμβου με αυτόν (εάν αυτοί υπάρχουν), απαιτείται η χειροκίνητη δημιουργία ενός path, με πηγή τον γονέα και προορισμό το παιδί (το ίδιο αποτέλεσμα θα παραχθεί και με αντίθετες τις αναθέσεις). Ο κώδικας που έχει γραφτεί σε αυτές τις γραμμές, εκτελεί αυτή τη χειροκίνητη δημιουργία path, για κάθε πατέρα του κόμβου (‘parent2’, ‘parent3’ κοκ.). Τελευταίο βήμα αυτού του κώδικα, είναι και η αναζήτηση και ανάθεση του κατάλληλου χρώματος στα paths που δημιουργούνται. Με αυτή τη διαδικασία κλείνει η σημαντικότερη συνάρτηση του αρχείου ‘code.js’.

code.js

```
533  // Show members on click.
534 function click(d) {
535      var mailPrint = [];
536      for(var i = 0; i<lists.length; i++){
537          if(lists[i]==d.name)
538              if(mails[i].includes("lists"))
539                  mailPrint.push({'mail' : mails[i], list: 1});
540              else{
541                  if(mails[i].includes(" ")){
542                      var str = "";
543                      for(var j=0; j<mails[i].split(" ").length - 3; j++)
544                          str = str + (mails[i].split(" ")[j]) + " ";
545                      str = str.trimEnd();
546                      mailPrint.push({'mail' : str + " &lt;" + mails[i].split(" ")[mails[i].split(" ").length - 2] + "&gt;"});
547                  }
548                  else mailPrint.push({'mail' : mails[i]});
549              }
550      }
551
552      // Prosthiki stoixeiwn sto panel
553      // Title
554      document.getElementById("showhead").innerHTML = "";
555      var el = document.createElement("h3");
556      el.style = "text-align : center; ";
557      el.innerHTML = d.name;
558      document.getElementById("showhead").appendChild(el);
559
560      // Members
561      document.getElementById("show").innerHTML = "";
562      el = document.createElement("h3");
563      el.style = "text-align : center; color: blue; cursor: default;";
564      el.innerHTML = "Members:";
565      document.getElementById("show").appendChild(el);
566
567      // An den uparxoun members, tote emfanizw katallilo minima, alliws emfanizw ta members
568      if(mailPrint.length == 0){
569          el = document.createElement("p");
570          el.style = "text-align : center";
571          el.innerHTML = "This list has no members";
572          document.getElementById("show").appendChild(el);
573      }
574      else{
```

Εικόνα 4.31: Ο κώδικας του αρχείου ‘code.js’ (12)

Η τελευταία συνάρτηση στο αρχείο ‘code.js’, είναι η ‘click’. Η συνάρτηση αυτή καλείται όταν γίνεται κλικ σε ένα κόμβο. Έχει σκοπό την εμφάνιση των μελών της επιλεγμένης λίστας, και τη προσθήκη μίας επιλογής για ανακατεύθυνση στη σελίδα διαχειριστή αυτής. Αρχικά, στις γραμμές 535 με 550, γίνεται ανάκτηση των μελών προς εμφάνιση. Αυτό γίνεται με την αναζήτηση του ονόματος της λίστας στον πίνακα ‘lists’ που δημιουργήθηκε στην αρχική ανάκτηση δεδομένων, και με την ανάκτηση του στοιχείου στον ίδιο δείκτη ‘i’ από τον πίνακα ‘mails’. Πιο απλά, επειδή οι σχέσεις έχουν αποθηκευτεί ως ‘lists[i] γονέας του mails[i]’, βρίσκοντας τα ‘i’ στον πρώτο πίνακα, ανακτάμε τα μέλη από τον δεύτερο. Αν το μέλος είναι λίστα προστίθεται ένα αναγνωριστικό, ώστε να προστεθεί η μπλε ένδειξη ‘(LIST)’ δίπλα του, κατά την εμφάνιση. Επίσης, σε περίπτωση που υπάρχει και όνομα και διεύθυνση email για το συγκεκριμένο μέλος, προστίθενται τα ‘<’ και ‘>’ που είχαν αφαιρεθεί κατά την ανάκτηση των δεδομένων, διότι προκαλούσαν λειτουργικά προβλήματα.

Στη συνέχεια δημιουργούνται και ενσωματώνονται κάποια στοιχεία στο της εμφάνισης μελών. Πρώτα προστίθεται ο τίτλος της λίστας στο ‘head’ του (γραμμές 554 με 558). Εάν δεν υπάρχουν μέλη στη λίστα, εμφανίζεται το αντίστοιχο μήνυμα. Αυτό ελέγχεται και εκτελείται στις γραμμές 568 με 573.

```
code.js
575         // Make the list
576         listElement = document.createElement('ol'),
577         document.getElementById("show").appendChild(listElement);
578         var listItem;
579
580         for (var i = 0; i < mailPrint.length; i++) {
581             // Create an item for each mail
582             listItem = document.createElement('li');
583
584             // Add the item text
585             if(mailPrint[i].list){
586                 listItem.innerHTML = mailPrint[i].mail;
587                 listItem.onmouseover =  function() {
588                     this.style = "color : red;";
589                 };
590                 listItem.onmouseout =  function() {
591                     this.style = "color : black;";
592                 };
593
594                 var sp = document.createElement("span");
595                 sp.innerHTML = " (LIST)";
596                 sp.style = "-webkit-user-select: none; -ms-user-select: none; user-select: none; color: blue;";
597                 listItem.appendChild(sp);
598             }
599             else {
600                 listItem.innerHTML = mailPrint[i].mail;
601                 listItem.onmouseover =  function() {
602                     this.style = "color : red;";
603                 };
604                 listItem.onmouseout =  function() {
605                     this.style = "color : black;";
606                 };
607             }
608
609             // Add listItem to the listElement
610             listElement.appendChild(listItem);
611         }
612     }
```

Εικόνα 4.32: Ο κώδικας του αρχείου ‘code.js’ (13)

Στις γραμμές 574 με 612, δημιουργείται μία ‘ordered list’ και μέσα σε αυτή προστίθενται τα μέλη, το καθένα σε ένα ‘li’ στοιχείο. Φυσικά δίνονται και ιδιότητες, όπως η ιδιότητα να γίνεται το χρώμα του κειμένου κόκκινο όταν το ποντίκι βρίσκεται επάνω του. Σε αυτό το σημείο τα μέλη έχουν ήδη φανεί στην ιστοσελίδα.

```
code.js
613
614         // Button gia redirect sto list admin
615         var a = document.createElement("a");
616         a.innerHTML = "Redirect to admin page";
617         a.href = admin_base + d.name.substring(0, d.name.lastIndexOf("@"));
618         a.style = "font-size: large";
619         el = document.createElement("p");
620         el.style = "text-align : center;";
621         el.appendChild(a);
622         document.getElementById("show").appendChild(el);
623     }
624
625     // Xtizontai xana ta trees se kathe window resize
626     window.onresize = function() {
627         listsMake();
628         treeMake();
629         tippy('[data-tippy-content]');
630     };
631
632     // For the tooltips
633     tippy('[data-tippy-content]');
634
```

Εικόνα 4.33: Ο κώδικας του αρχείου ‘code.js’ (14)

Η τελευταία προσθήκη στο πλαίσιο εμφάνισης των μελών, είναι αυτό του κειμένου για ανακατεύθυνση στη σελίδα διαχειριστή. Αυτό υλοποιείται στις γραμμές 615 με 622, και η συνάρτηση ‘click’ κλείνει με αυτό, στη γραμμή 623. Ο τελευταίος κώδικας του αρχείου είναι οι γραμμές 626 με 633. Αυτές περιέχουν δύο πολύ απλές εργασίες, το κάλεσμα των συναρτήσεων για το χτίσιμο των δέντρων, όταν παρατηρηθεί αλλαγή στο μέγεθος του παραθύρου, και το κάλεσμα της βιβλιοθήκης ‘tippy’ για την εμφάνιση των tooltip. Παρατηρείται ότι το κάλεσμα της ‘tippy’ γίνεται δύο φορές, μία στο ‘window resize’ και μία ακριβώς μετά. Η κλήση εντός του ‘window resize’ απαιτείται διότι εάν οι κόμβοι σβηστούν και δημιουργηθούν ξανά, πρέπει να κληθεί και η ‘tippy’ για να αναγνωρίσει τα νέα αυτά στοιχεία και να λειτουργεί επάνω τους. Φυσικά η κλήση εκτός του ‘window resize’ γίνεται για την αρχική κλήση της ‘tippy’.

4.5 Επίλογος

Στο κεφάλαιο αυτό παρουσιάστηκε η πρακτική υλοποίηση αυτής της Πτυχιακής Εργασίας. Η υλοποίηση αυτή αποτελεί το σημαντικότερο μέρος, καθώς η χρησιμότητα της είναι μεγάλη. Αρχικά έγινε μία παρουσίαση της εφαρμογής, δείχνοντας κάποιες από τις λειτουργίες της. Στη συνέχεια, για τη κατανόηση των λειτουργιών αυτών, έγινε μία εκτενής ανάλυση στα αρχεία δεδομένων, στα αρχεία ανάκτησης αυτών, στα αρχεία με κώδικα JavaScript και CSS και γενικά σε όλα τα απαιτούμενα αρχεία που κατασκευάστηκαν κατά την υλοποίηση αυτής της εφαρμογής. Επίσης, έγιναν αναφορές σε μειονεκτήματα και σε μη βολικά χαρακτηριστικά των βιβλιοθηκών που χρησιμοποιήθηκαν (κυρίως της d3), καθώς και το πως αυτά επιλύθηκαν, για την επίτευξη της τελικής, πλήρως λειτουργικής αυτής εφαρμογής.

Κεφάλαιο 5ο: Συμπεράσματα ή/και προτάσεις βελτίωσης

Σε αυτήν τη Πτυχιακή Εργασία υλοποιήθηκε μία πρακτική εφαρμογή Οπτικοποίησης Λιστών Ηλεκτρονικού Ταχυδρομείου. Η επιλογή των βιβλιοθηκών και των τεχνολογιών που χρησιμοποιήθηκαν, έγινε με σκοπό τη καλύτερη δυνατή υλοποίηση αλλά και εύκολη συντήρηση. Επίσης σε πολλά σημεία λήφθηκαν μέτρα για τη διαχείριση ενός μεγαλύτερου συνόλου από λίστες, για τη περίπτωση πιθανής προσθήκης πλήθους λιστών στο Mailman του τμήματος.

Η χρήση της HTML5 προσφέρει τρομερά πλεονεκτήματα, όπως σταθερή λειτουργικότητα ανεξαρτήτως συσκευής, ανεξαρτησία από πρόσθετες τεχνολογίες, και φυσικά τις νέες δυνατότητες που παρουσιάστηκαν στο 1ο κεφάλαιο αυτής της εργασίας. Οι υπόλοιπες βιβλιοθήκες επιλέχθηκαν με κριτήριο την ευκολία μάθησης σε σχέση με τις δυνατότητες. Η d3 είχε ελάχιστα κενά ως βιβλιοθήκη, τα οποία επιλύθηκαν χειροκίνητα στον κώδικα, ή με τη χρήση βοήθειας μίας άλλης βιβλιοθήκης, όπως η ‘graphviz’. Οι υπόλοιπες βιβλιοθήκες δεν είχα κάποια λειτουργική έλλειψη και η επιλογή τους έγινε σωστά, καθώς πρόκειται για βιβλιοθήκες που είναι απλές και λαμβάνουν υποστήριξη.

Μία πιθανή βελτίωση θα ήταν η εξ’ ολοκλήρου χρήση της βιβλιοθήκης ‘graphviz’ αντί της ‘d3’, διότι μετά από μελέτη ανακαλύφθηκε ότι πιθανώς να μπορεί να χτίσει τα αναγκαία δέντρα, χωρίς εξωτερική βοήθεια. Ωστόσο, αυτή η βιβλιοθήκη εντοπίστηκε κατά την αναζήτηση μίας λύσης για την ιεραρχία, σημείο στο οποίο η υλοποίηση με την ‘d3’ είχε σχεδόν ολοκληρωθεί. Αυτό το γεγονός καθιέρωσε τη χρήση της ‘d3’ τελικά. Επίσης, η ‘d3’ έχει αρκετά πλεονεκτήματα σε σχέση με τη ‘graphviz’, κυρίως στις δυνατότητες εμφάνισης των κόμβων, γεγονός που εκτιμάται στη συγκεκριμένη Πτυχιακή Εργασία.

Μία μικρή βελτίωση επίσης ίσως να ήταν η χρήση μίας συνάρτησης τυχαίας δημιουργίας χρώματος, στο σημείο που επιλέγονται χρώματα για τα path. Ωστόσο, επιλέχθηκε να δοθεί έτοιμη λίστα με χρώματα, για την αποφυγή της τυχαίας δημιουργίας παρόμοιων χρωμάτων, η οποία θα οδηγούσε στη στέρηση της δυνατότητας αντίληψης των ξεχωριστών σχέσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Raúl Tabarés Gutiérrez, Understanding the role of digital commons in the web; The making of HTML5, Telematics and Informatics, Volume 35, Issue 5, 2018, Pages 1438-1449, ISSN 0736-5853. Available: https://doi.org/10.1016/j.tele.2018.03.013

[2] McLaughlin, Brett. What is HTML5?. " O'Reilly Media, Inc.", 2011. Available: https://www.oricaminingservices.com/uploads/What.Is.HTML5.pdf

[3] PILGRIM, Mark. Dive into HTML5, 2010. Available: https://jbwyatt.com/270/dive.pdf

[4] Fulton, Steve, and Jeff Fulton. HTML5 canvas:

native interactivity and animation for the web. "O'Reilly Media, Inc.", 2013. Available: http://wallizard.com/eric/share/%5BHTML%5D%5BHTML5%20Canvas%5D.pdf

[5] Daoust, François, et al. "Towards Video on the Web with HTML5." NEM Summit (2010). Available: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.357.3415&rep=rep1&type=pdf

[6] Xing Yan, Lei Yang, Shanzhen Lan and Xiaolong Tong, "Application of HTML5 multimedia," 2012 International Conference on Computer Science and Information Processing (CSIP), 2012, pp. 871-874. Available: https://ieeexplore.ieee.org/abstract/document/6308992

[7] Lubbers P., Albers B., Salim F. (2010) Working with HTML5 Audio and Video. In: Pro HTML5 Programming. Apress. Available: https://link.springer.com/chapter/10.1007/978-1-4302-2791-5_3

[8] Bahor, Senad. “HTML 5 / WebGL vs Flash in 3 D Visualisation.” (2013). Available: https://www.semanticscholar.org/paper/HTML-5-%2F-WebGL-vs-Flash-in-3-D-Visualisation-Bahor/994f02fc9c3ae1d6b2d0e2ff8f2db3dc40aefcdb?p2df

[9] Johan Harjono, Gloria Ng, Ding Kong, and Jimmy Lo. 2010. Building smarter web applications with HTML5. In Proceedings of the 2010 Conference of the Center for Advanced Studies on Collaborative Research (CASCON '10). IBM Corp., USA, 402–403.

Available: https://dl.acm.org/doi/abs/10.1145/1923947.1924015

[10] Lubbers P., Albers B., Salim F. (2010) Using the HTML5 Web Workers API. In: Pro HTML5 Programming. Apress. Available: https://link.springer.com/chapter/10.1007/978-1-4302-2791-5_8

[11] Corcoran, Padraig and Mooney, Peter and Winstanley, Adam C. and Bertolotto, Michela (2011) Effective Vector Data Transmission and Visualization Using HTML5. In: Proceedings of the 19th GIS Research UK Annual Conference. University of Portsmouth with Ordnance Survey, pp. 179-183. Available: http://mural.maynoothuniversity.ie/4925/

[12] Eisenberg, J. David, and Amelia Bellamy-Royds. SVG Essentials: Producing Scalable Vector Graphics with XML. " O'Reilly Media, Inc.", 2014.

Available: http://www.tmapsonline.com/SVG_Essentials.pdf

[13] W. Hu, H. Yuan, J. Wang and L. Wang, "The research and application of power system visualization based on HTML5," 2011 4th International Conference on Electric Utility Deregulation and Restructuring and Power Technologies (DRPT), 2011, pp. 1562-1565. Available: https://ieeexplore.ieee.org/abstract/document/5994145

[14] Pfeiffer S. (2010) HTML5 Media and SVG. In: The Definitive Guide to HTML5 Video. Apress. Available: https://link.springer.com/chapter/10.1007/978-1-4302-3091-5_5

[15] Heydt, Michael. D3. js by example. Packt Publishing Ltd, 2015. Available: https://www.uplooder.net/ofiles/e7bc21c4acd55112f6291ec892a37242/D3.js-By-Example.pdf

[16] Körner, Christoph. Data Visualization with D3 and AngularJS. Packt Publishing Ltd, 2015. Available: https://services.math.duke.edu/courses/math_everywhere/assets/techRefs/Data%20Visualization%20with%20D3%20and%20AngularJS.pdf

[17] Nair, L., Shetty, S. & Shetty, S. (2016). Interactive visual analytics on Big Data: Tableau vs D3.js. Journal of e-Learning and Knowledge Society. 12 (4),. Italian e-Learning Association. Available: https://www.learntechlib.org/p/173675/

[18] Fan Bao and Jia Chen, "Visual framework for big data in d3.js," 2014 IEEE Workshop on Electronics, Computer and Applications, 2014, pp. 47-50.

Available: https://ieeexplore.ieee.org/abstract/document/6845553

[19] Iglesias, Marcos. Pro D3. js. Apress, 2019.

Available: https://link.springer.com/content/pdf/10.1007/978-1-4842-5203-1.pdf

[20] K. Takahashi, A. Sakai and K. Sakurai, "Invalidation of Mailing List Address to Block Spam Mails," 2008 IEEE Asia-Pacific Services Computing Conference, 2008, pp. 841-846. Available: https://ieeexplore.ieee.org/document/4780780

[21] Oda, Terri. "GNU Mailman-List Member Manual." Last accessed August 5th (2004). Available: https://www.exploratorium.edu/files/cmp/exnet/about/media/mailman-member.pdf

[22] Manheimer, Ken, and John Viega. "Mailman–An Extensible Mailing List Manager Using Python." Proceedings of the 7th International Python Conference. 1998.

Available: https://eastcoastjam.com/myr/software-and-systems/craft/mailman_ip7.pdf

[23] Warsaw, Barry A. "GNU Mailman, Internationalized." USENIX Annual Technical Conference, FREENIX Track. 2003.

Available: https://www.usenix.org/legacy/events/usenix03/tech/freenix03/warsaw.html