ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Πτυχιακή εργασία

Διαδικτυακά εργαλεία για μέτα-αναζήτηση στον Παγκόσμιο Ιστό

Του φοιτητή

Τσουντικίδη Ανδρέα

Αρ. Μητρώου: 03/2199

Επιβλέπων καθηγητής

Σιδηρόπουλος Αντώνης

Θεσσαλονίκη 2010

Πρόλογος

Τα τελευταία χρόνια το διαδίκτυο έχει εισέρθει στις ζωές μας και έχει επιφέρει πολύ σημαντικές αλλαγές. Το διαδίκτυο έχει αλλάξει τον τρόπο που εργαζόμαστε, διασκεδάζουμε, μορφωνόμαστε και επικοινωνούμε. Πολλοί ερευνητές υποστηρίζουν ότι η επιρροή του διαδικτύου έχει αλλάξει την ίδια τη δομή και φύση της κοινωνίας μας.

Η κυριότερη υπηρεσία που προσφέρει το διαδίκτυο και την οποία χρησιμοποιούν εκατομμύρια άτομα από όλο τον κόσμο είναι αυτή του web. Όλοι μας έχουμε «σερφάρει» σε ιστοσελίδες του διαδικτύου ψάχνοντας κάποια πληροφορία ή είδηση. Το σημαντικότερο πλεονέκτημα όμως του διαδικτύου, ο τεράστιος όγκος πληροφοριών που προσφέρει μας παρουσιάζει ένα σημαντικό πρόβλημα. Είναι πολύ δύσκολο να βρεις την πληροφορία που χρειάζεσαι μέσα σε έναν τεράστιο όγκο δεδομένων.

Για να επιλυθεί αυτό το πρόβλημα αναπτύχθηκαν, από τα πρώτα βήματα του διαδικτύου, ειδικά προγράμματα που μας βοηθούν στην αναζήτηση πληροφορίας, οι μηχανές αναζήτησης. Σε συνδυασμό με την σχετικά πρόσφατη τάση στο διαδίκτυο, αυτή της δυνατότητας του χρήστη να τροποποιεί την εμφάνιση και λειτουργικότητα του browser του, επιλέχθηκε το αντικείμενο αυτής της πτυχιακής. Η δημιουργία ενός toolbar για τον Mozilla Firefox που θα προσφέρει δυνατότητες μετα-αναζήτησης στον παγκόσμιο ιστό.

Περίληψη

Η μετα-αναζήτηση είναι μια ειδικής μορφής αναζήτησης κατά την οποία η μηχανή αναζήτησης στέλνει το ερώτημά της σε άλλες μηχανές και στη συνέχεια συμψηφίζει τα αποτελέσματα με βάση κάποιους κανόνες για να δημιουργήσει μια ενιαία λίστα αποτελεσμάτων. Στην εργασία αυτή παρουσιάζεται ο τρόπος δημιουργίας ενός extension για τον browser Mozilla Firefox. Συγκεκριμένα παρουσιάζονται όλες οι τεχνολογίες που χρησιμοποιούνται για αυτή τη διαδικασία όπως JavaScript, CSS, XUL και XPCOM. Στη συνέχεια γίνεται μια ανάλυση του εργαλείου μετα-αναζήτησης που θέλουμε να δημιουργήσουμε και η υλοποίησή ως toolbar για τον Mozilla Firefox.

Abstract

Meta-search is a special form of web search in which the search engine sends its search query to other search engines and afterwards combines the results using some rules, in order to create a unified list of results. In this paper is presented the way to create an extension for the browser Mozilla Firefox. Specifically we are examining the technologies used for this process like JavaScript, CSS, XUL and XPCOM. Finally there is an analysis of the search tool that we want to create and the development as a toolbar for Mozilla Firefox.

Ευχαριστίες

Αρχικά θέλω να ευχαριστήσω τους γονείς μου για την οικονομική και ηθική υποστήριξη που μου προσέφεραν κατά τη διάρκεια της φοιτητικής μου, και όχι μόνο, ζωής. Επίσης θα ήθελα να ευχαριστήσω τον κύριο Σιδηρόπουλο ο οποίος ήταν πάντα προσιτός και διαθέσιμος να με καθοδηγήσει και χωρίς την βοήθεια του οποίου η εργασία δεν θα είχε ολοκληρωθεί. Τέλος θέλω να ευχαριστήσω τον καλό μου φίλο Βασίλη για τις συμβουλές του στα προβλήματα που αντιμετώπισα.

Περιεχόμενα

Εισαγωγή

Ο στόχος αυτής της εργασίας είναι η δημιουργία ενός toolbar για τον Mozilla Firefox που θα εκτελεί λειτουργίες μετα-αναζήτησης. Στο πρώτο κεφάλαιο γίνεται μια παρουσίαση της υπηρεσίας αναζήτησης στον Ιστό και παρουσιάζονται οι βασικές αρχές λειτουργίας των μηχανών αναζήτησης και μετα- αναζήτησης.

Στο δεύτερο κεφάλαιο παρουσιάζεται η διαδικασία δημιουργίας ενός toolbar για τον Mozilla Firefox και οι επιμέορυς τεχνολογίες που χρησιμοποιούνται. Συγκεκριμένα παρουσιάζονται οι τεχνολογίες XUL και XPCOM της πλατφόρμας της Mozilla.

Στο τρίτο, τέταρτο, πέμπτο και έκτο κεφάλαιο αναλύονται οι τεχνολογίες JavaScript, XML, DOM και CSS οι οποίες χρησιμοποιήθηκαν εκτεταμένα για την υλοποίηση του στόχου της πτυχιακής.

Τέλος στο έβδομο κεφάλαιο γίνεται η ανάλυση και υλοποίηση της ζητούμενης εφαρμογής. Συγκεκριμένα αναλύονται οι απαιτήσεις της εφαρμογής και προσδιορίζονται οι στόχοι της. Στην συνέχεια παρουσιάζεται η υλοποίηση της εφαρμογής μαζί με κομμάτια κώδικα.

1. Μηχανές αναζήτησης

Όταν δημιουργήθηκε το web το 1991, οι διαθέσιμοι servers ήταν τόσο λίγοι ώστε ήταν δυνατό να κρατηθεί αρχείο με τις διευθύνσεις τους στον server του CERN. Καθώς όμως το web διαδόθηκε δημιουργήθηκε η ανάγκη για μια αποτελεσματικότερη μέθοδο. Έτσι δημιουργήθηκαν οι μηχανές αναζήτησης. Μηχανές αναζήτησης είναι προγράμματα που ψάχνουν για πληροφορία στο διαδίκτυο. Αυτή η πληροφορία μπορεί να είναι ιστοσελίδες εικόνες ή άλλοι τύποι αρχείων.

Η πρώτη μηχανή αναζήτησης δημιουργήθηκε το 1993 και ονομάζονταν W3Catalog. Αυτή η πρωτόγονη μηχανή αναζήτησης έψαχνε σε καταλόγους ιστοσελίδων και εμφάνιζε τα αποτελέσματα με συγκεκριμένη μορφοποίηση. Η πρώτη μηχανή αναζήτησης με τη σημερινή έννοια ήταν το Jumpstation. Αυτή η μηχανή χρησιμοποιούσε ένα ειδικό τύπο αυτόματου προγράμματος που λέγεται web robot για την εύρεση πληροφορίας στο web και την οποία χρησιμοποιούσε για δημιουργήσει τον κατάλογό του. Οι μηχανές αναζήτησης γνώρισαν μεγάλη επιτυχία και έτσι τα επόμενα χρόνια πολλές νέες μηχανές. Έτσι φτάνουμε στην σημερινή εποχή όπου πλέον το internet είναι συνώνυμο των μηχανών αναζήτησης.

1.1 Λειτουργία των μηχανών αναζήτησης

Οι μηχανές αναζήτησης λειτουργούν ως εξής. Αρχικά συλλέγουν πληροφορίες για ιστοσελίδες από τον HTML κώδικα. Στη συνέχεια αποθηκεύουν την πληροφορία σε καταλόγους. Αυτή η πληροφορία είναι ανά πάσα στιγμή διαθέσιμη στο χρήστη ο οποίος υποβάλει ερωτήματα στα οποία η μηχανή απαντά με πληροφορίες από τον κατάλογο. Άρα μια μηχανή αναζήτησης εκτελεί τρεις λειτουργίες, web crawling, indexing και searching.

Η πρώτη λειτουργία αυτή του web crawling, είναι η διαδικασία με την οποία η μηχανή αναζήτησης συλλέγει πληροφορίες από τις ιστοσελίδες. Η διαδικασία αυτή εκτελείται με τη χρήση ειδικών αυτόματων προγραμμάτων που ονομάζονται web crawlers. Τα προγράμματα αυτά έχουν ως στόχο να δημιουργήσουν λίστες με τις λέξεις που εμφανίζονται στις ιστοσελίδες που επισκέπτεται. Ένας crawler ξεκινά την αναζήτησή του με μια λίστα από URL που του δίνονται. Καθώς επισκέπτεται αυτές τις σελίδες προσθέτει στη λίστα αυτή όλα τα links που βρίσκει. Υπάρχουν κάποια χαρακτηριστικά του web που κάνουν τη διαδικασία του crawling αρκετά δύσκολη. Αυτά είναι ο μεγάλος όγκος δεδομένων του web, ο γρήγορος ρυθμός με τον οποίο αλλάζουν τα δεδομένα αυτά και το ότι πολλές ιστοσελίδες δημιουργούνται δυναμικά.

Η δεύτερη λειτουργία μιας μηχανής αναζήτησης είναι το indexing. Η λειτουργία αυτή δέχεται ως είσοδο τις λίστες που δημιουργεί ο web crawler. Κάθε μηχανή αναζήτησης με βάση τους δικούς της κανόνες δημιουργεί το index, το οποίο είναι ένας κατάλογος που συνδέει λέξεις κλειδιά με ιστοσελίδες και βαθμολογεί τις σχέσεις αυτές ανάλογα με τη συνάφειά τους. Στη συνέχεια τα δεδομένα αυτά κωδικοποιούνται ώστε να μειωθεί ο χώρος που καταλαμβάνουν στο δίσκο. Γενικά ο στόχος του indexing είναι η ταχύτητα. Πρέπει δηλαδή η πληροφορία να μπορεί να ανακτηθεί όσο το δυνατόν ταχύτερα.

Η τελευταία λειτουργία είναι αυτή της αναζήτησης από τον χρήστη. Όταν γίνεται από τον χρήστη υποβολή ενός ερωτήματος η μηχανή αναζήτησης ψάχνει στο index της και παρουσιάζει στο χρήστη μια λίστα με τα πιο σχετικά αποτελέσματα. Ο χρήστης πρέπει λοιπόν να δημιουργήσει ένα ερώτημα κάθε φορά που θέλει να ανακτήσει πληροφορία από το index. Οι μηχανές αναζήτησης υποστηρίζουν απλά ερωτήματα μιας λέξης ή σύνθετα ερωτήματα με τη χρήση τελεστών όπως AND, OR, και NOT.

1.2 Μηχανές μετα-αναζήτησης

Διάφορες σημαντικές μηχανές αναζήτησης όπως οι AltaVista και Google έχουν προσπαθήσει να συντάξουν ευρετήριο για ολόκληρο το Web και να παρέχουν τη δυνατότητα αναζήτησης στις σελίδες όλου του Web. Εντούτοις, αυτές οι μηχανές αναζήτησης πάσχουν από διάφορους περιορισμούς. Παραδείγματος χάριν, η κάλυψη του Web από κάθε μία τους είναι περιορισμένη λόγω διάφορων λόγων όπως ο αποκλεισμός ρομπότ και η έλλειψη κατάλληλων συνδέσεων. Ένα άλλο παράδειγμα είναι, ότι καθώς το index αυτών των μηχανών αναζήτησης γίνεται μεγαλύτερο, το μεγαλύτερο μέρος των πληροφοριών που περιέχουν γίνεται ξεπερασμένο.

Σαν απάντηση στο παραπάνω πρόβλημα δημιουργήθηκαν οι μηχανές μετα-αναζήτησης. Μια μηχανή μετα-αναζήτησης είναι ένα εργαλείο αναζήτησης που στέλνει τα αιτήματα χρηστών σε διάφορες άλλες μηχανές αναζήτησης ή βάσεις δεδομένων και αθροίζει τα αποτελέσματα σε έναν ενιαίο κατάλογο ή τα επιδεικνύει σύμφωνα με την πηγή τους. Οι μηχανές μετα-αναζήτησης λειτουργούν με βάση την υπόθεση ότι το Web είναι πολύ μεγάλο για να συντάξει ευρετήριο του μια μηχανή αναζήτησης και ότι ακριβέστερα αποτελέσματα αναζήτησης μπορούν να επιτευχθούν με το συνδυασμό των αποτελεσμάτων από διάφορες μηχανές αναζήτησης.

Οι μηχανές μετα-αναζήτησης δημιουργούν αυτό που είναι γνωστό ως εικονική βάση δεδομένων. Δεν συντάσσουν μια φυσική βάση δεδομένων ή έναν κατάλογο του Web. Αντίθετα δέχονται το αίτημα ενός χρήστη, το περνούν σε διάφορες άλλες βάσεις δεδομένων και στη συνέχεια συντάσσουν τα αποτελέσματα κατά τρόπο ομοιογενή βασισμένο σε έναν συγκεκριμένο αλγόριθμο.

Συγκεκριμένα οι μηχανές μετα-αναζήτησης εκτελούν τις παρακάτω υπολειτουργίες.

- Δέχονται το ερώτημα από το χρήστη.
- Εκτελούν μια προεργασία στο ερώτημα και το αποστέλλουν σε μηχανές αναζήτησης και βάσεις δεδομένων.
- Συλλέγουν τα αποτελέσματα από τις διάφορες πηγές.
- Συνδυάζουν τα αποτελέσματα σε μια ενιαία λίστα.
- Βαθμολογούν τα αποτελέσματα ως προς τη σχετικότητα.
- Παρουσιάζουν τα αποτελέσματα στο χρήστη.

2. Extensions για Mozilla Firefox

Όταν κυκλοφόρησε η έκδοση 2.0 του browser της mozilla ένα από τα σημαντικότερα χαρακτηριστικά του ήταν η χρήση add-ons. Τα add-ons έδιναν τη δυνατότητα να τροποποιήσουμε τον browser ανάλογα με τις ανάγκες μας. Μια από τις κατηγορίες add-ons που υποστηρίζει ο Firefox είναι τα extensions. Ένα extension είναι μια μικρή εφαρμογή που προσθέτει μια καινούργια λειτουργία στον Firefox. Σε αυτό το κεφάλαιο θα παρουσιάσουμε πως δημιουργείται ένα extension ξεκινώντας με τη δομή το είδος των αρχείων που αποτελούν το extension και συνεχίζοντας με τις τεχνολογίες που χρησιμοποιούνται.

2.1 Περιεχόμενα ενός extension

Τα extension του Firefox συμπιέζονται σε zip αρχεία με την κατάληξη xpi. Ο λόγος είναι ότι ο Firefox αναγνωρίζει την κατάληξη και ξεκινά την διαδικασία εγκατάστασης του extension. Ένα xpi αρχείο περιέχει συνήθως την ακόλουθη δομή αρχείων που αποτελούν το extension.

- myextesion
  - /install.rdf
  - /chrome.manifest
  - /chrome/
  - /chrome/content/
  - /chrome/content/myextesion.xul
  - / chrome/content/myextesion.js
  - /defaults/
  - /defaults/preferences/

Δύο από τα σημαντικότερα αρχεία του extension είναι τα chrome.manifest και install.rdf τα οποία παρουσιάζονται στην συνέχεια.

2.2 To install manifest

Το install.rdf, το οποίο λέγεται install manifest, περιέχει πληροφορίες σχετικά με το extension που χρησιμοποιούνται από τον browser κατά την εγκατάσταση. Ένα τυπικό αρχείο έχει την ακόλουθη μορφή.

```
<?xml version="1.0"?>

<RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:em="http://www.mozilla.org/2004/em-rdf#">

  <Description about="urn:mozilla:install-manifest">
    <em:id>sample@example.net</em:id>
    <em:version>1.0</em:version>
    <em:type>2</em:type>

    <em:targetApplication>
      <Description>
        <em:id>{ec8030f7-c20a-464f-9b0e-13a3a9e97384}</em:id>
        <em:minVersion>1.5</em:minVersion>
        <em:maxVersion>3.5.*</em:maxVersion>
      </Description>
    </em:targetApplication>

    <em:name>sample</em:name>
    <em:description>A test extension</em:description>
    <em:creator>Your Name Here</em:creator>
    <em:homepageURL>http://www.example.com/</em:homepageURL>
  </Description>
</RDF>
```

Όπως βλέπουμε η rdf είναι μια xml based γλώσσα. Τα σημαντικά πεδία τα οποία πρέπει να ορίσουμε είναι τονισμένα με έντονη γραφή. Τα τρία πρώτα πεδία

(em:id, em:version και em:type) περιέχουν πληροφορίες σχετικά με το extension. Το em:id είναι ο κωδικός του extension σε μορφή email διεύθυνσης. Το em:version είναι η έκδοση του extension και το em:type είναι ο τύπος του add-on που προσπαθούμε να εγκαταστήσουμε, η τιμή 2 σημαίνει extension.

Η δεύτερη ομάδα δεδομένων κάτω από το em:targetApplication περιέχει πληροφορίες για την εφαρμογή στην οποία μπορεί να εγκατασταθεί το extension. Το em:id είναι ο μοναδικός GUID κωδικός της εφαρμογής. Τα em:minVersion και em:maxVersion είναι η μικρότερη και μεγαλύτερη έκδοση αντίστοιχα της εφαρμογής με την οποία μπορεί να δουλέψει το extension.

Τα υπόλοιπα στοιχεία περιέχουν πληροφορίες το extension και τον δημιουργό του. Το em:name είναι το όνομα του extension όπως θα εμφανίζεται στο interface. Το em:description είναι μια περιγραφή του extension. Το em:creator είναι το όνομα του δημιουργού και το em:homepageURL η ιστοσελίδα του.

2.3 Το chrome manifest

Το chrome είναι το σύνολο των στοιχείων interface του Firefox τα οποία βρίσκονται εκτός του χώρου φόρτωσης περιεχομένου. Δηλαδή toolbars, menu bars και τα λοιπά είναι μέρος του chrome. Όταν κάνουμε εγκατάσταση ένα extension θέλουμε να κάνουμε register όλα τα αρχεία που το αποτελούν στο chrome ώστε να τα συμπεριλάβει ο browser στο interface του.

Το chrome manifest δηλώνει αυτά τα αρχεία στην chrome register του Firefox. Μια τέτοια δήλωση έχει την μορφή

content packagename filepath

Το packagename είναι το όνομα του extension και filepath είναι το directory του αρχείου. Για παράδειγμα

content myextension chrome/content/

Όταν θέλουμε να προσθέσουμε κάποια στοιχεία στο interface του firefox χρησιμοποιούμε xul overlays. Για να κάνουμε register ένα xul overlay χρησιμοποιούμε την παρακάτω εντολή.

overlay chrome://browser/content/browser.xul chrome://myextension/content/myextension.xul

Η πρώτη παράμετρος είναι το xul αρχείο στο οποίο θα προστεθεί το overlay. Η δεύτερη παράμετρος είναι το αρχείο που περιέχει το overlay, δηλαδή το interface που θέλουμε να προσθέσουμε. Περισσότερα για overlays και xul θα πούμε στη συνέχεια.

Για να έχουμε πρόσβαση σε αρχεία του extension, τα οποία είναι μέρος του chrome χρησιμοποιούμε chrome URIs όπως το chrome://myextension/content/myextension.xul.

2.4 Το κυρίως μέρος του extension

Τα αρχεία που αποτελούν το κυρίως σώμα του extension βρίσκονται στον φάκελο content. Ένα extension του Firefox αποτελείται από δύο μέρη. Το interface είναι γραμμένο σε xul και JavaScript.

2.4.1 XUL

Η xul είναι μια xml based γλωσσά η οποία μας δίνει πρόσβαση σε διάφορα interface elements όπως menus, buttons, toolbars, textboxes και άλλα. Για να επεκτείνουμε τον browser, με ένα extension, αρκεί να προσθέσουμε στοιχεία στο interface ή να τροποποιήσουμε ήδη υπάρχοντα. Για να προσθέσουμε στοιχεία στο interface εισάγουμε νέα XUL DOM στοιχεία στο παράθυρο του browser.

Ο ίδιος ο browser υλοποιείται σε ένα xul αρχείο το browser.xul. Αν ανοίξουμε το αρχείο θα δούμε διάφορα xul στοιχεία όπως το

```
<statusbar id="status-bar">
 ...
</statusbar>
```

Το <statusbar id="status-bar"> αποτελεί ένα merge point το οποίο μπορεί να χρησιμοποιηθεί σε ένα xul overlay. Δηλαδή θα μας δείχνει σε ποιο σημείο του browser.xul θα προστεθεί ο κώδικας του overlay.

2.4.1.1 XUL Overlays

Τα xul αρχεία περιέχουν δύο πράγματα παράθυρα (windows) ή overlays. Τα overlays ορίζονται σε ξεχωριστά αρχεία και χρησιμοποιούνται για να προσθέσουν ή να αλλάξουν περιεχόμενο σε ένα ήδη υπάρχον παράθυρο όπως αυτό του browser. Κάθε στοιχείο τοποθετείται στο αρχείο σε θέση που καθορίζεται συγκρίνοντας IDs. Για παράδειγμα ας δούμε το ακόλουθο overlay.

```
<?xml version="1.0"?>
<overlay id="myOverlay"
        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">
 <menu id="file_menu">
  <menuitem name="Super Stream Player"/>
 </menu>
</overlay>
```

Σε αυτό το παράδειγμα όλα τα στοιχεία που είναι άμεσα child nodes του overlay αποτελούν merge points. Άρα θα ψάξει το στοιχείο με id file_menu στο αρχείο που βάζουμε το ovelay. Αν βρει τέτοιο στοιχείο θα κάνει append όλα τα child nodes του αντίστοιχου στοιχείου στο overlay. Στην πράξη αν εφαρμόζαμε το

overlay αυτό στο παράθυρο του browser θα προσθέταμε στο τέλος του μενού file μια επιλογή με όνομα «Super Stream Player».

2.4.1.2 XUL Elements

Στη συνέχεια παρουσιάζονται μερικά από τα βασικότερα xul στοιχεία.

- window: Ορίζει ένα νέο παράθυρο. Είναι το root element ενός xul αρχείου.
- overlay: Ορίζει ένα νέο overlay. Είναι το root element ενός xul αρχείου.
- toolbox: Είναι ένα container για toolbar στοιχεία.
- toolbar: Είναι ένα container για διάφορα στοιχεία ελέγχου, συνήθως toolbarbutton
- toolbarbutton: Είναι ένα απλό button control που χρησιμοποιείται σε toolbars
- vbox: Είναι ένα container που στοιχίζει τα στοιχεία που περιέχει κάθετα
- hbox: Είναι ένα container που στοιχίζει τα στοιχεία που περιέχει οριζόντια
- textbox: Ένα κλασικό textbox control στο οποίο εισάγουμε κέιμενο.
- script: Χρησιμοποιείται για να ορίσουμε ένα script το οποίο θα χρησιμοποιηθεί από το παράθυρο.
- checkbox: Ένα κλασικό checkbox control.

2.4.2 Chrome Javascript

Όπως είδαμε δημιουργούμε το interface του extension με τη χρήση xul. Για να προσθέσουμε λειτουργικότητα στα controls του extension μας χρησιμοποιούμε JavaScript. Μπορούμε να χρησιμοποιήσουμε το Dom του Firefox chrome για να αποκτήσουμε πρόσβαση στα xul στοιχεία του browser όπως ακριβώς κάνουμε για ένα HTML έγγραφο.

Για να συνδέσουμε xul με JavaScript χρησιμοποιούμε το xul στοιχείο script. Για παράδειγμα.

```
<script type="application/javascript" src="chrome://helloworld/content/clock.js"/>
```

Έτσι ο JavaScript κώδικας είναι ορατός από το xul και μπορούμε να συνδέσουμε μεθόδους με συγκεκριμένα controls. Για παράδειγμα ο ακόλουθος κώδικας μας λέει πως όταν κάνουμε click στο button θα εκτελείται μια συγκεκριμένη μέθοδος.

<toolbarbutton id="button1" oncommand="myfunction()"/>

Πάντα ο JavaScript κώδικας του extension είναι οργανωμένος σε μεθόδους καθώς θέλουμε να εκτελείται όταν συμβαίνουν συγκεκριμένα γεγονότα. Έχουμε τη δυνατότητα να χρησιμοποιήσουμε όλες τις δυνατότητες της core JavaScript. Η γλώσσα JavaScript περιγράφεται αναλυτικά σε επόμενο κεφάλαιο. Εκτός όμως από τις βασικές λειτουργίες που μας προσφέρει η core JavaScript έχουμε τη δυνατότητα επιπλέον λειτουργιών κάνοντας χρήση του XPCOM.

2.4.3 XPCOM

Το XPCOM (cross platform component object model) είναι ένα component object model παρόμοιο με το COM της Microsoft. Στην ουσία είναι ένας τρόπος για να υλοποιήσουμε αντικείμενα ανεξάρτητα από συγκεκριμένη γλώσσα προγραμματισμού. Το XPCOM μας προσφέρει ένα σύνολο από κλάσεις με τις οποίες μπορούμε να κάνουμε πολλές λειτουργίες όπως διαχείριση αρχείων και μνήμης.

Γενικά ο Firefox αποτελείται από δύο κομμάτια. Το μεγαλύτερο είναι μια πλατφόρμα γραμμένη σε C++ . Με βάση αυτή την πλατφόρμα λειτουργεί το chrome το οποίο είναι γραμμένο κυρίως σε XML, JavaScript και CSS. Η βάση του chrome είναι λοιπόν μια ισχυρή πλατφόρμα ανάπτυξης λογισμικού που λέγεται XULRunner. Το XULRunner χρησιμοποιείται για τη δημιουργία λογισμικού που δουλεύει ανεξαρτήτως λειτουργικού συστήματος.

Το XPCOM είναι το σύστημα με το οποίο το XULRunner και το chrome επικοινωνούν. Τα περισσότερα αντικείμενα και μέθοδοι του XULRunner είναι

κρυμμένα από το chrome και ο τρόπος για να αποκτήσουμε πρόσβαση σε αυτά είναι μέσω των interfaces και components του XPCOM.

Για να καλέσουμε XPCOM components χρησιμοποιούμε κώδικα της ακόλουθης μορφής.

```
service = Cc[''@mozilla.org/observer-
service;1''].getService(Ci.nsIObserverService);
```

Το αντικείμενο Cc είναι ένας πίνακας με όλες τις κλάσεις που είναι διαθέσιμες μέσω του XPCOM. Το string ανάμεσα στις αγκύλες είναι ο κωδικός της υπηρεσίας που θέλουμε να χρησιμοποιήσουμε. Τα components μπορούν να είναι είτε υπηρεσίες (στατικά αντικείμενα) ή στιγμιότυπα κάποιας κλάσης. Αναλόγως καλούμε την αντίστοιχη μέθοδο getService για υπηρεσίες ή createInstance για στιγμιότυπα κλάσης. Τέλος το Ci είναι ένας πίνακας με όλα τα διαθέσιμα interfaces. Ένα interface είναι ένα σύνολο από χαρακτηριστικά και μεθόδους τα οποία το αντικείμενο που το υλοποιεί πρέπει να έχει.

2.4.3.1 Διαχείριση αρχείων

Το XPCOM μας παρέχει κάποια έτοιμα interfaces που μας επιτρέπουν τη διαχείριση φακέλων και αρχείων. Καταρχάς χρειάζεται να δημιουργήσουμε ένα αντικείμενο nsILocalFile το οποίο αναπαριστά ένα τοπικό αρχείο. Με τη μέθοδο initWithPath().

```
var file = Components.classes['@mozilla.org/file/local;1']
                    .createInstance(Components.interfaces.nsILocalFile);
file.initWithPath('C:\\temp\\temp.txt');
```

Για να μετακινηθούμε σε ένα νέο directory χρησιμοποιούμε τη μέθοδο append().

```
file.initWithPath('C:\\');
file.append('Documents and Settings');
file.append('All Users');
file.append('Documents');
```

Για να διαβάσουμε και να γράψουμε ένα δυαδικό αρχείο χρησιμοποιούμε streams.

```
file.initWithPath('C:\\temp\\temp.txt');

var fileStream = Components.classes['@mozilla.org/network/file-input-stream;1']

                .createInstance(Components.interfaces.nsIFileInputStream);

fileStream.init(file, 1, 0, false);

var binaryStream = Components.classes['@mozilla.org/binaryinputstream;1']

                  .createInstance(Components.interfaces.nsIBinaryInputStream);

binaryStream.setInputStream(fileStream);

var array = binaryStream.readByteArray(fileStream.available());

binaryStream.close();

fileStream.close();
```

Για να διαβάσουμε και να γράψουμε σε ένα αρχείο κειμένου χρησιμοποιούμε το converter stream.

```
file.initWithPath('C:\\temp\\temp.txt');
var fileStream = Components.classes["@mozilla.org/network/file-output-stream;1"].createInstance(Components.interfaces.nsIFileOutputStream);
fileStream.init(file, 1, 0, false);
var converterStream = Components.classes["@mozilla.org/intl/converter-output-stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);
converterStream .init(foStream,"UTF-8",0,0);
converterStream.writeString (“Hello”);
converterStream.close();
fileStream.close();
```

2.4.3.2 Preferences

Το σύστημα προτιμήσεων του Firefox χρησιμοποιείται για να αποθηκεύσει προτιμήσεις χρηστών. Μπορούμε να χρησιμοποιήσουμε προτιμήσεις στα extension μας για να μπορέσουμε να χειριστούμε εύκολα τις προτιμήσεις. Τα preferences είναι τριών τύπων. Αριθμοί, Booleans και string. Για να ορίσουμε προτιμήσεις στο extension μας δημιουργούμε ένα αρχείο javascript στο directory /defaults/preferences/ και γράφουμε τα preferences με τον ακόλουθο τρόπο.

```
pref("extensions.xulschoolhello.message.count", 0);
```

Για να αλλάξουμε και να διαβάσουμε προτιμήσεις χρησιμοποιούμε το Preferences service του XPCOM.

```
var prefService = Cc["@mozilla.org/preferences-service;1"]

.getService(Ci.nsIPrefBranch);

var get = prefService.getIntPref("extensions.xulschoolhello.message.count");

var set = prefService.setIntPref("extensions.xulschoolhello.message.count", 5);

}
```

3. JavaScript

3.1 Εισαγωγή

Η JavaScript είναι μια ελαφριά, interpreted γλώσσα προγραμματισμού με αντικειμενοστρεφείς δυνατότητες. Ο πυρήνας της γλώσσας είναι ενσωματωμένος σε όλους τους σημαντικούς browsers όπως οι Firefox, Internet Explorer, Opera, Konqueror και άλλοι. Η γλώσσα έχει εξελιχθεί με στόχο το προγραμματισμό στο διαδίκτυο (Web Programming) με την προσθήκη αντικειμένων που αντιπροσωπεύουν το παράθυρο του browser και το περιεχόμενό του. Το αποτέλεσμα αυτής της εξέλιξης είναι η λεγόμενη client side JavaScript.

Όταν ενσωματώνουμε ένα διερμηνέα JavaScript σε έναν browser το αποτέλεσμα είναι η client side JavaScript. Αυτή είναι η πιό κοινή μορφή της JavaScript που συναντάμε. Στην ουσία η client side JavaScript συνδυάζει τη δυνατότητα scripting που προσφέρει η core JavaScript με το document object model (DOM) που ορίζεται από τον browser. Ο συνδυασμός αυτών των δύο τεχνολογιών επιτρέπει την ύπαρξη δυναμικού περιεχομένου στις ιστοσελίδες και αποτελεί τον πυρήνα της Dynamic HTML.

Όπως αναφέραμε η JavaScript είναι μία scripting γλώσσα προγραμματισμού. Οι γλώσσες αυτού του τύπου έχουν ως στόχο τον έλεγχο εφαρμογών. Γενικά δεν είναι φτιαγμένες με στόχο την δημιουργία εφαρμογών απο το μηδέν αλλά χρησιμοποιούνται σε συνδυασμό με components γραμμένα σε κλασσικές γλώσσες προγραμματισμού. Η ποιο συχνή λειτουργία τους είναι να επεκτείνουν τέτοια components προσθέτοντας επιπλέον λειτουργικότητα. Ως αποτέλεσμα του σκοπού τους οι scripting γλώσσες παρουσιάζουν κάποια κοινά χαρακτηριστικά. Συνήθως είναι interpreted, είναι typeless δηλαδή δεν χρειάζεται να ορίζουμε τον τύπο των μεταβλητών και τέλος έχουν συντακτικό με χαλαρούς κανόνες.

Το συντακτικό της JavaScript παρουσιάζει πολλές ομοιότητες με τις γλώσσες C και Java όπως στους τελεστές, στη σύνταξη των επαναληπτικών βρόγχων for και while και στη χρήση του if else statement. Εκτός όμως από αυτή την ομοιότητα η JavaScript έχει περισσότερες διαφορές. Καταρχάς η JavaScript είναι untyped γλώσσα, δηλαδή οι μεταβλητές που ορίζουμε δεν χρειάζεται να έχουν κάποιο συγκεκριμένο τύπο. Επιπλέον τα αντικείμενα στη JavaScript δεν μοιάζουν ιδιαίτερα με τα αντικείμενα που συναντάμε στη Java και τη C++.

Η JavaScript μας παρέχει πολλές δυνατότητες οι βασικότερες των οποίων παρουσιάζονται παρακάτω.

- Έλεγχος της εμφάνισης και του περιεχομένου του εγγράφου. Με τη χρήση του αντικειμένου Document μπορούμε να προσθέσουμε δυναμικά περιεχόμενο σε ένα HTML έγγραφο κατά τη φόρτωσή του από τον browser. Επιπλέον μπορούμε να εμφανίσουμε διαφορετικό περιεχόμενο ανάλογα με τον browser.
- Έλεγχος του browser. Διάφορα αντικείμενα JavaScript επιτρέπουν τον έλεγχο της συμπεριφοράς του browser. Το αντικείμενο Window υποστηρίζει μεθόδους για την εμφάνιση popup διαλόγων με τους οποίους μπορεί να αλληλεπιδράσει ο χρήστης. Επιπλέον υπάρχει η δυνατότητα για άνοιγμα ή κλείσιμο παραθύρων του browser.
- Αλληλεπίδραση με HTML φόρμες. Η JavaScript μας παρέχει το αντικείμενο Form με τη χρήση του οποίου μπορεί να μεταβάλλει τις τιμές των στοιχείων μιας HTML φόρμας. Η πιο σημαντική χρήση αυτής της δυνατότητας είναι για την επαλήθευση των δεδομένων που εισάγωνται σε μία φόρμα.
- Αλληλεπίδραση με το χρήστη. Ένα σημαντικό χαρακτηριστικό της JavaScript είναι η δυνατότητα να ορίσουμε event handlers. Event handlers είναι κομμάτια κώδικα που εκτελούνται όταν συμβεί κάποιο γεγονός. Για

παράδειγμα όταν γίνεται μια ενέργεια με το ποντίκι. Τα γεγονότα είναι από τους πιο σημαντικούς μηχανισμούς της JavaScript και του Dynamic HTML.

3.2 Ιστορία

Όταν το World Wide Web δημιουργήθηκε αρχικά στις αρχές της δεκαετίας του 1990 όλες ιστοσελίδες ήταν στατικές. Όταν βλέπαμε ιστοσελίδες βλέπαμε απλώς τη σελίδα όπως ήταν σχεδιασμένη για να παρουσιαστεί και δεν υπήρχε κανένας τρόπος για να αλληλεπιδράσουμε με τη σελίδα. Η δυνατότητα να αλληλεπιδράσει ο χρήστης με ιστοσελίδες απαιτούσε την προσθήκη κάποιας μορφής γλώσσας προγραμματισμού που θα καθοδηγούσε τη σελίδα για το πως θα ανταποκριθεί στις ενέργειες του χρήστη. Προκειμένου η γλώσσα να μπορεί να αποκριθεί αμέσως χωρίς να πρέπει να ξαναφορτώσει την ιστοσελίδα έπρεπε να τρέχει στον ίδιο υπολογιστή με τον browser.

Ο Netscape ήταν ο πρώτος browser που έκανε χρήση μιας τέτοιας γλώσσας προγραμματισμού που επέτρεπε στις ιστοσελίδες να γίνουν διαδραστικές. Πιο συγκεκριμένα η ιστορία της JavaScript ξεκινάει το 1996 με την έκδοση του Netscape 2. Ο Netscape πρόσφερε αρκετές νέες τεχνολογίες οι σημαντικότερες των οποίων ήταν τα frames και η JavaScript. Η JavaScript γράφτηκε από το Brendan Eich και είχε τη δυνατότητα να ενσωματωθεί σε ιστοσελίδες. Μπορούσε να επεξεργαστεί αριθμούς, να τροποποιήσει το περιεχόμενο των HTML forms και ναεκτελέσει διάφορες άλλες λειτουργίες. Κατά τη διάρκεια της ανάπτυξης η JavaScript ήταν γνωστή ως Mocha, μετά LiveWire και έπειτα LiveScript. Επειδή όμως το συντακτικό της έμοιαζε πολύ με την Java και επειδή η Java ήταν εξαιρετικά δημοφιλής αποφασίστηκε η ονομασία JavaScript για λόγους marketing.

Τον ίδιο χρόνο η Netscape κατέθεσε τη JavaScript στην Ευρωπαϊκή Ένωση Κατασκευαστών Υπολογιστών (ECMA) για τυποποίηση. Η ECMA δημιούργησε

το πρότυπο ECMAscript που ενσωμάτωνε τη σύνταξη της core JavaScript, αλλά δεν διευκρίνιζε όλες τις πτυχές του level 0 DOM. Στα μετέπειτα χρόνια η JavaScript εξελίχθηκε κυρίως παράλληλα με την τεχνολογία DOM.

Σε απάντηση η Microsoft αναβάθμισε τον Internet Explorer προσθέτωντας υποστήριξη για δύο scripting γλώσσες, την VBscript και την Jscript. Η VBscript είχε πολλές ομοιότητες με το συντακτικό της γλώσσας BASIC. Αντίθετα η Jscript έμοιαζε πάρα πολύ με την JavaScript. Μάλιστα οι ομοιότητες στο συντακτικό ήταν τόσες πολλές ώστε αν κάποιος ήταν προσεκτικός μπορύσε να γράψει κώδικα που να είναι συμβατός με JavaScript και Jscript. Πλέον η JavaScript είναι η πιο δημοφιλής και ευρέως διαδεδομένη γλώσσα για client side προγραμματισμό.

3.3 Συντακτικό της JavaScript

Στις επόμενες σελίδες παρουσιάζεται η γλώσσα JavaScript ξεκινώντας από βασικές αρχές όπως συντακτικοί κανόνες, μεταβλητές, τιμές και συνεχίζοντας με έννοιες όπως μέθοδοι, αντικείμενα και άλλα.

3.3.1 Βασικοί κανόνες συντακτικού

Αυτό το υποκεφάλαιο παρουσιάζονται οι βασικοί συντακτικοί κανόνες της JavaScript. Αυτοί οι κανόνες καθορίζουν το χαμηλού επιπέδου συντακτικό που διευκρινίζει πράγματα όπως ποια μορφή έχουν τα ονόματα των μεταβλητών, ποιοι χαρακτήρες χρησιμοποιούνται για ορισμό σχόλιων, και πως μια εντολή του προγράμματος χωρίζεται από την επόμενη.

3.3.2 Σύνολο χαρακτήρων

Τα προγράμματα JavaScript γράφονται χρησιμοποιώντας το σύνολο χαρακτήρων Unicode. Σε αντίθεση με την κωδικωποίηση ASCII που υποστηρίζει μόνο αγγλικούς χαρακτήρες η κωδικωποίηση Unicode μπορεί να αναπαραστήσει κάθε γραπτή γλώσσα σε κοινή χρήση. Παρόλα αυτά η JavaScript είναι συμβατή και με προγράμματα που γράφονται σε ASCII κωδικωποίηση καθώς το ASCII είναι υποσύνολο του Unicode.

Η υποστήριξη του συνόλου χαρακτήρων Unicode υπάρχει μόνο στην έκδοση ECMAScript v1 και σε μετέπειτα εκδόσεις της JavaScript. Παλιότερες εκδόσεις δεν υποστηρίζουν Unicode. Επιπλέον μόνο η ECMAScript v3 έκδοση υποστηρίζει Unicode χαρακτήρες σε οποιοδήποτε σημείο του προγράμματος. Παλιότερες εκδόσεις που υποστηρίζουν Unicode επιτρέπουν τέτοιους χαρακτήρες μόνο σε σχόλια και σε αλφαριθμητικές τιμές.

3.3.3 Case Sensitive

Η JavaScript είναι μια Case Sensitive, δηλαδή αντιλαμβάνεται του πεζούς χαρακτήρες ως διαφορετικούς από τους κεφαλαίους. Αυτό σημαίνει ότι οι λέξεις κλειδιά της γλώσσας, τα ονόματα μεταβλητών και συναρτήσεων και οποιαδήποτε άλλα προσδιοριστικά πρέπει πάντα να γράφονται με συνεπή κεφαλαιοποίηση των γραμμάτων. Για παράδειγμα οι myvar και myVar είναι δυο διαφορετικές μεταβλητές.

3.3.4 Κενά και αλλαγή σειράς

Η JavaScript αγνοεί τα διαστήματα, τους στυλοθέτες, και τις αλλαγές γραμμής που εμφανίζονται μεταξύ των tokens στα προγράμματα. Εξαίρεση αποτελούν οι παραπάνω χαρακτήρες όταν εμφανίζονται σαν μέρος μιας κανονικής έκφρασης ή τιμής αλφαριθμητικού (string literal). Ένα token είναι μια λέξη κλειδί, όνομα

μεταβλητής, αριθμός, όνομα μεθόδου, ή κάποια άλλη οντότητα του προγράμματος.

Επειδή μπορούμε να χρησιμοποιήσουμε τα διαστήματα, τους στυλοθέτες, και τις αλλαγές γραμμής ελεύθερα στα προγράμματά μας (εκτός από τα string, τις κανονικές εκφράσεις, και τα token), έχουμε τη δυνατότητα να μορφοποιήσουμε και να δώσουμε τη κατάλληλη διάταξη στον κώδικα του προγράμματός μας έτσι ώστε να είναι τακτοποιημένος. Ως αποτέλεσμα θα διευκολύνεται η ανάγνωση και κατανόηση του κώδικα.

3.3.5 Διαχωρισμός εντολών

Οι απλές εντολές στην JavaScript ακολουθούνται από τον χαρακτήρα semicolon (;) που σηματοδοτεί το τέλος της εντολής. Δηλαδή ο χαρακτήρας αυτός χωρίζει μια εντολή από την επόμενη. Όταν όμως δύο εντολές είναι σε ξεχωριστή γραμμή, η JavaScript μας επιτρέπει να παραλείψουμε το semicolon. Για παράδειγμα οι παρακάτω εντολές μπορούν να γραφτούν χωρίς semicolon.

```
var i=0;

i++;
```

Αυτό που συμβαίνει όταν σε τέτοιες περιπτώσεις είναι ότι η JavaScript προσθέτει αυτόματα το semicolon στο τέλος της γραμμης αν ο κώδικας που προηγείται αποτελεί πλήρη εντολή. Επειδή όμως η JavaScript μπορεί να μπερδευτεί ειδικά στην περίπτωση των εντολών return, break και continue θεωρείται καλή πρακτική προγραμματισμού να μην αλλάζουμε γραμμή στη μέση μιας εντολής και να τοποθετούμ πάντα το semicolon στο τέλος μιας εντολής.

3.3.6 Σχόλια

Η JavaScript υποστηρίζει τα σχόλια σε δύο μορφές. Καταρχάς με τη χρήση του χαρακτήρα // μπορούμε να γράψουμε σχόλιο μίας γραμμής. Συγκεκριμένα ότι υπάρχει ανάμεσα στο // και σε χαρακτήρα αλλαγής γραμμής θεωρείτε σχόλιο. Για παράδειγμα

```
//σχόλιο μίας γραμμής
```

Ο άλλος τύπος σχολίων υποστηρίζεται από τους χαρακτήρες /* και */. Οτιδήποτε βρίσκεται ανάμεσα σε αυτούς τους δύο χαρακτήρες αποτελεί σχόλιο για τη JavaScript. Για παράδειγμα

```
/*σχόλιο
πολλαπλών
γραμμών*/
```

3.4 Τύποι δεδομένων και μεταβλητές

Τα προγράμματα υπολογιστών λειτουργούν με το χειρισμό τιμών, όπως ο αριθμός 3.14 ή το κείμενο «Hello». Οι τύποι τιμών που μια γλώσσα μπορεί να αναπαραστήσει και χειριστεί είναι γνωστοί ως τύποι δεδομένων, και ένα από τα πιο θεμελιώδη χαρακτηριστικά μιας γλώσσας προγραμματισμού είναι το σύνολο των τύπων δεδομένων που υποστηρίζει. Η JavaScript υποστηρίζει τρεις βασικούς τύπους δεδομένων. Τους αριθμούς, τα αλφαριθμητικά και τις τιμές Boolean. Επιπλέον η JavaScript καθορίζει δύο ειδικότερους τύπους δεδομένων,

τους null και undefined, κάθε ένας από τους οποίους μπορεί να έχει μόνο μια τιμή.

Εκτός από τους παραπάνω βασικούς τύπους δεδομένων η JavaScript υποστηρίζει και έναν πιο σύνθετο τύπο το αντικείμενο (object). Ένα αντικείμενο αποτελεί στην ουσία ένα σύνολο από τιμές που μπορούν να είναι οποιουδήποτε τύπου από αριθμό ως ένα άλλο αντικείμενο. Τα αντικείμενα στην JavaScript εμφανίζονται σε δύο μορφές. Μπορούν να αντιπροσωπεύουν μια συλλογή μη ταξινομημένων τιμών που προσδιορίζονται ονομαστικά. Η άλλη μορφή είναι μια συλλογή ταξινομημένων τιμών που προσδιορίζονται αριθμητικά. Αυτη η δεύτερη μορφή του αντικειμένου ονομάζεται πίνακας.

3.4.1 Αριθμοί

Οι αριθμοί είναι ο πιο βασικός τύπος δεδομένων και δεν απαιτούν πολύ εξήγηση. Η JavaScript διαφέρει από τις γλώσσες προγραμματισμού όπως Java δεδομένου ότι δεν κάνει διάκριση μεταξύ των τιμών ακέραιων αριθμών και κινητής υποδιαστολής. Στη JavaScript όλοι οι αριθμοί αναπαριστώνται σαν αριθμοί κινητής υποδιαστολής. Συγκεκριμένα η JavaScript χρησιμοποιεί το 64 bit format κινητής υποδιαστολής όπως αυτό ορίζεται από το IEEE 754 πρότυπο. Οι αριθμοί που η JavaScript μπορεί να αναπαραστήσει είναι μεταξύ 5×10^{-324} και $1.7976931348623157 \times 10^{30}$ και εκτός από αριθμούς στο δεκαδικό σύστημα (base-10) μπορεί να αναπαραστήσει και δεκαεξαδικούς (base-16).

Οι ακέραιοι αριθμοί γράφονται ως μια απλή ακολουθία ψηφίων. Για παράδειγμα 32 ή 3947. Οι δεκαδικοί αριθμοί γράφονται με τη βοήθεια υποδιαστολής. Για παράδειγμα 32.5. Επιπλέον υπάρχη η δυνατότητα να γραφτούν σε εκθετική μορφή. Ο αριθμός 5.3×10^{-324} μπορεί να γραφτεί στην JavaScript ως 5.3e-324. Αντίστοιχα ο 7.69×10^{17} αντιστοιχεί με 7.69e17. Οι δεκαεξαδικούς αριθμοί

ξεκινάνε με το χαρακτήρα “0x” και ακολουθούνται από τον αριθμό σε δεκαεξαδική μορφή. Για παράδειγμα ο αριθμός 3fd γράφεται ως 0x3fd.

Τέλος η JavaScript υποστηρίζει δύο ειδικές σταθερές. Η πρώτη σταθερά είναι η infinity και αντιπροσωπεύει το άπειρο. Αυτή η σταθερά προκύπτει όταν ένας αριθμός γίνεται μεγαλύτερος από το μέγιστο που μπορεί να αναπαραστήσει η JavaScript. Η δεύτερη σταθερά είναι η NaN από τα αρχικά not a number. Αυτή σταθερά είναι αποτέλεσμα σφάλματος μιας μαθηματικής πράξης, όπως για παράδειγμα η διαίρεση με το μηδέν.

3.4.2 Αλφαριθμητικά

Ένα αλφαριθμητικό (string) είναι μια ακολουθία Unicode γραμμάτων, ψηφίων, χαρακτήρων στίξης, και ούτω καθεξής. Είναι ο τύπος δεδομένων της JavaScript για την αντιπροσώπευση του κειμένου. Ένα string είναι μια ακολουθία μηδέν ή περισσότερων χαρακτήρων Unicode που εσωκλείονται μέσα σε μονά(‘) ή διπλά εισαγωγικά(“). Ο χαρακτήρας διπλά εισαγωγικά μπορεί να εισαχθεί μέσα σε strings που οριοθετούνται από τους χαρακτήρες μονά εισαγωγικά, και οι χαρακτήρες μονά εισαγωγικά μπορούν να εισαχθούν μέσα σε strings που οριοθετούνται από τα διπλά εισαγωγικά.

Υπάρχουν πολλοί ειδικοί χαρακτήρες που δεν μπορούν να εισαχθούν σε ένα string κατευθείαν. Για παράδειγμα ο χαρακτήρας αλλαγής γραμμής. Αν θέλουμε να εισάγουμε έναν τέτοιο χαρακτήρα πρέπει να χρησιμοποιήσουμε το χαρακτήρα backslash (\). Αυτός ο χαρακτήρας έχει μία ειδική χρήση στη JavaScript. Όταν συνδυάζεται με τον χαρακτήρα που τον ακολουθεί επιτρέπει την αναπαράσταση χαρακτήρων που κανονικά δεν μπορούν να εισαχθούν σε ένα string. Για παράδειγμα οι χαρακτήρες \n αντιστοιχούν στο χαρακτήρα αλλαγής γραμμής. Ακολουθίες σαν την παραπάνω ονομάζονται escape sequences.

3.4.3 Boolean τιμές

Σε αντίθεση με τους αριθμούς και τα string τα οποία μπορούν να έχουν πολλές διαφορετικές τιμές, ο τύπος δεδομένων Boolean μπορεί να έχει μόνο δύο. Οι τιμές αυτές είναι οι true και false. Οι τιμές Boolean προκύπτουν συνήθως όταν κάνουμε συγκρίσεις στη JavaScript και έτσι χρησιμοποιούνται κυρίως σε εντολές ελέγχου όπως η εντολή if.

3.4.4 Αντικείμενα

Ένα αντικείμενο είναι μια συλλογή των τιμών που προσδιορίζονται από κάποιο όνομα. Αυτές οι ονομασμένες τιμές αναφέρονται συνήθως ως ιδιότητες (properties) του αντικειμένου. Οι ιδιότητες των αντικειμένων είναι, από πολλές απόψεις, ακριβώς όπως οι μεταβλητές της JavaScript. Μπορούν να περιέχουν οποιοδήποτε τύπο δεδομένων, όπως string, αριθμούς ή άλλα αντικείμενα. Τα αντικείμενα είναι ένα σύνθετο και σημαντικό στοιχείο της JavaScript και για αυτό το λόγω θα γίνει πιο λεπτομερής αναφορά σε ξεχωριστό κεφάλαιο της πτυχιακής.

3.4.5 Null και undefined

Η λέξη κλειδί της JavaScript null είναι ένας τύπος δεδομένου που παίρνει μόνο μία τιμή. Αυτή η τιμή σημαίνει καμία τιμή (no value). Η τιμή null θεωρείται συνήθως ως μια ειδική τιμή του τύπου αντικειμένο, μια αξία που δεν αντιπροσωπεύει κανένα αντικείμενο.

Μια άλλη ειδική τιμή που χρησιμοποιείται από την JavaScript είναι η undefined τιμή. Αυτή επιστρέφεται όταν χρησιμοποιούμε μια μεταβλητή που έχει δηλωθεί αλλά δεν της έχει ανατεθεί τιμή ή μια ιδιότητα ενός αντικειμένου που δεν υπάρχει. Αν και η τιμή undefined μοιάζει πολύ με την τιμή null η χρήση τους είναι διαφορετική και άρα η JavaScript τις θεωρεί διαφορετικές.

3.4.6 Μεταβλητές

Μια μεταβλητή είναι στην ουσία ένα όνομα το οποίο το συνδέουμε με μία τιμή. Τις χρησιμοποιούμε για να χειριζόμαστε τιμές στο πρόγραμμά μας. Οι μεταβλητές στην JavaScript εκτελούν την ίδια λειτουργία με κάθε άλλη γλώσσα με κάποιες όμως διαφορές. Μια σημαντική διαφορά της JavaScript με άλλες γλώσσες όπως η C είναι ότι είναι untyped. Δηλαδή μια μεταβλητή μπορεί να περιέχει οποιοδήποτε τύπο δεδομένων. Μπορούμε δηλαδή να αναθέσουμε στη μεταβλητή έναν αριθμό και στη συνέχεια ένα string χωρίς πρόβλημα. Το πλεονέκτημα αυτής της ιδιότητας είναι ότι η JavaScript είναι πολύ πιο απλή στη σύνταξη.

Μια άλλη σημαντική διαφορά της JavaScript είναι ο τρόπος που χειρίζεται την εμβέλεια των μεταβλητών. Συγκεκριμένα δεν υπάρχει μεταβλητή που ορίζεται μόνο για ένα κομμάτι κώδικα. Όταν ορίσουμε μια μεταβλητή μέσα σε ένα βρόγχο for η μεταβλητή αυτή είναι ορισμένη για όλο το σώμα της συνάρτησης. Άρα η εμβέλεια μιας μεταβλητής εξαρτάται από το αν είναι ορίζεται μέσα σε συνάρτηση (local) ή όχι (global) και το σημείο που ορίζεται μέσα σε ένα block κώδικα.

Για να ορίσουμε μια μεταβλητή χρησιμοποιούμε τη λέξη κλειδί var. Για παράδειγμα var i = 0. Υπάρχει η δυνατότητα να ορίσουμε πολλές μεταβλητές με το ίδιο declaration. Για παράδειγμα var i, j k.

3.5 Συναρτήσεις

Μια συνάρτηση είναι ένα κομμάτι κώδικα που δεν εκτελείται μέχρι να ζητηθεί η λειτουργία του ή όπως λέμε να κληθεί. Εκτός από τον μεγάλο έλεγχο που μας

δίνει ως προς την εκτέλεση του κώδικα, οι συναρτήσεις μας γλυτώνουν πολύ χρόνο όταν έχουμε να κάνουμε με επαναλαμβανόμενα κομμάτια κώδικα.

Στη JavaScript η δήλωση των συναρτήσεων γίνεται ως εξής.

```
function asd(var1,var2) {
//declaration block
return var1 * var2;
}
```

Στο παραπάνω κομμάτι κώδικα έχουμε τα εξής. Ο τελεστής function δηλώνει μια καινούργια συνάρτηση. Στη συνέχεια ακολουθεί το όνομα της συνάρτησης και ένα ζεύγος παρενθέσεων που περικλείει τα ορίσματα της συνάρτησης. Τέλος σε άγκιστρα είναι το declaration block της συνάρτησης δηλαδή οι εντολές που εκτελούνται όταν καλούμε την συνάρτηση.

Για να καλέσουμε μια συνάρτηση χρησιμοποιούμε την ακόλουθη σύνταξη.

```
var a = asd(5,4);
```

Χρησιμοποιούμε δηλαδή το όνομα της μεθόδου και στη θέση των ορισμάτων βάζουμε τις επιθυμητές τιμές. Έτσι μπορούμε να εκτελούμε τις ίδιες εντολές για διαφορετικά κάθε φορά ορίσματα.

3.6 Αντικείμενα

Τα αντικείμενα είναι από τους πιο σημαντικούς τύπους δεδομένων στη JavaScript. Τα αντικείμενα είναι ένας σύνθετος τύπος δεδομένων. Αποτελούν συλλογές τιμών και μας επιτρέπουν πρόσβαση σε αυτές τις τιμές με βάση το όνομα. Τα properties του αντικειμένου μπορούν να είναι απλοί τύποι δεδομένων όπως αριθμοί ή strings, ή μπορούν να είναι άλλα αντικείμενα.

Η δημιουργία ενός νέου αντικειμένου γίνεται με τη χρήση του τελεστή new.

```
var a = new Object();
```

Η JavaScript μας δίνει πολλά έτοιμα αντικείμενα που μπορούμε να χρησιμοποιήσουμε. Επιπλέον όμως μας δίνει τη δυνατότητα να ορίσουμε τα δικά μας αντικείμενα. Αυτό γίνεται με τη χρήση μιας συνάρτησης δομητή. Για παράδειγμα

```
function myObject(a,b) {
this.varA = a;
this.varB = b;
}
```

Στην συνέχεια μπορούμε με τον τελεστή new να δημιουργήσουμε ένα στιγμιότυπο του αντικειμένου myObject με τις τιμές που θέλουμε.

```
var obj = new myObject(1,2);
```

Μπορούμε να έχουμε πρόσβαση στα properties του αντικειμένου με τη χρήση του τελεστή τελεία. Για παράδειγμα obj.varA μας επιστρέφει την τιμή 1. Εκτός από μεταβλητές ένα αντικείμενο μπορεί να περιέχει και συναρτήσεις. Για να ορίσουμε μια συνάρτηση για έναν constructor πρέπει να κάνουμε τα εξής. Καταρχάς ορίζουμε τη συνάρτηση κανονικά.

```
function multiply() {
return this.varA * this.varB;
}
```

Στη συνέχεια αναθέτουμε τη μέθοδο στον constructor ως εξής.

```
function myObject(a,b) {
this.varA = a;
this.varB = b;
this.mult = multiply;
}
```

Έπειτα καλούμε τη μέθοδο από το αντικείμενο. Το obj.mult() μας επιστρέφει την τιμή 2. Γενικά τα αντικείμενα μας επιτρέπουν να ομαδοποιήσουμε μεταβλητές και μεθόδους και να τις χρησιμοποιήσουμε ευέλικτα.

3.7 Πρωτότυπα

Γενικά μία μέθοδος constructor είναι ένας ανεπαρκής τρόπος για να ορίσουμε μεθόδους για το αντικείμενο που αρχικοποιεί. Όταν χρησιμοποιούμε αυτή τη μέθοδο κάθε αντικείμενο που αρχικοποιείται από τον constructor έχει τα ίδια αντίγραφα των ίδιων ιδιοτήτων μεθόδου. Υπάρχει ένας πολύ καλύτερος τρόπος να ορίσουμε μεθόδους, σταθερές, και άλλες ιδιότητες που μοιράζονται από όλα τα αντικείμενα μιας κλάσης. Αυτό γίνεται με τη χρήση της τεχνικής της κληρονομικότητας που υποστηρίζει η JavaScript.

Κάθε αντικείμενο στη JavaScript κληρονομεί ιδιότητες από ένα πρωτότυπο αντικείμενο. Όλες οι ιδιότητες του πρωτότυπου αντικειμένου εμφανίζονται αυτόματα σαν ιδιότητες όλων των αντικειμένων για τα οποία είναι πρωτότυπο. Δηλαδή κάθε αντικείμενο κληρονομεί τις ιδιότητες από το πρωτότυπό του.

Το πρωτότυπο ενός αντικειμένου ορίζεται από τη μέθοδο constructor που αρχικοποιεί το αντικείμενο. Όλες οι συναρτήσεις στη JavaScript μια ιδιότητα που ονομάζεται prototype και η οποία αναφέρεται σε ένα αντικείμενο. Αυτό το αντικείμενο είναι αρχικά κενό αλλά όσες ιδιότητες ορίσουμε σε αυτό κληρονομούνται από όλα τα αντικείμενα που δημιουργούνται από τον constructor. Ένας constructor καθορίζει μια κατηγορία αντικειμένων και αρχικοποιεί τις ιδιότητες του. Το πρωτότυπο αντικείμενο συνδέεται με τον constructor και έτσι κάθε μέλος της κλάσης κληρονομεί ακριβώς το ίδιο σύνολο ιδιοτήτων από το πρωτότυπο. Αυτό σημαίνει ότι το αντικείμενο πρωτοτύπων είναι μια ιδανική θέση για τις μεθόδους και άλλες σταθερές ιδιότητες.

Κατά τη διαδικασία της κληρονομικότητας δεν γίνεται αντιγραφή των ιδιοτήτων του πρωτοτύπου στο αντικείμενο που κληρονομεί. Αντίθετα οι ιδιότητες φαίνονται να είναι μέρος του αντικειμένου αλλά στην πραγματικότητα οι ιδιότητες βρίσκονται μόνο στο πρωτότυπο αντικείμενο και κάθε φορά που τις χρειαζόμαστε ανατρέχουμε σε αυτό. Έχει τις ακόλουθες επιπτώσεις. Καταρχάς μειώνεται σημαντικά ο χώρος που καταλαμβάνουν στη μνήμη τα αντικείμενα που κληρονομούν ιδιότητες από το πρωτότυπο. Η δεύτερη είναι ότι ένα αντικείμενο μπορεί να κληρονομήσει ιδιότητες που προστίθενται στο πρωτότυπο αφού το αντικείμενο έχει δημιουργηθεί.

Η κληρονομικότητα ισχύει μόνο όταν διαβάζουμε τις ιδιότητες ενός αντικειμένου. Όταν διαβάζουμε μια ιδιότητα η JavaScript ψάχνει αν το αντικείμενο έχει μια ιδιότητα με αυτό το όνομα και αν όχι τότε ψάχνει στο πρωτότυπό του. Αντίθετα όταν γράφουμε την τιμή μιας ιδιότητας που δεν υπάρχει στο αντικείμενο, η JavaScript τη δημιουργεί και δεν αλλάζει σε καμία περίπτωση την τιμή της ιδιότητας στο πρωτότυπο. Αυτό συμβαίνει γιατί αν αλλάζαμε την τιμή της ιδιότητας στο πρωτότυπο τότε θα επηρεάζονταν όλα τα αντικείμενα που κληρονομούν από το πρωτότυπο αυτό.

Για να αποκτήσουμε πρόσβαση στο πρωτότυπο ενός αντικειμένου χρησιμοποιούμε τον ακόλουθο κώδικα.

myObj.prototype

Μπορούμε να ορίσουμε ιδιότητες για το πρωτότυπο ως εξής.

myObj.prototype.a=15;

Αντίστοιχα μπορούμε να ορίσουμε και μεθόδους με τον ίδιο τρόπο.

myObj.prototype.myfunc=function () {return a*b;}

3.8 Ενσωματωμένα αντικείμενα της JavaScript

Στη συνέχεια γίνεται αναφορά σε μερικά από τα ενσωματωμένα αντικείμενα της JavaScript τα οποία χρησιμοποιήθηκαν στα πλαίσια της πτυχιακής. Συγκεκριμένα παρουσιάζονται τα αντικείμενα window, document και array.

3.8.1 Window

Το αντικείμενο window έχει κεντρικό ρόλο στη JavaScript καθώς βρίσκεται στην κορυφή της ιεραρχίας όλων των αντικειμένων. Επιπλέον έχει σημαντικό ρόλο γιατί αναπαριστά κάθε παράθυρο web browser και κάθε frame μέσα σε αυτό το παράθυρο.

Μερικές από τις πιο σημαντικές ιδιότητες του window είναι οι ακόλουθες.

- document: Είναι μια αναφορά στο αντικείμενο document που περιλαμβάνει το παράθυρο.
- frames[ ]: Είναι ένας πίνακας από window αντικείμενα που αναπαριστούν τα frames που περιέχει το παράθυρο.

- location: Είναι μια αναφορά στο αντικείμενο location που αναπαριστά το URL του document που περιέχεται στο παράθυρο.

Ορισμένες από τις πιο σημαντικές μεθόδους του window είναι οι ακόλουθες.

- alert(): Εμφανίζει ένα απλό παράθυρο στο χρήστη που περιέχει ένα μήνυμα.
- close(): Κλείνει το παράθυρο του browser που αναπαριστά το αντικείμενο.
- open(): Ανοίγει ένα καινούργιο παράθυρο του browser με συγκεκριμένες παραμέτρους που εμφανίζει το URL της επιλογής μας.
- setTimeout(): Προγραμματίζει την εκτέλεση μιας συνάρτησης μετά από την πάροδο συγκεκριμένου χρονικού διαστήματος.
- clearTimeout(): Χρησιμοποιείται για να ακυρώσει την εκτέλεση μιας προγραμματισμένης συνάρτησης.

3.8.2 Document

Ένα ακόμη σημαντικό αντικείμενο της JavaScript το αντικείμενο document αναπαριστά το HTML αρχείο που εμφανίζεται στο παράθυρο του browser. Το document είναι πιθανότατα το πιο συχνά χρησιμοποιημένο αντικείμενο της JavaScript καθώς μας δίνει πρόσβαση σε όλα τα HTML στοιχεία που περιέχονται στο έγγραφό μας. Η χρησιμότητά του αυξάνεται όταν χρησιμοποιείται μαζί με το DOM το οποίο αναλύεται σε επόμενο κεφάλαιο. Στη συνέχεια παρουσιάζονται μερικές από τις ιδιότητες και μεθόδους του αντικειμένου.

Μερικές από τις πιο σημαντικές ιδιότητες του document.

- links[ ]: Ένας πίνακας όλων των υπερσυνδέσμων που περιέχονται στο έγγραφο.
- URL: Είναι ένα string που περιέχει το URL από το οποίο φορτώθηκε το HTML έγγραφο.

Μερικές από τις πιο σημαντικές μεθόδους του document.

- open(): Ανοίγει ένα καινούργιο έγγραφο στο παράθυρο και διαγράφει το παλιό.
- close(): Κλείνει ένα έγγραφο που ανοίξαμε με την μέθοδο open().
- write(): Προσθέτει κείμενο της επιλογής μας στο τέλος του εγγράφου που είναι ανοιχτό.

3.8.3 Array

Το αντικείμενο array είναι ένας σύνθετος τύπος δεδομένων που περιέχει αριθμημένες τιμές. Κάθε τιμή που περιέχεται στο array ή πίνακα αποκαλείται στοιχείο του πίνακα. Ένα στοιχείο του πίνακα μπορεί να είναι οποιουδήποτε τύπου ακόμη και άλλος πίνακας.

Υπάρχουν τέσσερις τρόποι για να δημιουργήσουμε ένα πίνακα. Μπορούμε να καλέσουμε τον constructor του αντικειμένου χωρίς παραμέτρους οπότε δημιουργεί έναν άδειο πίνακα. Ο δεύτερος τρόπος είναι να καλέσουμε τον constructor με μια λίστα από ορίσματα όπου κάθε όρισμα αποτελεί ένα στοιχείο του πίνακα. Για παράδειγμα

```
var myArray = new Array(1,2,3);
```

Ο τρίτος τρόπος είναι να καλέσουμε τον constructor με ένα όρισμα το οποίο είναι ακέραιος αριθμός. Σε αυτή την περίπτωση δημιουργούμε ένα πίνακα με αυτόν τον αριθμό των στοιχείων τα οποία περιέχουν την τιμή undefined.

Ο τελευταίος τρόπος είναι να χρησιμοποιήσουμε ένα array literal (τιμή). Για να δημιουργήσουμε ένα array literal χρησιμοποιούμε τους χαρακτήρες αγκύλες. Για παράδειγμα

var myvar = [1,2,3];

Για να αποκτήσουμε πρόσβαση σε ένα στοιχείο του πίνακα χρησιμοποιούμε πάλι τους χαρακτήρες αγκύλες και μέσα βάζουμε τον αριθμό του στοιχείου που θέλουμε. Για παράδειγμα myvar[0]. Μπορούμε έτσι να διαβάσουμε την τιμή του στοιχείου ή να γράψουμε μια άλλη τιμή. Μια άλλη δυνατότητα που έχουμε είναι να προσθέσουμε ένα νέο στοιχείο στον πίνακα μετά τη δημιουργία του. Αυτό γίνεται αναθέτοντας μια τιμή στην επιθυμητή θέση. Για παράδειγμα αν έχουμε έναν πίνακα 15 θέσεων μπορούμε να βάλουμε ένα στοιχείο στην εικοστή θέση γράφοντας myArray[19] = 57.

Η πιο σημαντική ιδιότητα ενός πίνακα είναι η length. Αυτή η ιδιότητα μας επιστρέφει μια τιμή που μας δείχνει πόσα στοιχεία περιέχει ο πίνακας. Το length αλλάζει αυτόματα κάθε φορά που προσθέτουμε ένα στοιχείο στον πίνακα έτσι ώστε να περιέχει ανά πάσα στιγμή το σωστό μέγεθος του πίνακα. Η πιο συχνή χρήση του length είναι μαζί με το βρόγχο for όταν θέλουμε να περάσουμε διαδοχικά όλα τα στοιχεία του πίνακα.

Παρακάτω παρουσιάζονται μερικές από τις μεθόδους του array.

- join(): Η μέθοδος αυτή μετατρέπει όλα τα στοιχεία του πίνακα σε strings και τα ενώνει σε ένα ενιαίο. Προαιρετικά μπορεί να πάρει ως όρισμα string που χρησιμοποιείται για να χωρίσει τα στοιχεία του πίνακα στο string που προκύπτει.
- reverse(): Αντιστρέφει τη σειρά των στοιχείων του πίνακα. Δηλαδή ο πίνακας [1,2,3] θα πάρει τη μορφή [3,2,1].
- push(): Η μέθοδος αυτή μας επιτρέπει να δουλέψουμε με τους πίνακες σαν να είναι στοίβες. Συγκεκριμένα προσθέτει το στοιχείο που δίνουμε σαν όρισμα στο τέλος του πίνακα και μας επιστρέφει το length του πίνακα.
- pop(): Η αντίστροφη μέθοδος της push(). Διαγράφει το τελευταίο στοιχείο του πίνακα προσαρμόζει το length και μας επιστρέφει την τιμή που αφαίρεσε.

4. XML

Η XML είναι ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση εγγράφων. Είναι ένα ανοιχτό πρότυπο που δίνει έμφαση στην απλότητα και στην χρηστικότητα της πληροφορίας στο internet. Είναι μια μέθοδος μορφοποίησης δεδομένων σε μορφή κειμένου που παρέχει υποστήριξη για όλες τις γλώσσες που υπάρχουν μέσω Unicode. Αν και αρχικά προοριζόταν για κωδικοποίηση εγγράφων, πλέον χρησιμοποιείται ευρέως για αναπαράσταση δεδομένων σε web υπηρεσίες. Τέτοια δεδομένα μπορεί να είναι μία συλλογή στοιχείων δεδομένων όπως είναι για παράδειγμα τα λογιστικά φύλλα, οι κατάλογοι διευθύνσεων, οι παράμετροι διαμόρφωσης, οι οικονομικές συναλλαγές και τα τεχνικά σχέδια.

Η ανάπτυξη της XML ξεκίνησε το 1996. Από το Φεβρουάριο του 1998 η XML αποτελεί σύσταση του W3C. Πριν από την XML υπήρχε η SGML, η οποία αναπτύχθηκε στις αρχές της δεκαετίας του '80, τυποποιήθηκε από τον ISO το 1986, και χρησιμοποιήθηκε ευρέως σε προγράμματα με εκτεταμένη τεκμηρίωση. Η ανάπτυξη της HTML ξεκίνησε το 1990. Οι σχεδιαστές της XML επέλεξαν τα καλύτερα τμήματα της SGML, χρησιμοποίησαν την εμπειρία που είχαν αποκτήσει κατά την ανάπτυξη της HTML και παρήγαγαν μία γλώσσα η οποία δεν είναι λιγότερο ισχυρή από την SGML αλλά είναι πιο κανονικοποιημένη και πολύ πιο εύχρηστη.

Η XML δεν είναι γλώσσα προγραμματισμού, πράγμα που σημαίνει ότι δεν απαιτεί γνώσεις προγραμματισμού για να χρησιμοποιηθεί. Διευκολύνει τον υπολογιστή να παράγει δεδομένα, να διαβάζει δεδομένα και να εξασφαλίζει τη σαφήνεια της δομής των δεδομένων. Είναι επεκτάσιμη, ανεξάρτητη συστήματος υλικού και μπορεί να υποστηρίξει διεθνείς και τοπικές προσαρμογές.

Υπάρχει μεγάλο εύρος από API’s τα οποία μπορούν οι προγραμματιστές να χρησιμοποιήσουν για να χουν πρόσβαση σε δεδομένα XML. Επίσης έχει αναπτυχθεί μεγάλος αριθμός γλωσσών βασισμένων στην XML όπως RSS,

ATOM, SOAP και XHTML, ενώ η XML έχει υιοθετηθεί από τα περισσότερα office suites όπως το Microsoft Office, OpenOffice και Apple’s Iwork.

Μερικές βασικές ορολογίες είναι οι:

1. Χαρακτήρες: Εξ’ ορισμού κάθε XML έγγραφο είναι ένα σύνολο από strings χαρακτήρων. Σχεδόν κάθε χαρακτήρας του Unicode μπορεί να χρησιμοποιηθεί σε αυτά.
2. Επεξεργαστής και εφαρμογή: Επεξεργαστής είναι λογισμικό που επεξεργάζεται XML έγγραφα, το οποίο λειτουργεί σαν υπηρεσία μίας εφαρμογής. Υπάρχουν σαφείς κανόνες για τη λειτουργία ενός επεξεργαστή αλλά όχι για τη συμπεριφορά μιας εφαρμογής. Ο επεξεργαστής συχνά αναφέρεται ως XML Parser.
3. Markup και Content: Οι χαρακτήρες οι οποίοι απαρτίζουν ένα XML έγγραφο χωρίζονται σε markup και content. Αυτά διακρίνονται με την χρήση απλών συντακτικών κανόνων. Όλοι οι χαρακτήρες που αναπαριστούν markups περικλείονται από του χαρακτήρες “ < ” και “ > ” ή από “ & ” και “ ; ”. Οι υπόλοιποι χαρακτήρες που δεν περικλείονται από αυτούς τους χαρακτήρες αποτελούν το περιεχόμενο (content).
4. Tag: Μία δομή markup μπορεί να έχει 3 μορφές: start-tag (π.χ. <section>), end-tag (π.χ. </section>) και empty-element (π.χ. <line-break/>
5. Element: Είναι το λογικό συστατικό ενός XML εγγράφου που ξεκινά με ένα start-tag και τελειώνει με ένα end-tag. Οι χαρακτήρες που περικλείονται από start-tags και end-tags αποτελούν το περιεχόμενο του element. Ένα e;ement μπορεί να περιέχει άλλα elements τα οποία αποκαλούνται child-elements. Ένα παράδειγμα element είναι η γραμμή: <Greeting>Hello, world.</Greeting>.
6. Attribute: Μια δομή markup που περιλαμβάνει ένα η περισσότερα ζεύγη ονόματος/τιμής μέσα στο start-tag. Για παράδειγμα, ένα element img, το οποίο έχει δύο ορίσματα, τα src και alt, θα είχε την εξής μορφή: . Μία άλλη μορφή θα

μπορούσε να είναι το εξής attribute: <step number="3">Connect A to B.</step>.

7. XML Declaration: Ένα XML έγγραφο μπορεί να ξεκινάει δηλώνοντας μερικές πληροφορίες για το έγγραφο όπως η κωδικοποίηση ή η έκδοση της xml που χρησιμοποιεί. Παράδειγμα τέτοιας δήλωσης είναι η εξής: <?xml version=”1.0” encoding=”UTF-8” ?>

Ένα παράδειγμα XML κώδικα είναι το εξής:

```
<?xml version="1.0" encoding='UTF-8'?>
<painting>
  <img src="madonna.jpg" alt='Foligno Madonna, by Raphael'/>
  <caption>This is Raphael's "Foligno" Madonna, painted in
    <date>1511</date>–<date>1512</date>.
  </caption>
</painting>
```

Υπάρχουν 5 elements σε αυτό το παράδειγμα, τα painting, img, caption και 2 dates. Τα “dates” είναι “child” elements του “caption”, το οποίο με τη σειρά του είναι child element του root element “painting”.

Όπως μπορούμε να δούμε από το παράδειγμα η XML έχει αρκετά κοινά στοιχεία με την HTML. Η XML, όπως η HTML, χρησιμοποιεί ετικέτες (tags) (λέξεις μέσα σε γωνιακές αγκύλες '<' και '>') και γνωρίσματα (τύπου όνομα = "τιμή"). Σε αντίθεση με την HTML η οποία διευκρινίζει τη σημασία κάθε ετικέτας και γνωρίσματος και συχνά προσδιορίζει πως θα εμφανίζεται σε φυλλομετρητή το κείμενο το οποίο περιλαμβάνεται σε αυτά, η XML χρησιμοποιεί ετικέτες μόνο για να οριοθετήσει κομμάτια δεδομένων και αφήνει την ερμηνεία των δεδομένων στη εφαρμογή που τα διαβάζει.

Τα προγράμματα που παράγουν λογιστικά φύλλα, καταλόγους διευθύνσεων και άλλα δομημένα δεδομένα αποθηκεύουν, συχνά, τα εν λόγω δεδομένα στο σκληρό δίσκο, χρησιμοποιώντας δυαδική μορφή ή μορφή κειμένου. Ένα από τα πλεονεκτήματα της μορφής κειμένου είναι ότι επιτρέπει στο χρήστη, ένα είναι αναγκαίο, να δει τα δεδομένα χωρίς το πρόγραμμα που τα παρήγαγε. Οι μορφές κειμένου επιτρέπουν, επίσης, στους κατασκευαστές λογισμικού να εκσφαλματώνουν εφαρμογές με μεγαλύτερη ευκολία. Όπως και τα αρχεία HTML, τα αρχεία XML είναι αρχεία κειμένου τα οποία δεν προορίζονται για ανάγνωση αλλά προσφέρουν αυτή τη δυνατότητα στο χρήστη εάν προκύψει ανάγκη. Ωστόσο, οι κανόνες των αρχείων XML είναι αυστηροί σε αντίθεση με τα αρχεία HTML. Η παράληψη μίας ετικέτας ή ένα γνώρισμα δίχως αγκύλες καθιστά άχρηστο το αρχείο XML ενώ η HTML ανέχεται τέτοιου είδους παραλήψεις και συχνά τις επιτρέπει εξολοκλήρου.

Η XML εμφανίζεται υπό μορφή κειμένου και χρησιμοποιεί ετικέτες για την οριοθέτηση των δεδομένων και για τον λόγο αυτό τα αρχεία XML είναι σχεδόν πάντα μεγαλύτερα σε έκταση από συγκρίσιμα αρχεία σε δυαδική μορφή. Πρόκειται για συνειδητή επιλογή των σχεδιαστών της XML, καθώς Η χωρητικότητα του σκληρού δίσκου δεν είναι τόσο ακριβή όσο παλαιότερα και προγράμματα όπως το zip και το gzip μπορούν να συμπιέσουν αρχεία αποτελεσματικά και γρήγορα. Επιπρόσθετα, πρωτόκολλα επικοινωνίας όπως τα πρωτόκολλα μόντεμ και το HTTP/1.1, το οποίο είναι το πρωτόκολλο πυρήνας του Ιστού, μπορούν να συμπιέσουν πολύ εύκολα αρχεία με μεγάλη ταχύτητα μεταφοράς και το ίδιο αποτελεσματικά όσο και τα δυαδικά αρχεία.

Η *XML 1.0* είναι η προδιαγραφή που ορίζει τι είναι οι "ετικέτες" και τα "γνωρίσματα". Πέρα από την XML 1.0, "η οικογένεια XML" είναι ένα διαρκώς αναπτυσσόμενο σύνολο λειτουργικών μονάδων οι οποίες προσφέρουν χρήσιμες υπηρεσίες για τη διεκπεραίωση σημαντικών έργων τα οποία ανακύπτουν συχνά. Η *Xlink* περιγράφει έναν προκαθορισμένο τρόπο εισαγωγής υπερσυνδέσμων σε αρχεία XML. Τα *XPointer* και τα XFragments είναι συντακτικά υπό διαμόρφωση

για την υπόδειξη θέσεων ενός εγγράφου XML. Το XPointer μοιάζει λίγο με URL αλλά αντί να υποδεικνύει έγγραφα στον Ιστό, υποδεικνύει κομμάτια πληροφοριών ενός εγγράφου XML. Το *CSS*, η γλώσσα μορφοποίησης σελίδων, είναι δυνατό να εφαρμοστεί σε XML όπως και σε HTML. Το *XSL* είναι προηγμένη γλώσσα (advanced language) μορφοποίησης σελίδων. Βασίζεται στο *XSLT*, μία γλώσσα μετασχηματισμού η οποία χρησιμοποιείται για την αναδιάταξη, την πρόσθεση και την διαγραφή ετικετών και γνωρισμάτων. Το *DOM* είναι ένα προκαθορισμένο σύνολο λειτουργιών για τη διαχείριση αρχείων XML (και HTML) από μία γλώσσα προγραμματισμού. Τα *XML Schemas 1* και *2* επιτρέπουν στους κατασκευαστές λογισμικού να ορίσουν με ακρίβεια τις δομές των δικών τους μορφών XML. Υπάρχουν αρκετά εργαλεία και λειτουργικές μονάδες τα οποία βρίσκονται υπό διαμόρφωση ή είναι ήδη διαθέσιμα.

Μία από τις εφαρμογές XML υπάρχει υπό μορφή εγγράφου: πρόκειται για την XHTML του W3C, το διάδοχο της HTML. Η XHTML διαθέτει αρκετά κοινά στοιχεία με την HTML. Το συντακτικό, όμως, έχει αλλάξει έτσι ώστε να συμβαδίζει με τους κανόνες της XML. Τα έγγραφα με βάση την XML χρησιμοποιούν το συντακτικό της XML ,με ορισμένους, όμως, περιορισμούς (π.χ., η XHTML επιτρέπει "<p>", όχι όμως "<r>"); και πρόσθεση σημασίας στο συντακτικό (η XHTML λέει ότι το "<p>" σημαίνει "paragraph" (παράγραφος), και όχι "price" (τιμή) ή "person" (πρόσωπο) και όχι κάτι άλλο).

Η XML επιτρέπει στο χρήστη τον ορισμό νέας μορφής εγγράφου προσφέροντάς του τη δυνατότητα να συνδυάσει και να χρησιμοποιήσει άλλες μορφές. Ωστόσο, επειδή δύο διαφορετικές μορφές, οι οποίες έχουν αναπτυχθεί ανεξάρτητα, ενδέχεται να διαθέτουν στοιχεία ή γνωρίσματα με το ίδιο όνομα, πρέπει να αποδοθεί ιδιαίτερη προσοχή κατά το συνδυασμό των δύο μορφών (το "<p>" μπορεί να σημαίνει "paragraph" (παράγραφος) στη μία μορφή και "person" (πρόσωπο) στην άλλη). Για την αποφυγή σύγχυσης ονομάτων κατά το συνδυασμό μορφών, η XML παρέχει ένα μηχανισμό *namespace*. Παραδείγματα μορφών με βάση την XML οι οποίες χρησιμοποιούν namespaces είναι η XSL και

η RDF .Το *XML Schema* σχεδιάστηκε με στόχο να επιδείξει την ικανότητα υποστήριξης συνδυασμών στο επίπεδο ορισμού δομών εγγράφου XML καθιστώντας εφικτό το συνδυασμό δύο σχημάτων και την παραγωγή τρίτου το οποίο αντιπροσωπεύει δύο συγχωνευμένες δομές εγγράφου.

Ο Σκελετός Περιγραφής Πόρων του W3C (Resource Description Framework) (RDF) είναι μία μορφή κειμένου XML η οποία υποστηρίζει περιγραφή πόρων και εφαρμογές μεταδεδομένων, όπως οι κατάλογοι μουσικής, οι συλλογές φωτογραφιών και οι βιβλιογραφίες. Για παράδειγμα, το RDF έχει τη δυνατότητα αναγνώρισης προσώπων σε ένα άλμπουμ φωτογραφιών του Ιστού χρησιμοποιώντας πληροφορίες από μία προσωπική λίστα επαφών. Όπως συμβαίνει και στα συγχωνευμένα έγγραφα (integrated documents) HTML, τα συστήματα μενού και τις φόρμες αιτήσεων για την έναρξη του αρχικού Ιστού, το RDF συνδυάζει εφαρμογές και πράκτορες σε έναν ενιαίο Σημασιολογικό Ιστό. Και, βέβαια, όπως οι άνθρωποι έχουν συμφωνήσει να χρησιμοποιούν κοινές ονομασίες για τις σημασίες των λέξεων που χρησιμοποιούν όταν επικοινωνούν, έτσι και οι υπολογιστές χρειάζονται μηχανισμούς οι οποίοι να ορίζουν κοινά ονόματα για τους όρους ώστε να είναι εφικτή η αποτελεσματική επικοινωνία. Οι επίσημες περιγραφές όρων που ανήκουν σε ένα συγκεκριμένο νοηματικό πεδίο (για παράδειγμα αυτό των αγορών ή των κατασκευών) ονομάζονται οντολογίες και συνιστούν σημαντικό τμήμα του Σημασιολογικού Ιστού. Οι οντολογίες του RDF, και η αναπαράσταση των διαφόρων σημασιών ώστε οι υπολογιστές να διευκολύνουν τους ανθρώπους στην εκτέλεση διαφόρων εργασιών αποτελούν μέρος της Δραστηριότητας Σημασιολογικού Ιστού (Semantic Web Activity).

Η XML προσφέρει πρόσβαση σε μια μεγάλη και διαρκώς αναπτυσσόμενη κοινότητα εργαλείων και ειδικών με μεγάλη εμπειρία στις εν λόγω τεχνολογίες. Δεν χρειάζεται άδεια χρήσης και τυγχάνει ευρείας και ολοένα επεκτεινόμενης υποστήριξης.

5. DOM

Το DOM (Document Object Model) είναι μία cross-platform και ανεξάρτητή από την γλώσσα υλοποίησης σύμβαση για τν αναπαράσταση και αλληλεπίδραση των αντικειμένων στην HTML. Έχει δικό του API και τα στοιχεία της DOM μπορούν να διαμορφωθούν και να χρησιμοποιηθούν μέσα από τη σύνταξη της γλώσσας προγραμματισμού που χρησιμοποιείται.

Η ιστορία του DOM είναι αλληλένδετη με την ιστορία του πολέμου των φυλλομετρητών (“browser wars”), του πολέμου ανάμεσα του Netscape και του Internet Explorer καθώς επίσης και με τον πόλεμο μεταξύ της JavaScript και της Jscript, των πρώτων γλωσσών που υλοποιούνταν από τους browsers για την κατασκευή του layout.

Η JavaScript απελευθερώθηκε για πρώτη φορά από την Netscape Communications το 1996 μέσω του φυλλομετρητή της, του Netscape ενώ το Jscript, το οποίο είναι ένα port της JavaScript στον Internet Explorer, από την Microsoft στην ίδια χρονιά. Η JavaScript και η Jscript επέτρεψαν για πρώτη φορά στους web developers να δημιουργήσουν ιστοσελίδες με client-side αλληλεπίδραση. Τα πρώτα χαρακτηριστικά που επέτρεπαν την τροποποίηση του HTML εγγράφου και την ανίχνευση events που παράγονται από τον χρήστη αποτελούσαν το “DOM Level 0” ή το “Legacy DOM”. Δεν αναπτύχθηκε κάποιο standard για το DOM Level 0 αλλά η υλοποίησή του περιγράφεται μερικώς στο specification της HTML4.

Το Legacy DM είχε περιορισμένη δυνατότητα πρόσβασης σε αντικείμενα. Φόρμες, υπερσύνδεσμοι και αντικείμενα εικόνων αναφέρονταν με ένα ιεραρχικό όνομα που ξεκινούσε από το αντικείμενο ρίζας. Επίσης σαν όνομα αντικειμένου του DOM μπορούσε να χρησιμοποιηθεί και ο σειριακός δείκτης στοιχείων. Δηλαδή ένα στοιχείο μιας φόρμας μπορούσε να αναφερθεί είτε ως document.formName.inputName είτε ως document.forms[0].elements[0]. Το

Legacy DOM ήταν αυτό που εισήγαγε το validation μια φόρμας από τον χρήστη και το εφέ του rollover.

Το 1997, νέες εκδόσεις του Netscape και του Explorer απελευθερώθηκαν, οι οποίες υποστήριζαν την Dynamic HTML (DHTML). Νέα έκδοση DOM απελευθερώθηκε, η αποκαλούμενη Intermediate DOM. Ενώ η Legacy DOM ήταν συμβατή με όλους τους φυλλομετρητές, το Intermediate DOM αναπτύχθηκε ξεχωριστά από κάθε κατασκευαστή ιστοσελίδων με αποτέλεσμα να παρουσιάζει πολλά προβλήματα ασυμβατότητας. Το Intermediate DOM εισήγαγε την δυνατότητα χειρισμού των Cascading Style Sheet (CSS), τα οποία καθορίζουν την εμφάνιση μιας ιστοσελίδας.

Το World Wide Web Consortium (W3C), έφερε την Netscape και την Microsoft μαζί με άλλους κατασκευαστές ιστοσελίδων στο ίδιο τραπέζι, ώστε να αναπτύξουν ένα κοινό standard για μία ενιαία scripting γλώσσα για όλους τους φυλλομετρητές, η αποκαλούμενη ECMAScript. Η πρώτη έκδοση αυτής της γλώσσας δημοσιεύτηκε το 1997 και οι νέες εκδόσεις JavaScript και Jscript υλοποίησαν την ECMAScript για μεγαλύτερη συμβατότητα.

Μετά την δημοσίευση της ECMAScript, το W3C άρχισε να εργάζεται για την υλοποίηση ενός standard DOM. Αυτή η προσπάθεια είχε ως αποτέλεσμα το DOM LEVEL 1, το οποίο δημοσιεύτηκε το 1998 αλλά υπήρξε περιορισμένη υποστήριξη από τους browsers, ειδικά από τον Explorer. Η επόμενη έκδοση, η DOM LEVEL 2 δημοσιεύτηκε το 2000 και εισήγαγε στοιχεία όπως το “getElemetnById”, υποστήριξη για events models, XML namespaces και CSS. Το DOM LEVEL 3, η τρέχουσα έκδοση του DOM, υποστηρίζει επιπλέον το XPath (γλώσσα χειρισμού XML εγγράφων) και τον χειρισμό keyboard events.

Το DOM είναι ιδανικό για εφαρμογές όπου τα στοιχεία ενός εγγράφου πρέπει να χρησιμοποιηθούν επανηλλειμένως ή με όχι σειριακό τρόπο. Ένας φυλλομετρητής δεν είναι υποχρεωμένος να χρησιμοποιεί το DOM ώστε να υλοποιήσει το layout

μιας ιστοσελίδας αλλά είναι απαραίτητο για την JavaScript ώστε να έχει τη δυνατότητα πρόσβασης ή τροποποίησης μιας ιστοσελίδας δυναμικά.

Στο DOM κάθε αντικείμενο αποτελεί έναν κόμβο. Π.χ. η εξής σύνταξη HTML

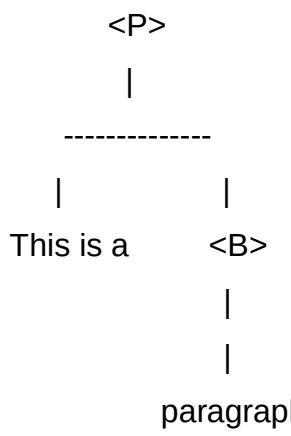
<P>This is a paragraph</P>

Δημιουργεί δύο κόμβους, ένα στοιχείο P και ένα κόμβο κειμένου με το περιεχόμενο “This is a paragraph”. Το περιεχόμενο βρίσκεται μέσα στο στοιχείο P οπότε το P θεωρείται ως parent node του child node του κόμβου κειμένου.

Με την εξής σύνταξη:

<P>This is a <B>paragraph</B></P>

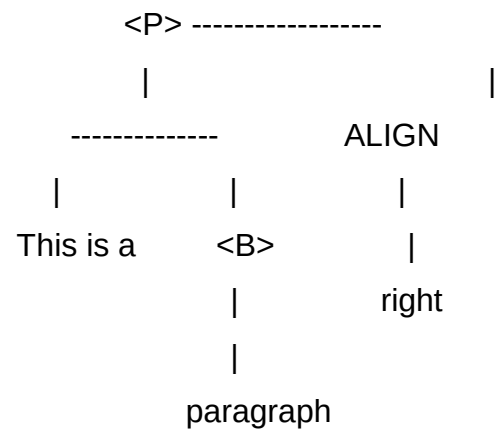
Ο κόμβος P έχει πλέον 2 child nodes, τον κόμβο κειμένου “This is a” και τον κόμβο B. Ο κόμβος B έχει δικό του child node, τον κόμβο κειμένου “paragraph”. Σχηματικά, η δομή της παραπάνω σύνταξης είναι:



Άλλη κατηγορία κόμβων είναι οι κόμβοι ιδιοτήτων, όπως το παράδειγμα:

<P ALIGN="right">This is a <B>paragraph</B></P>

το οποίο σχηματικά μας δίνει:



Το ίδιο το στοιχείο P έχει και αυτό parent node, το οποίο μπορεί να είναι ένα <DIV> ή το <BODY>.

Λόγω της δομής δέντρου που έχει το DOM, είναι εύκολο μα διατρεχθεί όλα το δέντρο έως ότου αποκτηθεί το επιθυμητό στοιχείο. Ας υποθέσουμε ότι ο τρέχων κόμβος είναι ο x. Για να χρησιμοποιήσουμε το parent node του η σύνταξη είναι:

x.parentNode

Αντίθετα για να χρησιμοποιήσουμε το child node:

x.childNodes[1]

Η αρίθμηση ξεκινάει από το 0 οπότε η παραπάνω σύνταξη αφορά το δεύτερο “παιδί” του x. Για πρόσβαση στο πρώτο η στο τελευταίο παιδί, μπορούν να χρησιμοποιηθούν αντίστοιχα και τα x.firstChild και x.lastChild.

Αν υποθέσουμε ότι το στοιχείο P των προηγούμενων διαγραμμάτων είναι ο πρώτος κόμβος του εγγράφου, τότε το στοιχείο B θα μπορούσε να αναφερθεί με τους εξής τρόπους, μεταξύ άλλων:

```
document.firstChild.firstChild.lastChild;
document.firstChild.childNodes[0].lastChild;
document.firstChild.childNodes[0].childNodes[1];
```

Διατρέχοντας όλο το δέντρο για να βρεθεί ένα στοιχεί δεν είναι πάντα ο καλύτερος τρόπος. Υπάρχουν και άλλοι πιο πρακτικοί τρόποι, ένας εκ των οποίων είναι και το getElementsByTagName, το οποίο κατασκευάζει έναν πίνακα με όλους τους κόμβους που έχουν ως tag το ζητούμενο. Στο προηγούμενο σχήμα, το B θα μπορούσε να αναφερθεί ως:

```
var x = document.getElementsByTagName('B')[0]
```

Χρησιμοποιούμε το 0 για να αναφερθούμε στο πρώτο στοιχείο του πίνακα με τους κόμβους που περιέχουν το tag B. Εναλλακτικά θα μπορούσαμε να χρησιμοποιήσουμε την εξής σύνταξη:

```
var x = document.getElementsByTagName('P')[0].lastChild;
```

καθώς το B είναι το πρώτο παιδί του πρώτου στοιχείου με tag P.

Ο καλύτερος τρόπος για να αναφερθεί ένας κόμβος είναι χρησιμοποιώντας το ID. Δίνοντας ένα μοναδικό ID στο στοιχείο B:

```
<P ALIGN="right">This is a <B ID=”here”>paragraph</B></P>
```

μπορούμε έπειτα να αναφερθούμε σε αυτό ως εξής:

```
var x = document.getElementById('here');
```

Με το DOM υπάρχει η δυνατότητα να αλλάξουμε την τιμή ενός κόμβου. Για να αλλάξουμε το περιεχόμενο του κόμβου Β μπορούμε να κάνουμε το εξής:

```
document.getElementById('here').firstChild.nodeValue=new text';
```

Με αυτήν την σύνταξη, εντοπίζουμε το στοιχείο με ID ‘here’ και αλλάζουμε το περιεχόμενό του με το κείμενο “new text”. Επίσης μπορούμε να αλλάξουμε τις ιδιότητες ενός κόμβου όπως το align:

```
node = document.getElementById('here').parentNode;
node.setAttribute('align', ’center’);
```

Επίσης μπορούμε να δημιουργήσουμε και να αφαιρέσουμε κόμβους. Για παράδειγμα μπορούμε να προσθέσουμε ένα <HR> στοιχείο:

```
var x = document.createElement('HR');
document.getElementById('here').appendChild(x);
```

Όπως μπορούμε να δούμε από τα παραδείγματα, η DOM βοηθάει στα μέγιστα στην πλήρη εκμετάλλευση των δυνατοτήτων των client_side scripting γλωσσών με αποτέλεσμα την μεγιστοποίηση του βαθμού διαδραστικότητας με τον χρήστη.

6. CSS

6.1 Εισαγωγή

Τα CSS είναι τα αρχικά του Cascading Style Sheets. Όπως το όνομα προτείνει, CSS είναι μια ειδικής χρήσης γλώσσα stylesheet που βοηθά στον καθορισμό της μορφής της παρουσίασης για οποιοδήποτε έγγραφο που έχει γραφτεί σε μια γλώσσα σήμανσης (markup language) όπως το HTML. Συνήθως, τα CSS χρησιμοποιούνται για τον καθορισμό της παρουσίασης των εγγράφων HTML και XHTML.

Ο βασικός σκοπός των CSS είναι να χωριστεί το περιεχόμενο ενός εγγράφου από την παρουσίασή του. Πριν τη δημιουργία των CSS, οι δημιουργοί ιστοσελίδων ήταν υποχρεωμένοι να τοποθετούν την πληροφορία για το πως μορφοποιείται ένα HTML έγγραφο μέσα στον HTML κώδικα. Το αποτέλεσμα ήταν η ανάπτυξη ιστόσελίδων να γίνεται συνεχώς πιο χρονοβόρα και δύσκολη, καθώς μια πληθώρα χαρακτηριστικών έπρεπε να οριστούν για κάθε στοιχείο HTML. Έτσι ο κώδικας ήταν ογκώδης και δυσνόητος.

Η λύση που προτάθηκε ήταν να αφαιρεθεί η πληροφορία για το πως θα εμφανίζεται το περιεχόμενο μιας ιστοσελίδας από τον HTML κώδικα. Πλέον τα HTML έγγραφα περιέχουν τη δομή της ιστοσελίδας, δηλαδή τι θα εμφανίζει η ιστοσελίδα. Το πως θα εμφανίζεται αυτό το περιεχόμενο, η μορφοποίηση, πλεον περιγράφεται με CSS. Σαν αποτέλεσμα έχουμε σημαντικά απλούστερο HTML κώδικα.

Ένα σημαντικό πλεονέκτημα που προκύπτει από τον διαχωρισμό περιεχομένου και παρουσίασης είναι ότι η ίδια σελίδα εμφανίζεται με διαφορετικό τρόπο αν εφαρμόσουμε διαφορετικά CSS. Αυτό δίνει στο χρήστη τη δυνατότητα να ρυθμίσει την εμφάνιση της σελίδας ανάλογα με τις ανάγκες του.

6.2 Ιστορία

Ο χωρισμός της δομής των εγγράφων από τη μορφοποίησή τους ήταν ένας στόχος του HTML από την σύλληψή του ως ιδέα το 1990. Ο πρώτος browser που εφάρμοζε style sheets για τον καθορισμό της μορφής του περιεχομένου ήταν ο NeXT. Στη συνέχεια ακολούθησαν και άλλοι browsers που εφάρμοζαν απλά style sheets όπως οι Viola και Harmony το 1993 και 1994 αντίστοιχα. Όλοι οι παραπάνω browsers παρείχαν περιορισμένες δυνατότητες μέσω των style sheets που εφάρμοζαν. Επιπλέον δεν γίνοταν πρόοδος ως προς τις δυνατότητες που παρέχονταν αφού πολύ συχνά καινούργοι browsers παρείχαν μειωμένες δυνατότητες σε σχέση με παλιότερους.

Τα CSS εμφανίστηκαν για πρώτη φορά το 1994 όταν ο Håkon Wium Lie δημοσίευσε την εργασία του με τίτλο Cascading HTML Style Sheets. Λίγους μήνες αργότερα ο Håkon σε συνεργασία με τον Bert Bos παρουσίασαν τα CSS στο συνέδριο για το Web στο Σικάγο. Τα CSS δεν ήταν η μόνη πρόταση για μια stylesheet γλώσσα. Ο λόγος που ξεχώρισε ήταν ότι πρότεινε μια ιδέα που δεν υπήρχε σε καμία από τις άλλες προτάσεις. Αυτή η ιδέα ήταν η ένοια του cascade. Σύμφωνα με την πρόταση του Håkon η μορφοποίηση ενός αρχείου δεν μπορούσε να γινεται μόνο από τον χρήστη ή μόνο από το δημιουργό. Αντίθετα το περιβάλλον του internet επέβαλε να γίνεται συνδυασμός των ρυθμίσεων του χρήστη και του δημιουργού.

Το επόμενο μεγάλο ορόσημο για τα CSS είναι το 1995 με τη δημιουργία του World Wide Web Consortium (W3C). Μία από τις πρώτες ομάδες που οργανώθηκαν από το W3C είχε σαν θέμα τα style sheets. Η ομάδα αυτή χρησιμοποίησε σαν βάση την προδιαγραφή των CSS toy Håkon και το Δεκέμβριο του 1996 δημοσιεύθηκε το CSS level 1 ως επίσημη πρόταση του

W3C. Δύο χρόνια αργότερα το 1998 δημοσιεύθηκε το CSS level 2 το οποίο προσέφερε πολλές επιπλέον δυνατότητες.

6.3 Συντακτικό

Τα CSS χρησιμοποιούν απλό συντακτικό. Συγκεκριμένα ένα style sheet αποτελείται από μια λίστα κανόνων. Κάθε κανόνας έχει δύο μέρη. Τον selector και το declaration block. Ένα απλό παράδειγμα CSS κανόνα είναι το ακόλουθο.

```
p { font-size: 2em }
```

Στο παραπάνω παράδειγμα ορίζουμε ότι το μέγεθος γραμματοσειράς για όλα τα P HTML στοιχεία θα είναι ίσο με 2em. Στο παράδειγμά μας το P αποτελεί τον selector και το { font-size: 2em } το declaration block. Αντίστοιχα το declaration block αποτελείται από ένα ή περισσότερα declarations που χωρίζονται μεταξύ τους με τον χαρακτήρα semicolon (;). Εκτός από τα declarations υπάρχει η δυνατότητα να ομαδοποιήσουμε και selectors χωρίζοντάς τους με κόμμα (,).

```
p, h1 {font-size: 2em }
```

Σε αυτό το παράδειγμα ορίζουμε το μέγεθος της γραμματοσειράς για όλα τα στοιχεία p και h1 με έναν κανόνα. Στην συνέχεια γίνεται πιο λεπτομερής αναφορά στους selectors και τα declarations.

6.3.1 Selectors

Οι selectors καθορίζουν ποια στοιχεία πρόκειται να επηρεαστούν από τον κανόνα CSS. Στην ουσία αποτελούν τον συνδετικό κρίκο ανάμεσα στο έγγραφο που θέλουμε να μορφοποιήσουμε και τους κανόνες μορφοποίησης. Τα CSS έχουν ένα επαρκές σύνολο από selectors που μας επιτρέπουν να προσδιορίσουμε εύκολα και με ακρίβεια για ποια στοιχεία θέλουμε να ισχύει ο κανόνας.

Οι selectors χωρίζονται σε δύο σημαντικές κατηγορίες. Στους απλούς (simple) και στους contextual selectors. Απλός selector είναι αυτός που ταιριάζει με ένα στοιχείο βασιζόμενος στον τύπο ή/και τις ιδιότητές του, αλλά όχι στη θέση του στοιχείου αυτού στη δομή του εγγράφου. Για παράδειγμα ο “p” είναι ένας απλός selector. Άλλος τύπος απλού selector είναι ο class selector ο οποίος μας επιτρέπει να επιλέξουμε στοιχεία ενός εγγράφου με βάση την τιμή του class attribute. Για παράδειγμα ο selector “.foo” επιλέγει όλα τα στοιχεία που ανήκουν στην κλάση foo. Αντίστοιχα ο “a.foo” επιλέγει όλα τα στοιχεία a που ανήκουν στην κλάση foo.

Αντίθετα contextual selector είναι αυτός που ταιριάζει με ένα στοιχείο βασιζόμενος στη θέση του στοιχείου αυτού στη δομή του εγγράφου. Οι contextual selectors προκύπτουν με την εμφώλευση απλών selectors μεταξύ τους. Για παράδειγμα ο “ul li” είναι ένας contextual selector ο οποίος ταιριάζει με όλα τα li στοιχεία που είναι απόγονοι ul στοιχείων. Όπως βλέπουμε λοιπον οι selectors αυτού του τύπου επιλέγουν στοιχεία με βάση τη θέση τους στο έγγραφο ή αλλιώς τη θέση τους σε σχέση με άλλα στοιχεία.

Μια τρίτη κατηγορία selectors είναι τα ψευδο-στοιχεία (pseudo-elements) και οι ψευδο-κλάσσεις (pseudo-classes). Ψευδο-στοιχείο είναι ένα μέρος ενός στοιχείου που δεν αντιστοιχεί σε ένα πραγματικό στοιχείο στο έγγραφο αλλά αντιστοιχεί σε

ένα αντικείμενο εμφάνισης. Τα ποιο γγνωστά ψευδο-στοιχεία που υποστηρίζουν τα CSS είναι τα first-letter και first-line τα οποία μας επιτρέπουν να ορίσουμε το στυλ για το πρώτο γράμμα και την πρώτη γραμμή αντλιστοιχα ενός στοιχείου. Για παράδειγμα p: first-letter {color:blue;}.

6.3.2 Declarations

Όπως είπαμε παραπάνω ένας κανόνας CSS αποτελείτε από δύο κομμάτια. Τον selector και το declaration block. Ο selector μας δείχνει για ποια στοιχεία θα ισχύει ο κανόνας. Τα declarations ορίζουν τι ακριβώς θα κάνει ο κανόνας, περιέχουν δηλαδή τη λειτουργικότητα του κανόνα.

Ένα declaration είναι της μορφής “font-size: 2em”. Όπως βλέπουμε αποτελείται από δύο τμήματα, την ιδιότητα (property) την τιμή (value). Γενικά λοιπόν όλα τα declarations ακολουθούν την παρακάτω σύνταξη: <property>: <value>.

Οι ιδιότητες στα CSS αντιπροσωπεύουν μια συνιστώσα μορφοποίησης ενός στοιχείου. Υπάρχουν αρκετές κατηγορίες ιδιοτήτων όπως για παράδειγμα η font που παρέχει ιδιότητες για τη μορφοποίηση της γραμματοσειράς ενός στοιχείου. Οι font-size και font-weight είναι λίγες μόνο από τις ιδιότητες αυτές και μας επιτρέπουν να ορίσουμε το μέγεθος και το βάρος της γραμματοσειράς αντίστοιχα.

Οι ιδιότητες ακολουθούνται πάντα από τις τιμές που τους αναθέτουμε. Οι τιμές αυτές μπορούν να έχουν διάφορες μορφές και εξαρτώνται αποκλειστικά από την ιδιότητα στην οποία αναφέρονται. Οι τιμές μπορούν να είναι αριθμοί ακολουθούμενοι από τις κατάλληλες μονάδες μέτρησης, ποσοστά %, αλφαριθμητικά και σε ορισμένες σπάνιες περιπτώσεις μέθοδοι.

Πλήρες reference για όλες της ιδιότητες του CSS και τις αντίστοιχες τιμές τους βρίσκεται online στην ιστοσελίδα του W3C.

6.4 Ανάθεση τιμών

Για να αποφύγουμε το να πρέπει να αναθέσουμε τιμές για όλες τις ιδιότητες σε όλα τα στοιχεία, τα CSS υλοποιούν μηχανισμούς που επιτρέπουν την ανάθεση τιμών αυτόματα (value propagation). Το κύριο όφελος της αυτόματης ανάθεσης τιμών είναι πολύ συνοπτικότερα αρχεία CSS, λιγότερος φόρτος εργασίας για τον δημιουργό και μικρή πιθανότητα για λάθη αφού κάθε ιδιότητα θα έχει τιμή ανά πάσα στιγμή.

Τα CSS έχουν τρεις βασικούς μηχανισμούς για την ανάθεση τιμής: cascading, κληρονομικότητα και αρχικές τιμές. Οι μηχανισμοί αυτοί εξασφαλίζουν ότι το σύνολο των στοιχείων ενός εγγράφου θα έχει μία τιμή για κάθε ιδιότητα οποιαδήποτε στιγμή. Η ισχύς των τριών μηχανισμών είναι διαφορετική. Το cascading είναι ο ισχυρότερος μηχάνισμός και επιβάλλεται των άλλων δύο. Αν η διαδικασία του cascading επιστρέψει μια τιμή τότε αυτή θα χρησιμοποιηθεί. Εάν το cascading δεν επιστρέψει τιμή ή επιστρέψει την τιμή inherit θα ενεργοποιηθεί ο μηχανισμός της κληρονομικότητας και θα χρησιμοποιηθεί η αντίστοιχη τιμή του στοιχείου γονέα. Ο μηχανισμός της αρχικής τιμής έρχεται τρίτος και θα χρησιμοποιηθεί μόνο εάν κανένας από τους άλλους μηχανισμούς δεν επιστρέψει κάποια τιμή ή αν επιστρέψουν την τιμή initial.

6.5 Cascading

Ο μηχανισμός cascading είναι ο πιο σύνθετος από τους τρεις και εξυπηρετεί διάφορους σκοπούς. Όταν υπάρχουν declarations σε περισσότερα από ένα style

sheets τα οποία προσπαθούν να θέσουν μια συγκεκριμένη τιμή σε μία ιδιότητα ενός στοιχείου, ο μηχανισμός cascading θα επιλέξει ένα κυρίαρχο declaration. Αυτό το declaration θα έχει πλήρη έλεγχο της τιμής της εν λόγω ιδιότητας.

Η επιλογή του κυρίαρχου κανόνα από το μηχανισμό cascading γίνεται ως εξής. Κάθε φορά που υπάρχει σύγκρουση κανόνων ελέγχονται τρεις παράγοντες. Η προέλευση (origin), η ακρίβεια (specificity) και η σειρά εμφάνισης. Οι παράγοντες αυτοί είναι διατεταγμένοι σε σειρά ισχυός. Δηλαδή η ακρίβεια λαμβάνεται υπόψιν μόνο αν η προέλευση δεν επιστρέψει το κυρίαρχο declaration. Αντίστοιχα η σειρά εμφάνισης λαμβάνεται υπόψιν μόνο αν οι άλλοι δύο παράγοντες δεν επιστρέψουν το κυρίαρχο declaration.

Υπάρχουν τρεις πιθανές προελεύσεις στα CSS. Δημιουργός, χρήστης και browser. Εξ ορισμού τα declarations του δημιουργού υπερισχύουν των declarations του χρήστη. Αντίστοιχα τα declarations του χρήστη υπερισχύουν των declarations του browser (default style sheet). Έχουμε τη δυνατότητα να δώσουμε αυξημένη βαρύτητα σε ένα declaration σημειώνοντάς το ως !important. Σε αυτή την περίπτωση το declaration θα υπερισχύει όλων των άλλων ασχέτως προέλευσης. Έτσι εξασφαλίζεται η δυνατότητα του χρήστη να ορίσει τη μορφωποίηση ενός αγγράφου σύμφωνα με τις ανάγκες του.

Ο δεύτερος παράγοντας, η ακρίβεια, έχει να κάνει με το πόσο συγκεκριμένος είναι ο στόχος ενός declaration. Ας πάρουμε το ακόλουθο παράδειγμα.

```
h3 {font-size: 2em}

p h3 {font-size: 3em}
```

Ο πρώτος κανόνας αναφέρεται σε όλα τα h3 στοιχεία. Ο δεύτερος κανόνας είναι πιο συγκεκριμένος γιατί αναφέρεται σε ένα υποσύνολο του πρώτου. Αναφέρεται σε όσα στοιχεία h3 είναι απόγονοι p στοιχείων. Άρα ο δεύτερος κανόνας υπερισχύει του πρώτου γιατί είναι πιο συγκεκριμένος.

Αν ο παράγοντας της ακρίβειας δεν μας δώσει κάποιο κυρίαρχο declaration, γιατί υπάρχουν δύο ή περισσότερα με την ίδια ακρίβεια, τότε λαμβάνεται υπόψιν η σειρά εμφάνισης. Τα declarations που εμφανίζονται αργότερα σε ένα style sheet έχουν μεγαλύτερη βαρύτητα από αυτά που εμφανίζονται νωρίτερα.

7. Ανάλυση και υλοποίηση του toolbar

Όπως αναφέρθηκε ο στόχος της πτυχιακής είναι η δημιουργία ενός toolbar που θα παρέχει δυνατότητες αναζήτησης στο web με τη βοήθεια γνωστών μηχανών αναζήτησης. Θα παρέχει λοιπόν λειτουργίες μετα-αναζήτησης. Σε αυτό το κεφάλαιο θα παρουσιαστούν οι απαιτήσεις του συστήματος που θέλουμε να υλοποιήσουμε και στη συνέχεια θα αναλυθούν ώστε να προσδιοριστούν τα προβλήματα που θα αντιμετωπίσουμε. Σε αυτά τα προβλήματα καλούμαστε να βρούμε λύσεις πάνω στις οποίες θα βασιστούμε όταν θα περάσουμε στο στάδιο του σχεδιασμού και της υλοποίησης.

7.1 Περιγραφή του συστήματος

Αρχικά παραθέτουμε την περιγραφή του συστήματος όπως δίνεται στην περιγραφή της πτυχιακής. Αυτό το κείμενο περιγράφει τους στόχους του συστήματός μας. Έτσι αποτελεί τη βάση από την οποία θα ξεκινήσουμε για να προσδιορίσουμε υποπροβλήματα και να βρούμε λύσεις.

«Μέσω του toolbar ο χρήστης θα έχει την δυνατότητα να κάνει συγχρόνως αναζήτηση σε όλες τις γνωστές μηχανές αναζήτησης (google, live search, yahoo, ask, κτλ). Η εφαρμογή θα συλλέγει όλα τα αποτελέσματα και θα παρουσιάζει στον χρήστη ενιαία λίστα αποτελεσμάτων. Για αυτό το βήμα απαιτείται «parsing» των αποτελεσμάτων αναζήτησης κάθε μηχανής. Το σύστημα θα πρέπει να είναι ευέλικτο ώστε να μπορούν να προστεθούν αργότερα νέες μηχανές αναζήτησης (νέοι κανόνες parsing).»

«Ο χρήστης θα μπορεί να αποθηκεύσει το ερώτημα αναζήτησης καθώς και τα αποτελέσματα της αναζήτησης. Θα πρέπει να γίνεται παρακολούθηση (tracking) των «click» του χρήστη. Ο χρήστης θα μπορεί να κάνει ανάκτηση ένα

αποθηκευμένο ερώτημα και με επιλογή του να του εμφανίζονται μόνο οι σελίδες που δεν έχει επισκεφτεί κατά την προηγούμενη αναζήτηση ή μόνο οι νέες σελίδες.

Σημείωση: Δεν θα πρέπει να γίνει ανάπτυξη ιστοσελίδας. Όλες οι πληροφορίες θα αποθηκεύονται στον υπολογιστή του χρήστη.»

Από την παραπάνω περιγραφή μπορούμε να διακρίνουμε δύο σημαντικούς στόχους που αποτελούν τις δύο λειτουργίες που η εφαρμογή μας καλείται να εκτελεί. Η πρώτη λειτουργία είναι αυτή της αναζήτησης. Θέλουμε η εφαρμογή μας να δέχεται έναν όρο αναζήτησης από το χρήστη και να επιστρέφει τα αποτελέσματα. Άρα θα πρέπει το toolbar να δέχεται έναν όρο αναζήτησης, να στέλνει τον όρο αυτό στις μηχανές αναζήτησης, να παίρνει τα αποτελέσματα και να παράγει μια ενιαία λίστα αποτελεσμάτων.

Η δεύτερη λειτουργία που θέλουμε είναι η αποθήκευση των αποτελεσμάτων μιας αναζήτησης. Συγκεκριμένα μας ενδιαφέρει όταν ο χρήστης φορτώνει μια αποθηκευμένη αναζήτηση να έχει τη δυνατότητα με επιλογή του να φιλτράρει τα αποτελέσματα με τα εξής κριτήρια. Καταρχάς να βλέπει μόνο όσα αποτελέσματα δεν έχει επισκευθεί και δεύτερον να βλέπει μόνο όσα αποτελέσματα δεν είχε επιστρέψει η αρχική αναζήτηση.

Τέλος η περιγραφή του συστήματος μας δίνει δύο περιορισμούς που πρέπει να ικανοποιήσουμε. Πρώτον πρέπει να είναι εύκολο να προσθέσουμε νέες μηχανές αναζήτησης στην εφαρμογή με τις λιγότερες δυνατές αλλαγές. Πρέπει στην ουσία να φροντίσουμε ο κώδικας της εφαρμογής να είναι ανεξάρτητος από την πληροφορία των μηχανών αναζήτησης. Ο δεύτερος περιορισμός είναι ότι η εφαρμογή πρέπει να είναι εξ ολοκλήρου client-side. Με εξαίρεση την συλλογή αποτελεσμάτων από τις μηχανές αναζήτησης, όλες οι άλλες λειτουργίες, όπως η

επεξεργασία των αποτελεσμάτων ή η αποθήκευση ερωτημάτων, πρέπει να γίνεται στον υπολογιστή του χρήστη.

7.1.2 Αναζήτηση

Θα ξεκινήσουμε αναλύοντας διεξοδικά τη λειτουργία της αναζήτησης. Όπως είδαμε νωρίτερα μια μηχανή μετα-αναζήτησης πρέπει να εκτελεί τις εξής λειτουργίες. Να δέχεται όρο αναζήτησης από το χρήστη, να κάνει μια επεξεργασία του όρου για να τον στείλει στις μηχανές αναζήτησης, να γίνει συλλογή των αποτελεσμάτων από τις μηχανές αναζήτησης, να δημιουργηθεί μια ενιαία λίστα αποτελεσμάτων ταξινομημένα ανάλογα με τη σχετικότητα και τέλος να παρουσιαστούν τα αποτελέσματα στο χρήστη. Γενικά διακρίνουμε τα παρακάτω βασικά βήματα.

1. Το πρόγραμμα δέχεται τον όρο αναζήτησης από το χρήστη.
2. Γίνεται επεξεργασία του όρου αναζήτησης.
3. Το πρόγραμμα στέλνει το ερώτημα στις μηχανές αναζήτησης που υποστηρίζει.
4. Γίνεται συλλογή των αποτελεσμάτων από το πρόγραμμα και δημιουργείται μια ενιαία λίστα αποτελεσμάτων.
5. Γίνεται ταξινόμηση των αποτελεσμάτων με βάση τα κριτήρια που ορίσαμε.
6. Εμφανίζεται η ταξινομημένη λίστα αποτελεσμάτων στον χρήστη.

Βήμα 1

Το πρώτο βήμα λοιπόν είναι να περάσουμε τον όρο αναζήτησης στο πρόγραμμα. Αυτό θα γίνεται μέσω της διεπαφής της εφαρμογής. Θα πρέπει δηλαδή το toolbar να έχει τα εξής controls. Ένα textbox στο οποίο ο χρήστης θα γράφει τον όρο αναζήτησης και ένα button το οποίο θα ξεκινάει την διαδικασία της αναζήτησης.

Βήμα 2

Η επεξεργασία του όρου αναζήτησης έχει να κάνει κυρίως με την απαλοιφή περιττών κενών και τη δημιουργία μιας παραμέτρου έτοιμης να δοθεί στις μηχανές αναζήτησης. Συγκεκριμένα αφαιρούμε τα κενά από το τέλος και την αρχή του όρου αναζήτησης και αντικαθιστούμε τυχών πολλαπλά κενά ανάμεσα στις λέξεις με ένα κενό. Στη συνέχεια ενώνουμε τις λέξεις με το χαρακτήρα '+'. Για παράδειγμα ο όρος αναζήτησης «firefox extension » θα μετατραπεί στον όρο «firefox+extension».

Βήμα 3

Το δεύτερο βήμα είναι πολύ πιο περίπλοκο. Καταρχάς πρέπει να δούμε με ποιο τρόπο θα γίνεται η αποστολή του ερωτήματος στην μηχανή αναζήτησης. Παίρνοντας ως παράδειγμα τη μηχανή αναζήτησης google και κάνοντας αναζήτηση για τον όρο toolbar βλέπουμε ότι η σελίδα αποτελεσμάτων έχει την εξής url διεύθυνση.

http://www.google.gr/#hl=el&source=hp**&q=toolbar**&btnG=%CE%91%CE%BD%CE%B1%CE%B6%CE%AE%CF%84%CE%B7%CF%83%CE%B7+Google&meta=&aq=1&oq=too&fp=29cbd9a5056d14fc

Με έντονη γραφή σημειώνεται το πιο σημαντικό τμήμα της παραπάνω διεύθυνσης. Όπως βλέπουμε τα ορίσματα της αναζήτησης στέλνονται στον server της google μέσω της διεύθυνσης url. Άρα αρκεί να δημιουργήσουμε μία διεύθυνση της μορφής http://www.google.gr/#q=<search_term> όπου <search_term> είναι ο όρος αναζήτησης που εισάγει ο χρήστης στο toolbar.

Με αυτή την προσέγγιση προκύπτουν δύο θέματα. Καταρχάς πρέπει να γίνει η κατάλληλη επεξεργασία του όρου αναζήτησης πριν ενωθεί με την διεύθυνση. Συγκεκριμένα πρέπει να εξαλειφθούν τα κενά και οι λέξεις να ενωθούν με το χαρακτήρα +. Για παράδειγμα ο όρος «toolbar αναζήτησης» θα πρέπει να πάρει την μορφή «toolbar+αναζήτησης». Το δεύτερο θέμα είναι που θα αποθηκεύουμε τα url της κάθε μηχανής αναζήτησης και το που θα φορτώνουμε τα αποτελέσματα.

Η λύση που επιλέχθηκε είναι να χρησιμοποιήσουμε την τεχνική των hidden frames. Συγκεκριμένα θα έχουμε μια HTML σελίδα τοπικά την οποία θα φορτώνουμε όταν θα κάνουμε αναζήτηση. Στο κύριο μέρος της σελίδας θα εμφανίζονται τα αποτελέσματα που θα επιστρέφει το toolbar. Τα αποτελέσματα κάθε μηχανής ξεχωριστά θα φορτώνονται σε ένα κρυμένο iframe μέσα στην σελίδα. Έτσι το πρόγραμμα θα μαζεύει στοιχεία από τα hidden iframes, θα τα επεξεργάζεται και θα εμφανίζει τα αποτελέσματα στην κύρια σελίδα. Ο HTML κώδικας της σελίδας θα είναι ο εξής.

```
<html>

<head>

<title>Anazitisi-Seach Results</title>

<link rel="stylesheet" type="text/css" href="results.css"/>

<body>

<iframe id="frame1" src="" width="30%" height="100" frameborder="0"></iframe>

<div id="results">

<ul id="mylist">
```

```
</ul>

</div>

</body>

</html>
```

Το ζήτημα λοιπόν είναι για κάθε μηχανή αναζήτησης να αλλάξουμε το src attribute του αντίστοιχου iframe. Η πληροφορία για κάθε μηχανή αναζήτησης θα πρέπει να βρίσκεται σε ένα αρχείο από όπου θα φορτώνεται. Για πρακτικούς λόγους η καλύτερη επιλογή είναι ένα xml αρχείο καθώς προσφέρει ευκολίες στην εύρεση πληροφορίας χάρη στην οργανωμένη δομή του. Για κάθε μηχανή αναζήτησης χρειαζόμαστε το όνομα και το url της.

```
<?xml version="1.0"?>

<search_engine_list>

  <search_engine>

    <name>Google</name>

    <url>http://www.google.com/search?q=</url>

  </search_engine>

</search_engine_list>
```

Με αυτή την προσέγγιση για να προσθέσουμε μια νέα μηχανή αναζήτησης χρειάζεται να κάνουμε δύο πράγματα. Να προσθέσουμε ένα iframe στην HTML σελίδα και να προσθέσουμε την πληροφορία της μηχανής στο xml αρχείο.

Βήμα 4

Αυτό το βήμα έχει να κάνει με τη συλλογή των αποτελεσμάτων από τις διάφορες μηχανές αναζήτησης και τη δημιουργία μιας ενιαίας λίστας αποτελεσμάτων. Η συλλογή των αποτελεσμάτων γίνεται σχετικά εύκολα σε συνάρτηση με τα iframes. Δηλαδή αφού έχουμε φορτώσει τις σελίδες με τα αποτελέσματα στα hidden frames το μόνο που μένει είναι να χρησιμοποιήσουμε το DOM για να έχουμε πρόσβαση σε αυτά.

Με έναν πρόγραμμα όπως το DOM inspector μπορούμε να δούμε τη δομή των ιστοσελίδων αναζήτησης. Αυτό που θα παρατηρήσουμε είναι ότι για όλες τις μηχανές αναζήτησης τα αποτελέσματα βρίσκονται μέσα σε μια HTML λίστα. Αν ξέρουμε τι τύπου λίστα είναι αυτή μπορούμε να έχουμε πρόσβαση στα αποτελέσματα. Επειδή μπορεί να υπάρχουν και άλλες λίστες ίδιου τύπου στην ιστοσελίδα θα χρησιμοποιούμε το πιο κοντινό στη λίστα στοιχείο που βρίσκεται ψηλότερα στην ιεραρχία του δέντρου και το οποίο έχει μοναδικό id. Έτσι στο xml αρχείο θα προσθέσουμε τα δύο παρακάτω χαρακτηριστικά για κάθε μηχανή.

```
<search_engine>
  <name>Google</name>
  <url>http://www.google.com/search?q=</url>
  <results_identifier>res</results_identifier>
  <results_container>ol</results_container>
</search_engine>
```

Το results_identifier είναι το id του στοιχείου που περιέχει τη λίστα αποτελεσμάτων. Το results_container είναι ο τύπος της λίστας αποτελεσμάτων.

Το άλλο ζήτημα είναι να δημιουργήσουμε μια ενιαία λίστα αποτελεσμάτων. Κάθε μηχανή αναζήτησης θα μας επιστρέφει τα δικά της αποτελέσματα και αναπόφευκτα κάποια URL θα εμφανίζονται στα αποτελέσματα δύο διαφορετικών μηχανών αναζήτησης. Εμείς θέλουμε να κάθε φορά που ένα URL εμφανίζεται πολλές φορές να το κρατάμε μια φορά στην τελική λίστα.

Βήμα 5

Στο στάδιο αυτό θέλουμε να βαθμολογήσουμε τα αποτελέσματα της τελικής λίστας ως προς τη σχετικότητά τους με το ερώτημα του χρήστη. Έπειτα με βάση τη βαθμολογία αυτή θα γίνει ταξινόμηση των στοιχείων της λίστας από το πιο σχετικό προς το λιγότερο σχετικό. Η βαθμολόγηση των αποτελεσμάτων αποφασίστηκε να γίνεται σε σχέση με τη σειρά που κάθε αποτέλεσμα εμφανίζεται στην εκάστοτε μηχανή αναζήτησης. Συγκεκριμένα η τελική βαθμολογία κάθε αποτελέσματος είναι το άθροισμα της θέσης που εμφανίζεται το URL σε κάθε μηχανή αναζήτησης, προς το πλήθος των μηχανών αυτών. Για παράδειγμα αν ένα αποτέλεσμα εμφανίζεται στο Google την πρώτη θέση και στο Yahoo στην τρίτη, η βαθμολογία του στην τελική λίστα θα είναι (1+3)/2 = 2. Όσο μικρότερος είναι αυτός ο αριθμός τόσο πιο σχετικό θα είναι το αποτέλεσμα.

Αυτός ο τρόπος βαθμολόγησης δίνει βαρύτητα στη θέση του αποτελέσματος σε κάθε μηχανή αναζήτησης. Όσο πιο ψηλά βρίσκεται στη λίστα τόσο πιο καλή βαθμολογία θα έχει το URL στην τελική λίστα. Επιπλέον χρησιμοποιούμε το μέσο όρο για να αποφύγουμε το εξής πρόβλημα. Ένα URL που εμφανίζεται σε μια μηχανή στην πρώτη θέση και σε μια άλλη στην τρίτη θα βγάζει βαθμολογία με απλό άθροισμα 4. Ένα άλλο URL που εμφανίζεται σε μια μηχανή στην τρίτη θέση

θα έχει βαθμολογία 3. Αυτό όμως είναι λάθος γιατί το πρώτο URL είναι προφανώς πιο σχετικό. Χρησιμοποιώντας το μέσο όρο οι βαθμολογίες θα είναι αντίστοιχα 2 και 3.

Αφού έχει λυθεί το ζήτημα της βαθμολόγησης των αποτελεσμάτων το μόνο που απομένει είναι ταξινομηθεί η λίστα ως προς το πιο σχετικό. Άρα σύμφωνα με τα παραπάνω θα πρέπει να γίνει ταξινόμηση με βάση τη βαθμολογία από το μικρότερο αριθμό προς το μεγαλύτερο.

Βήμα 6

Το τελευταίο στάδιο στη λειτουργία της αναζήτησης είναι αυτό της παρουσίασης των αποτελεσμάτων στο χρήστη. Αφού έχουμε τη τελική λίστα έτοιμη πρέπει να εμφανίσουμε τα αποτελέσματα. Όπως είπαμε, όταν κάνουμε αναζήτηση ανοίγει μια καινούργια σελίδα στην οποία υπάρχουν και τα hidden frames που χρησιμοποιούμε. Στο υπόλοιπο μέρος της σελίδας μπορούμε να φορτώσουμε τα αποτελέσματα μέσα σε μια λίστα.

Τα αποτελέσματα εμφανίζονται με τη κλασσική μορφή που χρησιμοποιείται από τις περισσότερες μηχανές αναζήτησης. Δηλαδή έχουμε σαν τίτλο το link που οδηγεί στην ιστοσελίδα και στη συνέχεια μία παράγραφο που περιγράφει την ιστοσελίδα. Η μορφοποίηση της σελίδας αποτελεσμάτων γίνεται με τη χρήση CSS. Παράμετροι που ορίζονται με τα CSS είναι οι αποστάσεις ανάμεσα στα HTML στοιχεία, μέγεθος και χρώμα γραμματοσειρών, τα iframes χρησιμοποιούνται ορίζεται να είναι hidden ώστε να μην εμφανίζονται στην σελίδα και τα λοιπά.

Μία επιπλέον απαίτηση είναι να φαίνεται δίπλα σε κάθε αποτέλεσμα πια μηχανή αναζήτησης το επέστρεψε και σε ποια θέση. Αυτό γίνεται προσθέτοντας δίπλα σε κάθε τίτλο ένα string της μορφής «Result returned by: <name>: <position>» όπου name είναι το όνομα της μηχανής αναζήτησης και position η θέση που βρισκόταν το αποτέλεσμα. Για παράδειγμα «Result returned by: Google: 3 Yahoo:5 Bing:3».

7.1.2 Αποθήκευση και φόρτωση αναζήτησης

Η δεύτερη λειτουργία που θέλουμε να εκτελεί η εφαρμογή μας είναι η αποθήκευση και φόρτωση αναζήτησης. Αναλυτικά θέλουμε η εφαρμογή να παρακολουθεί τα clicks του χρήστη για να βλέπει πια links επισκέφτηκε. Στην συνέχεια ο χρήστης θα μπορεί να κάνει αποθήκευση της αναζήτησης. Στη συνέχεια όταν ο χρήστης φορτώσει την αναζήτηση το πρόγραμμα θα μαρκάρει όσα links έχει επισκεφτεί ο χρήστης καθώς και όσα είχαν εμφανιστεί στην προηγούμενη αναζήτηση. Έπειτα ο χρήστης θα μπορεί να κρύβει όσα links έχει επισκεφτεί ή τα links που εμφανίστηκαν στην προηγούμενη αναζήτηση.

Αρχικά θα δούμε τη διαδικασία της αποθήκευσης. Όπως είδαμε αποτελείτε από δύο κομμάτια, την παρακολούθηση των clicks του χρήστη και την αποθήκευση των links αυτών. Το να βρούμε πια links επισκέφτηκε ο χρήστης μπορεί να γίνει χρησιμοποιώντας το click event. Κάθε φορά που ο χρήστης κάνει click στην ιστοσελίδα θα βλέπουμε σε τι αντικείμενο έγινε το click. Αν το στοιχείο αυτό ήταν link και το parent node του σύμφωνα με το DOM είναι h3 element τότε ο χρήστης επισκέφτηκε ένα αποτέλεσμα. Σε αυτήν την περίπτωση κρατάμε το URL αυτό.

Όταν ο χρήστης κάνει αποθήκευση της αναζήτησης η εφαρμογή θα σώσει όσα URL έχει επισκεφτεί ο χρήστης ως visited και στη συνέχεια θα σώσει τα υπόλοιπα URL ως old για να δείξει ότι αυτά τα links είχαν εμφανιστεί στην προηγούμενη αναζήτηση. Αυτά τα δεδομένα θέλουμε να αποθηκευτούν τοπικά

στον υπολογιστή του χρήστη. Η αποθήκευση επιλέξαμε να γίνεται σε αρχεία τύπου xml για τις ευκολίες που παρουσιάζουν κατά την ανάγνωση και τον καλά δομημένο χαρακτήρα τους. Το αρχείο θα έχει την ακόλουθη δομή.

<?xml version=\"1.0\" encoding=\"UTF-8\"?>

<saved_url>

<search_name>searchTerms</search_name>

<old>

<url>link1</url>

<url>link2</url>

<url>link3</url>

</old>

<visited>

<url>link1</url>

<url>link2</url>

</visited>

</saved_url>

Αφού έχει αποθηκεύσει μια αναζήτηση ο χρήστης θα μπορεί στη συνέχεια να φορτώσει την αναζήτηση αυτή. Στην ουσία αυτό που θα γίνεται είναι να διαβάζει η εφαρμογή το αρχείο που αποθηκεύσαμε. Θα κάνει μια νέα αναζήτηση για αυτόν τον όρο και θα μαρκάρει όσα από τα αποτελέσματα είναι μέσα στις λίστες visited και old. Δηλαδή τα αποτελέσματα που έχουν ίδιο URL με κάποιο URL από

αυτά που βρίσκονται στο αποθηκευμένο αρχείο ως visited, μαρκάρονται ανάλογα. Στη συνέχεια ο χρήστης μπορεί να κρύβει τα αποτελέσματα αυτά με τη χρήση ενός check box. Η απόκρυψη των αποτελεσμάτων θα γίνεται με τη χρήση CSS. Στα στοιχεία που είναι μαρκαρισμένα ως visited θα αλλάζουμε το χαρακτηριστικό visibility στην τιμή hidden όπως ακριβώς κάναμε για τα hidden frames.

7.1.3 Επιλογές αναζήτησης

Η εφαρμογή υποστηρίζει κάποια options για την αναζήτηση. Συγκεκριμένα υπάρχουν οι επιλογές για το πλήθος των αποτελεσμάτων που θα συλλέγει η εφαρμογή, το πλήθος των τελικών αποτελεσμάτων που θα επιστρέφει και η δυνατότητα να επιλέξει ο χρήστης ποιες μηχανές θα χρησιμοποιηθούν.

Το πόσα αποτελέσματα θα συλλέγει η εφαρμογή θα γίνεται με το να περνάμε παράμετρο στο URL που αποστέλλουμε στη μηχανή. Για παράδειγμα η google χρησιμοποιεί το & num=. Έτσι θα προσθέσουμε μια τελευταία παράμετρο στο αρχείο με τα στοιχεία κάθε μηχανής. Η παράμετρος θα είναι το ακόλουθο xml στοιχείο <results_number>&num=</results_number>.

Το πλήθος των τελικών αποτελεσμάτων μπορούμε να το ελέγξουμε πολύ απλά με το να εμφανίζουμε τα πρώτα αποτελέσματα από την τελική λίστα μέχρι το πλήθος που ορίζουμε. Τέλος οι μηχανές που θα χρησιμοποιηθούν θα επιλέγονται από το σύνολο των μηχανών που έχουμε εισάγει στην εφαρμογή.

7.2 Υλοποίηση του toolbar για Mozilla Firefox

Στη συνέχεια θα παρουσιάσουμε το πώς έγινε η ανάπτυξη του toolbar για τον browser Firefox της Mozilla. Αρχικά θα δείξουμε το interface της εφαρμογής και στη συνέχεια θα παρουσιάσουμε το πώς αντιμετωπίσαμε τα διάφορα προβλήματα που επισημάναμε στο στάδιο της ανάλυσης.

7.2.1 Το interface του toolbar

Το toolbar αποτελείται από τα παρακάτω controls.

- Ένα textbox στο οποίο ο χρήστης γράφει τον όρο αναζήτησης
- Ένα button με το οποίο ο χρήστης ξεκινά την διαδικασία της αναζήτησης
- Δύο check boxes με τα οποία ο χρήστης κρύβει όσα αποτελέσματα έχει επισκεφτεί ή όσα είναι παλιά, όταν έχει φορτώσει μια αναζήτηση.
- Δύο buttons για αποθήκευση και φόρτωση αναζήτησης
- Ένα button με το οποίο ο χρήστης επιλέγει διάφορες ρυθμίσεις για το toolbar.

Στη συνέχεια παρουσιάζεται ο xul κώδικας που υλοποιεί το παραπάνω interface.

```
<?xml version="1.0"?>

<overlay id="ptyxiaki_overlay"

        xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">

<script type="application/x-javascript" src="chrome://ptyxiaki/content/anazitisi.js"/>

<toolbox id="navigator-toolbox">

   <toolbar id="anazitisi_toolbar" toolbarname="Ptyxiaki Anazitisi" class="chromeclass-
toolbar"
```

```
context="toolbar-context-menu" insertafter="PersonalToolbar">
    <toolbaritem id="Anazitisi_SearchTermsContainer" persist="width">
      <menulist id="Anazitisi_SearchTerms" editable="true" flex="1"
          minwidth="100" width="250" onkeypress="Anazitisi_Search(event)">
      <menupopup id="Anazitisi_SearchTermsMenu"
onpopupshowing="Anazitisi_PopulateList()"/>
      </menulist>
    </toolbaritem>
    <toolbarbutton id="anazitisi_button" label="Search"
onclick="Anazitisi_Search(event)"/>
    <checkbox id="anazitisi_check1" label="Hide visited" checked="false"
onclick="Anazitisi_hideVisited()"/>
    <checkbox id="anazitisi_check2" label="Hide old" checked="false"
onclick="Anazitisi_hideOld()"/>
    <spacer flex="1"/>
    <toolbarbutton id="anazitisi_save" label="Save Search"
onclick="Anazitisi_saveSearch()"/>
    <toolbarbutton id="anazitisi_load" label="Load Search"
onclick="Anazitisi_loadSearch()"/>
    <toolbarbutton id="anazitisi_options" label="Options"
oncommand="Anazitisi_searchOptions()"/>
   </toolbar>
  </toolbox>
</overlay>
```

7.2.2 Αναζήτηση

Όπως έχουμε πει ο Firefox χρησιμοποιεί JavaScript για να δημιουργήσει τα extensions του. Σε αυτό το σημείο θα δείξουμε πως υλοποιούμε τη διαδικασία της αναζήτησης. Το πρώτο στάδιο ξεκινάει όταν καλείται η μέθοδος που έχουμε κάνει attach στο button αναζήτησης και στο textbox. Συγκεκριμένα όταν καλείται η μέθοδος ελέγχει τι τύπου ήταν το event που την κάλεσε. Έχουμε τρεις περιπτώσεις. Με απλό click ανοίγει η σελίδα της αναζήτησης στο tab που βρισκόμαστε τώρα. Με μεσαίο click ανοίγει σε νέο tab και τέλος αν πατήσουμε το πλήκτρο enter όταν το focus βρίσκεται στο textbox φορτώνει τη σελίδα στο tab που βρισκόμαστε.

```
if(event.button==0) {
        window.content.document.location = "chrome://ptyxiaki/content/results.html";
        t=window.setTimeout("Anazitisi_LoadFrames()",100);
    }
    else if(event.button==1) {
        gBrowser.selectedTab =
gBrowser.addTab("chrome://ptyxiaki/content/results.html");
        t=window.setTimeout("Anazitisi_LoadFrames()",100);
    }
    else if(event.keyCode == event.DOM_VK_RETURN) {
        window.content.document.location = "chrome://ptyxiaki/content/results.html";
        t=window.setTimeout("Anazitisi_LoadFrames()",100);
    }
```

Όπως βλέπουμε η μέθοδος που καλείται είναι η Anazitisi_LoadFrames() η οποία έχει ως στόχο να φορτώσει στα hidden frames τα αποτελέσματα από τις μηχανές αναζήτησης.

Εδώ διακρίνουμε τα εξής στάδια. Αρχικά διαβάζουμε από το xml αρχείο τα χαρακτηριστικά των μηχανών αναζήτησης. Αυτό περιλαμβάνει το όνομα της μηχανής αναζήτησης, το URL της και το όρισμα που ορίζει πόσα αποτελέσματα θα επιστρέψει η μηχανή αναζήτησης.

```
var search_engine_list = xmlDoc.getElementsByTagName("name");

var url_list = xmlDoc.getElementsByTagName("url");

var s_results_number = xmlDoc.getElementsByTagName("results_number");

var prefBranch = prefService.getBranch("extensions.anazitisiprefs.");

var engineResultsNum = prefBranch.getIntPref("searchEngineResultsNumber");
```

Στη συνέχεια διαβάζουμε το option για το ποιες μηχανές αναζήτησης θα χρησιμοποιηθούν.

```
var prefService = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefService);

var prefBranch2 = prefService.getBranch("extensions.anazitisiprefs.searchEngines.");
```

Στο επόμενο στάδιο ελέγχουμε αν οι μηχανές έχουν επιλεχθεί να πάρουν μέρος στην αναζήτηση και αν έχουν επιλεγεί φορτώνουμε το URL στα frames.

```
for(i=0;i<url_list.length;i++) {
        if(prefBranch2.getBoolPref(search_engine_list[i].childNodes[0].nodeValue)) {
            frames_list[i].id = search_engine_list[i].childNodes[0].nodeValue;
            var final_url = url_list[i].childNodes[0].nodeValue + searchTerms +
s_results_number[i].childNodes[0].nodeValue + engineResultsNum;
            frames_list[i].src = final_url;
            frames_list[i].className = "used";
        }
}
```

Τέλος καλούμε την επόμενη μέθοδο που συλλέγει πληροφορίες από τα frames. Επειδή όμως πρέπει να φορτώσουν πλήρως οι ιστοσελίδες πριν μπορούμε να έχουμε πρόσβαση στο DOM τους καλούμε τη μέθοδο με κάποια χρονική καθυστέρηση.

```
t=window.setTimeout("Anazitisi_getcontent()",2000);
```

Η μέθοδος Anazitisi_getcontent() κάνει την εξαγωγή αποτελεσμάτων από τα frames. Διαβάζοντας τις αντίστοιχες παραμέτρους από το xml αρχείο μαζεύουμε τα κατάλληλα DOM elements σε ένα δισδιάστατο πίνακα, τον results_per_engine που περιέχει τα αποτελέσματα ανά μηχανή αναζήτησης.

```
var results_per_engine = new Array();
for(i=0;i<frames_list.length;i++) {
```

```
    for(k=0;k<engine_names.length;k++) {


        if(frames_list[i].id == engine_names[k].childNodes[0].nodeValue) {


            results_per_engine[i] = new Array();

            results_per_engine[i][0] = frames_list[i].id;


            var get_result_id =
frames_list[i].contentDocument.getElementById(results_id[k].childNodes[0].nodeValue);

            var get_result_container =
get_result_id.getElementsByTagName(results_container[k].childNodes[0].nodeValue)[0];

            var search_nodes = get_result_container.getElementsByTagName("li");

        }

    }

}
```

Τέλος καλούνται τρεις μέθοδοι οι Anazitisi_combineResults(results_per_engine), Anazitisi_sortResults(combined_results) και Anazitisi_showResults(sorted_results).

Οι πρώτη μέθοδος δέχεται σαν όρισμα τον πίνακα results_per_engine που περιέχει τα αποτελέσματα ανά μηχανή και επιστρέφει ένα πίνακα που περιέχει την ενοποιημένη λίστα αποτελεσμάτων. Αυτή η λίστα είναι ένας πίνακας που

εκτός από τα αποτελέσματα περιέχει για κάθε αποτέλεσμα τη βαθμολογία του και μια λίστα με τις μηχανές που το επέστρεψαν και τη θέση εμφάνισης ανά μηχανή.

```
var combined_results = new Array();

for(j=0;j<results_per_engine[0].length-1;j++) {
    var temp = Anazitisi_createCombinedResult(results_per_engine[0],j+1);
    combined_results[j] = temp;
}

for(i=1;i<results_per_engine.length;i++) {
    for(j=1;j<results_per_engine[i].length;j++) {
        var result_exists = false;
        for(k=0;k<combined_results.length;k++) {
            var a = Anazitisi_getresultURL(results_per_engine[i][j]);
            var b = Anazitisi_getresultURL(combined_results[k][0]);
            if(a == b) {
                combined_results[k][1] += j;
                combined_results[k][2].push(new
search_engine_reference(results_per_engine[i][0],j));
                combined_results[k][1] /= combined_results[k][2].length;
                result_exists = true;
                break;
```

```
            }
        }
        if(!result_exists) {
            var temp = Anazitisi_createCombinedResult(results_per_engine[i],j);
            combined_results.push(temp);
        }
    }
}
return combined_results;
```

Η Anazitisi_sortResults(combined_results) ταξινομεί τον πίνακα combined_results ως προς τη βαθμολογία κάθε αποτελέσματος όπως αυτό ορίστηκε από την μέθοδο Anazitisi_combineResults.

```
for(i=0;i<combined_results.length;i++) {
    for(j=i+1;j<combined_results.length;j++) {
        if(combined_results[j][1]<combined_results[i][1]) {
            var temp = combined_results[i];
            combined_results[i] = combined_results[j];
            combined_results[j] = temp;
        }
    }
}
```

Η μέθοδος Anazitisi_showResults είναι η τελευταία που εκτελείται και είναι υπεύθυνη για να εμφανίσει τα αποτελέσματα στον χρήστη. Συγκεκριμένα ελέγχει το option για το πόσα αποτελέσματα θα εμφανίσει και έπειτα προσθέτει την πληροφορία για το ποιες μηχανές επέστρεψαν το αποτέλεσμα. Τέλος προσθέτει το αποτέλεσμα στη λίστα αποτελεσμάτων που βρίσκεται στην ιστοσελίδα.

```
var prefService = Components.classes["@mozilla.org/preferences-
service;1"].getService(Components.interfaces.nsIPrefService);
    var prefBranch = prefService.getBranch("extensions.anazitisiprefs.");
    var resultsNum = prefBranch.getIntPref("resultsNumber");
    if(resultsNum > sorted_results.length)
        resultsNum = sorted_results.length;

    for(i=0;i<resultsNum;i++) {
        var results_list = window.content.document.getElementById("mylist");
        var metrics_string = " - Result returned by: ";

        for(j=0;j<sorted_results[i][2].length;j++) {
            metrics_string += sorted_results[i][2][j].name;
            metrics_string += ": ";
            metrics_string += sorted_results[i][2][j].rank;
            metrics_string += " ";
        }
```

```
    var metrics_textnode = document.createTextNode(metrics_string);
    var metrics_elem = document.createElement("span");
    metrics_elem.className = "metrics_span";
    metrics_elem.appendChild(metrics_textnode);

    var result_header = sorted_results[i][0].getElementsByTagName("h3")[0];
    result_header.appendChild(metrics_elem);
    sorted_results[i][0].style.marginLeft="0px";
    sorted_results[i][0].className = "new";
    results_list.appendChild(sorted_results[i][0]);
  }
```

7.2.3 Αποθήκευση

Η διαδικασία της αποθήκευσης είναι σχετικά απλή. Έχουμε μια μέθοδο που εκτελείται κάθε φορά που κάνουμε click στο HTML έγγραφο. Η μέθοδος Anazitisi_trackClicks(event) ελέγχει αν το στοιχείο στο οποίο κάναμε click είναι τύπου a, δηλαδή link. Στη συνέχεια ελέγχει αν το parent element είναι τύπου h3. Έτσι ξέρουμε ότι ο χρήστης επισκέφτηκε κάποιο από τα αποτελέσματα. Σε αυτή την περίπτωση καλείται η μέθοδος Anazitisi_setNodeVisited(clicked_link) η οποία μαρκάρει το link ως visited. Ο κώδικας της Anazitisi_trackClicks φαίνεται παρακάτω.

```
var clicked_link = event.target;
```

```
var y = clicked_link.parentNode;
if(clicked_link.nodeName=="A") {
    if(y.nodeName=="H3") {
        Anazitisi_setNodeVisited(clicked_link);
    }
}
```

Όπως βλέπουμε στον κώδικα παρακάτω η Anazitisi_trackClicks μαρκάρει το αποτέλεσμα αλλάζοντας το className attribute του li στοιχείου που περιέχει το αποτέλεσμα.

```
var parent_node = clicked_link.parentNode;

if(parent_node.nodeName=="LI") {
    parent_node.className = "visited";
}
else {
    Anazitisi_setNodeVisited(parent_node);
}
```

Πρέπει να σημειωθεί ότι η δυνατότητα της εφαρμογής να θυμάται τα clicks του χρήστη υπάρχει μόνο όταν χρησιμοποιούμε tabbed browsing. Αυτό γίνεται γιατί η σελίδα αποτελεσμάτων δημιουργείται δυναμικά και σε περίπτωση που αλλάξουμε σελίδα δεν μπορούμε να επιστρέψουμε πίσω.

Για να αποθηκεύσουμε την κατάσταση της σελίδας χρησιμοποιούμε τη μέθοδο Anazitisi_saveSearch(). Η μέθοδος αυτή δημιουργεί ένα file output stream, χρησιμοποιώντας XPCOM αντικείμενα, για να δημιουργήσει ένα xml αρχείο στον υπολογιστή του χρήστη με όλα τα links αποτελεσμάτων και την κατάστασή τους.

```
file.append(search_filename);

var foStream = Components.classes["@mozilla.org/network/file-output-stream;1"].createInstance(Components.interfaces.nsIFileOutputStream);

foStream.init(file, 0x02 | 0x08 | 0x20,0666,0);

var converter = Components.classes["@mozilla.org/intl/converter-output-stream;1"].createInstance(Components.interfaces.nsIConverterOutputStream);

converter.init(foStream,"UTF-8",0,0);
```

Τη δομή του xml αρχείου τη δημιουργούμε γράφοντας τα κατάλληλα declarations.

```
converter.writeString("<?xml version=\"1.0\" encoding=\"UTF-8\"?>\n");

converter.writeString("<saved_url>\n");

converter.writeString("<search_name>" + searchTerms +"</search_name>\n");
```

7.2.4 Φόρτωση

Η τελευταία λειτουργία, αυτή της φόρτωσης είναι στην ουσία μια ειδική μορφή αναζήτησης. Χρησιμοποιούμε τις ίδιες μεθόδους με δύο σημαντικές διαφορές. Αρχικά καλείται η μέθοδος Anazitisi_loadSearch() η οποία μας εμφανίζει ένα παράθυρο διαλόγου από το οποίο επιλέγουμε ποια αποθηκευμένη αναζήτηση θα φορτώσουμε.

```
var nsIFilePicker = Components.interfaces.nsIFilePicker;
var fp = Components.classes["@mozilla.org/filepicker;1"].createInstance(nsIFilePicker);
fp.init(window,"Select a File",nsIFilePicker.modeOpen);
```

Αφού επιλέξουμε, η μέθοδος διαβάζει τον όρο αναζήτησης από το αρχείο και ξεκινάει μια αναζήτηση για αυτόν τον όρο καλώντας τη μέθοδο Anazitisi_LoadFrames(). Από εκεί και πέρα η διαδικασία αναζήτησης συνεχίζει κανονικά μέχρι να φτάσουμε στη μέθοδο Anazitisi_showResults(sorted_results). Η μέθοδος αυτή έχει μια επιπλέον προσθήκη για τη διαδικασία της φόρτωσης. Αφού γεμίσει τη λίστα αποτελεσμάτων, τα συγκρίνει με τα URL στο αποθηκευμένο αρχείο και αλλάζει ανάλογα το className attribute.

```
Anazitisi_updateNodes(sorted_results[i][0],old_links,"old");
Anazitisi_updateNodes(sorted_results[i][0],visited_links,"visited");

function Anazitisi_updateNodes(resultNode,xmlLinksArray,name)
{
   for(s=0;s<xmlLinksArray.length;s++) {
      a = Anazitisi_getresultURL(resultNode);
      b = xmlLinksArray[s].childNodes[0].nodeValue;
      b = decodeURIComponent(b);
      if(a==b)
         resultNode.className = name;
```

```
    }
}
```

Από τη στιγμή που τα αποτελέσματα είναι μαρκαρισμένα μπορούμε να αλλάξουμε εύκολα το visibility των στοιχείων χρησιμοποιώντας CSS τα οποία ενεργοποιούνται από την ανάλογη μέθοδο.

```
function Anazitisi_hideVisited()
{
    var check_box = document.getElementById("anazitisi_check1");
    var link_elem = window.content.document.getElementById("link_visited");
    if(!(check_box.checked)) {
        link_elem.href = "chrome://ptyxiaki/content/CSS/visited.css";
    }
    else if(check_box.checked) {
        link_elem.href = "";
    }
}
```

7.2.5 Επιλογές αναζήτησης

Η αποθήκευση των επιλογών αναζήτησης του χρήστη γίνεται πολύ εύκολα χρησιμοποιώντας το preference system του Firefox που παρουσιάσαμε σε προηγούμενο κεφάλαιο. Χρησιμοποιούμε ένα prefwindow αντικείμενο στο οποίο βρίσκεται όλο το interface που μας επιτρέπει να αλλάζουμε τα options.

```
<?xml version="1.0"?>

<?xml-stylesheet href="chrome://global/skin/" type="text/css"?>


<prefwindow title="Anazitisi Options"
xmlns="http://www.mozilla.org/keymaster/gatekeeper/there.is.only.xul">


  <prefpane id="Anazitisi_options" label="Backups">

    <vbox>

      <hbox>

        <vbox>

          <label value="Number of results" control="Anazitisi_resultNumbox"/>

          <label value="Results from search engines" control="Anazitisi_sEngineResults"/>

        </vbox>

        <vbox>

          <textbox id="Anazitisi_resultNumbox" preference="Anazitisi_resultNum"/>

          <textbox id="Anazitisi_sEngineResults"
preference="Anazitisi_sEngineResultNum"/>

        </vbox>

      </hbox>

      <groupbox flex="1">

      <caption label="Engine Selection"/>

      <listbox rows="5">
```

```
        <listitem label="Google"  type="checkbox" preference="Anazitisi_Google"/>
        <listitem label="Yahoo"  type="checkbox" preference="Anazitisi_Yahoo"/>
        <listitem label="Bing"  type="checkbox" preference="Anazitisi_Bing"/>
      </listbox>
      </groupbox>
    </vbox>
  </prefpane>

</prefwindow>
```

Οι default τιμές των preferences που χρησιμοποιούμε είναι οι ακόλουθες.

```
pref("extensions.anazitisiprefs.resultsNumber",10);
pref("extensions.anazitisiprefs.searchEngineResultsNumber",30);
pref("extensions.anazitisiprefs.searchEngines.google",true);
pref("extensions.anazitisiprefs.searchEngines.yahoo",true);
pref("extensions.anazitisiprefs.searchEngines.bing",true);
```

Συμπεράσματα

Οι βασικότεροι στόχοι της εφαρμογής που ζητήθηκε ήταν δυνατόν να επιτευχθούν. Η διαδικασία της αναζήτησης λειτουργεί για διάφορες μηχανές αναζήτησης ασχέτως του αριθμού τους. Επιπλέον υλοποιήθηκαν οι διαδικασίες αποθήκευσης και φόρτωσης αναζήτησης. Παρόλα αυτά υπάρχει ένα σημαντικό πρόβλημα.

Το πρόβλημα αυτό είναι ο τρόπος που το toolbar έχει πρόσβαση στις σελίδες αποτελεσμάτων. Η τεχνική των hidden frames μπορεί να εφαρμοστεί για έναν περιορισμένο αριθμό σελίδων. Ο λόγος είναι ότι η JavaScript είναι δύσκολο να ελέγξει πότε ολοκληρώνουν όλα τα frames τη φόρτωση της σελίδας. Το αποτέλεσμα είναι η μέθοδος χρονικής καθυστέρησης που εφαρμόστηκε να μην είναι αξιόπιστη

Βιβλιογραφία - Πηγές

1) *DOM Scripting: Web Design with JavaScript and the Document Object Model,* Jeremy Keith, friends of ED, 2005.

2) *JavaScript: The Definitive Guide*, Fifth Edition, David Flanagan, O'Reilly Media, 2006

3) *XML: The Complete Reference*, Heather Williamson, McGraw-Hill Companies, 2001

4) *Handcrafted CSS: More Bulletproof Web Design*, Dan Cederholm, New Riders Press, 2009

5) *Understanding Search Engines: Mathematical Modeling and Text Retrieval (Software, Environments, Tools)*, Second Edition, Michael W. Berry, SIAM, Society for Industrial and Applied Mathematics, 2005

6) https://developer.mozilla.org/en/Building_an_Extension

7) http://en.wikipedia.org/wiki/Document_Object_Model

8) http://en.wikipedia.org/wiki/XML

9) http://www.w3schools.com/css/css_reference.asp

10) http://www.w3schools.com/jsref/default.asp