ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάλυση της απόδοσης CSS τεχνικών

Μαριλίνα Σκαρπέτα
Χριστίνα Μυλωνά

Επιβλέπων: Σωτήρης Χριστοδούλου

ΑΝΤΙΡΡΙΟ
21-04-2016

Περίληψη

Στην παρούσα πτυχιακή εργασία θα κατανοήσουμε την έννοια της «ποιότητας λογισμικού» και τους παράγοντες που την επηρεάζουν.

Κατανόηση της έννοιας «ποιότητα λογισμικού» και τους παράγοντες που την επηρεάζουν. Γνωριμία με τις μετρικές λογισμικού. Διερεύνηση τρόπων αντιστοίχισής τους σε ανάλογες μετρικές, για την εκτίμηση της ποιότητας του κώδικα CSS και CSS3. Υλοποίηση εργαλείου μέτρησής τους.

Αντικείμενο:

- Εξοικείωση και μελέτη σε βάθος της CSS3.
- Μελέτη και καταγραφή των καλών πρακτικών, των πιο διαδεδομένων τεχνικών συγγραφής CSS κώδικα και των CSS προ-επεξεργαστών. Μέτρηση της απόδοσης τους.
- Μελέτη και καταγραφή των πιο συνηθισμένων «σφαλμάτων» σε κώδικα CSS/CSS3 (π.χ. με χρήση CSS Validators, βλ. http://www.w3.org/QA/Tools/).
- Κατηγοριοποίηση τους και συσχέτισή τους με μεγέθη του κώδικα που μπορούν να μετρηθούν.
- Εξοικείωση με εργαλεία στατικής ανάλυσης λογισμικού και μετρήσεων μετρικών.
- Υλοποίηση εργαλείου για την μέτρηση μετρικών σε CSS/CSS3 κώδικα.

Περιεχόμενα

Κεφάλαιο 1

1. Εισαγωγή

Ο ***παγκόσμιος ιστός (world wide web)*** πρωτοεμφανίστηκε το 1989 και οφείλει την δημοτικότητα του στην πολύ εύκολη χρήση από τους αρχάριους, καθώς και στον τεράστιο όγκο πληροφοριών που παρέχει για ένα ευρύ, σχεδόν καθολικό, φάσμα θεματολογίας.

Η συνεχόμενη και ραγδαία ανάπτυξης της ψηφιακής τεχνολογίας δημιούργησε μια αυξανόμενη ανάγκη για περισσότερη, ποιοτικότερη και πιο εύκολα προσβάσιμη γνώση. Το ***διαδίκτυο (internet)*** ήρθε για να καλύψει αυτή την ανάγκη, με ένα δίκτυο διασυνδεδεμένων υπολογιστών, που διευκολύνει την επικοινωνία μεταξύ των χρηστών και αποτελεί πηγή πληθώρας πληροφοριών.

Ο ιστός είναι ουσιαστικά μία πολύ μεγάλη παγκόσμια συλλογή εγγράφων, ή ιστοσελίδων. Κάθε σελίδα (σελίδα είναι η συντομογραφία της ιστοσελίδας) μπορεί να περιέχει συνδέσμους προς άλλες σελίδες που βρίσκονται σε οποιοδήποτε σημείο στον κόσμο, μέσω του ***υπερκειμένου (hyper-text)***. Οι σελίδες εμφανίζονται μέσω ενός προγράμματος που ονομάζεται ***φυλλομετρητής (browser)***. Ο φυλλομετρητής προσκομίζει τη ζητούμενη σελίδα, ερμηνεύει το κείμενο και τις εντολές μορφοποίησης που περιέχει και εμφανίζει την σελίδα, κατάλληλα μορφοποιημένη, στην οθόνη. Η επικοινωνία των φυλλομετρητών με τις ιστοσελίδες, καθώς και η περιγραφή αυτών, επιτυγχάνεται με μία τυποποιημένη γλώσσα, την ***HTML (Hyper Text Markup Language)***.

Όταν μια τοποθεσία ιστού είναι περίπλοκη και αποτελείται από πολλές σελίδες που έχουν παραχθεί από πολλούς συγγραφείς, είναι συχνά επιθυμητό να υπάρχει κάποιος τρόπος έτσι ώστε να αποφεύγονται οι διαφορές στην εμφάνιση των διαφόρων σελίδων. Αυτό το πρόβλημα μπορεί να λυθεί με τα ***φύλλα στυλ (style sheets)***.

Ένα βασικό εργαλείο για τη σχεδίαση και την κατασκευή των ιστοσελίδων το οποίο θα μελετήσουμε σε βάθος, είναι η τεχνολογία ***Διαδοχικών Φύλλων Στυλ (Cascading Style Sheets - CSS)***. Η CSS χρησιμοποιείται από τους σχεδιαστές ιστοσελίδων για την ανάπτυξη εύχρηστων και επεκτάσιμων εφαρμογών. Ο συνδυασμός της CSS με την HTML δίνει τη δυνατότητα ανάπτυξης εύχρηστων και καλαίσθητων ιστοσελίδων δυναμικού περιεχομένου, ακόμα και από χρήστες χωρίς ιδιαίτερη εμπειρία στο διαδικτυακό προγραμματισμό.

1.1.Η Γλώσσα HTML

Η HTML είναι η κύρια γλώσσα σήμανσης για τις ιστοσελίδες και τα στοιχεία της είναι τα βασικά δομικά στοιχεία των ιστοσελίδων. Προέρχεται από τα αρχικά των λέξεων *Hyper Text Markup Language* και είναι ένα προϊόν της SGML (Standard Generalized Markup Language). Η HTML είναι μία γλώσσα τύπου, δηλαδή χρησιμοποιεί ετικέτες (tags) για να περιγράψει την δομή ενός εγγράφου.

Στο CERN (Ευρωπαϊκό Κέντρο Πυρηνικών Ερευνών της Γενεύης) στη δεκαετία του 1990, η ανάγκη επικοινωνίας μεταξύ των επιστημόνων, οδήγησε το φυσικό Tim-Berners Lee στη δημιουργία του πρωτοκόλλου ***HTTP (Hyper Text Transfer Protocol)***, με το οποίο επιταχύνθηκε η μεταφορά αρχείων μέσω του διαδικτύου, γεγονός που αποτέλεσε την απαρχή του Παγκόσμιου Ιστού (www).

Η πραγματική όμως διάδοση του www και της HTML οφείλεται στην κατασκευή του *Mosaic*, τον πρώτο browser με γραφικό περιβάλλον. Ο Mosaic αναπτύχθηκε στα εργαστήρια του Πανεπιστημίου του Ιλινόις και παρουσιάστηκε το Φεβρουάριο του 1993. Ένα χρόνο αργότερα, ο δημιουργός του *Marc Andreeseen* μαζί με τον *Jim Clark*, ίδρυσαν την εταιρία *Netscape Communications Corp*, με σκοπό την ανάπτυξη εφαρμογών για το www. Την ίδια χρονιά, το CERN και το MIT (Massachusetts Institute of Technology) συμφώνησαν στη δημιουργία του οργανισμού ***World Wide Web Consortium (W3C)***. Ο σκοπός του W3C είναι η συνεχής ανάπτυξη του διαδικτύου, η ανάπτυξη προτύπων και η διαλειτουργικότητα μεταξύ των διαφόρων τοποθεσιών (sites) του διαδικτύου. Τον Ιούλιο του 1994 εκδόθηκε νέο πρότυπο και το 1995 κυκλοφόρησε ο browser Netscape. Την ίδια χρονιά η Microsoft άρχισε την ανάπτυξη του δικού της browser με

γραφικό περιβάλλον, του Internet Explorer, εκμεταλλευόμενη τις νέες δυνατότητες που είχαν προστεθεί στην 3η έκδοση της HTML. Η 4η έκδοση δημιουργήθηκε στα τέλη του 1999. Οι browsers σήμερα διαβάζουν πλέον αρχεία HTML που είναι γραμμένα βάσει της τελευταίας έκδοσης του προτύπου της HTML 5, που κυκλοφόρησε το 2011, ενώ αυτή την περίοδο η W3C αναπτύσσει μια σουίτα δοκιμών προς επίτευξη διαλειτουργικότητας για την πλήρη έκδοση.

1.1.1.Δομή γλώσσας HTML

Η HTML είναι μία γλώσσα που αποτελείται από ένα σύνολο ετικετών (tags), με την ακόλουθη δομή:

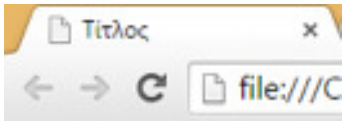
<tag>*περιεχόμενο****</tag>***

και χωρίζεται σε δύο κύρια τμήματα:

- ***Head***: οι πληροφορίες που αφορούν το αρχείο HTML αυτό καθ' εαυτό (δεν εμφανίζεται στο χρήστη)
- ***Body***: το περιεχόμενο που εμφανίζεται στη σελίδα.

```
<html>

        <head>
            <title>Τίτλος</title>
        </head>

        <body>...</body>

</html>
```

Με την ετικέτα <html> αρχίζουμε πάντα τον κώδικα μας και με την ετικέτα </html> τον τερματίζουμε. Έτσι ο browser καταλαβαίνει ότι οι γραμμές που βρίσκονται μέσα σε αυτές τις δύο ετικέτες είναι κώδικας γραμμένος σε γλώσσα HTML.


Τίτλος
file:///C

Η πρώτη ενότητα, δηλαδή η ενότητα head μίας HTML σελίδας ορίζεται από τις ετικέτες <head> και </head>. Οι ετικέτες εντός του head αποτελούν τον πρόλογο για τη σελίδα. Βασικότερη εξ ’αυτών είναι η ετικέτα <title>, όπου δηλώνεται ο τίτλος στην μπάρα τίτλου του φυλλομετρητή.

Η δεύτερη ενότητα, δηλαδή η ενότητα body μίας HTML σελίδας, ορίζεται από τις ετικέτες <body> και </body>. Οι ετικέτες εντός του body ορίζουν το κύριο περιεχόμενο της σελίδας, δηλαδή το κείμενο και τις ετικέτες μορφοποίησής του. Μπορούμε επίσης να προσθέσουμε περιεχόμενο εκτός κειμένου, δηλαδή πολυμέσα όπως εικόνες, ήχο και βίντεο.

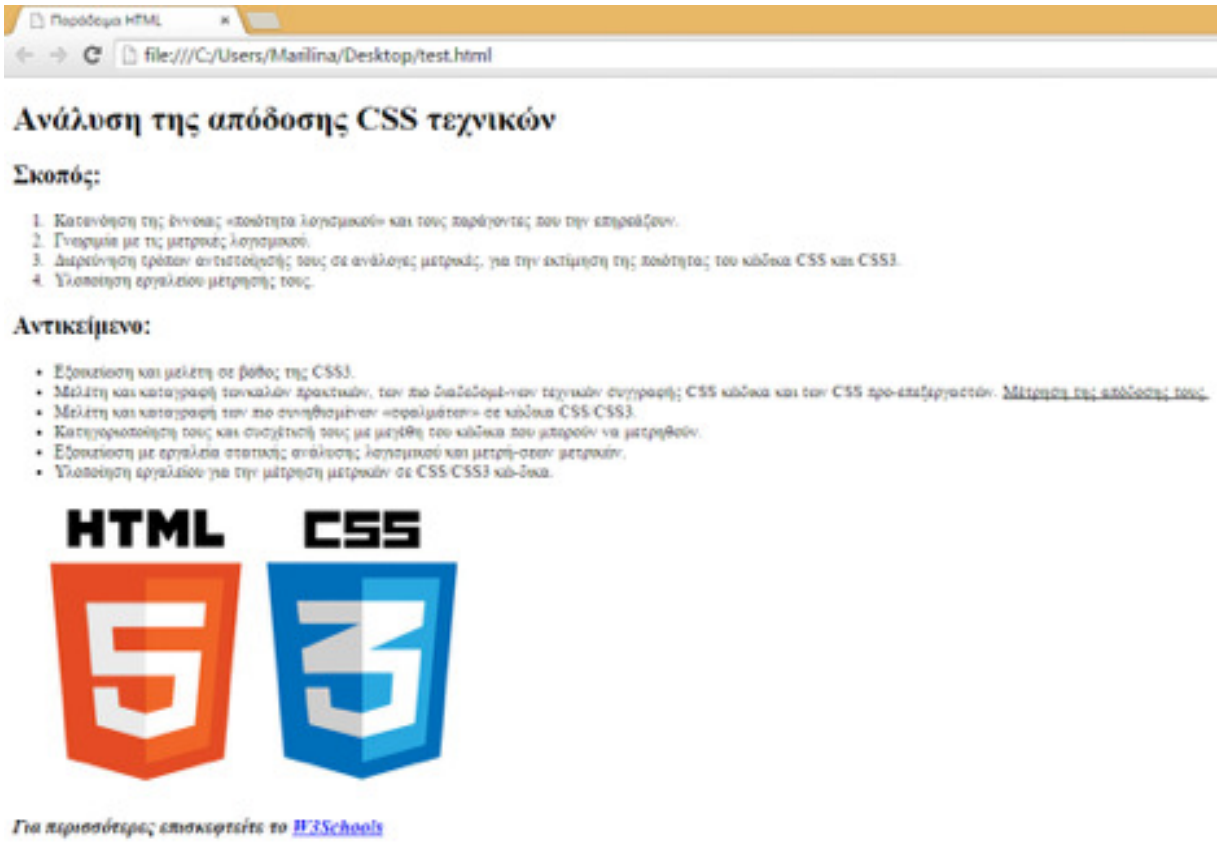
Ετικέτες	Λειτουργία
<h> … </h>	Επικεφαλίδες κειμένου
</br> *	Αλλαγή γραμμής στο κείμενο
** … **	Αλλαγή παραγράφου
<hr>	Οριζόντια γραμμή που χωρίζει τη σελίδα στα δύο
<u>…</u>	Υπογραμμισμένο κείμενο
…	Κείμενο με πλάγια γραφή
…	Κείμενο με έντονη γραφή
**<p align=“ left”> … **	Στοίχιση κειμένου αριστερά
**<p align=“ right”> … **	Στοίχιση κειμένου δεξιά
** … **	Στοίχιση κειμένου στο κέντρο
…	Αλλαγή γραμματοσειράς
...	Αλλαγή χρώματος γραμματοσειράς
...	Αλλαγή μεγέθους γραμματοσειράς
** … **	Μη αριθμημένη λίστα
** … **	Αριθμημένη λίστα
** … **	Στοιχείο λίστας
<img src =”photos/html.gif”>	Εμφάνιση εικόνας
** w3s **	Εμφάνιση link
<table> …</table>	Δημιουργία πίνακα

<tr>…</tr>	Σειρά πίνακα
<td>…</td>	Κελί πίνακα

1.1.2.Παράδειγμα χρήση HTML κώδικα

```
<!DOCTYPE html>
<html>
	<head>
		<title>Παράδειγμα HTML</title>
	</head>
	<body>
		<h1>Ανάλυση της απόδοσης CSS τεχνικών</h1>
		<h2>Σκοπός:</h2>
		<ol>
			<li>Κατανόηση της έννοιας «ποιότητα λογισμικού» και τους
			παράγοντες που την επηρεάζουν.</li>
			<li>Γνωριμία με τις μετρικές λογισμικού.</li>
			<li>Διερεύνηση τρόπων αντιστοίχισής τους σε ανάλογες μετρικές, για
			την εκτίμηση της ποιότητας του κώδικα CSS και CSS3.</li>
			<li>Υλοποίηση εργαλείου μέτρησής τους.</li>
		</ol>

		<h2>Αντικείμενο:</h2>
		<ul>
			<li>Εξοικείωση και μελέτη σε βάθος της CSS3.
			<li>Μελέτη και καταγραφή των καλών πρακτικών, των πιο
			διαδεδομένων τεχνικών συγγραφής CSS κώδικα και των CSS προ-
			επεξεργαστών. <u>Μέτρηση της απόδοσης τους.</u></li>
			<li>Μελέτη και καταγραφή των πιο συνηθισμένων «σφαλμάτων» σε
			κώδικα CSS/CSS3.</li>
			<li>Κατηγοριοποίηση τους και συσχέτισή τους με μεγέθη του κώδικα
			που μπορούν να μετρηθούν.</li>
			<li>Εξοικείωση με εργαλεία στατικής ανάλυσης λογισμικού και
			μετρήσεων μετρικών.</li>
			<li>Υλοποίηση εργαλείου για την μέτρηση μετρικών σε CSS/CSS3
			κώδικα.</li>
		</ul>
		<img src = css3.jpg></p>
		<p><h3><i>Για περισσότερες επισκεφτείτε το <a href="http://
www.w3schools .com/">W3Schools</a><i></h3></p>
	</body>
</html>
```

1.1.3.



1.1.3.Πλεονεκτήματα γλώσσας HTML

Τα πολλά πλεονεκτήματα της γλώσσας HTML είναι αυτά που την έχουν καταστίσει ευρέως διαδεδομένη από εκατομμύρια χρήστες:

1. **Εύκολη εκμάθηση/χρήση**. Ο προγραμματισμός σε HTML δεν προϋποθέτει ιδιαίτερες γνώσεις προγραμματισμού, γεγονός που την καθιστά χρηστική σε μη προγραμματιστές.
2. **Καθολική υποστήριξη**. Οι περισσότεροι browsers υποστηρίζουν την HTML.
3. **Ταχεία φόρτωση σελίδων**. Δεν χρησιμοποιείται άσκοπος κώδικας, συνεπώς δεν καθυστερεί την εμφάνιση της σελίδας.

4. **Αξιοπιστία**. Μεγάλη συνάφεια του περιεχομένου που προγραμματίζεται με το τελικό αποτέλεσμα για το χρήστη.
5. **Έξυπνη σχεδίαση**. Οι σελίδες προσαρμόζονται αυτόματα στις ιδιαιτερότητες της οθόνης του χρήστη.
6. **Έλεγχος του αποτελέσματος**. Ο προγραμματισμός σε HTML δεν ακολουθείται από μετάφραση για την εμφάνιση του περιεχομένου στο χρήστη, με αποτέλεσμα να μην υπάρχει κίνδυνος σημαντικά στοιχεία να απαλειφθούν από το τελικό αποτέλεσμα.
7. **Υψηλή ευελιξία**. Η γλώσσα της HTML έχει πληθώρα δυνατοτήτων.
8. **Εύκολη επίλυση προβλημάτων (debugging)**. Εύκολος εντοπισμός και διόρθωση προβλημάτων, καθώς γνωρίζουμε τα χαρακτηριστικά της σελίδας που δημιουργήσαμε.
9. **Επέκταση σε άλλα γνωστικά αντικείμενα**. Γνωρίζοντας κώδικα HTML μπορούμε ευκολότερα να κατανοήσουμε και να αξιοποιήσουμε άλλες τεχνολογίες.

1.1.4.Μειονεκτήματα γλώσσας HTML

Τα πολλά πλεονεκτήματα της γλώσσας HTML μπορούν να αποτελέσουν και τροχοπέδη για το χρήστη:

1. **Χρονοβόρα διαδικασία**. Σε άλλους τρόπους δημιουργίας ιστοσελίδων μπορούμε να βρούμε έτοιμα κομμάτια για να εισάγουμε, ενώ στην HTML ως επί το πλείστον πρέπει να δημιουργούμε εξ αρχής όλα τα κομμάτια του κώδικα.
2. **Δύσκολη για πολύπλοκες κατασκευές**.
3. **Εξειδικευμένη χρήση**. Είναι άσκοπη η μελέτη HTML για κάποιον που δεν θέλει επαγγελματική χρήση για web design αλλά περιστασιακή.
4. **Χρόνος κατασκευής**. Η κατασκευή ιστοσελίδας απαιτεί μεγάλο χρονικό διάστημα.

5. **Στατικό περιεχόμενο**. Στα αυτόματα προγράμματα γίνεται αυτόματη ενημέρωση των ιστοσελίδων του διακομιστή με όλες τις αλλαγές που έχουν γίνει μετά από κάθε ανανέωση, ενώ στην HTML οι αλλαγές πρέπει να γίνονται από τον προγραμματιστή.

1.2.Η Γλώσσα XML

Η XML (eXtensible Markup Language) είναι άλλη μία γλώσσα σήμανσης. Χρησιμοποιείται για την περιγραφή και ανταλλαγή δεδομένων, σε αντίθεση όμως με την HTML, οι ετικέτες καθορίζονται αποκλειστικά από το χρήστη. Η XML έχει αναπτυχθεί και συντηρείται από το W3C (World Wide Web Consortium), και αποτελεί υποσύνολο ενός προτύπου κατά ISO, της γλώσσας SGML (Standard Generalized Markup Language).

Η γλώσσα αυτή δίνει έμφαση στη μορφοποίηση (παρουσίαση) των δεδομένων σε μια ιστοσελίδα (μέγεθος και το χρώμα των γραμμάτων, γραμματοσειρά, θέση παραγράφων, κ.α.). Η XML δεν αντικαθιστά, αλλά συμπληρώνει τη γλώσσα HTML, δίνοντας έμφαση στο περιεχόμενο των δεδομένων μιας ιστοσελίδας, αγνοώντας τη μορφοποίησή τους.

1.2.1.Δομή γλώσσας HTML

Στη γλώσσα XML, παρόλο που οι ετικέτες καθορίζονται από το χρήστη, έχουμε αυστηρά προκαθορισμένη δομή. Μία απλή απεικόνιση της δομής είναι η δεντρική, δηλαδή το πρώτο στοιχείο αποτελεί τη ρίζα (root).

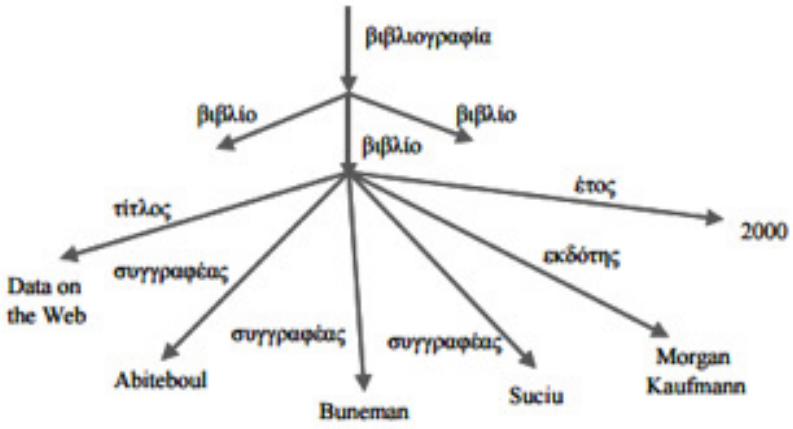
Για μεγαλύτερη ευκολία μπορεί να αναπαρασταθεί με δεντρική δομή στην οποία έχουμε τις ακόλουθες κατηγορίες:

- ***Ρίζα (root)***: απεικονίζει το πρώτο στοιχείο του εγγράφου.
    - ***Γονικές (parent)***: περιέχουν δεδομένα.
    - ***Παιδιά (child elements)***: τα στοιχεία τους περιέχονται μέσα σε άλλες κατηγορίες.

Τα στοιχεία που έχουν κοινό γονέα ονομάζονται ***αδέρφια (siblings)***. Ένα απλό παράδειγμα XML κώδικα:

```
<βιβλιογραφία>
   <βιβλίο>
      <τίτλος>Βάσεις Γεωγραφικών Δεδομένων</τίτλος>
      <συγγραφέας>Στεφανάκης</συγγραφέας>
      <εκδότης>Παπασωτηρίου</εκδότης>
      <έτος>2003</έτος>
   </βιβλίο>
   <βιβλίο>
      <τίτλος>Data on the Web</τίτλος>
      <συγγραφέας>Abiteboul</συγγραφέας>
      <συγγραφέας>Buneman</συγγραφέας>
      <συγγραφέας>Suciu</συγγραφέας>
      <εκδότης>Morgan Kaufmann</εκδότης>
      <έτος>2000</έτος>
   </βιβλίο>
</βιβλιογραφία>
```

Και η αντίστοιχη δεντρική απεικόνιση:



Η γλώσσα XML επιτρέπει την ανάθεση γνωρισμάτων (attributes) στα στοιχεία. Ένα παράδειγμα είναι το ακόλουθο:

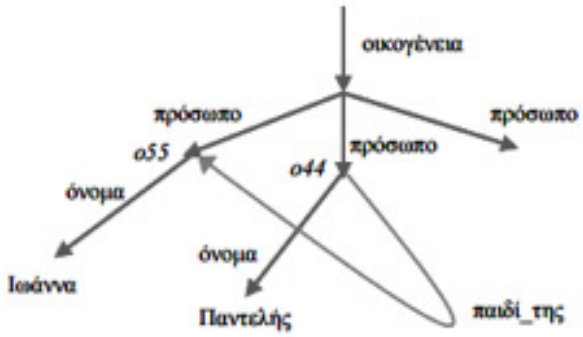
```
<βιβλίο κόστος="40" νόμισμα="Ευρώ">
```

Σύμφωνα με αυτό, στο στοιχείο βιβλίο ανατίθενται δύο γνωρίσματα. Το γνώρισμα *κόστος* με τιμή «40» και το γνώρισμα *νόμισμα* με τιμή «Ευρώ». Οι τιμές των γνωρισμάτων (σε αντίθεση με τα στοιχεία) αναγράφονται πάντα εντός διπλών εισαγωγικών. Τα γνωρίσματα αποτελούν ένα εναλλακτικό τρόπο αναπαράστασης δεδομένων σε ένα κείμενο XML.

Επιπλέον, η γλώσσα XML επιτρέπει την ανάθεση *μοναδικών ταυτοτήτων* στα στοιχεία ενός κειμένου. Η ανάθεση αυτή γίνεται σε ένα συγκεκριμένο γνώρισμα του στοιχείου, το οποίο καλείται *id*. Το γνώρισμα αυτό αξιοποιείται σαν *αναφορά* του στοιχείου από άλλα στοιχεία του κειμένου. Η αναφορά επιτυγχάνεται μέσω του γνωρίσματος *idref*. Ένα παράδειγμα είναι το ακόλουθο:

```
<οικογένεια>
    <πρόσωπο id="o55">
        <όνομα>Ιωάννα</όνομα>
    </πρόσωπο>
    <πρόσωπο id="o44">
        <όνομα>Παντελής</όνομα>
        <παιδί_της idref="o55"/>
    </πρόσωπο>
</οικογένεια>
```

Το παράδειγμα αυτό παρουσιάζει δύο πρόσωπα μιας οικογένειας, την *Ιωάννα* και τον *Παντελή*, με ταυτότητες (id) *o55* και *o44* αντίστοιχα. Το γνώρισμα *idref* στο στοιχείο *παιδί_της* συσχετίζει τα δύο πρόσωπα. Η σημασιολογία, που μεταφέρει η ετικέτα του στοιχείου αυτού, δείχνει ότι *«ο Παντελής είναι παιδί της Ιωάννας»*. Η χρήση των ταυτοτήτων και αναφορών αποτελούν το μηχανισμό της γλώσσας XML για την επέκταση των δένδρων σε γράφους (δικτυακές δομές), στα αντίστοιχα διαγράμματα. Το παρακάτω σχήμα βλέπουμε το διάγραμμα XML για το προηγούμενο παράδειγμα.



1.2.2.Πλεονεκτήματα γλώσσας XML

Τα πλεονεκτήματα της γλώσσας XML είναι αυτά που την έχουν καταστίσει ευρέως διαδεδομένη για ανταλλαγή πληροφοριών:

1. **Εύκολη εκμάθηση / λιγότερη ορολογία**. Η XML δεν περιέχει ιδιαίτερη ορολογία, εντρυφώντας στο περιερχόμενο και όχι στην επεξεργασία του.
2. **Καθολική υποστήριξη**. Μπορεί να εξάγει πληροφορίες ανεξάρτητα από την πλατφόρμα που χρησιμοποιείται και να τις μεταφέρει μεταξύ διαφορετικών συστημάτων.
3. **Επεκτασιμότητα / ευελιξία**. Ο χρήστης έχει την ευελιξία να ορίζει τις ετικέτες, χωρίς προκαθορισμένη ορολογία.

1.2.3.Μειονεκτήματα γλώσσας XML

Όπως και στην ευρύτατα χρησιμοποιούμενη γλώσσα HTML, έτσι και στην XML υπάρχουν χαρακτηριστικά που σε ορισμένες περιπτώσεις αποτρέπουν τη χρήση της:

1. **Η κωδικοποίηση σε XML οδηγεί σε μεγάλα αρχεία**, σε σχέση με το μέγεθός τους σε δυαδική μορφή, αφού όλα τα δεδομένα βρίσκονται μέσα σε ετικέτες.
2. **Η κωδικοποίηση των δεδομένων σε απλό κείμενο είναι χρονοβόρα**, πολλές φορές απαγορευτική για επιστημονικές εφαρμογές ή για εφαρμογές όπου η απόδοση έχει μεγάλη σημασία.

Κεφάλαιο 2

2. Η γλώσσα CSS

Η CSS (Cascading Style Sheets - Διαδοχικά Φύλλα Στυλ) είναι μία γλώσσα σήμανσης (markup), η οποία περιγράφει τον τρόπο απεικόνισης άλλων εγγράφων που έχουν γραφεί σε άλλες markup γλώσσες, όπως η HTML. Σήμερα η CSS χρησιμοποιείται σε πολύ μεγάλο βαθμό στο σχεδιασμό διαδικτυακών εφαρμογών. Συγκεκριμένα, ο οργανισμός W3C προωθεί τη χρήση style sheets στο διαδίκτυο από το 1994 για στοιχεία όπως είναι η σχεδίαση των διαστάσεων της ιστοσελίδας, τα χρώματα και οι γραμματοσειρές θα χρησιμοποιηθούν.

Με την επισύναψη αρχείων CSS σε δομημένα έγγραφα (HTML, XML κ.α.) ο χρήστης έχει τη δυνατότητα να τροποποιήσει την παρουσίαση του εγγράφου χωρίς να κάνει αλλαγές στη δομή του, όπως να προσθέσει κάποια ετικέτα. Βασικό χαρακτηριστικό της γλώσσας αυτής είναι ο διαχωρισμός του περιεχομένου από την εμφάνιση του εγγράφου, δίνοντας έτσι ένα πλεονέκτημα σε θέματα που αφορούν την πρόσβαση στο περιεχόμενο, την καλύτερη δόμηση του κώδικα και θέματα βελτιστοποίησης.

Χρησιμοποιώντας τη γλώσσα CSS στον προγραμματισμό ιστοσελίδων μπορούμε να εφαρμόσουμε την ίδια μορφοποίηση σε πολλαπλές σελίδες και να μειώσουμε την πολυπλοκότητα του HTML κώδικα, ορίζοντας τη μορφοποίηση των πινάκων από το CSS έγγραφο. Επίσης, υπάρχει η δυνατότητα εμφάνισης του ίδιου εγγράφου με πολλές μορφοποιήσεις. Με αυτό τον τρόπο έχουμε τη δυνατότητα να επιλέξουμε τον τρόπο με τον οποίο το ίδιο έγγραφο θα να απεικονιστεί είτε στην οθόνη, είτε στον εκτυπωτή είτε ακόμα (θα αναπαρασταθεί) και με ήχο.

2.1.Εισαγωγή στη CSS

2.1.1.Ιστορία

Τα style sheets δημιουργήθηκαν από τα πρώτα βήματα της SGML το 1970. Αναπτύχθηκαν προκειμένου να παρέχουν πληροφορίες για την απεικόνιση των εγγράφων στο διαδίκτυο. Μετά τη διάδοση της HTML, προστέθηκαν νέες δυνατότητες στις γλώσσες συγγραφής style sheets, έτσι ώστε να ικανοποιηθούν οι ανάγκες των προγραμματιστών σε εφαρμογές διαδικτύου. Αυτή η επανάσταση, έδωσε στους σχεδιαστές ιστοσελίδων τον απόλυτο έλεγχο γύρω από θέματα απεικόνισης του ιστότοπου. Παράλληλα όμως η πολυπλοκότητα της HTML γινόταν όλο και μμεγαλύτερη.

Ο Robert Cailliau ήταν ο άνθρωπος ο οποίος σκέφτηκε πως αν υπήρχε διαχωρισμός μμεταξύ της εμφάνισης και της δομής ενός εγγράφου τότε η ανάπτυξη και ο σχεδιασμός ιστοσελίδων θα γίνονταν αρκετά πιο πρακτικά. Ο ιδανικός τρόπος για να γίνει αυτό ήταν να δώσει στον χρήστη διαφορετικές επιλογές εμφάνισης ενός εγγράφου και έτσι δημιουργήθηκαν τρία είδη από style sheets: ένα για εκτύπωση εγγράφων, ένα για απεικόνιση στην οθόνη και τέλος, ένα για την επεξεργασία του εγγράφου.

Για να βελτιωθεί η απεικόνιση των ιστότοπων στο διαδίκτυο, εννέα διαφορετικές γλώσσες προτάθηκαν από τον οργανισμό W3C (World Wide Web Consortium) στις λίστες ηλεκτρονικού ταχυδρομείου (mailing list) που συζητούσαν αυτό το θέμα. Από αυτές τις εννέα γλώσσες, δύο ήταν αυτές που επιλέχθηκαν από τον οργανισμό: η CHSS (Cascading HTML Style Sheets) και η SSP (Stream-based Style Sheet Proposal) και από αυτές προέρχεται η CSS.

Η CHSS, ήταν μία γλώσσα η οποία έχει μεγάλες ομοιότητες με τη σημερινή μορφή της CSS και προτάθηκε από τον Hakom Wium Lie τον Οκτώβριο του 1994. Από την άλλη πλευρά η SSP προήλθε από τον Bert Bos ο οποίος είχε κατασκευάσει έναν περιηγητή (browser) ο οποίος για την απεικόνιση των εγγράφων χρησιμοποιούσε την SSP style sheet γλώσσα. Ο Lie και ο Yves Lafon συνεννοήθηκαν με τον Dave Raggett για την επέκταση του περιηγητή Arena, ώστε να υποστηρίζει τη CSS. Ο Lie και ο Bos εργαστήκαν από κοινού προκειμένου να αναπτύξουν το πρότυπο της CSS.

Σε αντίθεση με πολλές style sheet γλώσσες όπως η DSSSl και η FOSI, η CSS επέτρεψε τη μορφοποίηση ενός εγγράφου από διαφορετικά style sheets. Ένα style sheet μπορεί να επηρεαστεί από άλλο, επιτρέποντας έτσι τη μίξη διαφόρων μορφοποιήσεων τα οποία μπορούσαν να διαχειριστούν ισότιμα από το σχεδιαστή αλλά και από το χρήστη. Η πρόταση του Lie παρουσιάστηκε στο συνέδριο με τίτλο "Mosaic and the Web" στο Σικάγο το 1994. Εκείνη τη χρονική περίοδο μόλις είχε συσταθεί ο οργανισμός W3C και έδειξε ιδιαίτερο ενδιαφέρον για την ανάπτυξη της CSS. Ο ίδιος οργανισμός διοργάνωσε ένα workshop στο οποίο υπεύθυνος ήταν ο Steven Pemberton. Ως αποτέλεσμα, ανέκυπτε όλο και μεγαλύτερο ενδιαφέρον για την ανάπτυξη της CSS. Ο Lie και ο Bos ήταν τα πρώτα μέλη της τεχνικής επιτροπής αυτού του project μαζί με άλλους σημαντικούς μηχανικούς της εποχής, όπως ο Thomas Reardon από τη Microsoft. Τον Αύγουστο του 1996 η εταιρία Netscape Communication Corporation παρουσίασε μία εναλλακτική γλώσσα της CSS, που ονομαζόταν JSSS (Javascript Style Sheets). Οι προδιαγραφές της ποτέ δεν διεκπεραιώθηκαν με αποτέλεσμα να εγκαταλειφθεί.

2.1.2.Εκδόσεις

Ο οργανισμός W3C δημοσίευσε τις προτάσεις της για την 1η έκδοση το 1996 και τις αναθεώρησε το 1999. Η ανάπτυξη της HTML, CSS και DOM έγινε από μία ομάδα εργασίας που ονομάζεται ERB (HTML Editorial Review Board), που στη συνέχεια η χωρίστηκε σε τρεις άλλες ομάδες εργασίας.

Η γλώσσα CSS εξελίχθηκε τα τελευταία χρόνια υπό την καθοδήγηση της W3C, μέχρι να φτάσει στην σημερινή της μορφή. Τα περισσότερα προγράμματα περιήγησης της σημερινής εποχής υποστηρίζουν την 2η έκδοση της CSS, η οποία περιλαμβάνει την τόσο την 1η έκδοση όσο και την γλώσσα CSS-Positioning. Μετά από πολλά χρόνια ανάπτυξης, η νέα έκδοση CSS 3 είναι γεγονός, φέρνοντας, μαζί με την HTML 5, μία επανάσταση στο web design.

2.1.2.1.Έκδοση επιπέδου CSS 1

Αυτή η πρώτη έκδοση είχε ικανότητες όπως:

- Μορφοποίηση των γραμματοσειρών, όπως έντονα, πλάγια ή υπογραμμισμένοι χαρακτήρες.
- Χρωματισμός κειμένου, φόντο και άλλων στοιχείων.
- Μορφοποίηση κειμένου, όπως ρύθμιση του κενού μεταξύ των γραμμάτων, των λέξεων και των σειρών του κειμένου.
- Στοίχιση κειμένου, εικόνων πινάκων και άλλων στοιχείων.
- Ρύθμιση αποστάσεων μεταξύ των στοιχείων και της θέσης του κάθε στοιχείου.
- Δυνατότητα δημιουργίας μοναδικής ταυτότητας για κάθε στοιχείο ή για ένα σύνολο στοιχείων.

Ο περιηγητής Netscape Navigator 4 καθώς και οι 3η και 4η έκδοση του Internet Explorer υποστηρίζουν το επίπεδο CSS 1. Ο W3C δεν υποστηρίζει πλέον το πρότυπο CSS 1.

Η ανάγκη των σχεδιαστών διαδικτυακών εφαρμογών για στοίχιση αντικειμένων με μεγάλη ακρίβεια, σε συνδυασμό με την έλλειψη υποστήριξη των περιηγητών για τη 2η έκδοση, οδήγησε τον W3C στη δημοσίευση της προσωρινή λύσης της CSS-Positioning (CSS-P). Η έκδοση αυτή είχε σαν επιπρόσθετη λειτουργία (σε σχέση με τη CSS1) τη δυνατότητα εμφάνισης ενός στοιχείου πίσω από ένα δεύτερο. Τόσο ο Netscape όσο και ο Internet Explorer υποστήριξαν την νέα αυτή έκδοση.

2.1.2.2.Έκδοση επιπέδου CSS 2

Η CSS επιπέδου 2 αναπτύχθηκε από τον οργανισμό W3C και δημοσιεύθηκε σαν πρόταση το Μάιο του 1998. Το δεύτερο επίπεδο της γλώσσας αποτελεί ένα υπερσύνολο της CSS επιπέδου 1 και περιέχει ένα σύνολο νέων χαρακτηριστικών, όπως:

- Ρύθμιση απόλυτης, σχετικής και προκαθορισμένης θέσης των στοιχείων.
- Το νέο χαρακτηριστικό z-index με το οποίο δίνεται η δυνατότητα δημιουργίας επίπεδων (layers) στην απεικόνιση.

- Υποστήριξη αρχείων πολυμέσων, υποστήριξη για aural style sheets και κάποιες νέες ιδιότητες στις γραμματοσειρές, όπως η σκιά.

Ο W3C δεν υποστηρίζει πλέον το πρότυπο CSS 2.

Η έκδοση 2.1 της CSS διορθώνει προβλήματα που εντοπίστηκαν στην έκδοση 2, διακόπτει την υποστήριξη κάποιων χαρακτηριστικών που δεν υποστηρίζονται πλέον από τους περιηγητές, ενώ προσθέτει στα χαρακτηριστικά της γλώσσας κάποια στοιχεία που υποστηρίζονταν μέχρι τώρα από επεκτάσεις (extensions) των περιηγητών. Ενώ η έκδοση αυτή ήταν η πρόταση του οργανισμού W3C για πολλούς μήνες, τον Ιούνιο του 2005 μετατράπηκε σε ένα project το οποίο θα επεξεργαζόταν περισσότερο. Ορίστηκε ως υποψήφια πρόταση στις 19 Ιουλίου του 2007 και τέλος η επιτροπή του W3C εξέδωσε το τελικό πρότυπο στις 7 Ιουνίου του 2011.

2.1.2.3.Έκδοση επιπέδου CSS 3

Η CSS επιπέδου 3 βρίσκεται υπό ανάπτυξη από τις 15 Δεκεμβρίου του 2005. Ο οργανισμός W3C διατηρεί συνεχώς μία αναφορά που περιέχει πληροφορίες για τη διαδικασία της ανάπτυξης προκειμένου να τη μελετούν οι σχεδιαστές. Η CSS επιπέδου 3 αποτελείται από διάφορα τμήματα (modules) και έχουν κατατεθεί για αυτή ένα σύνολο ξεχωριστών προτάσεων για την εξέλιξή της. Κάθε module προσθέτει νέες δυνατότητες ή επεκτείνει τα χαρακτηριστικά που ορίζονται από την CSS 2ου επιπέδου, διατηρώντας παράλληλα συμβατότητα με παλαιότερες εκδόσεις. Τον Ιούνιο του 2012 καταμετρήθηκαν πάνω από πενήντα modules CSS, ενώ η Ομάδα Εργασίας CSS δημοσίευσε τέσσερις από αυτές επίσημα και αφορούσαν:

- Τα *Media Queries*, που επιτρέπουν το περιεχόμενο της σελίδας να προσαρμοστεί σε διαφορετικές συνθήκες, όπως η ανάλυση της οθόνης σε διαφορετικούς τύπους οθονών (π.χ. smartphones, tablets).
- Τα *Namespaces*, που επιτρέπουν το ίδιο αρχείο να χρησιμοποιείται από πολλούς χρήστες χωρίς να υπάρχει σύγκρουση μεταξύ τους.

- Τα *Selectors*, που περιγράφουν στοιχεία επιλογής που χρησιμοποιεί η CSS για να επισυνάψει ιδιότητες στυλ. Τα στοιχεία αυτά μπορεί να επιλεγούν βάση του ονόματός τους, των γνωρισμάτων τους, του περιεχόμενού τους και άλλων ιδιοτήτων.
- Τα *Colors*, που προσδιορίζουν όλες τις πτυχές που σχετίζονται με το χρώμα των CSS, συμπεριλαμβανομένης της ιδιότητας της διαφάνειας.

2.1.2.4.Έκδοση επιπέδου CSS 4

Μέχρι σήμερα δεν έχει εκδοθεί επίσημη έκδοση επιπέδου CSS 4. Ωστόσο, δεδομένης της δομής της γλώσσας σε modules τα οποία αναπτύσσονται ανεξάρτητα, υφίσταται μία ανεπίσημη έκδοση 4ου επιπέδου, που βασίζεται σε χαρακτηριστικά του 3ου επιπέδου. Υπό αυτή την έννοια, υπάρχουν modules που μπορούν να αναφέρονται συλλογικά ως CSS4.

2.1.3.Πλεονεκτήματα

Το κύριο πλεονέκτημα της CSS όπως αναφέρθηκε παραπάνω είναι το ότι μία μορφοποίηση μπορεί να εφαρμοστεί σε ένα σύνολο ιστοσελίδων. Αν απαιτούνται αλλαγές στη μορφοποίηση, τότε με μία επεξεργασία του CSS εγγράφου όλες οι σελίδες του ιστότοπου θα επηρεαστούν από την ίδια αλλαγή. Η εναλλακτική αυτού θα ήταν η επεξεργασία κάθε σελίδας μεμονωμένα προκειμένου να πραγματοποιηθεί η οποιαδήποτε αλλαγή. Ως αποτέλεσμα, χρησιμοποιώντας τη CSS και τα υποστηριζόμενα από τη γλώσσα style sheets, όλη η ευθύνη για τη μορφοποίηση των ιστοσελίδων αναλαμβάνεται από μία ομάδα (εταιρεία) ή πολλές φορές και από ένα άτομο, ακόμη και για πολυεθνικές εταιρίες με δεκάδες ιστότοπους.

Άμεση συνέπεια των παραπάνω, είναι η ελαχιστοποίηση του μεγέθους του αρχείου μορφοποίησης, δεδομένου ότι ο κάθε κανόνας που χρησιμοποιείται αναφέρεται μόνο μια φορά στον κώδικα, ανεξάρτητα από το πόσες φορές θα εφαρμοστεί.

Επίσης, οι σελίδες που σχεδιάζουμε έχουμε πολύ καλύτερη απόκριση, καθώς με τη χρήση αρχείου CSS, ο browser αποθηκεύει τοπικά το αρχείο στην προσωρινή μνήμη, με αποτέλεσμα κάθε φορά που επαναφορτώνεται η σελίδα, δεν χρειάζεται να φορτωθεί ξανά το αρχείο, ακόμα και αν πρόκειται για άλλη σελίδα του ιστότοπού μας.

Τέλος, η χρήση των CSS καθιστά τις σελίδες πιο φιλικές προς τις μηχανές αναζήτησης (Search Engine Optimization - SEO). Οι μηχανές αναζήτησης διαχωρίζουν το περιεχόμενο από τη μορφοποίησή του, συνεπώς για μία αναζήτηση στη σελίδα αρκεί να αναζητήσουμε απλά το περιεχόμενο. Καλύτερα SEO δίνουν μεγαλύτερες πιθανότητες επισκεψιμότητας μιας σελίδας, άρα και επίτευξη του στόχου ύπαρξής της.

2.1.4.Μειονεκτήματα

Πέρα από τα πλεονεκτήματα της CSS, παρατηρούμε και τα εξής μειονεκτήματα τα οποία θα πρέπει να αντιμετωπίζουν με σοβαρότητα και προσοχή οι σχεδιαστές:

- **Ταχύτητα**. Το να κατεβάσει ο περιηγητής ταυτόχρονα ένα HTML και ένα CSS αρχείο είναι πάντα πιο χρονοβόρο από το να κατεβάσει ένα μόνο αρχείο HTML. Παρόλα αυτά αυτό δεν είναι τόσο σημαντικό όταν ο χρήστης θέλει να περιηγηθεί σε ένα σύνολο σελίδων οι οποίες χρησιμοποιούν την ίδια μορφοποίηση (ίδιο αρχείο CSS).
- **Διαφορετική σύνταξη από την HTML**. Η CSS αναπτύχθηκε εντελώς ανεξάρτητα από την HTML και χρησιμοποιεί εντελώς διαφορετική σύνταξη. Συνεπώς, ένας προγραμματιστής που αναπτύσσει εφαρμογές διαδικτύου θα πρέπει να γνωρίζει δύο διαφορετικούς τρόπους μορφοποίησης αντί για έναν. Επίσης, η σύνταξη της CSS δεν είναι ιδιαίτερα φιλική προς το χρήστη.
- **Απαραίτητη η πρόσβαση σε εξωτερικά αρχεία**. Εάν αποθηκευτεί ένα έγγραφο HTML στο σκληρό δίσκο δίχως το αρχείο CSS που ορίζει τη μορφοποίησή του, τότε η αυτή θα χαθεί. Με αυτό τον τρόπο, το έγγραφο δεν θα εμφανίζεται σωστά όταν ο χρήστης είναι αποσυνδεδεμένος από το διαδίκτυο. Με τον ίδιο τρόπο, όλες οι σελίδες των οποίων η μορφοποίηση εξαρτάται από ένα αρχείο CSS, όταν χάσουν τη σύνδεση με το αρχείο αυτό, δε θα εμφανίζονται σωστά. Επίσης, οποιοδήποτε άλλο υλικό (εικόνες, χρώμα φόντο, κ.α.) ορίζονται στο αρχείο της μορφοποίησης, δε θα εμφανίζονται.

- **Εύκολη επεξεργασία και επαναποθήκευση**. Μπορεί εύκολα να επεξεργαστεί και να επαναποθηκευτεί από τον οποιοδήποτε. Αν και αυτό μπορεί να θεωρηθεί και ως προσόν, λαμβάνοντας υπόψη την έλλειψη δικλείδων ασφαλείας στη CSS (μια και είναι ένα σύστημα βασισμένο σε κείμενο), δε θα ήταν υπερβολή να το θεωρούσαμε σοβαρό μειονέκτημα. Οποιοσδήποτε που έχει δικαίωμα να διαβάσει ή να εγγράψει στο αρχείο μπορεί να διακόψει κάποιες λειτουργίες ή να αλλάξει υπερσυνδέσμους στις ιστοσελίδες.
- **Πολυπλοκότητα σε συστήματα δημιουργίας ιστοσελίδων με templates**. Τα CSS αρχεία είναι ενοχλητικά σε πακέτα λογισμικού, όπως το Dreamweaver, που χρησιμοποιούν templates. Αυτό οφείλεται στο ότι ενώ κατά τη δημιουργία ιστοσελίδων οι υπερσύνδεσμοι είναι οι σωστοί, όταν οι σελίδες ανεβαίνουν στο server οι διαδρομές αυτές ενδέχεται να τροποποιηθούν, προκαλώντας προβλήματα στην εμφάνισή τους.
- **Πολυπλοκότητα σε συστήματα διαχείρισης περιεχομένου** (CMS – Content Management Systems) όπως το Joomla, Drupal, WordPress κ.α. Η πολυπλοκότητα αυτή οφείλεται στο ότι πρέπει να διατηρούνται σωστά ένα σύνολο από CSS links σε ένα μεγάλο εύρος φακέλων και αρχείων τα οποία αλλάζουν και ανανεώνονται συνεχώς. Τα περισσότερα από τα συστήματα τέτοιου είδους χρησιμοποιούν CSS αρχεία για τη διαμόρφωση των σελίδων, αλλά είναι πολύπλοκα και μερικές φορές κάποια αλλαγή σε αυτά μπορεί να προκαλέσει δυσάρεστες συνέπειες.

2.2.Σύνδεση αρχείου HTML με CSS

Όπως αναφέραμε παραπάνω, ένα αρχείο HTML φέρει το περιεχόμενο που θέλουμε να επιδείξουμε. Δείξαμε επίσης πώς μπορούμε να κάνουμε μία βασική μορφοποίηση του περιεχομένου με HTML (μορφοποίηση γραμματοσειράς, δομή δεδομένων). Σε περίπτωση που θέλαμε μία πιο περίπλοκη διαδικασία, όπως η αλλαγή γραμματοσειράς ή του φόντο της σελίδας, θα δημιουργούσαμε ξεχωριστό CSS αρχείο γι' αυτό το σκοπό, όπου θα περιγράφεται η μορφοποίηση των διαφόρων τμημάτων της σελίδας, με τον ακόλουθο τρόπο:

- Δημιουργία νέου CSS αρχείου (π.χ. template.css) με τον επιθυμητό κώδικα μορφοποίησης περιεχομένου.
- Σύνδεση του περιεχομένου μες τη μορφοποίηση μέσα από το αρχείο HTML. Συγκεκριμένα, προσθέτουμε μία ετικέτα της παρακάτω μορφής:

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

Να σημειωθεί ότι για να επιτυγχανθεί η παραπάνω σύνδεση, τα δύο αυτά αρχεία θα πρέπει να βρίσκονται στην ίδια διαδρομή (φάκελο).

Αυτός ο τρόπος όμως δεν είναι ο μοναδικός. Άλλες δύο μέθοδοι για να επιτύχουμε το ίδιο αποτέλεσμα περιγράφονται παρακάτω, δε χρησιμοποιούνται όμως πάρα μόνο σε μικρούς ιστότοπους, λόγω της αύξησης της πολυπλοκότητας του κώδικα που συνεπάγονται:

- **Ενσωμάτωση στο HTML αρχείο**. Στην περίπτωση που η μορφοποίηση αφορά ένα (ή γενικά λίγα) στοιχείο και δεν προορίζεται να επαναχρησιμοποιηθεί, μπορούμε να το εντάξουμε μέσα σε ένα *attribute style*. Για παράδειγμα, για να κάνουμε μια συγκεκριμένη παράγραφο να έχει μαύρο φόντο, προσθέτουμε ***<p style = "color: #000;">***. Με αυτά τα δεδομένα, εύλογο είναι το ερώτημα τι θα συμβεί όταν η μορφοποίηση ενός στοιχείου ορίζεται τόσο από ένα attribute style στο HTML αρχείο, όσο και από ένα ξεχωριστό CSS αρχείο. Σε αυτή την περίπτωση, ο περιηγητής είναι προγραμματισμένος να δίνει προτεραιότητα στη μορφοποίηση εντός του attribute style για το στοιχείο αυτό.
- **CSS για μια συγκεκριμένη σελίδα**. Υπάρχουν περιπτώσεις που θα θελήσουμε να εφαρμόσουμε κάποιους κανόνες CSS μόνο σε μία συγκεκριμένη σελίδα και όχι στο σύνολο του ιστότοπου. Ένας τρόπος να το κάνουμε αυτό είναι να εισάγουμε εντός των ετικετών <head>...</head> της σελίδας τους κανόνες CSS μας μέσα σε <style> ετικέτες. Για παράδειγμα, για να κάνουμε το χρώμα γραμμάτων μιας συγκεκριμένης σελίδας μαύρο, θα γράφαμε στο head της: ***<style type = "text/css"> body { color: #000; } </style>.***

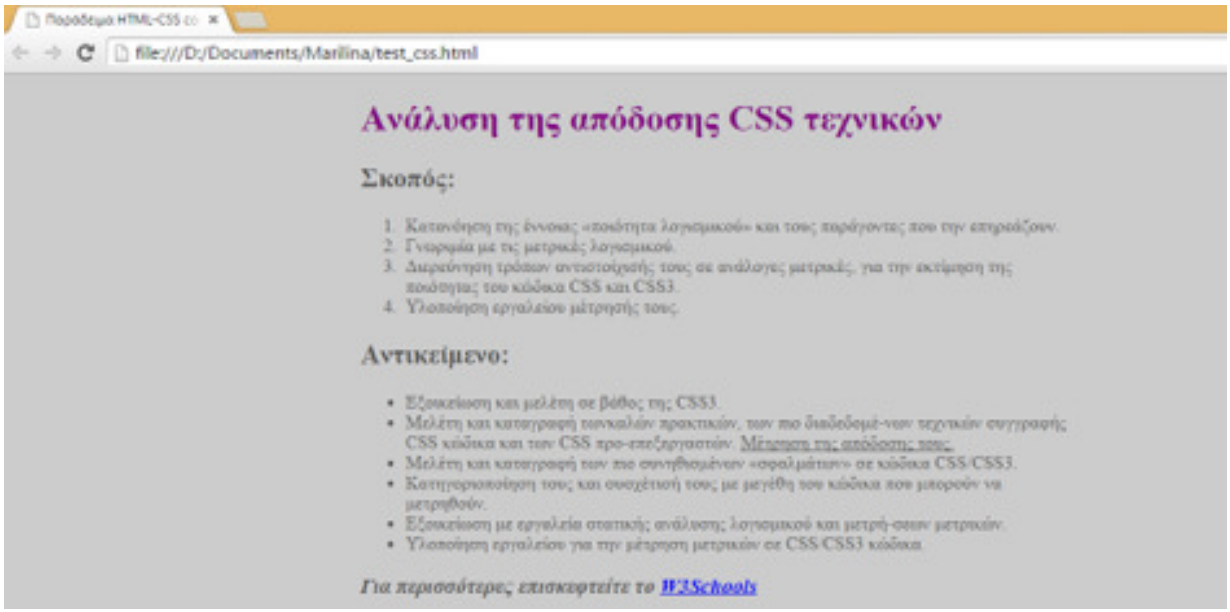
Παράδειγμα κώδικα HTML που αναφέρει το αρχείο CSS:

```
<!DOCTYPE html>
<html>
	<head>
		<title>Παράδειγμα HTML-CSS connection</title>
		<link rel="stylesheet" type="text/css" href="template.css" >
	</head>
	<body>
		<h1>Ανάλυση της απόδοσης CSS τεχνικών</h1>
		<h2>Σκοπός:</h2>
		<ol>
			<li>Κατανόηση της έννοιας «ποιότητα λογισμικού» και τους παράγοντες που την επηρεάζουν.</li>
			<li>Γνωριμία με τις μετρικές λογισμικού.</li>
			<li>Διερεύνηση τρόπων αντιστοίχισής τους σε ανάλογες μετρικές, για την εκτίμηση της ποιότητας του κώδικα CSS και CSS3.</li>
			<li>Υλοποίηση εργαλείου μέτρησής τους.</li>
		</ol>

		<h2>Αντικείμενο:</h2>
		<ul>
			<li>Εξοικείωση και μελέτη σε βάθος της CSS3.
			<li>Μελέτη και καταγραφή τωνκαλών πρακτικών, των πιο διαδεδομένων τεχνικών συγγραφής CSS κώδικα και των CSS προ-επεξεργαστών. <u>Μέτρηση της απόδοσης τους.</u></li>
			<li>Μελέτη και καταγραφή των πιο συνηθισμένων «σφαλμάτων» σε κώδικα CSS/CSS3.</li>
			<li>Κατηγοριοποίηση τους και συσχέτισή τους με μεγέθη του κώδικα που μπορούν να μετρηθούν.</li>
			<li>Εξοικείωση με εργαλεία στατικής ανάλυσης λογισμικού και μετρήσεων μετρικών.</li>
			<li>Υλοποίηση εργαλείου για την μέτρηση μετρικών σε CSS/CSS3 κώδικα.</li>
		</ul>
		<img src = css3.jpg></p>
		<p><h3><i>Για περισσότερες επισκεφτείτε το <a href="http://
www.w3schools .com/">W3Schools</a><i></h3></p>
	</body>
</html>
```

Και το αντίστοιχο αρχείο CSS:

```
body
{
    background-color:#ccc;
    margin:auto;
    width:50%;
    color: #555;
}

h1{
    color: purple;
}

p {
    font: Tahoma
}
```

Με τη χρήση ενός browser βλέπουμε το αποτέλεσμα (φόντο, χρώμα γραμματοσειράς, μέγεθος γραμματοσειράς, πλαίσιο κειμένου):



2.3.Κανόνες CSS

Ένα από τα βασικά πλεονεκτήματα της γλώσσας CSS, όπως αναφέραμε παραπάνω, είναι η ευκολότερη συντήρηση ιστοσελίδων μέσω της επεξεργασίας του συνόλου της μορφοποίησης από το αρχείο CSS. Για παράδειγμα για αλλαγή του χρώματος της γραμματοσειράς ή του μεγέθους των τίτλων, αρκεί μία μοναδική αλλαγή σε αυτό το αρχείο, αντί για την επεξεργασία πολλών σημείων σε κάθε σελίδα που υπάρχει στο site.

Το σύνολο των εντολών το οποίο μας δίνει τη δυνατότητα να αλλάξουμε ένα σύνολο στοιχείων, ονομάζεται κανόνας και αποτελείται από δύο μέρη:

- Ο *επιλογέας (CSS selector)* ο οποίος ορίζει το ποιο στοιχείο θα μορφοποιηθεί (*τι* θα μορφοποιηθεί) και
- Οι *ιδιότητες (CSS properties)* οι οποίες ορίζουν τον τρόπο (χρώμα γραμματοσειράς, είδος γραμματοσειράς κ.α.) με τον οποίο θα μορφοποιηθεί το κάθε στοιχείο (*πώς* θα μορφοποιηθεί).

Η σύνταξη έχει ως εξής:

```
selector
{
        property1:  value1;
        property2:  value2;
        property3: value3;
         ...
}
```

2.4.Επιλογείς

Μέσω των *επιλογέων CSS (CSS selectors)* επιλέγουμε ποια στοιχεία θα μορφοποιήσουμε. Είναι ουσιαστικά κάποιοι «κανόνες» ώστε να καταλάβει ο browser

που θέλουμε να εφαρμοστούν τα όσα γράφουμε κάθε φορά. Υπάρχουν αρκετά περίπλοκοι επιλογείς CSS στις δυνατότητες της γλώσσας, οι οποίοι προσφέρουν τεράστια ευελιξία, ωστόσο εδώ θα εξετάσουμε τους πιο βασικούς, τόσο για λόγους απλούστευσης, όσο και επειδή αρκετοί από τους πιο περίπλοκους δεν υποστηρίζονται από το σύνολο των browsers.

2.4.1.Καθολικός Επιλογέας (*)

Όταν ο επιλογέας είναι ένας χαρακτήρας αστερίσκου, τότε οι ιδιότητες που θα γράψουμε σε αυτόν τον κανόνα CSS εφαρμόζονται σε κάθε στοιχείο της σελίδας μας. Για παράδειγμα αν θέλουμε όλες ετικέτες στην ιστοσελίδα μας (παράγραφος, επικεφαλίδα κ.α) να εμφανίζονται με μέγεθος 12, τότε προσθέτουμε στον CSS κώδικα *{ *font-size: 12pt; }* και το μέγεθος όλων των στοιχείων της σελίδας αλλάζει. Όπως είναι κατανοητό, συνήθως δεν είναι ιδιαίτερα χρήσιμος επιλογέας από μόνος του και χρησιμοποιείται κυρίως σε συνδυασμό με άλλους.

2.4.2.Επιλογείς Ετικέτας ή Στοιχείου

Όταν ο επιλογέας αποτελείται από το όνομα μίας ετικέτας HTML, τότε οι ιδιότητες που θα γράψουμε σε αυτόν τον κανόνα CSS εφαρμόζονται σε κάθε τέτοιο στοιχείο HTML. Για παράδειγμα, ο επιλογέας *p* θα εφαρμοστεί σε οτιδήποτε στη σελίδα μας περιλαμβάνεται εντός των ετικετών <*p*>...</*p*>, ο επιλογέας *h1* θα εφαρμοστεί σε όλες της επικεφαλίδες τύπου 1, ο επιλογέας *table* θα εφαρμοστεί σε όλους τους πίνακες στη σελίδα μας, ο επιλογέας *img* θα αφορά όλες τις εικόνες στη σελίδα κ.ο.κ. Προφανώς όταν θέλουμε να εφαρμόσουμε κάποιες ιδιότητες CSS σε ολόκληρη τη σελίδα, χρησιμοποιούμε τον επιλογέα *body,* καθώς όλο το ορατό τμήμα της σελίδας περιέχεται εντός των tags <*body*>...</*body*>.

Πριν τη χρήση των CSS, προκειμένου να αλλάξουμε τη μορφοποίηση στο κείμενο μιας παραγράφου, θα έπρεπε να περικλύσουμε το κάθε κείμενο σε μια ετικέτα *<font>*, η οποία όριζε την ίδια εμφάνιση σε κάθε παράγραφο σε μια σελίδα. Η υπερβολική χρήση αυτών των ετικετών μέσα στην HTML, ωστόσο, έκανε την λήψη και την εμφάνιση των σελίδων πολύ αργή ενώ η διαδικασία ενημέρωσης ήταν ακόμη πιο χρονοβόρα.

Η χρήση επιλογέων δεν αποτέλεσε πανάκεια για τους σχεδιαστές. Εάν, για παράδειγμα, θέλαμε μόνο μια παράγραφος της σελίδας μας να έχει διαφορετική μορφοποίηση, ο browser θα εφάρμοζε ίδια μορφοποίηση, καθώς θα αδυνατούσε να διακρίνει τη διαφορά. Λύση σε αυτό το πρόβλημα δόθηκε με τους επιλογείς κλάσης

2.4.3.Επιλογείς Κλάσης (.όνομα_κλάσης)

Όταν ο επιλογέας μας περιλαμβάνει μια τελεία (.) στην αρχή του, τότε ο browser ψάχνει όσα στοιχεία στη σελίδα μας περιλαμβάνουν την ιδιότητα κλάση (class) και εφαρμόζει τις ιδιότητες που θα γράψουμε στον κανόνα CSS αυτό σε οποιοδήποτε στοιχείο περιλαμβάνει την κλάση «όνομα_κλάσης» στην ιδιότητα class το υ. Φυσικά ως όνομα_κλάσης μπορούμε να γράψουμε οτιδήποτε αποτελείται από γράμματα, αριθμούς, παύλες και χαρακτήρες underscore (_) και να ξεκινάει με γράμμα. Αξίζει να σημειωθεί ότι μπορεί το ίδιο στοιχείο να ανήκει σε περισσότερες από μια κλάσεις, διαχωρισμένες με κενά μέσα στο HTML χαρακτηριστικό κλάσης του (class html attribute). Π.χ. * ... *.

Για παράδειγμα, ο παρακάτω κανόνας CSS:

```
.emphasis
{
 color: red;
}
```

θα κάνει κόκκινη τη γραμματοσειρά τόσο στο στοιχείο:

* Random text *

όσο και στο στοιχείο:

*<div class=”emphasis otherclass”> Random text *

αλλά όχι στο στοιχείο:

<h1 class=”otherclass”> Random text </h1>

Έτσι, στο παράδειγμα που δημιουργήσαμε στο προηγούμενο κεφάλαιο, έστω ότι θέλουμε η υποσημείωση να εμφανίζεται με άσπρα γράμματα. Δημιουργούμε μία κλάση .note μέσω στο αρχείο CSS ως εξής:

```
.note
{
color: white;
}
```

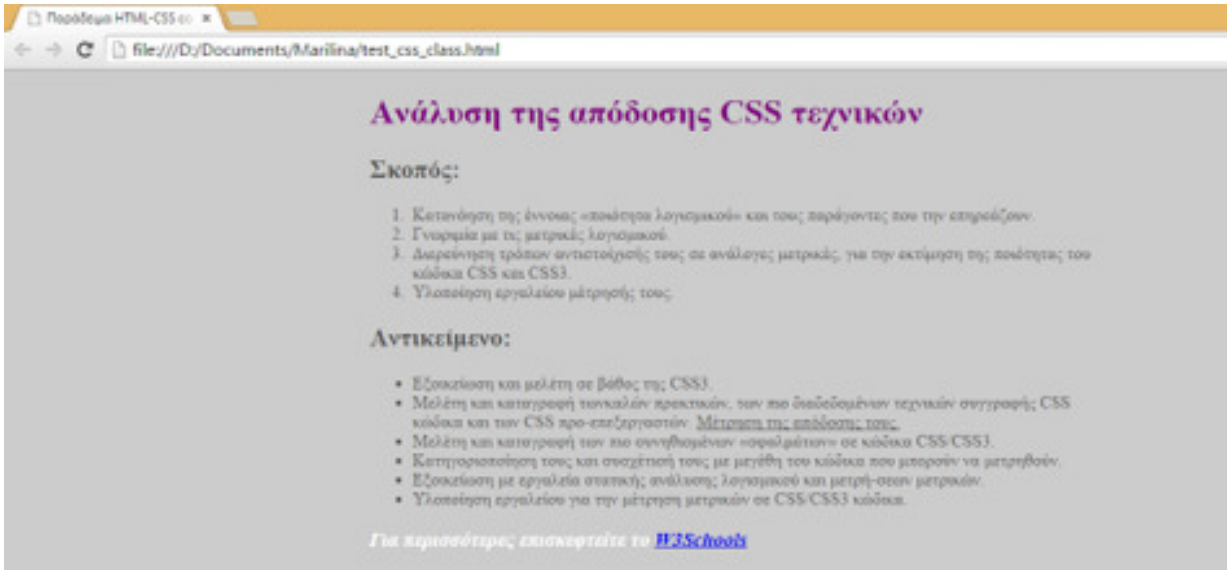
Στον HTML κώδικα τροποποιούμε το στοιχείο της σημείωσης, οπότε το:

```
<p><h3><i>Για περισσότερες επισκεφτείτε το <a href="http://www.w3schools .com/">
W3Schools</a><i></h3></p>
```

γίνεται:

```
<p><h3 class="note"><i>Για περισσότερες επισκεφτείτε το <a href="http://
www.w3schools .com/"> W3Schools</a><i></h3></p>
```

και το αποτέλεσμα είναι:



Οι κλάσεις γενικά χρησιμοποιούνται όταν θέλουμε να ομαδοποιήσουμε κάποια στοιχεία HTML για τα οποία μπορούμε να βρούμε κάποιον άλλο επιλογέα που να αφορά όλα αυτά και αποκλειστικά αυτά, οπότε τους προσδίδουμε μια συγκεκριμένη κλάση, ώστε να μπορούμε στο CSS μας να αναφερθούμε μόνο σε αυτά και να τα μορφοποιήσουμε αναλόγως.

2.4.4.Επιλογείς Στοιχείου - Κλάσης (στοιχείο.όνομα_κλάσης)

Αποτελεί ουσιαστικά συνδυασμό των δύο παραπάνω επιλογέων. Εφαρμόζεται σε όσα στοιχεία αποτελούνται από την ετικέτα HTML *<στοιχείο>* και ανήκουν στην κλάση *όνομα_κλάσης*. Για παράδειγμα, ο επιλογέας *p.emphasis* εφαρμόζεται σε όλο το περιεχόμενο που βρίσκεται σε ετικέτες της μορφής:

* ... *.

Ο επιλογέας αυτός είναι χρήσιμος όταν έχουμε πολλά διαφορετικού τύπου στοιχεία με την ίδια κλάση και επιθυμούμε να εφαρμόσουμε διαφορετική μορφοποίηση ανάλογα με τον τύπο του στοιχείου (επικεφαλίδα, παράγραφος, αρίθμηση, κ.α.).

2.4.5.Επιλογείς ID (#όνομα_id)

Όταν ο επιλογέας περιλαμβάνει ένα χαρακτήρα δίεσης *(#)* στην αρχή του, τότε ο browser εφαρμόζει τις ιδιότητες που θα γράψουμε στο στοιχείο το οποίο περιλαμβάνει την ιδιότητα *id =”όνομα_id”*. Δεν πρέπει να υπάρχουν δύο (ή περισσότερα) στοιχεία στη σελίδα μας με το ίδιο ID. Τα IDs διέπονται από τους ίδιους κανόνες ονοματολογίας με τις κλάσεις. Ουσιαστικά, ό,τι μπορούμε να κάνουμε με τα IDs μπορούμε να το κάνουμε και με τη χρήση κλάσεων, απλά όταν το στοιχείο που θέλουμε να μορφοποιήσουμε είναι μοναδικό, είναι γενικά καλύτερο να χρησιμοποιούμε IDs. Ιδανική επιλογή αποτελεί για ιστοσελίδες μεγάλου μεγέθους ή κατασκευασμένες με τη γλώσσα προγραμματισμού JavaScript.

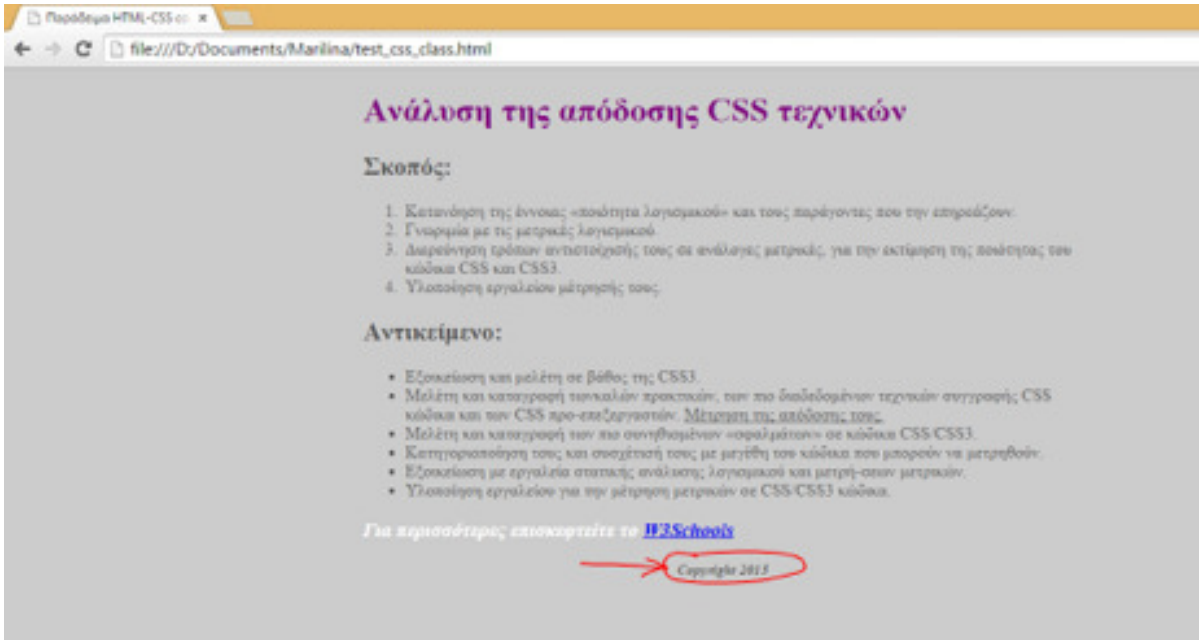
Η χρησιμότητά του είναι για τροποποίηση ενός μοναδικού μέρους της ιστοσελίδας, όπως το πάνω ή το κάτω μέρος της σελίδας. Εάν, για παράδειγμα θέλουμε να προσθέσουμε ένα footer (κάτω μέρος) με μικρά γράμματα, προσθέτουμε στο αρχείο CSS:

```
#footer
{
text-align: center;
font-size: 10pt;
font-weight: bold;
}
```

στον HTML κώδικα προσθέτουμε στο τέλος:

```
<p id="footer">Copyright 2015</p>
```

και το αποτέλεσμα είναι:



2.4.6.Επιλογείς Στοιχείου - Ιδιότητας (στοιχείο[attribute=”value”])

Αποτελεί ουσιαστικά μια «επέκταση» του επιλογέα ετικέτας ή στοιχείου που αναλύθηκε παραπάνω. Ο εν λόγω επιλογέας υποδεικνύει στον browser να εφαρμόζει τις ιδιότητες που θα γράψουμε σε αυτόν σε κάθε στοιχεί ο με ετικέτα *<στοιχείο>* το οποίο επιπροσθέτως έχει την τιμή *value* στην HTML ιδιότητα *attribute*.

Παραδείγματος χάριν, ο επιλογέας *input[type=”submit”]* αφορά όλα τα κουμπιά υποβολής φόρμας που υπάρχουν στη σελίδα μας, χωρίς ωστόσο να εφαρμόζεται σε άλλα στοιχεία φορμών όπως τα πεδία κειμένου (στα οποία η ιδιότητα *type* είναι *text*).

Άλλο ένα παράδειγμα: Έστω ότι θέλουμε να μορφοποιήσουμε μόνο όσους πίνακες στη σελίδα μας είναι κεντραρισμένοι. Αν χρησιμοποιούσαμε ως επιλογέα το *table*, τότε οι ιδιότητες που θα γράφαμε σε αυτόν τον επιλογέα θα εφαρμόζονταν σε όλους τους

πίνακες ανεξαιρέτως. Ενώ αν χρησιμοποιήσουμε τον επιλογέα *table[align=”center”]* τότε ο καν όνας CSS που θα γράψουμε, θα εφαρμοστεί μόνο σε όσους πίνακες έχουν την ιδιότητα *align=”c enter”*.

2.5.Ψευδο-κλάσεις και Ψευδο-στοιχεία

Ορισμένες φορές χρησιμοποιούμε τις λεγόμενες ψευδό-κλάσεις (pseudo-classes) ή ψευδό-στοιχεία (pseudo-elements), τα οποία μας επιτρέπουν να επιλέγουμε στοιχεία τα οποία δεν αποτελούν στοιχεία HTML, αλλά κομμάτια τους ή συγκεκριμένες καταστάσεις τους. Ουσιαστικά αποτελού ν κάποιες λέξεις-κλειδιά που γράφουμε μετά από έναν επιλογέα του τύπου στοιχείο και ξεκινούν με άνω-κάτω τελεία (:).

Παρακάτω παρατίθενται οι συνηθέστερες ψευδο-κλάσεις και ψευδο-στοιχεία που χρησιμοποιούνται:

- **a:link**. Χρησιμοποιείται αποκλειστικά για στοιχεία a (Δηλαδή: *a:link*) και αφορά τους συνδέσμους που ο χρήστης δεν έχει ακόμη επισκεφθεί.
- **a:visited**. Επίσης χρησιμοποιείται αποκλειστικά για στοιχεία a (Δηλαδή: *a:visited*) και αφορά τους συνδέσμους που ο χρήστης έχει επισκεφθεί.
- **στοιχείο:active**. Αφορά τα στοιχεία τύπου <*στοιχείο*> τη στιγμή που ο χρήστης έχει πατημένο το ποντίκι πάνω σε αυτά. Για παράδειγμα, ο επιλογέας *a:active* εφαρμόζεται σε συνδέσμους την ώρα που ο χρήστης έχει πατημένο το ποντίκι πάνω τους.
- **στοιχείο:hover**. Από τις πιο συχνά χρησιμοποιούμενες ψευδό-κλάσεις. Αφορά τα στοιχεία τύπου <*στοιχείο*> τη στιγμή που ο χρήστης έχει το δείκτη του ποντικιού πάνω σε κάποιο από αυτά (χωρίς να πατάει το

ποντίκι). Μπορεί να μας βοηθήσει να δημιουργήσουμε διάφορα όμορφα εφέ, τα οποί α παλιότερα ήταν εφικτά μόνο με javascript.

- **σ τ ο ι χ ε ί ο : f o c u s** . Χρησιμοποιείται κυρίως για στοιχεία φορμών και εφαρμόζεται στα στοιχεία τύπου <*στοιχείο*> που εκείνη τη στιγμή έχουν *«focus»,* παραδείγματος χάριν, ένα πεδίο κειμέ νου στο οποίο ο χρήστης έκανε κλικ για να εισάγει κείμενο.

- **σ τ ο ι χ ε ί ο : f i r s t − l e t t e r** . Αφορά το πρώτο γράμμα του κειμένου εντός κάποιου στοιχείου τύπου <*στοιχείο*>. Τ ο ψ ε υ δ ό − στοιχείο αυτό μπορεί να μας βοηθήσει να δημιουργήσουμε αρχιγράμματα. Για παράδειγμα, ο επιλογέας *p:first−letter* αφορά το πρώτο γράμμα κάθε παραγράφου.

- **στοιχείο:first−line**. Ψευδό−στοιχείο παρόμοιο με το παραπάνω, με τη διαφορά ότι αντί να αφορά μόνο το πρώτο γράμμα του κειμένου μέσα στο στοιχείο τύπου <*στ οιχείο*>, αφορά ολόκληρη την πρώτη γραμμή.

- **στοιχείο:before**. Διευκολύνει την προσθήκη μικρών κομματιών κώδικα. Για παράδειγμα αν είχαμε μια ιστοσελίδα ανακοινώσεων μιας γραμματείας θα ήταν πολύ χρήσιμο οι ανακοινώσεις σχετικά με την εξεταστική περίοδο να είχαν μπροστά μια μικρή επισήμανση που θα το όριζε. Για την κατασκευή της θα δημιουργούσαμε ένα στοιχείο με το όνομα *category* και θα το ορίζαμε στο CSS αρχείο ως:

  p.category: before { content: "[ΕΞΕΤΑΣΤΙΚΗ]" }

  Με αυτό τον τρόπο μειώνεται η πολυπλοκότητα του κώδικα HTML της σελίδας μας.

- **στοιχείο:after**. Όμοια λειτουργία με το *before* μόνο που αντί να προσθέτει την επισήμανση πριν το στοιχείο (π.χ. παράγραφο) το προσθέτει μετά από αυτό. Ευρεία η χρήση του είναι για την προσθήκη εισαγωγικών σε μία παράθεση.

- **::selection**. Επιλογέας που εισήχθη στη CSS3 και που χρησιμοποιείται για τη μορφοποίηση επιλεγμένου κειμένου. Πρόκειται για την επιλογή τμήματος κειμένου που κάνει ο χρήστης με το ποντίκι (π.χ. για να το αντιγράψει). Οι περισσότεροι

περιηγητές από προεπιλογή αλλάζουν το φόντο του επιλεγμένου κειμένου σε σκούρο μπλε, αλλά αν θέλουμε το φόντο να έχει διαφορετικό χρώμα (κυρίως το χρώμα φόντου και γραμματοσειράς τροποποιούμε) τότε αρκεί να προσθέσουμε μία εντολή της μορφής:

::selection { color: red; background-color: yellow; }

2.6.Μοντέλο Κουτιού CSS

Τα στοιχεία μιας σελίδας HTML μπορεί να θεωρηθούν ως ένα σύνολο κουτιών με διαφορετικές διατάξεις (το ένα εμφωλευμένο στο άλλο, το ένα δίπλα/πάνω/κάτω από κάποιο άλλο). Στην CSS ο όρος *«μοντέλο κουτιού» (CSS Box Model)* χρησιμοποιείται όταν αναφερόμαστε στο σχεδιασμό και τη διάταξη των στοιχείων. Το μοντέλο κουτιού CSS είναι ουσιαστικά ένα κουτί που περιβάλλει τα στοιχεία HTML και αποτελείται από περιθώρια,

περιγράμματα και το περιεχόμενο της ιστοσελίδας. Εξωτερικά υπάρχει το κουτί margin, που μέσα του περικλείει το κουτί border, που με την σειρά του περικλείει το κουτί padding. Εσωτερικά όλων υπάρχει το περιεχόμενο που είναι γραμμένο σε HTML. Οι παραπάνω ιδιότητες αναλύονται διεξοδικά στην επόμενη ενότητα.

2.7.Ιδιότητες CSS

Σε προηγούμενες ενότητες αναφερθήκαμε σε διάφορες ιδιότητες μορφοποίησης μεμονωμένα (γραμματοσειρά, χρώμα, φόντο, ιδιότητες μοντέλου κουτιού), προκειμένου να περιγράψουμε άλλα χαρακτηριστικά της γλώσσας CSS. Σε αυτή την ενότητα θα εμβαθύνουμε σε αυτά τα χαρακτηριστικά ξεχωριστά.

2.7.1.Γενικές Ιδιότητες

- **margin**. Ορίζει τον χώρο μεταξύ των ορίων ενός στοιχείου και όσων το περιβάλλουν. Είναι ιδιαίτερα σημαντικό να ορίζουμε margin σε εικόνες, ώστε να έχουν απόσταση από τα περιεχόμενα τους, μιας και είναι ιδιαίτερα αντιαισθητικό να εφάπτονται στο κ είμενο. Φυσικά και εδώ μπορούμε να χρησιμοποιήσουμε είτε τις ιδιότητες *margin-top*, *margin-right*, *margin-bottom*, *margin-left* για να ορίσουμε διαφορετικές τιμές margin ανά πλευρά, είτε να εισάγουμε τις τέσ σερις διαφορετικές τιμές margin με τη σειρά *[top] [right] [bottom] [left]*, ή *[top,bottom] [right,left]*, ή *[top,bottom,right,left]*.

  Παραδείγματα:

  - margin: 4px;
  - margin: 2px 4px;
  - margin: 0px 8px 2px 2px;

- **padding**. Ορίζει το κενό που θα υπάρχει μεταξύ των ορίων ενός στοιχείου και των περιεχομέν ων του. Είναι πολύ σημαντικό να ορίζουμε padding σε στοιχεία στα οποία έχουμε ορί σει κάποιο περίγραμμα, ώστε να μην εφάπτονται τα περιεχόμενα τους στο περίγραμμ α, κάτι που φαίνεται ιδιαίτερα άσχημο και ερασιτεχνικό. Το padding χρησιμοποιείται σε μεγάλη κλίμακα, καθώς όσο περισσότερο τόσο πιο επαγγελματική εμφάνιση (μέχρι κάποιων ορίων φυσικά). Για ν α ορίσουμε διαφορετικό padding ανά πλευρά,

μπορούμε ομοίως με το margin, είτε να χρησιμοποιήσουμε τις ιδιότητες *padding-top, padding-right, padding-bottom, padding-left*, είτε να εισάγουμε και τις τέσσερις τιμές στην ιδιότητα padding με τη σειρά *[top] [right] [bottom] [left]*, είτε *[top,bottom] [right,left],* είτε *[top,bottom,right,left]*.

Παραδείγματα:

- padding: 8px;
- padding: 2px 6px;
- padding: 0px 6px 6px 6px;

- **width, height**. Οι δύο αυτές ιδιότητες ορίζουν το πλάτος και το ύψος ενός στοιχείου αντίστοιχα, είτε σε ποσοστό (το οποίο υπολογίζεται βάσει του στοιχείου που το περιέχει), είτε σε κάποια μονάδα μήκους (π.χ. px, pt, cm).

  Παραδείγματα:

  - width: 100px;
  - width: 90%;
  - height: 30pt;
  - height: 100%;

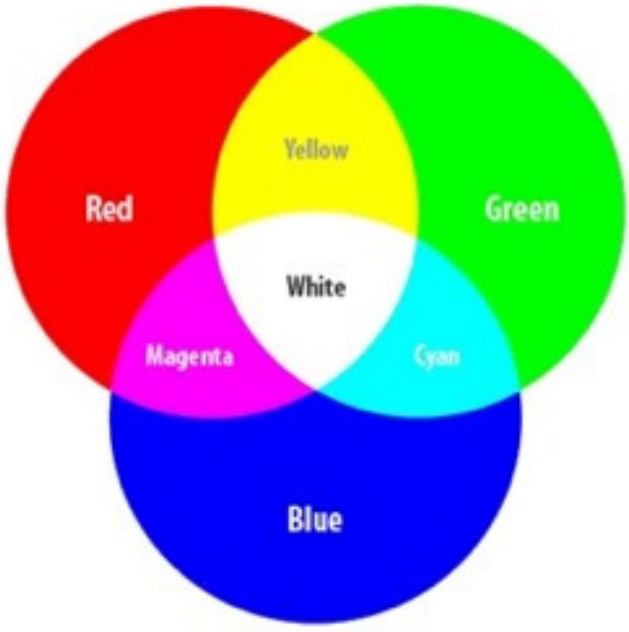
  Υπάρχουν και οι ιδιότητες *min-width, min-height, max-width, max-height* οι οποίες ορίζουν τα όρια στα οποία μπορούν να κινούνται οι διαστάσεις ενός στοιχείου, όταν δεν θέλουμε να ορίσουμε συγκεκριμένες διαστάσεις, αλλά καλό είναι να μην χρησιμοποιούνται ακόμα, όσο βολικές κι αν φαίνονται, μιας και δυστυχώς η υποστήριξη τους από τους υπάρχοντες browsers είναι περιορισμένη.

- **float**. Η ιδιότητα αυτή είναι η πιο χρήσιμη στη χρήση CSS και στο layout της σελίδας. Δέχεται τις τιμές *right, left* και *none* και επιτρέπει τα στοιχεία που περιβάλλουν το στοιχείο εκείνο στο οποίο εφαρμόζουμε την ιδιότητα αυτή να «ρέουν» τριγύρω του. Μια συνηθισμένη χρήση της ιδιότητας

είναι στις εικόνες που συνοδεύουν ένα άρθρο. Πολλές φορές χρησιμοποιούμε μετά από floated στοιχεία κάποιο στοιχείο (συνήθως ένα άδειο div) με την ιδιότητα *clear: both;* για να «καθαρίσει» τα floats για το μετέπειτα περιεχόμενο.

2.7.2.Ιδιότητες Μορφοποίησης Κειμένου

- **color**. Αφορά το χρώμα του κειμένου στο πρότυπο RGB (Red, Green, Blue), δηλαδή όλους τους δυνατούς συνδυασμούς που προκύπτουν από τα χρώματα κόκκινο, πράσινο και μπλε, και αναφέρεται με τρεις τρόπους:
    - σε μορφή RGB, π.χ. *color: rgb (255,0,0);*,
    - σε δεκαεξαδική μορφή, π.χ. *color: #550000;,*
    - με τη μορφή κάποιου keyword, π.χ. *color:red;*.



- **font-size**. Αφορά το μέγεθος της γραμματοσειράς. Οι τιμές που δέχεται μπορούν να είναι εκφρα

σμένες σε ένα μεγάλο πλήθος μονάδων μεγέθους, από τις οποίες οι πιο ευρέως διαδεδ ομένες είναι τα pixels (π.χ. *font-size:12px;*) και οι στιγμές (π.χ. *font-size:10pt*).

- **font-family**.
  Η ιδιότητα αυτή μας επιτρέπει να ορίσουμε ένα πλήθος γραμματοσειρών που θα χρη σιμοποιηθούν για το κείμενο, κατά σειρά προτίμησης. Ουσιαστικά μέσω αυτής ορίζο υμε τη γραμματοσειρά του κειμένου, απλά μας επιτρέπει να ορίσουμε και εναλλακτικ ές επιλογές, ώστε αν η γραμματοσειρά που ορίσαμε δεν υπάρχει στον υπολογιστή του χρήστη, να μην μας καταστρέψει ο browser την εμφάνιση της σελίδας επιλέγοντας μί α

  τυχαία, αλλά την επόμενη επιλογή μας. Αν καμία από όσες γραμματοσειρές ορίσαμε δεν υπάρχει στον υπολογιστή του χρήστη, τότε ο browser επιλέγει κάποια που να ανή κει στη γενική οικογένεια γραμματοσειρών (generic font-family) που θα ορίσουμε στο τέλος με το κατάλληλο keyword. Τέτοια keywords είναι τα εξής:

  - *serif*: Γραμματοσειρές με «πατούρες» όπως η Georgia.
  - *sans-serif*: Γραμματοσειρές χωρίς «πατούρες» όπως η Trebuchet MS.
  - *monospace*: Γραμματοσειρές όπου το κάθε γράμμα καταλαμβάνει ίσο πλάτος με τα υπόλοιπα, όπως η Courier New.
  - *cursive*: Καλλιγραφικές γραμματοσειρές όπως η Monotype Corsiva
  - *fantasy*: «Διακοσμητικές» γραμματοσειρές.

  Οι τελευταίες 2 γενικές οικογένειες γραμματοσειρών χρησιμοποιούνται πολύ σπάν ια. Παράδειγμα χρήσης της ιδιότητας:

  font-family: Calibri, Trebuchet MS, Verdana, sansserif;

- **font-style**.
  Σε αντίθεση με αυτό που θα περίμενε κανείς από μια ιδιότητα με αυτό το όνομα, δηλ αδή την εφαρμογή πολλών και διαφόρων «εφέ» στο κείμενο, στην πραγματικότητα α

φορά μόνο την περίπτωση όπου το κείμενο θα είναι πλάγιο. Για άλλα εφέ, χρησιμοποιούνται άλλες ιδιότητες, οι οποίες θα εξεταστούν παρακάτω. Οι πιθανές τιμές της είναι *normal*, *italic* και *oblique*. Οι δύο τελευταίες συνήθως κάνουν το ίδιο πράγμα (και είναι προτιμότερη η χρήση της italic μιας και υποστηρίζεται και από τον Internet Explorer 3, ενώ η υποστήριξη για την oblique ξεκινά από Internet Explorer 4), ενώ είναι προφανής η λειτουργία της τιμής normal.

- **font-weight**. Αφορά το «βάρος» της γραμματοσειράς και στην πράξη χρησιμοποιείται για να ορίσει αν το κείμενο μας θα είναι έντονο ή όχι, δεδομένου ότι οι περισσότερες γραμματοσειρές που χρησιμοποιούνται στο διαδίκτυο διατίθενται μόνο σε δύο βάρη: κανονικό και έντονο, σε αντίθεση με πιο εξειδικευμένες γραμματοσειρές που χρησιμοποιούνται από γραφίστες, οι οποίες πολλές φορές διατίθενται σε διάφορα βάρη. Οπότε οι τιμές που συνήθως χρησιμοποιούνται σε αυτή την ιδιότητα είναι οι *normal* και *bold*.

- **text-decoration**. Μας επιτρέπει να εφαρμόσουμε στο κείμενό μας διάφορα εφέ, συμπεριλαμβανόμενης και της υπογράμμισης. Οι τιμές που δέχεται είναι οι εξής:
  - *none*: Καμία διακόσμηση
  - *underline*: Υπογράμμιση
  - *overline*: Γραμμή πάνω από το κείμενο (ουσιαστικά το αντίθετο της υπογράμμισης)
  - *line-through*: Διαγράμμιση (διαγραφή κειμένου)
  - *blink*: Κάνει το κείμενο να αναβοσβήνει. Δεν πρέπει να χρησιμοποιείται, παρά μόνο σε εξαιρετικά σπάνιες περιπτώσεις, μιας και η εμπειρία έχει δείξει ότι εί

ναι εξαιρετικά κουραστικό για τον αναγνώστη και αφαιρεί επαγγελματικότητα από το σχεδιασμό της ιστοσελίδας.

Οι παραπάνω τιμές μπορούν να χρησιμοποιηθούν και συνδυαστικά, όταν επιθυμούμε να εφαρμόσουμε πάνω από ένα τέτοιο εφέ στο κείμενο μας. Παραδείγματος χάριν:

text-decoration: underline overline;

- **text−transformation**. Σχετίζεται με τον τρόπο γραφής μιας γραμματοσειράς σε κεφαλαία (uppercase) ή πεζά (lowcase). Μια επιπλέον λειτουργία είναι και η capitalize κατά την οποία το πρώτο γράμμα από κάθε λέξη τυπώνεται με κεφαλαία.
- **text−align**. Μας επιτρέπει να καθορίσουμε τη στοίχιση του κειμένου μας. Φυσικά, οι πιθανές τιμές είναι *left*, *center*, *right* και *justify*.
- **font**. Μας επιτρέπει να καθορίσουμε αρκετές ιδιότητες κειμένου με μια μόνο ιδιότητα (συγκεκριμένα τις *font-size, font-family, font-weight* και *font-style* από όσες αναφέρθηκαν παραπάνω, καθώς και κάποιες άλλες που δεν αναφέρθηκαν). Αν κάποια ιδιότητα δεν συμπεριληφθεί, ο browser θα «υποθέσει» την προεπιλεγμένη τιμή της.

  Παραδείγματα:

  - *font: 12px Calibri, Trebuchet MS, Verdana, sans-serif;*
  - *font: italic bold 12px Calibri, Trebuchet MS, Verdana, sans-serif;*

  Εάν, όμως, δεν ορίσουμε κάποια επιμέρους ιδιότητα όταν χρησιμοποιούμε την συγκεντρωτική ιδιότητα font, ο browser «υποθέτει» την προεπιλεγμένη τιμή για αυτή. Αυτό σημαίνει ότι αν γράψουμε για παράδειγμα:

  - *font-weight:bold;*
  - *font: 12px Calibri, Trebuchet MS, Verdana, sans-serif;*

τότε το κείμενο μας δεν θα είναι έντονο, αφού δεν ορίσαμε στη font και την font-weight και τότε ο browser «υποθέτει» την προεπιλεγμένη τιμή για αυτή την ιδιότητα, η οποία είναι η normal, ενώ δεδομένου ότι η font βρίσκεται κάτω από την font-weight, έχει μεγαλύτερη προτεραιότητα (αναλύεται σε άλλη ενότητα), οπότε επικρατεί για τη font-weight η τιμή normal.

- **Effects**. Στη CSS3 έχουν εισαχθεί δύο εφέ. Το πρώτο είναι η δημιουργία σκιάς σε ένα κείμενο (text-shadow) καθώς και το ταίριασμα λέξεων σε πλαίσιο χωρίς να βγαίνουν εκτός ορίων (word-wrap) και το χωρισμό αυτών προκειμένω να χωρέσουν.

2.7.3.Ιδιότητες Μορφοποίησης Φόντου

Το φόντο είναι βασικό στοιχείο μιας ιστοσελίδας. Η επιλογή του από το σχεδιαστή θα πρέπει να γίνει προσεκτικά, προκειμένου να συνδυάσει καλαισθησία (αρμονία με γραμματοσειρά και λοιπό περιεχόμενο) και πρακτικότητα (να μην κουράζει το χρήστη). Γι' αυτό το λόγο στο σύγχρονο σχεδιασμό ιστοσελίδων χρησιμοποιούνται λιγότερο έντονα χρώματα, γεγονός φυσικά που έγκειται στην κρίση του σχεδιαστή.

- **background-color**. Χρησιμοποιείται για να ορίσει χρώμα φόντου στα στοιχεία που αφορά ο επιλογέας. Το χρώμα μπορεί να γραφεί σε οποιαδήποτε από τις μορφές που περιγράφηκαν για την ιδιότητα color.

- **background-image**. Χρησιμοποιείται για να ορίσει μια εικόνα φόντου. Η τιμή που δέχεται είναι της μορφής *url(διεύθυνση_εικόνας)*, για παράδειγμα *background-image: url(http://www.mysite.gr/logo.gif);*. Γενικά όταν εισάγουμε μια διεύθυνση URL ως τιμή (ή τμήμα τιμής) κάποιας ιδιότητας CSS, πάντα τη γράφουμε εντός των παρενθέσεων του *url()*. Φυσικά μπορούμε να εισάγουμε και σχετικά URLs, π.χ. *background-image:url(logo.gif);*.

Στη CSS3 μπορούμε να έχουμε για φόντο πολλές εικόνες απλά εισάγοντας τις διευθύνσεις των εικόνων την μία πίσω από την άλλη.

- **background-repeat**. Αφορά τον τρόπο με τον οποίο θα επαναλαμβάνεται η εικόνα που ορίσαμε στην παραπάνω ιδιότητα. Οι τιμές που δέχεται είναι:
    - *no-repeat*: Καμία επανάληψη
    - *repeat-x*: Η εικόνα επαναλαμβάνεται στον οριζόντιο άξονα
    - *repeat-y*: Η εικόνα επαναλαμβάνεται στον κάθετο άξονα
    - *repeat*: Η εικόνα επαναλαμβάνεται και οριζόντια και κάθετα (η τιμή *repeat* είναι και η προεπιλεγμένη τιμή της ιδιότητας *background-repeat*).
- **background-position**. Η ιδιότητα αυτή μας επιτρέπει να ορίσουμε τη θέση που θα τοποθετηθεί η εικόνα φόντου που ορίσαμε με την ιδιότητα *background-image* τόσο κάθετα, όσο και οριζόντια. Συνήθως ως τιμές της χρησιμοποιούμε κάποια keywords, τα οποία είναι τα εξής:
    - *left, center, right* για την οριζόντια θέση της εικόνας
    - *top, center, bottom* για την κάθετη θέση της εικόνας

Τα παραπάνω μπορούμε να τα χρησιμοποιήσουμε και συνδυαστικά (το οποίο είναι και η πιο συνήθης τακτική), όπως στα παρακάτω παραδείγματα:

- *background-position: left bottom;*
- *background-position: top right;*
- *background-position: center top;*

ή μπορούμε να ορίσουμε την τοποθέτηση μόνο κατά τον ένα άξονα (για τον άλλο υποτίθεται η προεπιλεγμένη τιμή, η οποία είναι συνήθως *top* για τον κάθετο άξονα και *left* για τον οριζόντιο), όπως στα παρακάτω παραδείγματα:

- *background-position: left;*
- *background-position: bottom;*

Αντί για τα παραπάνω keywords, μπορούμε να ορίσουμε τη θέση της εικόνας φόντου και αριθμητικά, είτε με ποσοστό ή με κάποια μονάδα μεγέθους, όπως στα παρακάτω παραδείγματα:

- *background-position*: 10% 25%;
- *background-position*: 60px 100px;

αν και τέτοιες τιμές χρησιμοποιούνται πολύ πιο σπάνια.

Η ιδιότητα αυτή χρησιμοποιείται συνήθως όταν έχουμε ορίσει να μην επαναλαμβάνεται η εικόνα φόντου (μέσω της ιδιότητας *background-repeat*). Κάποιοι browsers υποστηρίζουν τη χωριστή δήλωση θέσης της εικόνας στον κάθετο και τον οριζόντιο άξονα, μέσω δύο ιδιοτήτων με όνομα *background-position-x* και *background-position-y*, αλλά καλύτερα να αποφεύγεται η χρήση αυτών των ιδιοτήτων, μιας και δεν υποστηρίζονται από όλους τους browsers.

- **background-size**: Πρόκειται ιδιότητα της CSS3, που καθορίζει το μέγεθος της εικόνας φόντου. Πριν την CSS3, το μέγεθος της εικόνας φόντου προσδιοριζόταν από το πραγματικό μέγεθος της εικόνας. Με την εντολή αυτή, κάθε εικόνα μπορεί να μεταβάλει το μέγεθός της το οποίο καθορίζεται σε pixels (*px*) ή ποσοστά (%).

- **background-origin**: Προσδιορίζει το χώρο τοποθέτησης των εικόνων φόντου. Η εικόνα φόντου μπορεί να τοποθετηθεί σε μία από τις τρεις παρακάτω περιοχές:
    - *content-box*
    - *padding-box*
    - *border-box*
- **background**. Η ιδιότητα αυτή αποτελεί ουσιαστικά συντόμευση για να ορίσουμε με μία μόνο ιδιότητα όλες τις παραπάνω (όπως είναι η ιδιότητα font για τα χαρακτηριστικά κειμένου).

  Παραδείγματα χρήσης:
    - *background: #c0c0c0 url(background.gif) no-repeat top center;*
    - *background: #c0c0c0 url(background.gif);*

  Αν δεν ορίσουμε κάποια επιμέρους ιδιότητα όταν χρησιμοποιούμε την συγκεντρωτική ιδιότητα background, ο browser «υποθέτει» την προεπιλεγμένη τιμή για αυτή και όχι ότι δεν ορίστηκε. Αυτό σημαίνει ότι αν γράψουμε για παράδειγμα:
    - *background-image: url(background.gif);*
    - *background: #CCCCCC no-repeat top center;*

  τότε δεν θα εμφανίζεται κάποια εικόνα φόντου, μιας και αφού δεν ορίσαμε στη *background* και την *background-image*, τότε ο browser «υποθέτει» την προεπιλεγμένη τιμή για αυτή την ιδιότητα, η οποία είναι η *none*, και μιας και η *background* βρίσκεται κάτω από την *background-image*, έχει μεγαλύτερη προτεραιότητα, οπότε επικρατεί για τη *background-image* η τιμή *none*.

2.7.4.Ιδιότητες Περιγράμματος

Η CSS μας δίνει την δυνατότητα εισαγωγής πλαισίων, που αποτελούν τόσο αισθητικό εργαλεία, όσο και πρακτικό, για την ανάδειξη σημαντικών τμημάτων της ιστοσελίδας μας.

- **border-color**. Ρυθμίζει το χρώμα περιγράμματος. Το χρώμα μπορεί να γραφεί σε οποιαδήποτε από τις μορφές που περιγράφηκαν για την ιδιότητα color. Αν δεν οριστεί αυτή η ιδιότητα, χρησιμοποιείται το χρώμα που ορίστηκε στην ιδιότητα color.
- **border-width**. Ρυθμίζει το πάχος του περιγράμματος σε κάποια από τις μονάδες μέτρησης που μπορούν να χρησιμοποιηθούν στο CSS, συνηθέστερα σε pixels.

  Παράδειγμα:

  - *border-width: 10px;*
- **border-style**. Ορίζει το στυλ του περιγράμματος. Οι τιμές που χρησιμοποιούνται συνήθως για την ιδιότητα αυτή είναι οι εξής:
  - *solid*: «Συμπαγές» περίγραμμα, δηλαδή χωρίς κάποια διακόσμηση, μία ενιαία γραμμή.
  - *dashed*: Περίγραμμα που αποτελείται από παύλες.
  - *dotted*: Περίγραμμα που αποτελείται από τελείες. Αξίζει να σημειωθεί ότι τα αποτελέσματα της χρήσης της τιμής αυτής διαφέρουν ελαφρώς ανά τους browsers. Για παράδειγμα ο Internet Explorer εμφανίζει τα *dotted* περιγράμματα ως αποτελούμενα από μικρούς κύκλους, ενώ ο Mozilla Firefox τα εμφανίζει ως αποτελούμενα από μικρά τετράγωνα.
  - *double*: Σαν την τιμή *solid*, μόνο που δημιουργεί δύο περιγράμματα, η απόσταση και το πάχος των οποίων δεν μπορούν να ρυθμιστούν χωριστά, αλλά ρυθμίζονται αυτόματα ώστε το πάχος τους να είναι

συνολικά όσο η τιμή που ορίσαμε στην ιδιότητα *border-width*. Συνήθως χρησιμοποιείται σε συνδυασμό με πάχος περιγράμματος *3px*, για το οποίο δίνει ένα αρκετά καλαίσθητο αποτέλεσμα.

Υπάρχουν και οι τιμές *groove*, *ridge*, *inset*, *outset* οι οποίες δημιουργούν διαφόρων ειδών «ψευδό-3D» περιγράμματα, ωστόσο στο σύγχρονο σχεδιασμό ιστοσελίδων αποφεύγονται για λόγους αισθητικής.

- **border-radius**. Ορίζει περιγράμματα με στρογγυλεμένες άκρες.
- **box-shadow**. Ορίζει πλαίσια σε σχήμα τετράγωνου με την προσθήκη του εφέ σκίασης.
- **border-image**. Με την ιδιότητα αυτή μπορούμε να χρησιμοποιήσουμε μια εικόνα για να δημιουργήσουμε ένα πλαίσιο.
- **border**. Η ιδιότητα αυτή αποτελεί ουσιαστικά συντόμευση για να ορίσουμε με μία μόνο ιδιότητα όλες τις παραπάνω (όπως είναι η ιδιότητα *font* για τα χαρακτηριστικά κειμένου).

  Παραδείγματα χρήσης:

  - *border: 10px solid #999999;*
  - *border: 2px dashed;*

  Αν δεν ορίσουμε κάποια επιμέρους ιδιότητα όταν χρησιμοποιούμε την συγκεντρωτική ιδιότητα *border*, ο browser «υποθέτει» την προεπιλεγμένη τιμή για αυτή, όχι ότι δεν ορίστηκε. Αυτό σημαίνει ότι αν γράψουμε για παράδειγμα:

  - *border-color: orange;*
  - *border: 2px dashed;*

  τότε το περίγραμμα μας δεν θα είναι πορτοκαλί, τουλάχιστον όχι σε όλους τους browsers (εκτός αν είναι πορτοκαλί και η ιδιότητα color), μιας και αφού δεν ορίσαμε στη *border* και την *border-color*, τότε ο browser «υποθέτει» την προεπιλεγμένη τιμή

για αυτή την ιδιότητα, και μιας και η *border* βρίσκεται κάτω από την *border-color*, έχει μεγαλύτερη προτεραιότητα, οπότε επικρατεί για τη *border-color* η προεπιλεγμένη τιμή.

- **border-x**. Κάθε μια από τις ιδιότητες που αναφέρθηκαν παραπάνω, αποτελείται από τέσσερις επιμέρους ιδιότητες, οι οποίες μας επιτρέπουν να ορίσουμε διαφορετικές τιμές για την ιδιότητα αυτή όσον αφορά το πάνω/δεξί/κάτω/αριστερό περίγραμμα. Κάθε ιδιότητα της μορφής border-κάτι αντιστοιχεί σε τέσσερις ιδιότητες της μορφής border-top-κάτι, border-right-κάτι, border-bottom-κάτι, border-left-κάτι. Επίσης, κάθε ιδιότητα από αυτές που αναφέρθηκαν παραπάνω δέχεται και τιμές της μορφής *[top] [right] [bottom] [left]* (ένας εύκολος μνημονικός κανόνας για τη σειρά των τιμών είναι ότι η σειρά αυτή είναι σύμφωνη με την πορεία των δεικτών του ρολογιού) ή της μορφής *[top,bottom] [right,left]* αντί για μόνο μια τιμή που θα εφαρμοστεί και για τις τέσσερις πλευρές του περιγράμματος.

  Για παράδειγμα, για να ορίσουμε ένα γκρι συμπαγές περίγραμμα με πάχος αριστερά 1px, δεξιά 2px, πάνω 3px και κάτω 4px θα μπορούσαμε να γράψουμε είτε:

  - border-left-width: 1px;
  - border-right-width: 2px;
  - border-top-width: 3px;
  - border-bottom-width: 4px;
  - border-style: gray;
  - border-style: solid;

  είτε:

  - border-width: 3px 2px 4px 1px;
  - border-style: gray;

- border-style: solid;

2.8.Πρόσθετες Ιδιότητες CSS3

Η CSS3 έχει εισαγάγει πληθώρα νέων δυνατοτήτων για τους σχεδιαστές ιστοσελίδων που οι προηγούμενες εκδόσεις δεν διέθεταν. Ήδη έχουν αναφερθεί σε διάφορα σημεία παραπάνω μερικές στον σχεδιασμό περιγραμμάτων και κειμένων, οι μεγάλες καινοτομίες όμως μπορούν να προσδώσουν μέχρι και το εφέ κίνησης στο περιεχόμενο της σελίδας, μόνο με τη χρήση CSS3.

2.8.1.Μετασχηματισμοί

Οι *μετασχηματισμοί* μας δίνουν τη δυνατότητα να μετακινήσουμε, να αλλάξουμε την κλίμακα, να περιστρέψουμε και να εκτείνουμε στοιχεία. Μπορούμε να μεταβάλλουμε τα στοιχεία χρησιμοποιώντας είτε δισδιάστατο (2D) είτε τρισδιάστατο (3D) μετασχηματισμό.

2.8.1.1.2D Μετασχηματισμός

Ο δισδιάστατος μετασχηματισμός διαθέτει πέντε (5) μεθόδους οι οποίες μπορούν να τροποποιήσουν ένα στοιχείο:

- **Translate**. Ένα στοιχείο μπορεί να κινηθεί από την τρέχουσα θέση του προς μια άλλη (στους άξονες X και Y) η οποία δίνεται ως παράμετρος.
- **Rotate**. Ένα στοιχείο μπορεί να περιστραφεί δεξιόστροφα ή αριστερόστροφα.
- **Scale**. Ένα στοιχείο μπορεί να αυξομειώσει το μέγεθός του βάσει των παραμέτρων που δίνονται.
- **Skew**. Ένα στοιχείο μπορεί να περιστραφεί κατά μια γωνία που δίνεται ως παράμετρος.
- **Matrix**. Συνδυάζει όλες τις προηγούμενες ιδιότητες σε μία.

2.8.1.2. 3D Μετασχηματισμός

Ο τρισδιάστατος μετασχηματισμός διαθέτει δύο (2) μεθόδους οι οποίες μπορούν να τροποποιήσουν ένα στοιχείο.

- **rotateX**. Με την μέθοδο αυτή ένα αντικείμενο μπορεί να περιστραφεί γύρω από τον άξονα X βάσει μιας δοθείσας γωνίας.
- **rotateY**. Όμοια με τη rotate, με την παραλλαγή ότι η περιστροφή γίνεται γύρω από τον άξονα Y.

2.8.2.Μεταβάσεις

Στη CSS3 μπορούμε να εισάγουμε ένα εφέ όταν μεταβαίνουμε από ένα στυλ μορφοποίησης σε ένα άλλο χωρίς να κάνουμε χρήση Flash ή Javascript. Η *μετάβαση* (transition) έχει τέσσερις ιδιότητες:

- **transition-property**. Συγκεκριμενοποιεί το όνομα της CSS ιδιότητας πάνω στην οποία εφαρμόζεται η transition.
- **transition-duration**. Η χρονική διάρκεια της μετάβασης.
- **transition-timing-function**. Δηλώνει πως υπολογίζεται η ταχύτητα μετάβασης.
- **transition-delay**. Η χρονική στιγμή στην οποία αρχίζει η μετάβαση.

2.8.3.Κινούμενες εικόνες

Στη CSS3 μπορούμε να δημιουργήσουμε *κινούμενες εικόνες* (animations), οι οποίες μπορούν να αντικαταστήσουν animations που έχουν γίνει με Flash ή JavaScripts σε υπάρχουσες ιστοσελίδες. Η δημιουργία animations στην CSS3 γίνεται βάσει ενός κανόνα *@keyframes* ο οποίος καθορίζει ένα στυλ CSS στο εσωτερικό του, καθώς και το animation που αλλάζει σταδιακά από το τρέχον στυλ στο νέο στυλ. Όταν ένα animation δημιουργείται από τον κανόνα *@keyframe*, θα πρέπει να συνδεθεί με έναν επιλογέα, διαφορετικά η κίνηση δε θα έχει καμία επίδραση στο αντικείμενο.

2.9.Προτεραιότητα

Πολλές φορές για το ίδιο στοιχείο μπορεί να αναφέρονται αρκετοί κανόνες CSS, ορισμέν οι εκ των οποίων να περιέχουν ιδιότητες που «συγκρούονται» (conflict).

Για παράδειγμα, έστω ότι έχουμε στο αρχείο CSS μας τους παρακάτω κανόνες:

```
.testclass
{
 color: red;
}

div
{
 color: gray;
}
```

και στην HTML μας έχουμε κάποιο στοιχείο:

* test text *

Σε αυτή την περίπτωση το κείμενο θα είναι κόκκινο, διότι ο επιλογέας .testclass είναι πιο «συγκεκριμένος» απ'ότι ο επιλογέας div.

Ωστόσο, αν στο αρχείο CSS μας είχαμε τους κανόνες:

```
.testclass
{
 color: red;
}

.testclass2
{
 color: gray;
}
```

και στην HTML μας είχαμε το στοιχείο:

* test text *

τότε το κείμενο θα ήταν γκρι, διότι η δήλωση της κλάσης testclass2 διαβάστηκε τελευταία από τ ον browser, οπότε έχει μεγαλύτερη προτεραιότητα.

Γενικά οι κανόνες προτεραιότητας στο CSS είναι αρκετά περίπλοκοι. Οι βασικότεροι είν αι οι εξής:

- Οι ιδιότητες CSS που βρίσκονται μέσα στην ιδιότητα style ενός στοιχείου έχουν πάντ α μεγαλύτερη προτεραιότητα από οποιεσδήποτε άλλες (εκτός από όσες χρησιμοποιού ν !important, το οποίο θα αναλυθεί παρακάτω).
- Οι κανόνες CSS που βρίσκονται μέσα στα tags *<style>...</style>* έχουν μεγαλύτερη προτεραιότητα από αυτούς που βρίσκονται σε ένα εξωτερικό αρχείο CSS.
- Όταν δύο ή περισσότερα αρχεία CSS συμπεριλαμβάνονται στην ίδια ιστοσελίδα, οι κ ανόνες αυτού που αναφέρθηκε τελευταίο στη σελίδα έχουν μεγαλύτερη προτεραιότητ α από αυτούς του πρώτου.
- Ένας κανόνας CSS έχει μεγαλύτερη προτεραιότητα από όσους κανόνες CSS βρίσκον ται γραμμένοι πιο πάνω από αυτόν, εφόσον δε συντρέχει άλλος λόγος για να μετρήσει παραπάνω κάποιος από αυτούς.
- Οι κανόνες με επιλογέα του τύπου *.όνομα_κλάσης* έχουν μεγαλύτερη προτεραιότητα από κανόνες με επιλογέα του τύπου στοιχείο, μιας και είναι πιο συγκεκριμένοι.
- Οι επιλογείς του τύπου *#όνομα_id* έχουν μεγαλύτερη προτεραιότητα από επιλογείς του τύπου *.όνομα_κλάσης*, μιας και είναι πιο συγκεκριμένοι.

Σε κάθε περίπτωση, αν επιθυμούμε μια ιδιότητα ενός κανόνα να έχει μεγαλύτερη προτερ αιότητα από αυτή που της αντιστοιχεί, μπορούμε πριν το *;* να γράψουμε *!important*.

Για παράδειγμα, στο πρώτο παράδειγμα αν είχαμε γράψει στο αρχείο CSS μας:

```
.testclass
{
        color: red;
}

div
{
       color: gray !important;
}
```

τότε τα γράμματα στο στοιχείο:

* test text *

θα ήταν γκρι, αντί για κόκκινα.

Ωστόσο γενικά πρέπει να χρησιμοποιείται αυτή η τεχνική σε μεμονωμένες περιπτώσεις, μιας και μπορεί να οδηγήσει σε αρκετά περίπλοκο κώδικα CSS και να δημιουργήσει προβλήματα στην αποσφαλμάτωση για το χρήστη.

2.10.Κληρονομικότητα

Στην HTML, κάθε στοιχείο αποτελεί *απόγονο* (descendant) κάποιου άλλου στοιχείου, εκτός από το στοιχείο *<html>* που αποτελεί *πρόγονο* (anchestor) για κάθε στοιχείο στη σελίδα μας. Το κάθε στοιχείο μπορεί να έχει και δικούς του απογόνους, εκτός αν πρόκειται για *self-closing* στοιχεία, όπως το *img* ή το *input*, τα οποία δεν διαθέτουν tag κλεισίματος (end tag). Έτσι διαμορφώνεται μια ιεραρχία η οποία αποτελεί το *DOM tree (Document Object Model tree)*.

Όταν εφαρμόζουμε μια ιδιότητα CSS σε ένα στοιχείο, μπορεί να κληρονομηθεί και στα στοιχεία-απογόνους του ή μπορεί και να μην κληρονομηθεί, αναλόγως την ιδιότητα. Όσον αφορά τις ιδιότητες που αναφέραμε παραπάνω, αυτές που κληρονομούνται και στα στοιχεία απογόνους είναι τα ακόλουθα:

- *color*
- *font-size*
- *font-family*
- *font-style*
- *font-weight*
- *text-align*
- *text-decoration*

Φυσικά αν ορίσουμε μέσω κάποιου κανόνα CSS μια διαφορετική τιμή για μια συγκεκριμένη ιδιότητα, θα εφαρμοστεί αυτή η τιμή και όχι η κληρονομημένη (inherited). Όταν επιθυμούμε ένα στοιχείο να αποκτήσει για μια ιδιότητα την τιμή της ιδιότητας αυτής του γονέα του, ενώ αυτή κανονικά δεν κληρονομείται, μπορούμε ως τιμή της ιδιότητας για το στοιχείο αυτό να ορίσουμε την *inherit*.

Κεφάλαιο 3

3. CSS Τεχνικές & Καλές Πρακτικές

Η CSS περιλαμβάνει πληθώρα αντικειμένων στα οποία θα πρέπει ο σχεδιαστής να εντρυφήσει (στοιχεία, κανόνες, ιδιότητες, τιμές). Μέχρι τώρα αναφερθήκαμε στην επεξήγηση των διαφόρων χαρακτηριστικών της CSS και της HTML, προκειμένου να μπορούμε να δομήσουμε μία απλή ιστοσελίδα. Ωστόσο, στο σύγχρονο σχεδιασμό ιστοσελίδων αυτή η γνώση δεν είναι αρκετή. Θα πρέπει ο σχεδιαστής να γράφει εύχρηστο, επαναχρησιμοποιήσιμο και αποδοτικό κώδικα, γεγονός το οποίο αποτελεί πρόκληση με μεγάλο βαθμό δυσκολίας. Χρησιμοποιούνται οι καλές πρακτικές ή απλά περιπλέκεται ο κώδικας διαφοροποιώντας την ποιότητα της ιστοσελίδας στους διαφόρους browsers; Η χρήση σχολίων, η παράθεση στοιχείων επικοινωνίας, η επικύρωση του κώδικα και η χρήση προεπεξεργαστών είναι μερικές από τις πρακτικές που ο σχεδιαστής συνίσταται να χρησιμοποιεί κατά τη συγγραφή CSS κώδικα.

3.1.Καλές Πρακτικές Συγγραφής CSS

Σε αυτή την ενότητα θα εξετάσουμε κάποιες πρακτικές που θα πρέπει να χρησιμοποιεί ο σχεδιαστής, προκειμένου τόσο να βελτιώσει την απόδοση του κώδικα, όσο και να συμβάλλει στην επαναχρησιμοποίησή του.

3.1.1.Χρήση CSS Reset

Η αστάθεια κάποιων browsers και οι διαφοροποιήσεις μεταξύ τους είναι από τα πιο σημαντικά προβλήματα των σχεδιαστών. Μορφοποιήσεις όπως τα margins, paddings, ύψος γραμμών, επικεφαλίδες, μέγεθος γραμματοσειρών κ.α., είναι πολύ πιθανό να εμφανίζονται με διαφορετικό τρόπο σε διαφορετικούς browsers, ακόμα και στους πιο διαδεδομένους.

Με τη χρήση ενός reset style sheet αμβλύνονται οι διαφοροποιήσεις μεταξύ των browsers, καθώς ορίζει generic μορφοποιήσεις που μπορούν να επεξεργαστούν, αλλά και να επεκταθούν. Στο διαδίκτυο υπάρχουν reset style sheets που είναι διαδεδομένα (MeyerWeb, Normalize) και η πρόσβασή σε αυτά και η χρήση τους είναι ελεύθερη, ωστόσο μπορεί ο καθένας να δημιουργήσει το δικό του, ανάλογα με τις ανάγκες και τις απαιτήσεις του.

3.1.2.Παράθεση Στοιχείων Επικοινωνίας

Η παράθεση ενός τίτλου του CSS αρχείου, του συγγραφέα του, keywords, περιγραφής, link κ.ο.κ. αποτελεί πλέον αναγκαία πρακτική των σχεδιαστών. Αυτό είναι ιδιαίτερα χρήσιμο αν αναλογιστεί κανείς τον τρόπο που δημιουργούνται σήμερα πολλές ιστοσελίδες.

```
/*
Theme Name: Simple Parallax Website
Description: Simple Parallax Scrolling Effect
Author: Samuel Norton
Author URI: http://www.1stwebdesigner.com/
Tags: Parallax, Website
*/
```

Υπάρχουν σχεδιαστές που κατασκευάζουν templates για ιστοσελίδες συγκεκριμένης πλατφόρμας (Joomla!, Wordpress, Plone, κ.α.) και τα ανεβάζουν στο διαδίκτυο. Πρόσβαση σε αυτά έχει κανείς είτε ελεύθερα είτε έναντι αμοιβής, προκειμένου να χρησιμοποιήσει το template στο δικό του site, γεγονός που σημαίνει ότι πολλές φορές είναι απαραίτητη η επεξεργασία του αρχείου CSS όχι από το σχεδιαστή, ούτε από το χρήστη, αλλά από έναν τρίτο.

3.1.3.Οργάνωση Στοιχείων (από πάνω προς τα κάτω)

Συνήθως οι αρχάριοι σχεδιαστές τοποθετούν τα στοιχεία στον κώδικα CSS σύμφωνα με αυτό που επιθυμούν πρώτο. Ωστόσο, αυτό δεν αποτελεί καλή πρακτική για τη δόμηση CSS κώδικα, καθώς δυσκολεύει την εύρεση τμημάτων κώδικα που θέλουμε σε αυτό. Η ταξινόμηση από τις μορφοποιήσεις (όπως body, h1, p, a, κ.τ.λ.) ακολουθούμενα από την κεφαλίδα προς στο

υποσέλιδο, δημιουργεί μία πιο λογική δομή, μέσα στην οποία εύκολα κανείς μπορεί να εντοπίσει αυτό που ψάχνει.

Για παράδειγμα:

```
/****** General Styles *********/

body {...}
h1, h2, h3 {..}
p {...}
a {...}

/****** Header Style *********/
#header {...}

/****** Navigation Style *********/
#nav {...}

/****** Footer Style *********/
#footer {...}
```

3.1.4.Σμίκρυνση CSS αρχείων με CSS Compressors

Υπάρχει η δυνατότητα σμίκρυνση του CSS αρχείου μέσω ενός εργαλείου που λέγεται CSS Compressor. Τέτοια εργαλεία είναι το CSS Compressor και το CSS Compressor & Minifier.

Αυτό που κάνει αυτό το εργαλείο ουσιαστικά, είναι η απομάκρυνση των κενών, των αλλαγών γραμμών και του πλεονάζοντα CSS κώδικα. Το θετικό σε αυτή την πρακτική φυσικά είναι η γρηγορότερη φόρτωση του στυλ από τον browser.

3.1.5.Ομαδοποίηση IDs & Κλάσεων κάτω από Κοινό Στοιχείο

Εάν έχουμε ένα στοιχείο που περιέχει διάφορα IDs και κλάσεις, είναι καλή πρακτική να τα ομαδοποιήσουμε, προκειμένου του είναι οργανωμένα με τέτοιο τρόπο, ώστε η αποσφαλμάτωση να μην απαιτεί ιδιαίτερο χρόνο όταν χρειαστεί.

Για παράδειγμα, εάν έχουμε μία κλάση *container* που περιέχει ένα div tag με ID *logo* και άλλο ένα div tag με ID *icons*:

```
<div id="logo"></div>
<div id="tagline">< /div></div>
```

μπορούμε να ομαδοποιήσουμε τον CSS κώδικά μας ως ακολούθως:

```
.container {width: 960px; margin: 0;  padding: 0;}
.container #logo {font-family: Arial, sans-serif;  font-size: 30px;  color:
red;}
.container #tagline {font-family: Verdana; font-size: 10px;}
```

3.1.6.Αποφυγή Ενσωματωμένου Κώδικα Στυλ (HTML)

Όπως αναφέραμε παραπάνω, ένα από τα σημαντικά πλεονεκτήματα που προσφέρει η γλώσσα CSS, είναι ο διαχωρισμός του περιεχομένου από τη μορφοποίηση. Αυτό όμως δεν αναιρεί τη δυνατότητα να καθορίζουμε τη μορφοποίηση και μέσα από την HTML. Ωστόσο, η μορφοποίηση μέσα από την HTML δυσκολεύει ιδιαίτερα τη χρήση και συντήρηση της ιστοσελίδας, καθώς για κάθε ενημέρωση που θα θελήσουμε να κάνουμε, θα πρέπει να ενημερώσουμε κάθε στοιχείο ξεχωριστά, αντί μίας εντολής που πρέπει με τη χρήση CSS.

Για παράδειγμα ο παρακάτω κώδικας HTML:

```
<div style="background: red;">
Test text
</div>
```

θα πρέπει να γίνει:

```
/* CSS code */
.infobox {
   background: red;
}

/* HTML code */
<div class="infobox">
Test text
</div>
```

3.1.7.Χρήση Σχολίων για Τμηματοποίηση του κώδικα

Μία ακόμα καλή πρακτική είναι η χρήση σχολίων σε κάθε τμήμα κώδικα CSS. Με αυτό τον τρόπο θα διευκολυνθεί ο εντοπισμός συγκεκριμένων τμημάτων κώδικα όταν θα αναζητούμε κάποιο σφάλμα.

Για παράδειγμα:

```
/****** General Styles *********/

body{
    margin: 0;
    padding: 0;
    width: 100%;
}

h1, h2, h3 {
    font-family: Arial, sans-serif;
    font-weight: normal;
    font-size: 55px;
    text-align: center;
    color: #fff;
    margin: 0;
    padding: 0;
}
```

3.1.8.Κατάλληλη Ονομασία Μεταβλητών

Η χρήση της κατάλληλης ονοματολογίας στα IDs και τις κλάσεις θα καταστήσει πολύ ποιοτικότερο τον κώδικα. Με αυτό τον τρόπο θα είναι πολύ πιο εύκολη και γρήγορη διαδικασία η προσθήκη στοιχείων ή ακόμα και ο επανασχεδιασμός της ιστοσελίδας.

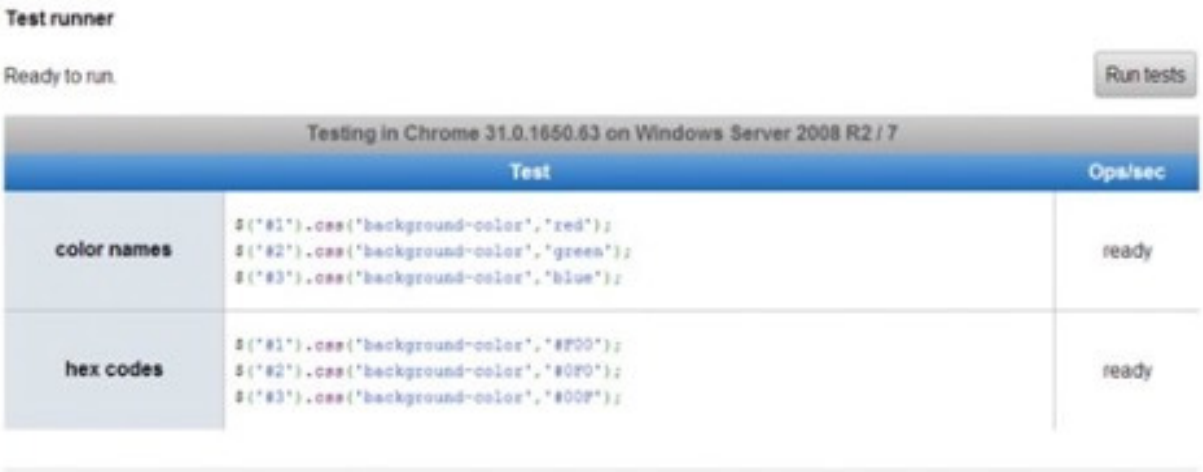
Για παράδειγμα, ονοματίζοντας μία κλάση *title-red* δε θα έχει κάποιο νόημα εάν αποφασίσουμε να αλλάξουμε το χρώμα του τίτλου. Συνεπώς η κατάλληλη ονομασία εν προκειμένω θα ήταν *title*.

Συνίσταται, λοιπόν, η χρήση ονομάτων μεταβλητών βάσει της χρήσης τους και όχι των ιδιοτήτων τους, όπως το χρώμα ή το μέγεθος του στοιχείου στο οποίο αναφέρονται, εφόσον αυτά είναι μεταβλητά.

3.1.9.Χρώμα: Χρήση hex code αντί για Όνομα

Σύμφωνα με έγκυρες μετρήσεις απόδοσης, με τον καθορισμό των χρωμάτων με τη δεκαεξαδική μορφή του (αναφέραμε παραπάνω ότι υπάρχουν τρεις τρόποι αναφοράς χρώματος, όνομα χρώματος, RGB και hex value) ο κώδικας τρέχει πιο γρήγορα.

Συγκεκριμένα, στις μετρήσεις του Sean Connon, Senior Web Developer της Alien Creations Inc, ο κώδικας έτρεξε πιο γρήγορα στο 80% των πειραμάτων.



3.1.10.Χρήση Κλάσεων Γενικού Σκοπού

Πολλές φορές καταλήγουμε να χρησιμοποιούμε τις ίδιες μορφοποιήσεις επανειλημμένα. Σε τέτοιες περιπτώσεις, θα μπορούσαμε αντί να καθορίζουμε τη μορφοποίηση για κάθε ID, να δημιουργήσουμε κλάσεις γενικού σκοπού και να τις εντάξουμε στα IDs ή σε άλλες κλάσεις.

Για παράδειγμα, πολλές φορές μπορεί να χρησιμοποιούμε τις μορφοποιήσεις *float:right* και *float:left* επανειλημμένα. Το ίδιο αποτέλεσμα μπορούμε να έχουμε προσθέτοντας τις κλάσεις *.left* και *.right* στον κώδικα CSS και να τις αναφέρουμε στα στοιχεία του κώδικα HTML:

```
/* CSS code */
.left
{
      float: left
}
.right
{
      float: right
}

 /* HTML code */
<div id="coolbox" class="left">...</div>
```

3.1.11.Χρήση CSS Vendor Προθέματος για κάθε browser

Σε πολλές μορφοποιήσεις, ιδιαίτερα σε αυτές που εισήχθησαν στη CSS3, κάθε browser έχει το δικό του προσδιορισμό (specification). Γι' αυτό το λόγο χρησιμοποιούνται το *προθέματα περιηγητών (browser prefixes)*, προκειμένου να εξασφαλίσουν ότι ο browser υποστηρίζει τη συγκεκριμένη δυνατότητα ή μορφοποίηση που θέλουμε. Είναι συχνό φαινόμενο να υπάρχουν σφάλματα, μόνο και μόνο επειδή έχει παραληφθεί η αναφορά σε συγκεκριμένους browsers.

Τα προθέματα αυτά είναι:

- **Chrome**: *-webkit-*
- **Firefox**: *-moz-*
- **iOS**: *-webkit-*

- **Opera**: *-o-*
- **Safari**: *-webkit-*

Για παράδειγμα, εάν θέλουμε να προσθέσουμε τη δυνατότητα της CSS3 transition στον κώδικά μας, θα χρησιμοποιήσουμε τον όρο *transition* μαζί με τα προθέματα για κάθε browser:

```
-webkit-transition: all 1s ease;
-moz-transition: all 1s ease;
-ms-transition: all 1s ease;
-o-transition: all 1s ease;
transition: all 1s ease;
```

1.CSS3 transition

-webkit-transition: background-color 500ms ease-out 1s;

-moz-transition: background-color 500ms ease-out 1s;

-o-transition: background-color 500ms ease-out 1s;

transition: background-color 500ms ease-out 1s;

2.CSS background

background: red; /* For browsers that do not support gradients */

background: -webkit-linear-gradient(red, yellow); /* For Safari 5.1 to 6.0 */

background: -o-linear-gradient(red, yellow); /* For Opera 11.1 to 12.0 */

background: -moz-linear-gradient(red, yellow); /* For Firefox 3.6 to 15 */

background: linear-gradient(red, yellow); /* Standard syntax */

3.CSS3 border

```
border: 10px solid transparent;
  padding: 15px;
  -webkit-border-image: url(border.png) 30 stretch; /* Safari 3.1-5 */
  -o-border-image: url(border.png) 30 stretch; /* Opera 11-12.1 */
  border-image: url(border.png) 30 stretch;
```

3.1.12.Χρήση πολλαπλών CSS αρχείων για μεγάλο Κώδικα

Ανάλογα με την πολυπλοκότητα του σχεδιασμού και του μεγέθους της ιστοσελίδας, είναι πολλές φορές πιο εύχρηστο να δημιουργούμε πολλά μικρότερα CSS αρχεία αντί για ένα μεγάλο που περιέχει το σύνολο του κώδικα. Εκτός του ότι με αυτό τον τρόπο καθίσταται πιο εύκολη η διαχείριση της μορφοποίησης από το σχεδιαστή, η χρήση πολλαπλών αρχείων CSS επιτρέπει να παραλείψουμε τη φόρτωσή τους σε σελίδες που δεν είναι αναγκαίο.

Για παράδειγμα, έστω ότι έχουμε μία εφαρμογή μηχανής αναζήτησης στην ιστοσελίδα μας, η οποία (εφαρμογή) χρησιμοποιεί συγκεκριμένη μορφοποίηση για την εμφάνιση των αποτελεσμάτων της αναζήτησης. Αντί να συμπεριλάβουμε αυτά τα στοιχεία στον κύριο κώδικα CSS, μπορούμε να δημιουργήσουμε ένα αρχείο *search.css* και το αρχείο αυτό θα συνδέεται μόνο με τις σελίδες που θα εμφανίζουν τα αποτελέσματα της αναζήτησης.

3.1.13.Επικύρωση του Κώδικα

Χρησιμοποιώντας *επικυρωτές (validators)*, όπως τον W3C CSS Validator, μπορούμε να μάθουμε εάν ο CSS κώδικάς μας είναι ορθά δομημένος. Επιπλέον, μπορούμε να μάθουμε εάν υπάρχει κάποιο σφάλμα στον κώδικα, γεγονός που μπορεί να εξοικονομήσει πολύ χρόνο σε σχέση με τον να προσπαθούσαμε να εντοπίσουμε εμείς τα σφάλματα.

3.2.Δημοφιλείς Τεχνικές Συγγραφής CSS

Σε αυτή την ενότητα θα εξετάσουμε κάποιες από τις πιο δημοφιλείς τεχνικές συγγραφής CSS κώδικα που χρησιμοποιούν οι περισσότεροι σχεδιαστές. Πρόκειται ουσιαστικά για τεχνικές

που δημιουργούν την «ψευδαίσθηση» του αποτελέσματος που επιθυμούμε μέσα από τους κανόνες και τις δυνατότητες της CSS.

3.2.1.CSS Sprites

Ένας τρόπος για να γίνει το site μας πιο γρήγορο μειώνοντας τα HTTP requests είναι να ενώσουμε τα αρχεία. Για τις εικόνες που χρησιμοποιούνται μέσα σε ένα site, μπορούν να χρησιμοποιηθούν CSS Sprites, όπου όλα τα εικονίδια και φόντα που χρησιμοποιούνται σε ένα site σερβίρονται σαν μια εικόνα, και στην συνέχεια με τις ιδιότητες background-image και background-position της CSS, εμφανίζουμε μόνο ένα κομμάτι της συνολικής εικόνας μας.

Σκοπός είναι η δημιουργία ενός μενού πλοήγησης το οποίο απαιτεί πολλή ώρα για να φορτωθεί και δεν χρειάζεται καθόλου JavaScript για το εφέ στην ενέργεια hover. Χρησιμοποιείται επίσης και γενικότερα όταν θέλουμε την εμφάνιση διαφορετικής εικόνας (συνήθως διαφορετική εκδοχή της εικόνας, διαφορετικά χρώματα) όταν τοποθετούμε το ποντίκι πάνω σε αυτή (*onmouseover*).

Η τεχνική βασίζεται στη φόρτωση μίας εικόνας και αποθήκευσή της στη μνήμη γραφικών, προκειμένου να φορτώνεται κάθε φορά ένα κομμάτι της. Αυτή η τεχνική είναι αρκετά πιο γρήγορη από τη φόρτωση κάθε εικόνας ξεχωριστά όταν πρέπει να εμφανιστεί.

Για παράδειγμα, έστω ότι έχουμε τις τρεις σημαίες συνδυασμένες σε μία:

προσθέτουμε τον παρακάτω κώδικα στο CSS αρχείο:

```
.flags-canada, .flags-mexico, .flags-usa {
  background-image: url('../images/flags.png');
  background-repeat: no-repeat;
}

.flags-canada {
  height: 128px;
  background-position: -5px -5px;
}

.flags-usa {
  height: 135px;
  background-position: -5px -143px;
}

.flags-mexico {
  height: 147px;
  background-position: -5px -288px;
}
```

και όταν θέλουμε να εμφανίσουμε μία από τις τρεις εικόνες, αναφερόμαστε στην αντίστοιχη κλάση (για οποιαδήποτε χρήση και αν θέλουμε αυτές τις εικόνες).

3.2.2.CSS Rounded Corners

Με την τεχνική *rounded corners*, μπορούμε να εμφανίσουμε μία εικόνα με στρογγυλεμένες γωνίες. Ο λόγος που είναι τόσο διαδεδομένη αυτή η τεχνική, είναι γιατί απαλλάσσει το σχεδιαστή από τη διαδικασία επεξεργασίας της εικόνας, η οποία δεν αλλάζει εύκολα, με γραφιστικά εργαλεία. Με CSS3 μπορεί να υπάρχουν πιο πολλές δυνατότητες, όμως με CSS γίνεται πιο γρήγορα και θα τα προτιμήσει ο πελάτης.

Για παράδειγμα, εάν θέλουμε γωνίες με ακτίνα κύκλου (με αυτό τον τρόπο υπολογίζεται ο τρόπος που θα στρογγυλοποιηθεί η εικόνα) 10 pixels για την παρακάτω εικόνα:

θα γράψουμε:

```
border-radius: 10px;
```

και η εικόνα θα γίνει:

ή εάν θέλουμε για κάθε γωνία ξεχωριστά:

```
border-radius-topleft: 10px;
border-radius-topright: 20px;
border-radius-bottomright: 30px;
border-radius-bottomleft: 0;
```


```
.btn {
  background: #3498db;
  background-image: -webkit-linear-gradient(top, #3498db, #2980b9);
  background-image: -moz-linear-gradient(top, #3498db, #2980b9);
  background-image: -ms-linear-gradient(top, #3498db, #2980b9);
  background-image: -o-linear-gradient(top, #3498db, #2980b9);
```

```
 background-image: linear-gradient(to bottom, #3498db, #2980b9);
 -webkit-border-radius: 20;
 -moz-border-radius: 20;
 border-radius: 20px;
 font-family: Arial;
 color: #ffffff;
 font-size: 16px;
 padding: 10px 20px 10px 20px;
 text-decoration: none;
}
```

3.2.3.Image Repla

Η τεχνική *CSS image replacement* είναι μία τεχνική αντικατάστασης ενός στοιχείου κειμένου (συνήθως μία ετικέτα επικεφαλίδας) με μία εικόνα. Ένα παράδειγμα είναι η εισαγωγή λογότυπου σε μία ιστοσελίδα. Μπορεί ο σχεδιαστής να θέλει να χρησιμοποιήσει το tag <*h1*> με κείμενο για να βελτιστοποιήσει την αναζήτηση της ιστοσελίδας από τις αντίστοιχες μηχανές, αλλά ο χρήστης να βλέπει το λογότυπο.Για καλύτερη αναζήτηση χρησιμοποιούμε λέξεις-κλειδιά.

Για παράδειγμα:

```
<!-- HTML code -->
<h1 id="technique-one">
  <span>CSS-Tricks</span>
</h1>


/* CSS code */
h1#technique-one {
  width: 250px;
  height: 25px;
  background-image: url(logo.gif);
}
h1#technique-one span {
  display: none;
}
```

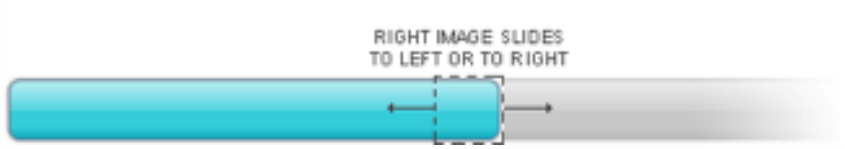
3.2.4.Sliding Doors

Η τεχνική *sliding doors* χρησιμοποιείται κυρίως για τη δημιουργία μενού με καρτέλες. Πριν την εμφάνιση αυτής της μεθόδου, ο τρόπος δημιουργίας τέτοιων μενού ήταν η τοποθέτηση εικόνων δίπλα δίπλα σε κελιά ενός πίνακα. Έπειτα, με την εμφάνιση της CSS, χρησιμοποιώντας λίστες και ιδιότητες της CSS με λίγες εικόνες μπορούσαμε να δημιουργήσουμε μία «ελαφριά», γρήγορη ιστοσελίδα, έχοντας όμως τους περιορισμούς των δυνατοτήτων της CSS.

Με την έλευση της CSS3, συνδυάστηκαν και τα δύο παραπάνω, λίγες εικόνες και επαναχρησιμοποιήσιμες μέθοδοι λόγω της CSS για ένα καλαίσθητο μενού. Η τεχνική είναι απλή και λύνει το πρόβλημα του σταθερού πλάτους των εικόνων για τις καρτέλες. Κατά τη δημιουργία των γραφικών , δηλαδή των καρτελών εν προκειμένω, δημιουργούμε μία καρτέλα πολύ πιο πλατιά απ' ό,τι χρειαζόμαστε. Στην τεχνική αυτή χρησιμοποιούνται δύο εικόνες η μία κολλημένη δίπλα στην άλλη. Η μία είναι πολύ πλατιά, πάνω στην οποία εμφανίζεται το κείμενο και η άλλη «κλείνει» την άλλη πλευρά. Χρησιμοποιώντας την ιδιότητα *background-image* κρύβουμε το κομμάτι που δε χρειαζόμαστε από τη μία εικόνα και η άλλη εικόνα καθορίζει το τέλος (εξού και η ονομασία).

Έχοντας λοιπόν τις δύο παρακάτω εικόνες:

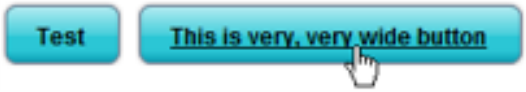
δημιουργούμε αυτή:



ο αντίστοιχος κώδικας HTML και CSS:

```
<!-- HTML code -->
<a class="button" href="#"><span>Submit</span></a>


/* CSS code */

a.button {
/* Background rigth image */
background: transparent url('button_right.png') no-repeat scroll top right;
display: block;
float: left;
height: 32px;
margin-right: 6px;
padding-right: 20px;
/* Text */
text-decoration: none;
color: #000000;
font-family: Arial, Helvetica, sans-serif;
font-size:12px;
font-weight:bold;

}

a.button span {
/* Background left image */
background: transparent url('button_left.png') no-repeat;
display: block;
line-height: 22px;
padding: 7px 0 5px 18px;
}

a.button:hover span{
text-decoration:underline;
}
```

και το αποτέλεσμα:



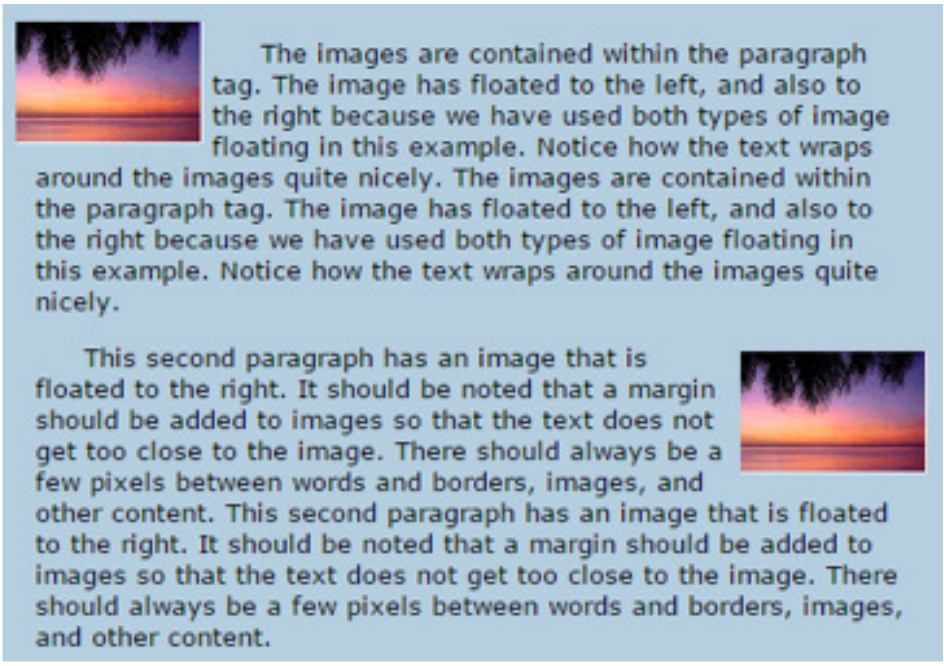
3.2.5.Αναδίπλωση Κειμένου σε Εικόνα

Μία μέθοδος που είναι συνυφασμένη με την ιδιότητα *float* της CSS (κλασσική και στην CSS2) και αποτελεί μία από τις πιο διαδεδομένες τεχνικές εμφάνισης εικόνων μέσα σε ένο κείμενο. Η μη εφαρμογή του *float*, έχει σαν αποτέλεσμα την εμφάνιση του κειμένου πάνω στην εικόνα, σαν η εικόνα να μην υπήρχε. Αντίθετα, ορίζοντας την τιμή του *float* σε *right* ή *left*, η εικόνα εμφανίζεται στην αντίστοιχη θέση και το κείμενο περιβάλει την εικόνα.

Ένα παράδειγμα:

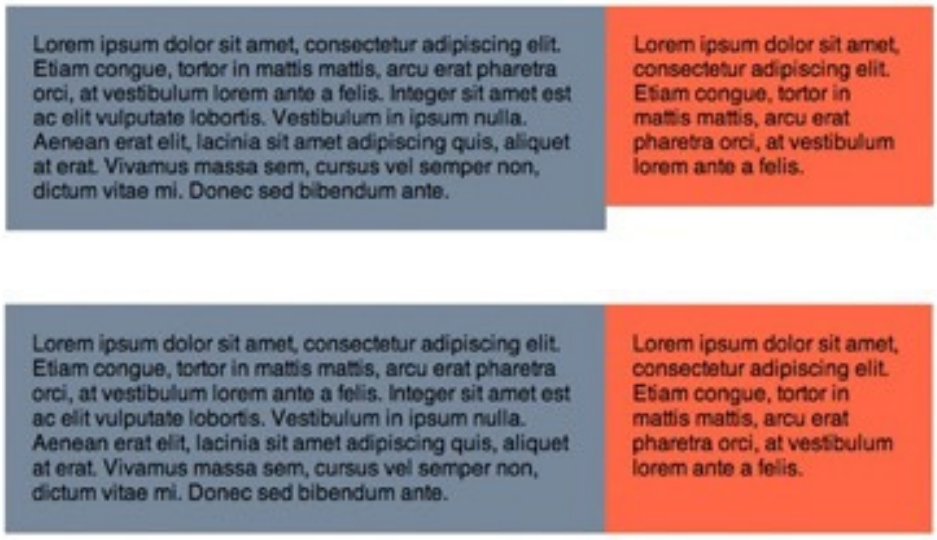
```
<!-- HTML code -->
<body>
    <img src="sunset.gif" class="floatLeft">
    <p>The images are contained with...</p>

    <img src="sunset.gif" class="floatRight">
    <p>This second paragraph has an...</p>
</body>


/* CSS code */
img.floatLeft {
    float: left;
    margin: 4px;
}
img.floatRight {
    float: right;
    margin: 4px;
}
```

με αποτέλεσμα:



3.2.6.Equal Height

Κατά κανόνα, μία ιστοσελίδα αποτελείται από το κυρίως μέρος και μία πλάγια στήλη. Είναι σύνηθες το ύψος του κυρίως μέρους που καθορίζεται από το περιεχόμενό του, να είναι πιο μεγάλο ή πιο μικρό από το ύψος που αντιστοιχεί στο περιεχόμενο της πλάγιας στήλης. Για αισθητικούς λόγους λοιπόν, πολλές φορές θέλουμε αυτά τα δύο ύψη να ταυτίζονται, δηλαδή να μετατρέψουμε αυτό:



σε αυτό:

Ένας τρόπος να το επιτύχουμε αυτό είναι να ορίζουμε τα στοιχεία να συμπεριφέρονται σαν πίνακες. Έτσι, θα έχουμε τον παρακάτω κώδικα:

```
<!-- HTML code -->
<div class="container clearfix">
    <div class="main">Lorem ipsum dolor sit amet, consectetur adipiscing
elit.
    Etiam congue, tortor in mattis mattis, arcu erat pharetra orci, at
vestibulum lorem ante a felis.
    Integer sit amet est ac elit vulputate lobortis. Vestibulum in ipsum
nulla. Aenean erat elit,
    lacinia sit amet adipiscing quis, aliquet at erat. Vivamus massa sem,
cursus vel semper non,
    dictum vitae mi. Donec sed bibendum ante.</div>

    <div class="sidebar">Lorem ipsum dolor sit amet, consectetur adipiscing
elit.
    Etiam congue, tortor in mattis mattis, arcu erat pharetra orci, at
vestibulum lorem ante a felis.</div>
</div>


/* CSS code */
.main, .sidebar {
    float: none;
    padding: 20px;
    vertical-align: top;
}
.container {
    display: table;
}
.main {
    width: 400px;
    background-color: LightSlateGrey;
    display: table-cell;
}
.sidebar {
    width: 200px;
    display: table-cell;
    background-color: Tomato;
}
```

3.2.7.Σκιές

Με την τεχνική box-shadow, μπορούμε να δημιουργήσουμε το εφέ της σκιάς σε μία εικόνα, ή ακόμα σε ένα κουτί γενικότερα (π.χ. πίνακας). Αυτή η τεχνική, όπως και η rounded corners, είναι ιδιαίτερη γιατί απαλλάσσει το σχεδιαστή από τη διαδικασία επεξεργασίας της εικόνας με γραφιστικά εργαλεία.

Ένα παράδειγμα:

```
.shadow {
  box-shadow: 3px 3px 5px 6px #ccc;
}
```

με αποτέλεσμα:

3.2.8.Διαφάνεια

Ακόμα μία τεχνική που απαλλάσσει από την ανάγκη γραφιστική επεξεργασίας είναι η δυνατότητα της διαφάνειας μέσω της ιδιότητας *opacity*. Ορίζοντας το ποσοστό της ορατότητας που επιθυμούμε, αρχίζει να «εξαφανίζεται» η εικόνα και να εμφανίζεται το background.

Για παράδειγμα, ο παρακάτω κώδικας CSS:

```
div {
  opacity: 0.5;
}
```

ανάλογα με την τιμή του opacity (από 0: απόλυτη διαφάνεια, έως 1: απόλυτη συμπάγεια), θα έχει τα παρακάτω αποτελέσματα:

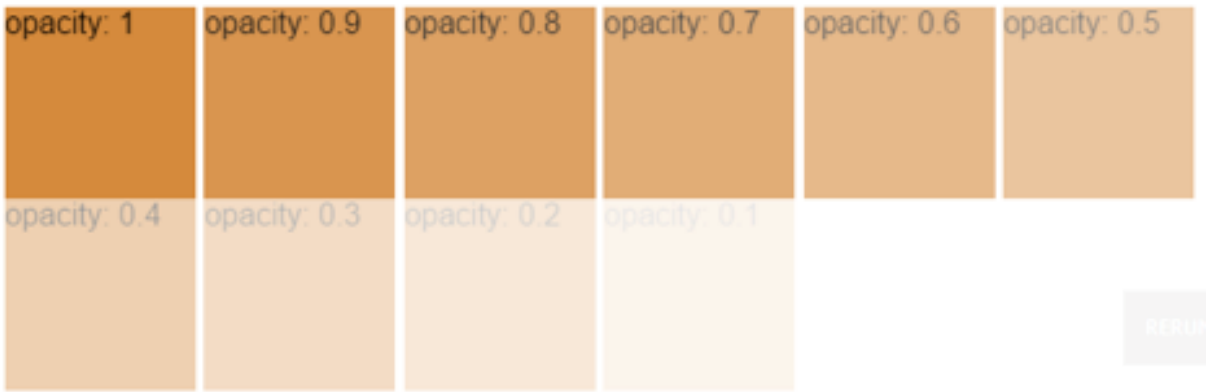
3.3.Προεπεξεργαστές

Έπειτα από αρκετό καιρό ενασχόλησης με HTML και CSS, είναι σύνηθες και λογικό ο προγραμματισμός να γίνει λίγο κουραστικό, με πολλές διαδικασίες να επαναλαμβάνονται συνεχώς. Διαδικασίες τέτοιες αφορούν τους βασικούς κανόνες των δύο αυτών γλωσσών προγραμματισμού, όπως να ανοίγουμε και να κλείνουμε τα tags στην HTML ή να αναζητούμε τις δεκαεξαδικές τιμές των χρωμάτων για τη CSS.

Αυτές οι διαφορετικές διεργασίες, παρόλο που είναι μικρές και θέμα χρόνου (ρουτίνα), επιδρούν αρνητικά στην αποτελεσματικότητα / αποδοτικότητα του προγραμματιστή / σχεδιαστή. Ευτυχώς, αυτές, καθώς και αρκετές άλλες χρονοβόρες ρουτίνες, έχουν καταγραφεί και έχουν αναπτυχθεί λύσεις που σχετίζονται με την προεπεξεργασία αυτών.

Ένας *προεπεξεργαστής (preprocessor)* είναι ένα πρόγραμμα που παίρνει σαν είσοδο ένα τύπο δεδομένων και δίνει σαν έξοδο έναν άλλο τύπο δεδομένων. Στην περίπτωση της CSS, κάποιες από τις πιο διαδεδομένες γλώσσες προεπεξεργαστών είναι η *Sass* και η *Less*, που μετατρέπουν τον κώδικά τους σε κώδικα CSS.

Στην προσπάθεια επίλυσης κάποιων συνηθισμένων προβλημάτων, γλώσσες



προεπεξεργαστών όπως η Sass και η Less βρήκαν τρόπο να ενισχύσουν τον προγραμματισμό σε

CSS, όχι μόνο απομακρύνοντας τις αναποτελεσματικότητες, αλλά και δημιουργώντας νέες τεχνικές για πιο εύκολης και λογική κατασκευή ιστοσελίδων. Η ευρεία χρήση προεπεξεργαστών έχει αυξήσει τις ανάγκες και για διαφορετικά frameworks που τους υποστηρίζουν, με ένα από τα πιο γνωστά να είναι το Compass.

Παρακάτω παρατίθενται κάποια βασικά πλεονεκτήματα της χρήσης προεπεξεργαστών, για να γίνει πιο κατανοητός ο λόγος εδραίωσής τους μεταξύ των προγραμματιστών διαδικτύου:

- Οι προεπεξεργαστές επιτρέπουν τον ορισμό μεταβλητών, με την έννοια ενός «κουτιού» με πληροφορίες, οι οποίες χρησιμοποιούνται σαν τιμές στις CSS. Για παράδειγμα, έστω ότι έχουμε μία απόχρωση του μπλε χρώματος, που θέλουμε να χρησιμοποιήσουμε στην ιστοσελίδα μας. Κανονικά θα πρέπει να χρησιμοποιούμε στον κώδικα CSS τη δεκαεξαδική τιμή του. Όταν θα θέλουμε να αλλάξουμε το χρώμα αυτό, θα πρέπει να αλλάξουμε τη δεκαεξαδική τιμή σε κάθε σημείο του κώδικα που αναφέρεται. Με χρήση προεπεξεργαστών, μπορούμε να ορίσουμε το χρώμα αυτό μία φορά στην αρχή του αρχείου και να αναφέρουμε κάθε φορά τη μεταβλητή. Συνεπώς, όταν θα θέλουμε να αλλάξουμε την τιμή του χρώματος, θα αλλάξουμε την τιμή της μεταβλητής μία φορά.

- Όπως έχει αναφερθεί σε προηγούμενη ενότητα, πολλές φορές είναι απαραίτητη η χρήση προθεμάτων για κάθε browser που θέλουμε να είναι συμβατή η ιστοσελίδα μας. Αυτό συνεπάγεται τη συγγραφή τεσσάρων ή παραπάνω γραμμών κώδικα CSS για την ίδια ιδιότητα, όπως π.χ. *linear-gradient*. Στους προεπεξεργαστές αρκεί να γράψουμε π.χ. *linear-gradient(red,blue);* και ο προεπεξεργαστής θα προσθέσει αυτόματα τις απαραίτητες γραμμές συμβατότητας στον κώδικα CSS.

- Όπως επίσης αναφέραμε παραπάνω, είναι σύνηθες και καλή πρακτική ο διαχωρισμός του κώδικα CSS σε επιμέρους αρχεία. Από τη σκοπιά του διακομιστή ωστόσο, είναι καλύτερο μία ιστοσελίδα να στέλνει τον ελάχιστο δυνατό αριθμό αρχείων. Με τους προεπεξεργαστές υπάρχει η δυνατότητα να έχουμε τον κώδικα CSS σε διαφορετικά αρχεία, ανάλογα με την κατηγορία της μορφοποίησης που περιγράφουν, και έπειτα να συνδυάζονται σε ένα αρχείο που θα χρησιμοποιεί η ιστοσελίδα για να

προσπελάσει την εκάστοτε μορφοποίηση. Επιπλέον, αφαιρούνται κενές γραμμές, τα σχόλια και άλλοι αχρείαστοι χαρακτήρες που ανεβάζουν το μέγεθος του CSS αρχείου.

Όπως είναι φυσικό, υπάρχουν και κάποιες αρνητικές πτυχές ή απαιτήσεις που δυσκολεύουν τη χρήση προεπεξεργαστών:

- Για τη χρήση τους θεωρείται αυτονόητο ότι έχει ο χρήστης βαθιά γνώση της CSS. Εάν δεν είναι εξοικειωμένος, είναι εξαιρετικά δύσκολη η εκμάθηση μίας γλώσσας σαν τη Sass και τη Less.
- Οι προεπεξεργαστές είναι ουσιαστικά μικρά προγράμματα που τρέχουν στον υπολογιστή στο παρασκήνιο. Συνεπώς, απαιτούν μία στοιχειώδη διαδικασία εγκατάστασης και ρύθμισης πριν τη χρήση τους

3.3.1.Sass

Η *Sass* έχει αναπτυχθεί σε γλώσσα Ruby. Επεκτείνει τη CSS προσθέτοντας εμφωλευμένους κανόνες, μεταβλητές, mixins, συναρτήσεις και πολλά άλλα. Πολλά γνωστά frameworks είναι δομημένα με τη Sass, όπως το Foundation 3. Ένα από τα πιο ενδιαφέροντα κομμάτια της χρήσης της Sass είναι ότι μπορεί να χρησιμοποιηθεί με το Compass, ένα framework από συναρτήσεις και πρόσθετα με τα οποία εξορθολογίζει τη διαδικασία, δημιουργεί sprites εικόνων δυναμικά, γράφει πιο καθαρό κώδικα και συνδυάζεται εξαιρετικά με δυνατότητες της CSS3.

Η Sass έχει δύο τρόπους σύνταξης:

- *τη SCSS*, δηλαδή Sassy CSS και
- τη *Sass*, την κλασική έκδοση σύνταξης.

Η σύνταξη της Sass χρησιμοποιεί εσοχές αντί για αγκύλες για να ορίσει τμήματα κώδικα μορφοποίησης. Χρησιμοποιεί επίσης αλλαγές γραμμών αντί για το διαχωριστικό *;* για το διαχωρισμό εντολών. Έτσι, ο κώδικας είναι πιο συνοπτικός και ευανάγνωστος.

3.3.1.1.Μεταβλητές

Η Sass επιτρέπει τον ορισμό μεταβλητών με το σύμβολο του δολαρίου ($) και ανάθεση αυτών με άνω κάτω τελεία (:). Υποστηρίζονται τέσσερις τύποι δεδομένων:

- Αριθμοί
- Χαρακτήρες
- Χρώματα
- Λογικοί

Οι μεταβλητές μπορούν να είναι είσοδοι ή αποτελέσματα μίας ή παραπάνω συναρτήσεων. Κατά τη μετάφραση, οι τιμές των μεταβλητών εκχωρούνται στο παραγόμενο CSS που καταλαβαίνει ο browser.

```
$blue: #3bbfce
$margin: 16px

.content-navigation
  border-color: $blue
  color: darken($blue, 9%)

.border
  padding: $margin/2
  margin:  $margin/2
  border-color: $blue
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
.content-navigation {
  border-color: #3bbfce;
  color: #2b9eab;
}

.border {
  padding: 8px;
  margin: 8px;
  border-color: #3bbfce;
}
```

3.3.1.2.Εμφωλευμένα μπλοκ με στοίχιση

Η Sass υποστηρίζει εμφωλευμένα μπλοκς επιλογέων, αλλά τα μπλοκς κώδικα αυτά καθαυτά δεν είναι εμφωλευμένα. Επιτρέπει στα τμήματα εμφωλευμένου κώδικα να εισάγονται το ένα μέσα στο άλλο.

```
table.hl {
  margin: 2em 0;
  td.ln {
    text-align: right;
  }
}

li {
  font: {
    family: serif;
    weight: bold;
    size: 1.3em;
  }
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
table.hl {
  margin: 2em 0;
}
table.hl td.ln {
  text-align: right;
}

li {
  font-family: serif;
  font-weight: bold;
  font-size: 1.3em;
}
```

3.3.1.3.Mixins

Τα mixins επιτρέπουν την ενσωμάτωση όλων των ιδιοτήτων μιας κλάσης σε μια άλλη περιέχοντας το όνομά της σαν μια ιδιότητά της, με συμπεριφορά σαν σταθερά ή μεταβλητή. Μπορούν, επίσης να λειτουργούν ως συναρτήσεις και να δέχονται παραμέτρους. Όπως και παραπάνω στα εμφωλευμένα μπλοκ, έτσι κι εδώ αυτή η δυνατότητα δεν υπάρχει στην απλή CSS, στην οποία πρέπει να επαναλαμβάνουμε κώδικα σε κάθε επιλογέα. Άρα, τα mixins καθιστούν τον επαναλαμβανόμενο κώδικα πιο αποδοτικό και επαναχρησιμοποιήσιμο.

Από τα πιο χρήσιμα και χρησιμοποιούμενα mixins είναι οι επαναλήψεις *@for*, *@each* και *@while*.

```
@mixin table-base {
  th {
    text-align: center;
    font-weight: bold;
  }
  td, th {padding: 2px}
}

#data {
  @include table-base;
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
#data th {
  text-align: center;
  font-weight: bold;
}
#data td, #data th {
  padding: 2px;
}
```

3.3.1.4.Κληρονομικότητα επιλογέων

Παρόλο που η CSS3 υποστηρίζει την ιεραρχία DOM (Document Object Model), δεν επιτρέπει την κληρονομικότητα επιλογέων. Αυτή επιτυγχάνεται εισάγοντας μία γραμμή σε ένα μπλοκ κώδικα με το σύμβολο @ και αναφέρεται σε έναν άλλο επιλογές. Οι ιδιότητες του επεκταμένου επιλογέα εφαρμόζονται στον επιλογέα που καλείται.

```
.error {
  border: 1px #f00;
  background: #fdd;
}
.error.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  @extend .error;
  border-width: 3px;
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
.error, .badError {
  border: 1px #f00;
  background: #fdd;
}

.error.intrusion,
.badError.intrusion {
  font-size: 1.3em;
  font-weight: bold;
}

.badError {
  border-width: 3px;
}
```

3.3.2.Less

Η *Less (Leaner CSS)* είναι μία δυναμική γλώσσα προεπεξεργαστή CSS ανοιχτού λογισμικού. Η πρώτη έκδοση ήταν γραμμένη στη γλώσσα προγραμματισμού Ruby και στη συνέχεια αντικαταστάθηκε από τη JavaScript. Η σύνταξή της βασίζεται σε εμφωλευμένη ιεραρχία. Περιλαμβάνει μηχανισμούς, όπως μεταβλητές, εμφωλευμένα μπλοκ με στοίχιση, mix-ins, τελεστές και συναρτήσεις. Η γενικότερη δομή της Less είναι αρκετά προσιτή σε ένα σχεδιαστή και η σύνταξή της αρκετά κοντά στην κλασική CSS, καθιστώντας την μία πιο ομαλή μετάβαση από τη CSS στους προεπεξεργαστές. Frameworks όπως το Bootstrap τη χρησιμοποιούν για να εκμεταλλευθούν τις δυναμικές δυνατότητες που προσφέρει.

Ένα από τα πιο σημαντικά χαρακτηριστικά της, είναι ότι επιτρέπει την επαναχρησιμοποίηση των κλάσεων οπουδήποτε στον κώδικα μορφοποίησης, όπως ακριβώς και τα mixins.

3.3.2.1.Μεταβλητές

Η Less επιτρέπει τον ορισμό μεταβλητών με το σύμβολο παπάκι (@) και ανάθεση αυτών με άνω κάτω τελεία (:). Κατά τη μετάφραση, οι τιμές των μεταβλητών εκχωρούνται στο παραγόμενο CSS.

```
@color: #4D926F;

#header {
  color: @color;
}

h2 {
  color: @color;
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
#header {
  color: #4D926F;
}

h2 {
  color: #4D926F;
}
```

3.3.2.2.Εμφωλευμένα μπλοκ με στοίχιση

Η Less υποστηρίζει εμφωλευμένα μπλοκς επιλογέων, κάτι που δεν υποστηρίζει η CSS. Έτσι, η κληρονομικότητα έχει περισσότερη σαφήνεια και τα έγγραφα στυλ διατηρούνται μικρότερα.

```
#header {

  h1 {
    font-size: 26px;
    font-weight: bold;
  }

  p {
    font-size: 12px;
    a {
      text-decoration: none;
      &:hover {
        border-width: 1px;
      }
    }
  }
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
#header h1 {
  font-size: 26px;
  font-weight: bold;
}

#header p {
  font-size: 12px;
}

#header p a {
  text-decoration: none;
}

#header p a:hover {
  border-width: 1px;
}
```

3.3.2.3.Mixins

Τα mixins επιτρέπουν την ενσωμάτωση όλων των ιδιοτήτων μιας κλάσης σε μια άλλη περιέχοντας το όνομά της σαν μια ιδιότητά της, με συμπεριφορά σαν σταθερά ή μεταβλητή. Μπορούν, επίσης να λειτουργούν ως συναρτήσεις και να δέχονται παραμέτρους. Όπως και παραπάνω στα εμφωλευμένα μπλοκ, έτσι κι εδώ αυτή η δυνατότητα δεν υπάρχει στην απλή CSS, στην οποία πρέπει να επαναλαμβάνουμε κώδικα σε κάθε επιλογέα. Άρα, τα mixins καθιστούν τον επαναλαμβανόμενο κώδικα πιο αποδοτικό και επαναχρησιμοποιήσιμο.

```
.rounded-corners (@radius: 5px) {
  border-radius: @radius;
  -webkit-border-radius: @radius;
  -moz-border-radius: @radius;
}

#header {
  .rounded-corners;
}

#footer {
  .rounded-corners(10px);
}
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
#header {
  border-radius: 5px;
  -webkit-border-radius: 5px;
  -moz-border-radius: 5px;
}

#footer {
  border-radius: 10px;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
}
```

3.3.2.4.Πράξεις και συναρτήσεις

Η πρόσθεση, η αφαίρεση, η διαίρεση και ο πολλαπλασιασμός των τιμών και των χρωμάτων χρησιμοποιούνται για να δημιουργήσουν πολύπλοκες σχέσεις μεταξύ των ιδιοτήτων. Οι συναρτήσεις αντιστοιχίζονται μια προς μια με τον κώδικα JavaScript, επιτρέποντας το χειρισμό των τιμών.

```
@the-border: 1px;
@base-color: #111;
@red:        #842210;

#header {
  color: @base-color * 3;
  border-left: @the-border;
  border-right: @the-border * 2;
}

#footer {
  color: @base-color + #003300;
  border-color: desaturate(@red, 10%);
}
```

```
#header { color:
  #333; border-
  left: 1px;
  border-right: 2px;
}

#footer {
  color: #114411;
  border-color: #7d2717;
}
```

3.3.3.Stylus

Η *Stylus*, η τρίτη πιο διαδεδομένη γλώσσα προεπεξεργαστών, μετά τη Sass και τη Less, έχει αναπτυχθεί σε Javascript και υποστηρίζει πολλές από τις δυνατότητες που υποστηρίζουν οι προηγούμενες δύο, όπως σύνταξη στοίχισης. Μπορεί να χρησιμοποιηθεί με τη *Nib*, μία βιβλιοθήκη παρόμοια με το Compass, που παρέχει CSS3 mixins συμβατά με πολλαπλούς browsers. Το συντακτικό είναι απλό και ευέλικτο, με αγκύλες, άνω και κάτω τελεία, ερωτηματικό και χωρίς παρενθέσεις.

3.3.3.1.Μεταβλητές

Η Stylus επιτρέπει τον ορισμό μεταβλητών με το σύμβολο παπάκι και ανάθεση αυτών χωρίς σύμβολα. Κατά τη μετάφραση, οι τιμές των μεταβλητών εκχωρούνται στο παραγόμενο CSS.

```
font-size = 14px

body
      font font-size Arial, sans-serif
```

```
body {
   font: 14px Arial, sans-serif;
 }
```

3.3.3.2.Mixins

Τα *mixins* επιτρέπουν την ενσωμάτωση όλων των ιδιοτήτων μιας κλάσης σε μια άλλη περιέχοντας το όνομά της σαν μια ιδιότητά της, με συμπεριφορά σαν σταθερά ή μεταβλητή. Μπορούν, επίσης να λειτουργούν ως συναρτήσεις και να δέχονται παραμέτρους. Όπως και παραπάνω στα εμφωλευμένα μπλοκ, έτσι κι εδώ αυτή η δυνατότητα δεν υπάρχει στην απλή CSS, στην οποία πρέπει να επαναλαμβάνουμε κώδικα σε κάθε επιλογέα. Άρα, τα mixins καθιστούν τον επαναλαμβανόμενο κώδικα πιο αποδοτικό και επαναχρησιμοποιήσιμο.

```
border-radius(n)
    -webkit-border-radius n
    -moz-border-radius n
    border-radius n

form input[type=button]
    border-radius(5px)
```

Αποτέλεσμα του παραπάνω κώδικα σε CSS:

```
form input[type=button] {
    -webkit-border-radius: 5px;
    -moz-border-radius: 5px;
    border-radius: 5px;
}
```

Συμπέρασμα

Κάθε προεπεξεργαστής css έχει το δικό του μοναδικό τρόπο επίτευξης του στόχου. Αν και δεν είναι απαραίτητοι είναι πολλές φορές πολύ χρήσιμοι γιατί μειώνουν πολύ το χρόνο παραγωγής.

Η Less σε σχέση με τη Stylus και τη Sass

-Περιλαμβάνει μηχανισμούς, όπως μεταβλητές, εμφωλευμένα μπλοκ με στοίχιση, mix-ins, τελεστές και συναρτήσεις.

-Η σύνταξή της αρκετά κοντά στην κλασική CSS

-Frameworks όπως το Bootstrap τη χρησιμοποιούν για να εκμεταλλευθούν τις δυναμικές δυνατότητες που προσφέρει.

-Επιτρέπει την επαναχρησιμοποίηση των κλάσεων

Η Sass σε σχέση με τη Less και τη Stylus

-Εχει αναπτυχθεί σε γλώσσα Ruby.

-Επεκτείνει τη CSS προσθέτοντας εμφωλευμένους κανόνες, μεταβλητές, mixins, συναρτήσεις και πολλά άλλα.

-Πολλά γνωστά frameworks είναι δομημένα με τη Sass, όπως το Foundation 3.

-Μπορεί να χρησιμοποιηθεί με το Compass

-Γράφει πιο καθαρό κώδικα και συνδυάζεται εξαιρετικά με δυνατότητες της CSS3.

Η Stylus σε σχέση με τη Less και τη Sass

-Υποστηρίζει πολλές από τις δυνατότητες που υποστηρίζουν οι άλλες 2

-Δυνατότητα να τρέξει μέσω Node.js / JavaScript

-Απλό και ευέλικτο συντακτικό

Κεφάλαιο 4

4. Ποιότητα & Μετρικές

Η έννοια της ποιότητας ξεκινάει από την αρχαία Ελλάδα στα χρόνια του Αριστοτέλη, ο οποίος ξεχώρισε το «προϊόν» με τα χαρακτηριστικά του και η μελέτη της έννοιας συνεχίστηκε τα επόμενα χρόνια από τους φιλοσόφους.

Ο λόγος για τον οποίο επιθυμούμε να μετρήσουμε την ποιότητα των προϊόντων είναι προκειμένου να συγκρίνουμε τα προϊόντα μεταξύ τους. Το ίδιο ισχύει και στο λογισμικό και αποτελεί την κύρια αιτία που ασχολούμαστε με την ποιότητα και αξιολόγησή του.

4.1.Ποιότητα Λογισμικού

Για να μπορέσουμε να μετρήσουμε την ποιότητα του λογισμικού, θα πρέπει να εξετάσουμε την ποιότητα από τρεις τουλάχιστον διαφορετικές πλευρές:

- την ποιότητα του προϊόντος,
- την ποιότητα της διεργασίας, που οδήγησε στην παραγωγή του προϊόντος και
- την ποιότητα του προϊόντος στα πλαίσια του επιχειρηματικού περιβάλλοντος στο οποίο πρόκειται να χρησιμοποιείται.

Είναι εμφανές ότι η ποιότητα λογισμικού είναι μία έννοια, τα χαρακτηριστικά της οποίας μπορεί να μεταβάλλονται ανάλογα με το σύστημα στο οποίο αναφερόμαστε. Εάν δηλαδή εντοπίσουμε ορισμένα ελαττώματα που θεωρούνται ανεκτά και μικρότερης σημασίας σε ένα λογισμικό επεξεργασίας κειμένου, κατά πάσα πιθανότητα τα ίδια ελαττώματα θα θεωρηθούν απαράδεκτα σε ένα σύστημα που είναι κρίσιμο για την ασφάλεια ή της επίτευξη ενός στόχου. Αυτό φαίνεται και από την προσπάθεια των χρηστών να αξιολογήσουν την ποιότητα του λογισμικού. Η πρώτη εντύπωση που θα λάβουν οι χρήστες από τα *εξωτερικά χαρακτηριστικά*,

όπως είναι η διεπαφή χρήστη, η αποδοτικότητα και η συχνότητα των αστοχιών, θα καθορίσει και την εικόνα που θα αποκτήσουν γι' αυτό το λογισμικό.

Εκτός από τους χρήστες, το λογισμικό αξιολογείται και από τους ανθρώπους που το δημιούργησαν και αυτούς που θα έχουν την ευθύνη να συντηρούν το λογισμικό μετά την ολοκλήρωσή του. Οι ειδικοί δηλαδή δεν ασχολούνται με τα εξωτερικά χαρακτηριστικά όπως οι απλοί χρήστες, αλλά με τα *εσωτερικά χαρακτηριστικά* της ποιότητας, όπως είναι το πλήθος και τα είδη των ελαττωμάτων μέσα από τα οποία εξάγουν την πιθανή ποιότητα λογισμικού. Τα εσωτερικά χαρακτηριστικά μπορούν να εξαχθούν από τον ίδιο τον κώδικα του προγράμματος και να μετρηθούν με τη χρήση των κατάλληλων μετρικών.

Γι' αυτό το λόγο έχουν δημιουργηθεί μοντέλα ποιότητας. Προκειμένου να συσχετισθεί η εξωτερική άποψη που έχει ο χρήστης για το λογισμικό, με την εσωτερική άποψη που έχει γι' αυτό ο δημιουργός του.

Παρόλο που όλες οι εταιρείες παραγωγής λογισμικού αφιερώνουν πολύ χρόνο και χρήμα προκειμένου να παραδώσουν στους πελάτες τους λογισμικό χωρίς σφάλματα, είναι πρακτικά αδύνατο αυτό να είναι απόλυτο. Για να γίνει αντιληπτό ο λόγος που δεν μπορεί να εξασφαλιστεί ένα αλάνθαστο λογισμικό, αρκεί να σκεφτούμε τον παρακάτω συλλογισμό. Ένα λογισμικό που έχει κριθεί κρίσιμο για την ασφάλεια, είναι ένα λογισμικό του οποίου η αστοχία (που μπορεί να προκληθεί είτε από κακή χρήση είτε από οποιοδήποτε άλλο λόγο) θα μπορούσε να θέσει σε κίνδυνο τη ζωή ή και την υγεία πολλών ανθρώπων. Η αξιοπιστία ενός τέτοιου συστήματος θα πρέπει να είναι τουλάχιστον 1.000.000.000. Εάν την τιμή της αξιοπιστίας τη μεταφράσουμε σε ώρες, η χρονική τιμή που πρέπει να καλύπτει ένα σύστημα κρίσιμο σε ασφάλεια, είναι ίσο με 140.000 χρόνια λειτουργίας. Ένα χρονικό διάστημα αυτού του μεγέθους δεν είναι δυνατό να αφιερωθεί μόνο για τη φάση των δοκιμών του συστήματος, όπως γίνεται εύκολα αντιληπτό.

Συνεπώς, είναι εμφανής η δυσκολία των κατάλληλων ελέγχων στο σύστημα, αν αναλογιστούμε την παγκόσμια αγορά και τους ρυθμούς με τους οποίους οι μεγάλες εταιρίες παράγουν λογισμικό. Ο συνεχής και έντονος ανταγωνισμός της αγοράς αναγκάζει τους τεχνολόγους λογισμικού να παραδίδουν τα προϊόντα τους όσο πιο γρήγορα είναι αυτό δυνατό. Αυτό οδηγεί σε μειωμένο χρόνο που αφιερώνουν στις δοκιμές του προϊόντος και σε έλεγχο μόνο

των απολύτως απαραίτητων λειτουργιών του. Οι λειτουργίες που θα δοκιμαστούν θα περιοριστούν σε αυτές που χρησιμοποιεί περισσότερο συχνά ο χρήστης και σε αυτές που πιθανότερα μπορούν να επηρεάσουν το σωστό χειρισμό των δεδομένων του προγράμματος.

Όπως κάθε κατασκευαστής οποιουδήποτε προϊόντος θέλει να διασφαλίσει τη μέγιστη ποιότητα, το ίδιο ισχύει και για τους μηχανικούς λογισμικού, που θέλουν να παράγουν το καλύτερο ποιοτικά λογισμικό σε σχέση με αυτό των ανταγωνιστών τους. Η διαφορά έγκειται στη στρατηγική αξιολόγησης που ακολουθείται. Η ανάπτυξη λογισμικού υψηλής ποιότητας συνεχίζει να είναι μία δύσκολη εργασία στις μέρες μας. Για πολλούς η ποιότητα είναι ένα απροσδιόριστο χαρακτηριστικό γνώρισμα του λογισμικού που δεν μπορεί μετρηθεί και να αξιολογηθεί, καθώς δεν πρόκειται για μια χειροπιαστή έννοια. Άρα ο καθένας έχει τη δική του γνώμη και αναπαράσταση της ποιότητας σαν έννοια στο μυαλό του. Αυτό κάνει ακόμα πιο δύσκολη την προσπάθεια διατύπωσης ενός καθολικού ορισμού που θα εκφράζει κάθε πτυχή ποιότητας του λογισμικού.

Μία από τις πιο σημαντικές παραμέτρους που χρειάζονται ποσοτικό προσδιορισμό κατά την ανάπτυξη των πληροφοριακών συστημάτων, είναι η ποιότητα λογισμικού. Η επίτευξη της υψηλής ποιότητας του λογισμικού είναι ο πιο σημαντικό στόχος και απαιτεί τη μελέτη και ανάλυση πολλών ιδιοτήτων του.

Υπάρχουν διάφορες απόψεις για το πώς μπορούμε να ορίσουμε το ποιοτικό λογισμικό. Πέντε διαφορετικές έχουν εκφραστεί από τον Garvin το 1984:

- **Υπερβατική** άποψη (transcendental view): Σύμφωνα με αυτή την άποψη, η ποιότητα είναι κάτι που αναγνωρίζουμε, αλλά δεν μπορούμε να ορίσουμε. Με την παραπάνω περιγραφή μοιάζει η περιγραφή που έκανε ο Πλάτωνας για την έννοια του ιδανικού, καθώς και με την έννοια που έδινε ο Αριστοτέλης στη μορφή. Δηλαδή όπως ακριβώς κάθε πραγματικό τραπέζι είναι μια προσέγγιση του ιδανικού τραπεζιού, μπορούμε να φανταστούμε την ποιότητα του λογισμικού ως ένα ιδανικό το οποίο θα θέλαμε πάρα πολύ να προσεγγίσουμε, αλλά ποτέ δε θα είμαστε σε θέση να υλοποιήσουμε πλήρως.

- Άποψη του **χρήστη** (user view): Σε αυτή την άποψη υποστηρίζεται ότι η ποιότητα είναι η καλή προσαρμογή σε ένα σκοπό. Σε αντίθεση με την υπερβατική, η άποψη του χρήστη είναι πιο χειροπιαστή και μπορεί να χρησιμοποιηθεί για τη μέτρηση των χαρακτηριστικών ενός προϊόντος, όπως η αξιοπιστία και η συχνότητα των ελαττωμάτων, με στόχο να κατανοήσουμε τη συνολική ποιότητα του προϊόντος.
- **Κατασκευαστική** άποψη (manufacturing view): Πρόκειται για την οπτική γωνία που υποστηρίζει ότι η ποιότητα είναι η συμμόρφωση στις προδιαγραφές. Εξετάζει την ποιότητα κατά τη διάρκεια της παραγωγής και μετά την παράδοση του λογισμικού σε συνδυασμό με τις απαιτήσεις που έχουν καθοριστεί. Εξετάζεται αν το προϊόν κατασκευάστηκε σωστά από την αρχή, ώστε να περιοριστεί το κόστος με αποφυγή δαπανηρών συμπληρωματικών εργασιών για την επιδιόρθωση ελαττωμάτων μετά την ολοκλήρωση της παραγωγής. Πάντα βέβαια υπάρχει περίπτωση όπου η συμμόρφωση στη διεργασία θα οδηγήσει σίγουρα σε λογισμικό με λιγότερα ελαττώματα και αστοχίες.
- Άποψη του **προϊόντος** (product view): Σύμφωνα με αυτή την άποψη, η ποιότητα είναι συνδεδεμένη με έμφυτα χαρακτηριστικά του προϊόντος. Πρόκειται για την αντίθετη άποψη από αυτή της κατασκευαστικής, αφού το συμπέρασμα για την ποιότητα του προϊόντος προκύπτει από τα βαθύτερα, εσωτερικά χαρακτηριστικά του. Αυτή η άποψη εκφράζει συνήθως τους περισσότερους επαγγελματίες στην αξιολόγηση λογισμικού, αφού πιστεύουν ότι αν ένα λογισμικό έχει ενδείξεις καλής εσωτερικής ποιότητας, κατά πάσα πιθανότητα θα έχει και καλές εξωτερικές ενδείξεις, όπως η αξιοπιστία και η ευκολία στη συντήρηση.
- Άποψη με βάση την **αξία** (value-based view): Η ποιότητα εξαρτάται από το ποσό των χρημάτων που είναι διατεθειμένος να καταβάλει γι' αυτή ο πελάτης. Η άποψη της αξίας συνδυάζει την άποψη του προϊόντος και την κατασκευαστική άποψη. Το αδύναμο σημείο αυτής της άποψης είναι η δυσκολία καθορισμού ξεκάθαρων ορίων, καθώς και η εφαρμογή της στην πράξη. Αν καταφέρουμε να εξισώσουμε την ποιότητα του λογισμικού με το ποσό που είναι διατεθειμένος να δώσει ο πελάτης για

την αγορά του, έχουμε μία πολύ σημαντική σχέση, που συνδέει το κόστος με την ποιότητα και αν παρουσιαστεί κάποιο πρόβλημα μπορούμε να την αξιοποιήσουμε. Τέλος, με την προσέγγιση αυτή οι πελάτες συνδέουν το κόστος με την ποιότητα του λογισμικού και κάθε φορά που θέλουν να αγοράσουν ένα προϊόν, εξετάζουν τα οφέλη που θα προκύψουν από ένα λογισμικό ποιότητας.

Σημαντικοί λόγοι για τους οποίους οι προγραμματιστές ενδιαφέρονται για τη διασφάλιση της υψηλής ποιότητας, πέρα από την ικανοποίηση των πελατών και την αποφορά κέρδους, είναι για την καλύτερη κατανόηση του κώδικα και για την αξιολόγηση και τη βελτίωση το κώδικα που αποφέρει μεταφέρσιμο και ευέλικτο σχεδιασμό.

Κατά κανόνα όμως ο τρόπος που μετράμε την ποιότητα του λογισμικού, συνδέεται άμεσα με την προστιθέμενη αξία που προσφέρει στο χρήστη, κάτι που προκύπτει από την εσωτερική του ποιότητα. Αν η προστιθέμενη αξία είναι υψηλή, υπάρχει μεγάλη πιθανότητα να συνοδεύεται και από υψηλή ποιότητα. Αντίθετα, έστω ένα σύστημα που προσφέρει μηδενική προστιθέμενη αξία, αλλά παρόλα αυτά είναι σχεδόν άριστο και ελαχιστοποιεί τα λάθη και τα σφάλματα που μπορεί να προκύψουν. Δυστυχώς, το παραπάνω παράδειγμα δεν μπορεί να αντιμετωπιστεί σαν λογισμικό υψηλής ποιότητας, καθώς πρόκειται για ένα άχρηστο λογισμικό. Το συμπέρασμα που προκύπτει λοιπόν είναι ότι για οποιοδήποτε λογισμικό πρέπει να δίνουμε ιδιαίτερη σημασία στην εσωτερική ποιότητα του κώδικα και να μην την παραμελούμε για χάρη της εξωτερικής ποιότητας. Μπορεί να είναι πολύ αργά για να βελτιώσουμε την εξωτερική ποιότητα αν αφιερώσουμε περισσότερο χρόνο και κόπο στην εξωτερική. Στη βελτίωση της εσωτερική ποιότητας μπορούν να συμβάλλουν κατάλληλα εργαλεία ανάλυσης, που έχουν τη δυνατότητα να εξάγουν χρήσιμα συμπεράσματα και βοηθούν στη βελτίωση της ποιότητας του κώδικα.

Οι σημαντικότεροι παράγοντες που επηρεάζουν την ποιότητα του λογισμικού, μπορούν να συμπτυχθούν στα:

- Ασφάλεια
- Προστασία

- Αξιοπιστία
- Διαθεσιμότητα
- Ευρωστία
- Συντηρησιμότητα
- Επαναφερσιμότητα
- Κατανοησιμότητα
- Πολυπλοκότητα
- Μεταφερσιμότητα
- Επεκτασιμότητα
- Αποδοτικότητα
- Ευχρηστία
- Προσαρμοστικότητα
- Ελεγξιμότητα
- Τμηματοποίηση

4.2.Μετρικές Λογισμικού

Στην Τεχνολογία Λογισμικού οι όροι *μέτρηση (measure)* και *μετρική (metric)* χρησιμοποιούνται πολλές φορές ως συνώνυμες. Μέτρηση είναι η αντιστοίχιση ενός μεγέθους με μία τιμή, ενώ μετρική είναι ποσοτική μέτρηση του βαθμού στον οποίο ένα σύστημα ή τμήμα αυτού έχει ένα χαρακτηριστικό. Με μία συλλογή μετρήσεων σχεδιάζονται και αναπτύσσονται οι μετρικές. Ένα σύνολο μετρικών που παρέχουν γνώση για το λογισμικό αποτελεί έναν *ενδείκτη (indicator)*.

Ο βασικότερος στόχος όλων των μετρικών που διεξάγονται σε ένα λογισμικό είναι η διασφάλιση της ποιότητάς του. Για να επιτευχθεί αυτό, όμως, θα πρέπει να υπάρχει ένα πλάνο ποιότητας συνολικά σε ολόκληρο το έργο ή ακόμη καλύτερα σε ολόκληρη την εταιρεία.

Έτσι, η διασφάλιση της ποιότητας, σε συνδυασμό με τις μετρικές που χρησιμοποιούνται, μπορεί να τεθεί από διαφορετικές πλευρές:

- *διασφάλιση ποιότητας προϊόντος (product quality assurance)*: στόχος είναι η ποιότητα του προϊόντος και κατ' επέκταση η ικανοποίηση του χρήστη.
- *διασφάλιση ποιότητας έργου (project quality assurance)*: που στοχεύει
    - στην αποτίμηση της κατάστασης του έργου,
    - την επίλυση προβλημάτων πριν αυτά γίνουν κρίσιμα και επηρεάσουν την πορεία του έργου,
    - τη ρύθμιση και το διαχωρισμό των εργασιών και
    - την εκτίμηση της ικανότητας των ομάδων να παράγουν ποιοτικό λογισμικό.
- *διασφάλιση ποιότητας διαδικασιών (process quality assurance)*: που βοηθά την εταιρεία να έχει γνώση για την αποτελεσματικότητα των διαδικασιών.

Πιο συγκεκριμένα, η μέτρηση κάποιων χαρακτηριστικών του λογισμικού είναι μία διαδικασία απαραίτητη για την εκτίμηση της κατάστασης των έργων, των προϊόντων, των διαδικασιών και των πόρων παραγωγής λογισμικού. Κάθε μέτρηση, όμως, πρέπει να εξυπηρετεί μία συγκεκριμένη ανάγκη, η οποία έχει σαφώς κατανοηθεί και καθοριστεί.

Οι μετρικές λογισμικού χωρίζονται σε κατηγορίες, ανάλογα με τη φάση ανάπτυξης του λογισμικού που εφαρμόζονται:

- Μετρικές για το *μοντέλο της ανάλυσης απαιτήσεων*, που εξετάζουν την ορθότητα και την πληρότητα της ανάλυσης των απαιτήσεων, προκειμένου να γίνει ο σχεδιασμός του τελικού συστήματος. Τέτοιες μετρικές αριθμούνται ελάχιστες στη βιβλιογραφία.

- Μετρικές για το *μοντέλο του σχεδιασμού*, που αναπτύσσονται για τον έλεγχο του σχεδιασμού του τελικού συστήματος. Εστιάζουν τόσο στο τμήματα (components) του συστήματος, όσο και στη διεπαφή (interface).
- Μετρικές για τον *πηγαίο κώδικα*, που αφορούν μετρήσεις πάνω στον κώδικα και αναλύονται στη συνέχεια της ενότητας.
- Μετρικές για τον *έλεγχο* (testing), που αφορούν στον έλεγχο του λογισμικού. Οι περισσότερες μετρικές εστιάζουν στη διαδικασία του ελέγχου και όχι στα τεχνικά χαρακτηριστικά. Οι μετρικές του ελέγχου προκύπτουν ουσιαστικά από τις μετρικές των τριών προηγούμενων φάσεων.
- Μετρικές για τη *συντήρηση*, που μετρούν το βαθμό στον οποίο είναι συντηρήσιμο το λογισμικό.

Ακολούθως παρουσιάζονται ορισμένες από τις πιο γνωστές μετρικές προϊόντος που εφαρμόζονται στον τελικό κώδικα του λογισμικού:

- *Average Module Length:* Πρόκειται για μία μετρική της τμηματοποίησης ενός προγράμματος. Ορίζεται ως το μέσο μήκος των τμημάτων (modules) του.
- *Chen Metric:* Εξετάζει την εντροπία ενός προγράμματος.
- *Decision Count:* Μετρά τη λογική δομή ενός προγράμματος. Ορίζεται από τον αριθμό των δηλώσεων ελέγχου (υπό-συνθήκη δηλώσεις και ανακυκλώσεις).
- *Executable Statements:* Μετρική μεγέθους. Ορίζεται ως τον αριθμό των εκτελέσιμων εντολών (executable statements) ενός προγράμματος. Μετρά τις διαφορετικές εντολές στην ίδια γραμμή ως διαφορετικές και αγνοεί τα σχόλια, τις δηλώσεις δεδομένων και τα headings.
- *Extent Of Reuse:* Μετρά το ποσοστό επαναχρησιμοποίησης κώδικα σε ένα υπάρχον πρόγραμμα. Ορίζονται τέσσερα διαφορετικά επίπεδα: reused verbatim (κώδικας που επαναχρησιμοποιήθηκε χωρίς καμία αλλαγή), slightly modified (λιγότερο από 25% κώδικα έχει αλλαχθεί), extensively modified (25% και περισσότερο των γραμμών του

κώδικα έχει αλλαχθεί) και new (εντελώς νέος κώδικας). Με τον τρόπο αυτό, για ένα συγκεκριμένο πρόγραμμα ορίζεται το μέγεθος (size) που τροποποιήθηκε σε σχέση με το συνολικό και, επομένως, το ποσοστό επαναχρησιμοποίησης στα τέσσερα επίπεδα.

- *Function Count:* Ορίζεται από τον αριθμό των συναρτήσεων (functions) ενός προγράμματος. Συνάρτηση θεωρείται μία συλλογή εκτελέσιμων εντολών που πραγματοποιούν μία συγκεκριμένη εργασία και οι δηλώσεις των παραμέτρων που διαχειρίζονται οι εντολές αυτές.
- *Hausen Metric On Modularity:* Μετρική της τμηματοποίησης ενός προγράμματος που περιγράφει την ολική τμηματοποίηση σε σχέση με ειδικές της θεωρήσεις. Για παράδειγμα: M1=modules/procedures, M2=modules/variables.

4.3.Μετρικές στη CSS

Σε αυτή την ενότητα θα αναφέρουμε σημαντικούς δείκτες που μπορούν να βελτιώσουν την ποιότητα του κώδικα CSS και κατ' επέκταση την ποιότητα μιας ιστοσελίδας.

4.3.1.Απόδοση

Η απόδοση (Performance) μιας ιστοσελίδας ήταν πάντα ένας παράγοντας ζωτικής σημασίας για την επιτυχία μιας ιστοσελίδας. Οι ερευνητές έχουν καταλήξει πως ακόμα και η πιο μικρή βελτίωση στον χρόνο φόρτωσης μιας ιστοσελίδας, μπορεί να οδηγήσει σε αύξηση πωλήσεων των προϊόντων, σε περισσότερα έσοδα από τις διαφημίσεις που περιέχονται σε αυτή και γενικότερα να προσφέρουν μεγαλύτερη ικανοποίηση στους πελάτες.

Τα προηγούμενα χρόνια οι προγραμματιστές ιστοσελίδων κατέφευγαν στην βελτίωση του υλικού τους συστήματος στο οποίο έτρεχε η ιστοσελίδα, αλλά και στην αύξηση του εύρους ζώνης (bandwidth) προκειμένω να προσφέρουν την βέλτιστη εμπειρία στον χρήστη. Τα τελευταία χρόνια όμως, ιδίως λόγω της εκτεταμένης περιήγησης στο διαδίκτυο μέσω κινητών συσκευών, αυτός ο τρόπος δεν ακολουθείτε. Το χαμηλό εύρος ζώνης καθώς και η μικρή επεξεργαστική ισχύς των κινητών συσκευών, κάνουν επιτακτική την ανάγκη για βελτιστοποίηση

μιας ιστοσελίδας στο front-end επίπεδο, προκειμένω να ικανοποιήσουν τις προσδοκίες των χρηστών.

Δεν έχει σημασία πόσο ενδιαφέρουσα, όμορφη ή διαδραστική είναι μια ιστοσελίδα. Εάν ο χρόνος φόρτωσής της, από έναν επιτραπέζιο υπολογιστή ή από μια κινητή συσκευή, διαρκέσει περισσότερο από δύο με τρία δευτερόλεπτα, ο χρήστης γίνεται ανυπόμονος και είναι πολύ πιθανόν να την εγκαταλείψει γρήγορα. Έχει υπολογισθεί πως ο μέσος χρήστης μπορεί να μεταβεί σε νέα σελίδα, ή να πατήσει το κουμπί back για να μεταβεί σε προηγούμενη σελίδα, ακόμα και να κλείσει το πρόγραμμα περιήγησης πριν καν φορτώσει η ιστοσελίδα που είχε ζητήσει. Ακόμη και καθυστερήσεις που διαρκούν λιγότερο από ένα δευτερόλεπτο μπορούν να επηρεάσουν σημαντικά τα έσοδα.

Το 2006 ο Marissa Mayer, πειραματίστηκε με τον δείκτη απόδοση μιας ιστοσελίδας και συγκεκριμένα με την σελίδα επιστροφής αποτελεσμάτων στην μηχανή Google. Οι χρήστες είχαν αναφέρει σε σχόλιά τους ότι θα επιθυμούσαν να παίρνουν πάνω από 10 αποτελέσματα ανά σελίδα κατά την αναζήτησή τους. Έτσι η Google αύξησε αυτό το όριο σε 30 αποτελέσματα ανά σελίδα. Προς μεγάλη τους έκπληξη είδαν πως τα έσοδα της εταιρίας μειώθηκαν κατά 20% και αυτό γιατί η σελίδα απαιτούσε μισό δευτερόλεπτο για να φορτώσει τα επιπλέον αποτελέσματα.

Οι προσδοκίες των χρηστών έχουν κλιμακωθεί από τότε. Μια μελέτη του 2009 από τη Forrester Research για λογαριασμό της τεχνολογίας Akamai, εντόπισε ως κατώτατο όριο αποδεκτού χρόνου απόκρισης μιας ιστοσελίδας στο ιντερνέτ, τα δύο δευτερόλεπτα και διαπίστωσε ότι το 40% των χρηστών εγκαταλείπουν μια σελίδα που η διάρκεια φόρτωσης του περιεχομένου της είναι περισσότερο από τρία δευτερόλεπτα. Μόλις ένα χρόνο αργότερα, στην ίδια έρευνα, το ποσοστό αυτό αυξήθηκε στο 57% .

4.3.2.Προσβασιμότητα

Η δυνατότητα πρόσβασης στο διαδίκτυο σημαίνει ότι οι ανάπηροι άνθρωποι μπορούν να χρησιμοποιήσουν το διαδίκτυο. Πιο συγκεκριμένα, η δυνατότητα πρόσβασης στο διαδίκτυο σημαίνει ότι οι ανάπηροι άνθρωποι μπορούν να αντιληφθούν, να καταλάβουν, να προηγηθούν, και να αλληλοεπιδράσουν στο Διαδίκτυο και μπορούν να συμμετέχουν σ’ αυτό. Η δυνατότητα

πρόσβασης στο Διαδίκτυο ωφελεί επίσης και άλλους, συμπεριλαμβανομένων και των ηλικιωμένων ανθρώπων που λόγω της γήρανσης οι δυνατότητές τους είναι διαφορετικές.

Η δυνατότητα πρόσβασης στο Διαδίκτυο καλύπτει όλες τις ανικανότητες που επηρεάζουν την πρόσβαση στο Διαδίκτυο και περιλαμβάνει: οπτικές, ακουστικές, σωματικές, δυσκολίες ομιλίας, γνωστικές, και νευρολογικές ανικανότητες. Εκατομμύρια ανθρώπων με ανικανότητες έχουν επιπτώσεις στη χρήση του Διαδικτύου. Αυτήν την περίοδο, οι περισσότερες ιστοσελίδες και λογισμικά παρουσιάζουν πολλές δυσκολίες στη δυνατότητα πρόσβασης και η χρήση τους γίνεται πιο δύσκολη ή αδύνατη για πολλούς ανάπηρους ανθρώπους. Δεδομένου ότι οι ιστοσελίδες και τα λογισμικά θα διατίθενται πιο εύκολα, οι ανάπηροι άνθρωποι θα είναι σε θέση να χρησιμοποιήσουν και να συμμετέχουν στο Διαδίκτυο αποτελεσματικότερα.

Η δυνατότητα πρόσβασης στο Διαδίκτυο ωφελεί επίσης και τους ανθρώπους χωρίς ανικανότητες. Παραδείγματος χάριν, μια βασική αρχή της δυνατότητας πρόσβασης Διαδικτύου είναι να σχεδιάζει ιστοσελίδες και λογισμικά που είναι ευέλικτα για να ικανοποιήσουν τις διαφορετικές ανάγκες, τις προτιμήσεις, και τις καταστάσεις των χρηστών. Αυτή η ευελιξία ωφελεί επίσης τους ανθρώπους χωρίς ανικανότητες σε ορισμένες καταστάσεις, όπως οι άνθρωποι που χρησιμοποιούν μια αργή σύνδεση στο Διαδίκτυο, οι άνθρωποι με τις «προσωρινές ειδικές ανάγκες» όπως ένας σπασμένος βραχίονας, και οι άνθρωποι με τις διαφορετικές δυνατότητες λόγω της γήρανσης.

Το πρότυπο WCAG , Web Content Accessibility Guidelines, ορίζει τις παραμέτρους ώστε μια ιστοσελίδα να χαρακτηρίζεται ως φιλική για ΑΜΕΑ. Η ύπαρξη μιας τέτοιας ιστοσελίδας διευκολύνει της πρόσβαση σε ανθρώπους με ειδικές ανάγκες , οι οποίο μπορούν να πλοηγηθούν στην ιστοσελίδα σας με εύκολο τρόπο. Τα τελευταια χρόνια επιδοτούμενες ιστοσελίδες μέσω ΕΣΠΑ ζητάνε την συμβατότητα της ιστοσελίδας σύμφωνα με το πρότυπο WCAG.

Έλεγχος Προσβασιμότητας: http://www.achecker.ca/

4.3.3.Αποτελεσματικότητα

Τα αρχεία CSS τα οποία έχουν την ίδια επίδραση στο περιεχόμενο ενός αρχείου HTML, μπορεί να διαφέρουν σημαντικά ως προς τον τρόπο με τον οποίο εκχωρούν τιμές στις ιδιότητές. Το μήκος του κώδικα είναι ένα κριτήριο εκχώρησης τιμής, αλλά το πιο σημαντικό είναι οι εκφράσεις που χρησιμοποιούνται για να αναφερθούν στα στοιχεία του αρχείου HTML για την ανάθεση ιδιοτήτων. Μεγιστοποίηση της αποδοτικότητας του κώδικα CSS σημαίνει εφαρμογή εκχώρησης ιδιοτήτων κατά τρόπο που να ελαχιστοποιείται η προσπάθεια σύνταξης κώδικα, καθώς και να επιτυγχάνεται η διατήρηση και επαναχρησιμοποίησή του. Στις επιλογείς CSS που αναφέραμε στο 2ο κεφάλαιο διαδραματίζουν βασικό ρόλο στην πολυπλοκότητα του κώδικα καθώς και στην ποιότητα του εγγράφου HTML που διαθέτει το περιεχόμενο.

4.3.4.Συμβατότητα

Η αιτία των προβλημάτων συμβατότητας (Compatibility) είναι ότι τα προγράμματα περιήγησης αναλύουν τα CSS αρχεία με διαφορετικό τρόπο. Ακόμη και προγράμματα περιήγησης των ίδιων κατασκευαστών, τα οποία όμως ανήκουν σε άλλη έκδοση όπως για παράδειγμα ο IE6.0 και IE7 μπορεί να εμφανίσουν προβλήματα συμβατότητας. Ένας άλλος λόγος, είναι ότι η έκδοση της CSS και οι εκδόσεις των προγραμμάτων περιήγησης είναι σε κατάσταση συνεχής ενημέρωσης και όσο συμβαίνει αυτό οι δύο πλευρές αλλάζουν δυναμικά αυτό πλήττει την συμβατότητα.

4.3.5.Αφαιρετικότητα

Οι παρουσιάσεις προδιαγραφών με υψηλό επίπεδο αφαιρετικότητας βασίζονται σε δομικά στοιχεία καθορισμένης σημασιολογίας, πέρα από το πλαίσιο του εγγράφου και δεν απαιτούν πρόσθετη δομή στο περιεχόμενο του εγγράφου. Έτσι, μια πιο αφηρημένη παρουσίαση προδιαγραφών κάνει το περιεχόμενο του εγγράφου εύκολα επαναχρησιμοποιήσιμο. Όσον αφορά την αφαιρετικότητα σε επίπεδο γλώσσας CSS, αυτή εκφράζει το πόσο έντονα ανατίθενται ιδιότητες παρουσίασης σε στοιχεία HTML.

Ένα υψηλό επίπεδο αφαίρεσης σημαίνει πως έχουμε έναν κώδικα που περιέχει πιο γενικές δηλώσεις και κατ' επέκταση εμφανίζει μεγαλύτερη ανοχή στις αλλαγές που γίνονται στο περιεχόμενο του εγγράφου.

4.3.6.Πολυπλοκότητα

Η πολυπλοκότητα (Complexity) της γλώσσας CSS αναφέρεται στο πόσο μπορεί να γίνει κατανοητή και εύκολα διαχειρίσιμη. Όλοι οι παράγοντες που κάνουν την χρήση της CSS δύσκολη είναι υπεύθυνοι για την αύξηση της πολυπλοκότητα του κώδικα. Τέτοιοι παράγοντες είναι το μέγεθος, η ποικιλία στην χρήση κανόνων, η πολλαπλή χρήση κανόνων κ.α.

4.4.Σφάλματα σε CSS και Html

Στην ενότητα αυτή θα μελετήσουμε τα πιο συχνά σφάλματα που εντοπίζουμε σε κώδικα CSS με χρήση *επικυρωτών CSS (CSS validators)*. Οι validators εντοπίζουν ζητήματα συντακτικής φύσης. Ελέγχουν με στατικό τρόπο, αν ένα αρχείο CSS ακολουθεί αυστηρά την CSS γραμματική. Ένα χαρακτηριστικό παράδειγμα τέτοιων εργαλείων είναι ο W3C CSS validator. Τα εργαλεία αυτά είναι πολύ χρήσιμα, αλλά δεν έχουν την δυνατότητα να εντοπίζουν τους κανόνες που δεν συμπεριφέρονται όπως αναμένεται.

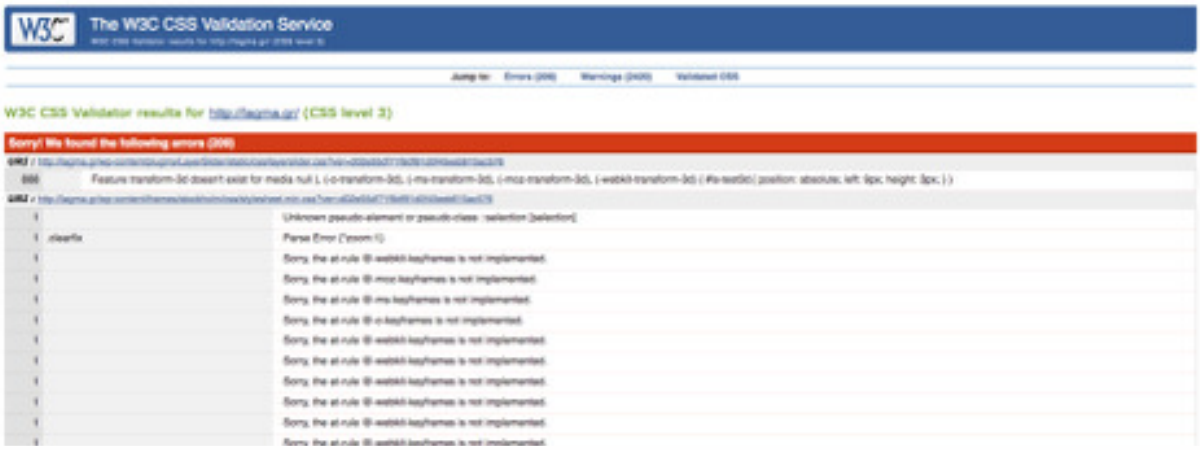
4.4.1.Συνήθη σφάλματα

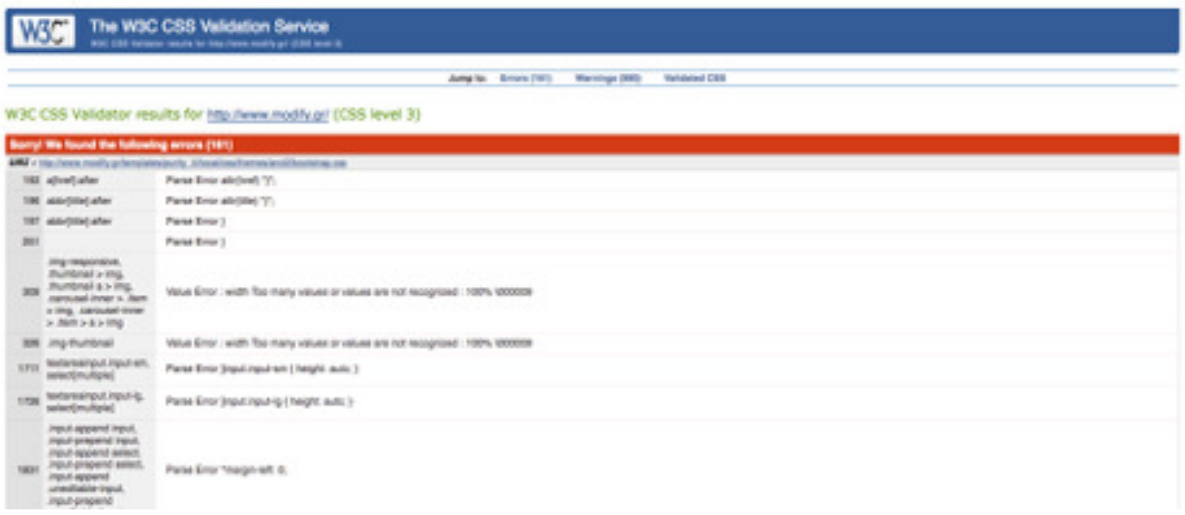
Σε αυτή την ενότητα θα καταγράψουμε κάποια συνήθη σφάλματα επικύρωσης CSS κώδικα. Παρόλο που μία ιστοσελίδα μπορεί να φαίνεται καλαίσθητη, η ποιότητα του κώδικα δε συνάδει πάντα με την ποιότητα του σχεδιασμού. Ωστόσο, τέτοια σφάλματα διορθώνονται αρκετά εύκολα.

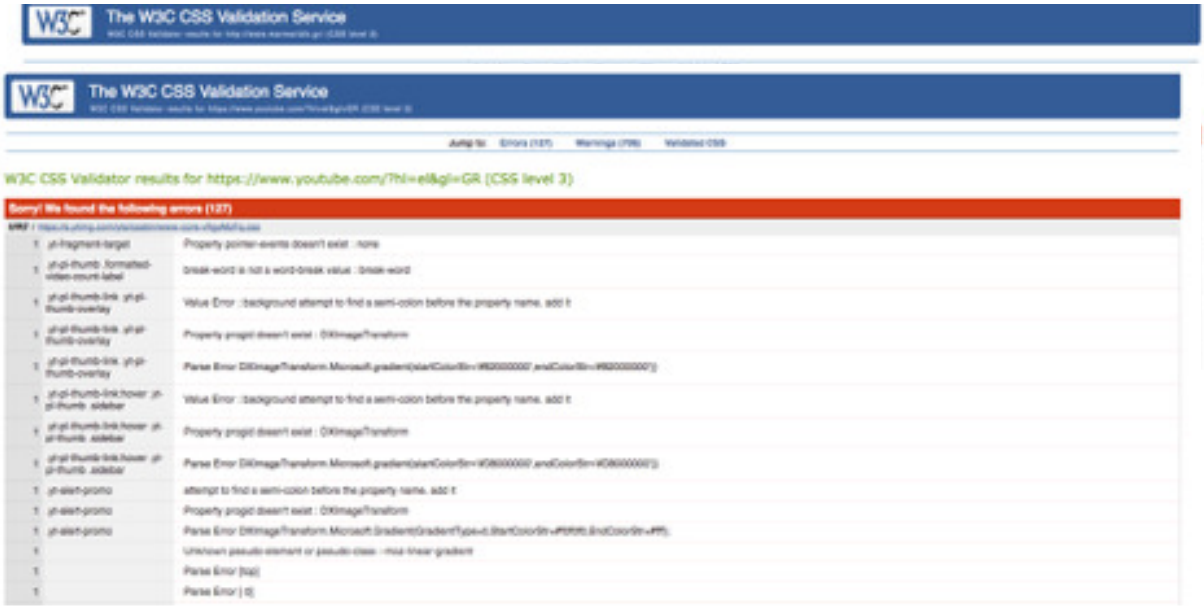
4.4.1.1.Απουσία Doctype

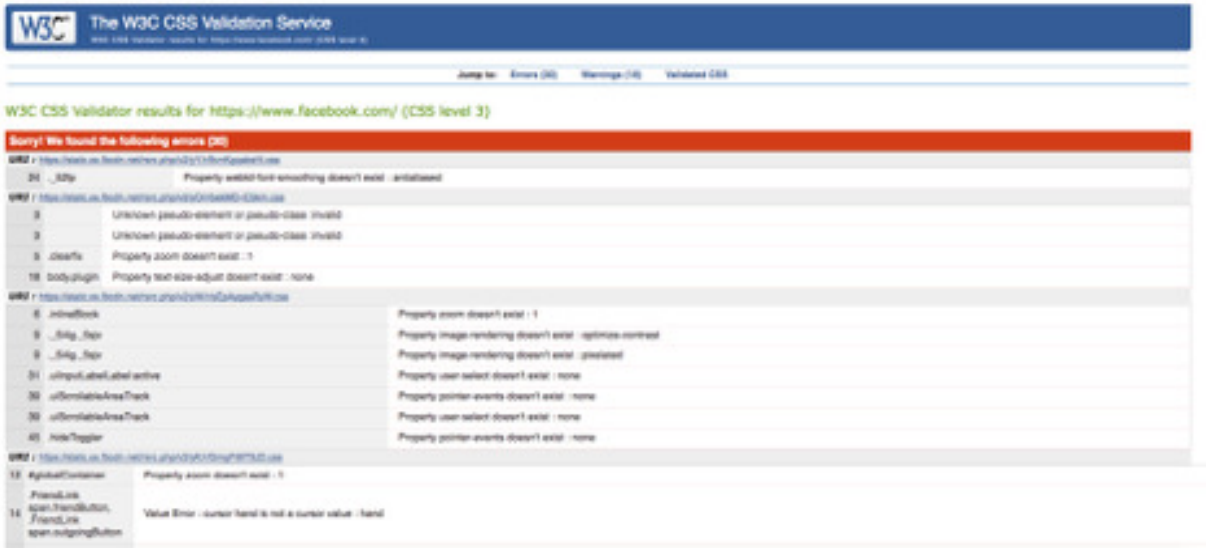
Από τα χειρότερα σφάλματα επικύρωσης είναι η απουσία Doctype στην αρχή του αρχείου. Απουσία Doctype σημαίνει ότι οι browsers θα πρέπει να μαντέψουν σε ποια γλώσσα

έχει γραφτεί η ιστοσελίδα. Αυτό που πρέπει να κάνει λοιπόν ο σχεδιαστής, είναι να προσθέτει πάντα ένα HTML ή XHTML doctype.









4.4.1.2.Αμέλεια κλεισίματος του στοιχείου

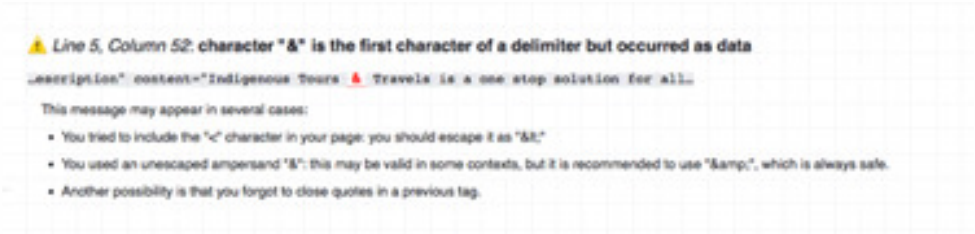
Εάν ανοίξουμε ένα tag σε κάποιο σημείο του HTML θα πρέπει οπωσδήποτε να θυμηθούμε να το κλείσουμε στο κατάλληλο σημείο. Αμελώντας το, όχι απλά δε θα είναι επικυρωμένος ο κώδικας, αλλά μπορεί να προκαλέσει σοβαρά προβλήματα στην εμφάνιση. Η προειδοποίηση γι' αυτό στον W3C validator είναι ένα μήνυμα *«end tag for "p" omitted, but OMITTAG NO was specified»*, που με λίγα λόγια σημαίνει ότι ξεχάσαμε να κλείσουμε το "p" tag.

4.4.1.3.Απουσία «/» στο κλείσιμο στοιχείου

Τα περισσότερα στοιχεία HTML έχουν συγκεκριμένο τρόπο κλεισίματος των tag τους, όπως π.χ. * ! *, αλλά άλλα στοιχεία όπως *input*, *img*, *meta* κλείνουν αυτόματα, που σημαίνει ότι έχουν απλά μία κάθετο «/» πριν κλείσουν οι αγκύλες «<>». Ξεχνώντας αυτή την κάθετο, είναι σαν να ξεχνάμε να κλείσουμε το tag, όπως παραπάνω.

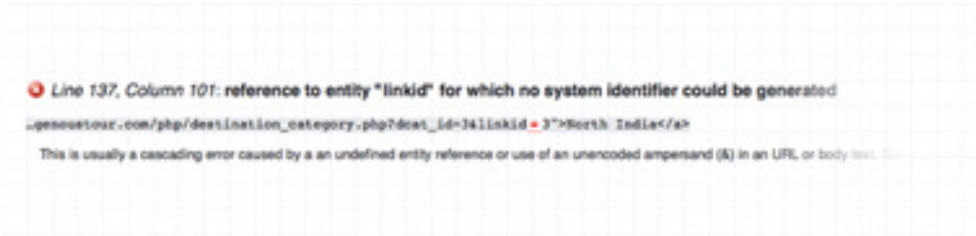
4.4.1.4.Αμέλεια μετατροπής ειδικών χαρακτήρων

Το σύμβολο *"&"* είναι ίσως το πιο δημοφιλής αιτία για τέτοιου είδους σφάλματα, μαζί με τα εισαγωγικά που έχουν αντιγραφεί από έναν άλλο κειμενογράφο. Θα πρέπει πάντα να θυμόμαστε να ελέγχουμε τον κώδικα του body και να μετατρέπουμε κάθε σύμβολα *"&"* στην αντιστοιχία τους σε HTML *"&amp"*, καθώς και όλους του άλλους ειδικούς χαρακτήρες στο αντίστοιχο σύμβολό τους.



4.4.1.5.Μη κωδικοποιημένοι χαρακτήρες στα URLs

Σε συνέχεια της προηγούμενης παραγράφου, οι ειδικοί χαρακτήρες, ειδικά το "&", θα πρέπει να μετατρέπονται στην αντίστοιχη HTML μορφή τους όταν εμφανίζονται στα URLs. Σύνδεσμοι προς ιστοσελίδες που προέρχονται από PHP αρχεία είναι πολύ πιθανό να περιέχουν



ένα χαρακτήρες όπως το "&", οπότε θα πρέπει να τους γράφουμε εξ αρχής χρησιμοποιώντας το στοιχείο "&amp".

4.4.1.6.Στοιχεία block σε ενσωματωμένα στοιχεία

Ένας θεμελιώδης κανόνας της HTML είναι ότι τα στοιχεία block δε θα πρέπει ποτέ να βρίσκονται εντός ενσωματωμένων στοιχείων. Ένα σύνηθες παράδειγμα είναι ένα <a> tag μέσα σε επικεφαλίδα:

```
<a href="#"><h3>random text</h3></a>
```

Το <h3> είναι ένα στοιχείο block το οποίο θα πρέπει να περικλείει το <a> (ενσωματωμένο στοιχείο) με τον παρακάτω τρόπο:

```
<h3><a href="#">random text</a></h3>
```

4.4.1.7.Απουσία «alt» σε εικόνες

Κάθε εικόνα που έχει εισαχθεί σε ένα HTML αρχείο θα πρέπει να έχει ένα γνώρισμα "alt", που θα περιγράφει τα περιεχόμενά τους. Εάν η εικόνα είναι μόνο για σκοπούς

παρουσίασης, το "alt" θα πρέπει ακόμα να υφίσταται, αλλά μπορεί να αφεθεί κενό, δηλαδή π.χ. *alt=*". Ειδάλλως, μία λεπτομερής περιγραφή των περιεχομένων της εικόνας θα πρέπει να δοθεί.

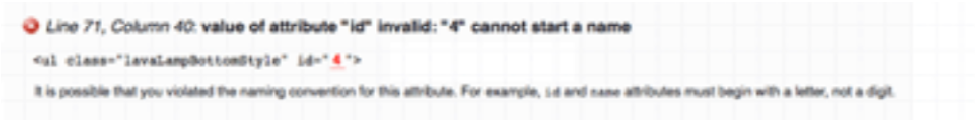
4.4.1.8.Χρήση χαρακτηριστικών όπως width, height

Αυτό είναι πολλές φορές αποτέλεσμα της χρήσης WYSIWYG επεξεργαστών κειμένου (πλατφόρμα εξαγωγής κώδικα από φιλική προς το χρήστη διεπαφή, όπως του Word), που πολλές φορές εισάγει HTML κώδικα που δε θα έπρεπε να υπάρχει. Τα γνωρίσματα *width* και *height* είναι αρκετά χρήσιμα όταν χρησιμοποιούνται σε μεταβατικά Doctypes, αλλά εάν επιθυμούμε το τέλειο αποτέλεσμα με ένα αυστηρό Doctype, θα πρέπει να ξέρουμε ότι η μορφοποίηση

παρουσίασης θα πρέπει να καθοριστεί στο CSS αρχείο.

4.4.1.9.Αρχή κλάσης ή ID με αριθμό

Οι κλάσεις, τα IDs και τα ονόματα γνωρισμάτων δεν μπορούν να ξεκινούν με αριθμό. Μπορούν να περιέχουν, αλλά δεν μπορούν να βρίσκονται στην αρχή της λέξης.



Κεφάλαιο 5

5. Εργαλεία Στατικής Ανάλυσης Λογισμικού & Μέτρησης Μετρικών

Στην πληροφορική, στατική ανάλυση λογισμικού / κώδικα, είναι η ανάλυση ενός προγράμματος που γίνεται χωρίς να εκτελεστεί το πρόγραμμα (η ανάλυση προγράμματος που γίνεται με εκτέλεσή του λέγεται δυναμική ανάλυση). Στις περισσότερες περιπτώσεις η ανάλυση γίνεται σε κάποιο επίπεδο πηγαίου κώδικα και πολλές φορές σε αντικειμενικό κώδικα. Ο όρος συνήθως αναφέρεται στην ανάλυση που γίνεται με κάποιο αυτόματο εργαλείο, ενώ η ανάλυση προγράμματος από άνθρωπο λέγεται κατανόηση προγράμματος.

Η ανάλυση που μπορεί να γίνει από αυτόματα εργαλεία ποικίλλει από αυτά που εξετάζουν τη συμπεριφορά μόνο συγκεκριμένων εκφράσεων ή δηλώσεων στο πρόγραμμα, έως αυτά που συμπεριλαμβάνουν ολόκληρο τον πηγαίο κώδικα ενός προγράμματος στην ανάλυση. Οι χρήσεις των πληροφοριών που πηγάζουν από την ανάλυση συμπεριλαμβάνουν από τον εντοπισμό πιθανών λαθών στο πρόγραμμα, έως τυπικές μεθόδους που αποδεικνύουν μαθηματικά ιδιότητες για κάποιο δεδομένο πρόγραμμα (π.χ. ότι η συμπεριφορά του όντως ακολουθεί την τεχνική του περιγραφή).

Η μέτρηση μιας οποιασδήποτε ιδιότητας του προγράμματος, όπως και ο αντίστροφος σχεδιασμός (reverse engineering) θεωρούνται επίσης είδη στατικής ανάλυσης.

Μια αυξανόμενη εμπορική χρήση της στατικής ανάλυσης είναι η επαλήθευση ιδιοτήτων του λογισμικού που χρησιμοποιείται σε υπολογιστικά συστήματα όπου η ασφάλεια είναι κρίσιμη, για την αυτόματη ανακάλυψη πιθανών τρωτών σημείων στον κώδικα.

5.1.Εργαλεία Στατικής Ανάλυσης Λογισμικού

Σήμερα υπάρχουν στην βιομηχανία πολλά εργαλεία στατικής ανάλυσης κώδικα, εμπορικά η μη. Για την παρούσα διπλωματική εργασία επιλέχθηκε ένας μικρός αριθμός

εργαλείων και πραγματοποιήθηκε μια σύγκριση μεταξύ τους. Για την επιλογή των εργαλείων λήφθηκαν υπόψη τα παρακάτω κριτήρια:

- κάλυψη διαδεδομένων γλωσσών προγραμματισμού
- βάθος της ανάλυσης
- κανόνες ελέγχου
- μετρικές ποιότητας κώδικα

Η επιλογή των περισσότερων εργαλείων βασίστηκε επίσης στο πόσο διαδεδομένα είναι τα εργαλεία, και ποια από αυτά χρησιμοποιούνται περισσότερο από χρήστες και οργανισμούς. Επίσης για να είναι αποδεκτό ένα εργαλείο θα πρέπει να μην επιστρέφει μεγάλο ποσοστό λανθασμένων αποτελεσμάτων (false positives).

Για κάθε εργαλείο είναι διαθέσιμες οι παρακάτω πληροφορίες: το όνομα, ο κατασκευαστής, η επίσημη του ιστοσελίδα, ο τύπος άδειας λογισμικού, οι γλώσσες προγραμματισμού που υποστηρίζει και το λειτουργικό σύστημα στο οποίο μπορεί να εκτελεστεί. Πραγματοποιώντας μια αναζήτηση στο διαδίκτυο ανακαλύψαμε ότι τα περισσότερα εργαλεία αναλύουν κώδικα γραμμένο στις πιο διαδεδομένες γλώσσες Java, C και C++. Με βάση τα κριτήρια που αναφέρθηκαν πιο πάνω καταλήξαμε σε 5 εργαλεία λόγω της περιορισμένης εμπορικής άδειας που είχαν κάποια εργαλεία (που δε μας επέτρεπε την ανάλυσή του σε βάθος), καθώς και λόγω της παλαιάς ημερομηνίας δημοσίευσης ορισμένων εργαλείων (για τα οποία δεν υπήρχε διαθέσιμη κάποια πρόσφατη έκδοση).

Τα 10 εργαλεία καλύπτουν συνολικά μερικές από τις πιο κάτω γλώσσες προγραμματισμού:

- Java
- C
- C++
- C#
- PHP

5.1.1.Metrics

Το εργαλείο Metrics με έκδοση είναι ένα ανοικτού κώδικα λογισμικό το οποίο διατίθεται ως plugin για το περιβάλλον προγραμματισμού Eclipse. Αναπτύχθηκε από τον Frank Sauer και είναι διαθέσιμο από τον Σεπτέμβριο του 2007 από το SourceForge.

Το Metrics αναλύει στατικά πηγαίο κώδικα σε Java και μπορεί να υπολογίσει όλες τις διαδεδομένες μετρικές. Επίσης το εργαλείο υπολογίζει τον γράφο (dependency graph) με τις εξαρτώμενες συνδέσεις ανάμεσα στις κλάσεις και στα πακέτα του λογισμικού.

Αφού ενεργοποιηθεί η ανάλυση, υπολογίζονται οι μετρικές για κάθε μέθοδο και κλάση και εμφανίζονται κάτω από την όψη Metrics (Metrics View). Οι μετρήσεις που υπολογίζονται είναι οι εξής:

- Lines of code (LOC)
- Number of Static Methods (NSM)
- Afferent Coupling (CA)
- Normalized Distance (RMD)
- Number of Classes
- Specialization Index (SIX)
- Instability (RMI)
- Number of Attributes (NOF)
- Number of Packages (NOP)
- Method Lines of Code (MLOC)
- Weighted Methods per Class (WMC)
- Number of Overridden Methods (NORM)
- Number of Static Attributes (NSF)
- Nested Block Depth (NBD)

- Number of Methods (NOM)
- Lack of Cohesion of Methods (LCOM)
- McCabe Cyclomatic Complexity (VG or CC)
- Number of Parameters (PAR)
- Abstractness (RMA)
- Number of Interfaces (NOI)
- Efferent Coupling (CE)
- Number of children (NOC)
- Depth of Inheritance Tree (DIT)

Για ένα παράδειγμα της εκτέλεσης της ανάλυσης μέσω του εργαλείου Metrics εάν η τιμή της μετρικής είναι έξω από τα όρια του επιθυμητού πεδίου τιμών (range) τότε εμφανίζεται με κόκκινο χρώμα, διαφορετικά εμφανίζεται με μπλε χρώμα. Για κάθε μετρική υπολογίζεται η τιμή για κάθε μέθοδο του προγράμματος (εάν είναι μετρήσιμη σε αυτό το επίπεδο), η τιμή για κάθε κλάση (εάν είναι μετρήσιμη σε αυτό το επίπεδο), η συνολική τιμή της (στο πρόγραμμα), η μικρότερη τιμή της, η μεγαλύτερη τιμή της και η τυπική απόκλιση της (standard deviation). Επίσης, όπως αναφέραμε και προηγουμένως, το εργαλείο δημιουργεί το δυναμικό γράφο με τις εξαρτήσεις των κλάσεων και των πακέτων του έργου.

5.1.2.LocMetrics

Το εργαλείο LocMetrics είναι ένα δωρεάν εργαλείο στατικής ανάλυσης κώδικα το οποίο μπορεί να τρέξει και από την γραμμή εντολών. Παρέχει μια εύκολη και απλή διεπαφή και υπολογίζει 10 μετρικές σχετικές με τις γραμμές του κώδικα σε πηγαίο κώδικα στις γλώσσες C#, Java, C++, Sql.

Οι μετρικές που υπολογίζονται είναι οι εξής:

- Total lines of code (LOC): Συνολικές γραμμές κώδικα

- Blank lines of code (BLOC): Κενές γραμμές κώδικα
- Comment lines of code (CLOC): Σχολιασμένες γραμμές κώδικα
- Lines with both code and comments (C&SLOC): Γραμμές με κώδικα και σχόλια
- Logical source lines of code (SLOC-L): Λογικές γραμμές κώδικα
- McCabe VG complexity (CC): Κυκλωματική πολυπλοκότητα
- Number of comment words (CWORDS): Αριθμός σχόλιων λέξεων
- Physical executable source lines of code (SLOC-P): Εκτελέσιμες γραμμές κώδικα (οι συνολικές γραμμές κώδικα εκτός των κενών γραμμών και των σχολίων).
- Header Comment Lines of Code (HCLOC): Σχολιασμένες γραμμές στην επικεφαλίδα
- Header Commentary Words (HCWORD): Σχολιασμένες λέξεις στην επικεφαλίδα

Επίσης ο χρήστης μπορεί να εξάγει αναφορά σε μορφή HTML ή CSV μορφή με τις λεπτομέρειες της ανάλυσης καθώς και ένα διάγραμμα βασισμένο στην μετρική SLOCP.

5.1.3.SourceMonitor

Το εργαλείο SourceMonitor είναι ένα δωρεάν λογισμικό που αναπτύχθηκε από την εταιρεία Campwood Software και συγκεκριμένα από τον Jim Wanner το 1997. Το εργαλείο αναλύει στατικά λέξη-λέξη του κώδικα στις γλώσσες C++, C, C#, VB.NET, Java, Delphi, Visual Basic (VB6), HTML.

Οι μετρικές που υπολογίζονται είναι οι εξής:

- Number of Lines (LOC): ο αριθμός των φυσικών γραμμών σ' ένα αρχείο.
- Percent comments: Οι γραμμές που περιλαμβάνουν σχόλια με βάση τον συνολικό αριθμό γραμμών του κώδικα.
- Percent branches: Το ποσοστό εντολών διακλάδωσης.

- Average methods per class: Ο συνολικός μέσος όρος όλων των μεθόδων των κλάσεων ενός αρχείου ή ενός checkpoint. Το πηλίκο του συνολικού αριθμού μεθόδων δια το συνολικό αριθμό για τις οποίες βρίσκονται οι εφαρμογές μεθόδου.
- Average statements per method: Ο μέσος όρος εντολών ανά μέθοδο.
- Number of functions: Ο αριθμός των συναρτήσεων.
- Max Complexity: Η μέγιστη πολυπλοκότητα όλων των μεθόδων ή των συναρτήσεων μιας ρουτίνας.
- Max depth: Το μέγιστο βάθος όλων των μεθόδων.
- Average block depth: Το μέσο βάθος μίας ρουτίνας, δηλαδή το μέσο όρο των βαθών όλων των κλάσεων της ρουτίνας.
- Average Complexity: Ο μέσος όρος της πολυπλοκότητας όλων των μεθόδων ή των συναρτήσεων μιας ρουτίνας.

Οι μετρικές για όλα τα αρχεία ενός checkpoint εμφανίζονται σε έναν πίνακα. Με τον όρο checkpoint αναφερόμαστε σε μια συλλογή από αρχεία με τις μετρικές για όλα τα αρχεία που έχουν επιλεγεί κατά την δημιουργία του checkpoint. Ένας δεύτερος πίνακας παρουσιάζει τις τιμές των μετρικών για κάθε αρχείο ενός checkpoint. Επίσης, υπάρχει ένας πίνακας που παρουσιάζει τις μετρικές για όλες τις μεθόδους ενός checkpoint ή ενός επιλεγμένου υποσυνόλου αρχείων ενός checkpoint.. Τέλος, ένας λεπτομερής πίνακας περιέχει όλα τα δεδομένα που έχουν συλλεχθεί από ένα checkpoint ή ένα αρχείο. Ακόμη, ο χρήστης μπορεί να εξάγει αναφορά σε xml και csv μορφή, με τις λεπτομέρειες της ανάλυσης.

Επίσης υπάρχει η δυνατότητα αποθήκευσης του έργου σε αρχείο (project file).

5.1.4.PMD

Το PMD είναι ένα λογισμικό ανοικτού κώδικα για κώδικα γραμμένο σε Java το οποίο διατίθεται ως plugin για τα περιβάλλοντα προγραμματισμού Eclipse, Maven, NetBeans, JBuilder, JDeveloper, IntelliJ IDEA. Επίσης μπορεί να τρέξει και από την γραμμή εντολών. Αναπτύχθηκε από τον Tom Copeland και είναι ένα δημοφιλής εργαλείο στην Java κοινότητα.

Το PMD αναλύει στατικά πηγαίο κώδικα σε Java με βάση ενός συνόλου κανόνων με σκοπό να ανακαλυφτούν κοινά προγραμματιστικά λάθη (violations) που αναφέρουμε πιο κάτω. Αφού ενεργοποιηθεί η ανάλυση, υπολογίζονται αυτόματα οι ενεργοποιημένοι κανόνες και εμφανίζονται οι παραβιάσεις στην όψη (view) Violations Overview και στην όψη Violationsi. Τα λάθη επίσης επισημαίνονται και στον κώδικα στο Eclipse. Επίσης το plugin διαθέτει και δικό του Perspective για την καλύτερη επισκόπηση των αποτελεσμάτων της ανάλυσης.

Το PMD αναλύει τον κώδικα με βάση 25 σύνολα (set) από κανόνες. Συνολικά οι κανόνες είναι 260 και εντοπίζουν σημαντικά προβλήματα όπως: προβλήματα απόδοσης (‘νεκρός’ κώδικας κ.λπ.), υπερβολικά σύνθετες εκφράσεις, προβλήματα που αφορούν καλές πρακτικές (κανόνες σχεδιασμού, κανόνες ονομασίας κ.λπ.), προβλήματα που αφορούν την ορθότητα του κώδικα όπως η κλωνοποίηση του κώδικα κ.λπ.

Παρακάτω κάτω παρουσιάζονται τα 25 σύνολα κανόνων:

- Android: Κανόνες (3) που σχετίζονται με το εργαλείο Android SDK [3] .
- Basic: Κανόνες (23) που αφορούν καλές πρακτικές συγγραφής κώδικα όπως αποφυγή κενών catch μπλοκς (blocks), κενών if εντολών, αχρείαστων return εντολών, κ.λπ.
- Braces: Κανόνες (4) που αφορούν παραβιάσεις σχετικές με τη μη χρησιμοποίηση curly braces ({}) στις εντολές διακλάδωσης.
- Clone Implementation: Κανόνες (3) για τον εντοπισμό λανθασμένων χρήσεων της μεθόδου clone() όπως για παράδειγμα όταν η μέθοδος δεν κάνει throw το exception CloneNotSupportedException ή δεν υλοποιεί (implement) την διεπαφή Cloneable.
- Code Size: Κανόνες (11) που εντοπίζουν παραβιάσεις σχετικές με το μέγεθος του κώδικα ή την πολυπλοκότητα του. Το σύνολο αυτό εφαρμόζει μετρικές όπως η κυκλωματική πολυπλοκότητα (CC), ο αριθμός των δημόσιων (public) μεθόδων (NPM), ο αριθμός των μεθόδων (NOM), ο αριθμός των παραμέτρων (NOP), το μέγεθος της κλάσης, ο αριθμός γραμμών των κατασκευαστών (constructor), ο αριθμός γραμμών των μεθόδων (MLOC).

- Comments: Κανόνες (3) που εντοπίζουν παραβιάσεις σχετιζόμενες με τα σχόλια του κώδικα όπως ο αριθμός γραμμών των σχολίων (CLOC).
- Controversial: Κανόνες (23) που εντοπίζουν αμφιλεγόμενες παραβιάσεις όπως κάθε κλάση πρέπει να δηλώνει τουλάχιστον έναν κατασκευαστή (constructor), κάθε μέθοδος που δεν είναι void πρέπει να επιστρέφει μόνο μια τιμή (return) κ.λπ.
- Coupling: Κανόνες (5) σχετικοί με υψηλές συζεύξεις (coupling) μεταξύ αντικειμένων και πακέτων όπως η μετρική CBO.
- Design: Κανόνες (54) σχετικοί με τον σχεδιασμό του κώδικα. Για παράδειγμα δήλωση αφηρημένης (abstract) κλάσης χωρίς να υπάρχει αφηρημένη (abstract) μέθοδος, κλάσεις με μόνο ιδιωτικούς (private) κατασκευαστές να δηλώνονται final, κλάσεις με τον όρο 'god' που είναι αρκετά μεγάλες και πολύπλοκες πρέπει να διασπαστούν κ.λπ.
- Empty Code: Κανόνες (11) που αφορούν κενά μπλοκς (blocks), κενές εντολές διακλάδωσης, κενά try μπλοκς (blocks).
- Finalizer: Κανόνες (6) που αφορούν την μέθοδο finalize().
- Import Statements: Κανόνες (6) που αφορούν την εισαγωγή import εντολών. Για παράδειγμα πρέπει να αποφεύγεται η δήλωση 2 φορών του ίδιου import.
- J2EE: Κανόνες (9) σχετικοί με την J2EE αρχιτεκτονική.
- JavaBeans: Κανόνες (2) σχετικοί με το μοντέλο JavaBeans.
- Junit: Κανόνες (12) για την βιβλιοθήκη Junit. Για παράδειγμα οι κλάσεις που περιέχουν σενάρια ελέγχου (test cases) πρέπει να τελειώνουν με Test.
- Jakarta Commons Logging: Κανόνες (4) σχετικοί με το framework Jakarta Commons Logging.
- Java Logging: Κανόνες (5) για το Java Logging framework. Για παράδειγμα οι logger μεταβλητές πρέπει να δηλώνονται στατικές (static) και final.

- Migration: Κανόνες (14) σχετικοί με περιπτώσεις μετακίνησης από ένα JDK (Java Development Kid) σε άλλο.
- Naming: Κανόνες (20) σχετικοί με το πως πρέπει να ονομάζονται οι κλάσεις, οι μέθοδοι, οι μεταβλητές, τα πακέτα σύμφωνα με το PMD.
- Optimization: Κανόνες (12) για την βελτιστοποίηση του κώδικα βασισμένοι στις καλές πρακτικές. Για παράδειγμα μια μεταβλητή που της ανατίθεται τιμή μόνο μια φορά θα μπορούσε να είναι final.
- Strict Exceptions: Κανόνες (12) σχετικοί με το throw και catch exceptions. Για παράδειγμα αποφυγή του catch του exception Throwable.
- String and StringBuffer: Κανόνες (16) σχετικοί με τον τύπο String.
- Security Code Guidelines: Κανόνες (2) σχετικοί με την σωστή αποθήκευση πινάκων.
- Unnecessary: Κανόνες (7) για αχρείαστο κώδικα. Για παράδειγμα αχρείαστες παρενθέσεις θα πρέπει να αφαιρούνται.
- Unused Code: Κανόνες (5) για κώδικα που δεν χρησιμοποιείται.

Οι παραβάσεις που εντοπίζονται κατηγοριοποιούνται σε 5 επίπεδα προτεραιότητας ανάλογα με την σοβαρότητα της παραβίασης:

- Προτεραιότητα 1: Blocker (high error)
- Προτεραιότητα 2: Critical (medium high error)
- Προτεραιότητα 3: Urgent (medium error)
- Προτεραιότητα 4: Important (medium low error)
- Προτεραιότητα 5: Warning (low error)

Παράλληλα ο χρήστης έχει την δυνατότητα να καθορίσει ο ίδιος την προτεραιότητα για κάθε κανόνα.

Για κάθε στοιχείο (element) στην όψη Violations Overview για ένα παράδειγμα της εκτέλεσης της ανάλυσης, υπολογίζεται το πλήθος των παραβιάσεων, ο αριθμός των παραβάσεων ανά τις γραμμές του κώδικα (LOC) και ο αριθμός των παραβάσεων ανά τις μεθόδους (NOM).

Η όψη Violations Overview δείχνει το βαθμό προτεραιότητας της παραβίασης, την γραμμή που εντοπίστηκε την ημερομηνία που έγινε η ανάλυση, το όνομα του κανόνα της παραβίασης και ένα μήνυμα (error message) για το λάθος που εντοπίστηκε.

Επίσης το PMD δίνει τη δυνατότητα οι υψηλής προτεραιότητας παραβιάσεις να θεωρούνται σφάλματα του Eclipse, να απαιτείται δηλαδή η διόρθωση τους για να εκτελεστεί το πρόγραμμα., τον καθορισμό working set δηλαδή έργα που ακολουθούν τις ρυθμίσεις που καθορίζονται καθώς επίσης και την δυνατότητα ο χρήστης να γράψει τους δικούς του κανόνες χρησιμοποιώντας Xpath ή Java κλάσεις.

Επιπλέον, το PMD παρέχει τη δυνατότητα εντοπισμού κλώνων κώδικα με το Copy/Paste Detector που εφαρμόζει τον Karp-Rabin αλγόριθμο, κάτι που βέβαια ξεφεύγει από την στατική ανάλυση κώδικα.

Ο χρήστης πέρα από την ανάλυση, στην οποία μπορεί να επιλέξει ποιες παραβιάσεις να ενεργοποιηθούν, μπορεί να εξάγει αναφορές (reports) με τις λεπτομέρειες της ανάλυσης στις μορφές HTML, XML, csv.

5.1.5.Checkstyle

Το εργαλείο Checkstyle είναι ένα λογισμικό ανοικτού κώδικα το οποίο διατίθεται ως plugin για τα περιβάλλοντα προγραμματισμού Eclipse, Maven, NetBeans, IntelliJ IDEA κ.λπ. Αναπτύχθηκε από τον Oliver Burn το 2001.

Το Checkstyle αναλύει στατικά πηγαίο κώδικα σε Java με βάση ενός συνόλου κανόνων με σκοπό να ανακαλυφτούν κοινά προγραμματιστικά λάθη (violations) που αναφέρουμε πιο κάτω. Αφού ενεργοποιηθεί η ανάλυση οι παραβιάσεις εμφανίζονται στην όψη (view) Checkstyle Violations. Τα λάθη επίσης επισημαίνονται και στον κώδικα στο eclipse.

Το Checkstyle αναλύει τον κώδικα με βάση 126 ελέγχους που κατηγοριοποιούνται σε 14 κατηγορίες (modules). Οι προκαθορισμένες (default) ρυθμίσεις δεν έχουν ενεργοποιημένους όλους τους ελέγχους. Ο χρήστης μπορεί να επιλέξει τους ελέγχους που επιθυμεί δημιουργώντας το δικό του configuration.

Παρακάτω παρουσιάζονται οι 14 κατηγορίες ελέγχου:

- Annotations: Έλεγχοι (5) σχετικοί με annotations.
- Block Checks: Έλεγχοι (5) σχετικοί με κενά μπλοκς (blocks).
- Class design: Έλεγχοι (8) για την ορατότητα των μελών της κλάσης. Για παράδειγμα μόνο τα μέλη (μέθοδοι, μεταβλητές) που είναι static final πρέπει να είναι δημόσια (public) ή μια κλάση που έχει μόνο ένα ιδιωτικό (private) κατασκευαστή πρέπει να δηλωθεί final κλπ.
- Coding: Έλεγχοι (43) σχετικοί με το πως πρέπει να συγγράφεται ο κώδικας. Για παράδειγμα, κάθε δήλωση switch πρέπει να δηλώνει και default πεδίο πάντα στο τέλος (μετά από όλα τα cases), κάθε μεταβλητή που η τιμής της δεν έχει αλλάξει ποτέ στο πρόγραμμα πρέπει να δηλώνεται final, δεν πρέπει να δηλώνονται 'magic numbers', δηλαδή μεταβλητές που έχουν μια ακέραια σταθερή τιμή θα πρέπει να δηλώνονται ως σταθερές, κάθε κλάση πρέπει να έχει κατασκευαστή, σε κάθε γραμμή πρέπει να είναι δηλωμένη μόνο μια γραμμή κ.λπ.
- Duplicate code: Έλεγχος γραμμή γραμμή του κώδικα για διπλότυπο κώδικα.
- Headers: Έλεγχοι (2) σχετικά με τα headers. Για παράδειγμα κάθε αρχείο ξεκινά με ένα συγκεκριμένο Header.
- Imports: Έλεγχοι (7) σχετικά με την εισαγωγή imports. Για παράδειγμα καμιά import εντολή δεν χρησιμοποιεί το σύμβολο *.
- Javadoc comments: Έλεγχοι (6) σχετικά με τα Javadoc σχόλια. Για παράδειγμα κάθε πακέτο πρέπει να περιέχει Javadoc σχόλια.
- Metrics: Έλεγχος για τις εξής μετρικές:

- Boolean Expression Complexity: Περιορισμός των αριθμών των &&, | |, &, | και ^ σε μια έκφραση.
- Class Data Abstraction Coupling: Η πολυπλοκότητα της σύζευξης που προκαλείται από Αφηρημένους Τύπους Δεδομένων (Abstract Data Types, ADTs).
- Class Fan Out Complexity: Ο αριθμός των υπόλοιπων κλάσεων που εξαρτώνται από την συγκεκριμένη κλάση.
- Cyclomatic Complexity (CC ή VG): Κυκλωματική Πολυπλοκότητα.
- N path Complexity: Ο αριθμός των πιθανών μονοπατιών εκτέλεσης σε μια συνάρτηση (μέγιστη default τιμή=200).
- Non Commenting Source Statements (NCLOC): Ο αριθμός των μη σχολιασμένων γραμμών για κάθε μέθοδο (μέγιστη default τιμή=50), κλάση (μέγιστη default τιμή=1500) και αρχείο (μέγιστη default τιμή=2000).

- Miscellaneous: Διάφοροι έλεγχοι (13) όπως αν το αρχείο τελειώνει με τον χαρακτήρα '\n' (καινούργια γραμμή), αν η δήλωση ενός πίνακα είναι με το στυλ όπως καθορίζει η Java, String[] args (true) ή το στυλ της C String args[] (false).
- Modifiers: Έλεγχοι (2) για τους τύπους. Για παράδειγμα αν ακολουθούν την ιεραρχική δομή που καθορίζεται από το Java Language specification [18].
- Naming Conventions: Έλεγχοι (12) για το αν το ονόματα των μεταβλητών, μεθόδων, παραμέτρων, κλάσεων, διεπαφών και πακέτων ακολουθούν μια συγκεκριμένη δομή. Για παράδειγμα τα ονόματα των μεταβλητών πρέπει να είναι στο όριο ^[a-z][a-zA-Z0-9]*$.
- Regexp: Έλεγχοι (3) σχετικοί με τις κανονικές εκφράσεις. Για παράδειγμα έλεγχος για την ανίχνευση μιας γραμμής που ταιριάζει με μια συγκεκριμένη κανονική έκφραση (regular expression).

- Size violations: Έλεγχοι (8) για μετρικές σχετικές με το μέγεθος. Για παράδειγμα το μέγεθος του αρχείου (μέγιστος αριθμός γραμμών, LOC default τιμή = 2000), το μέγεθος της γραμμής (μέγιστος αριθμός γραμμών, LOC default τιμή = 80), το μέγεθος της μεθόδου (μέγιστος αριθμός γραμμών, MLOC default τιμή = 150), ο αριθμός των παραμέτρων σε μια μέθοδο (μέγιστος αριθμός παραμέτρων, default τιμή = 7).
- Whitespace: Έλεγχοι (12) για κενά διαστήματα. Για παράδειγμα δεν πρέπει να υπάρχει ο χαρακτήρας tab (‘\t’) στο αρχείο, πρέπει να υπάρχει κενό διάστημα μετά ή πριν από κάποιο σημείο (token).

Για κάθε παραβίαση εμφανίζεται ο τύπος της παραβίασης και ο συνολικός αριθμός που εμφανίστηκε η παραβίαση. Επιλέγοντας την παραβίαση εμφανίζεται το αρχείο που βρέθηκε η παραβίαση, ο φάκελος, η γραμμή και το μήνυμα της παραβίασης.

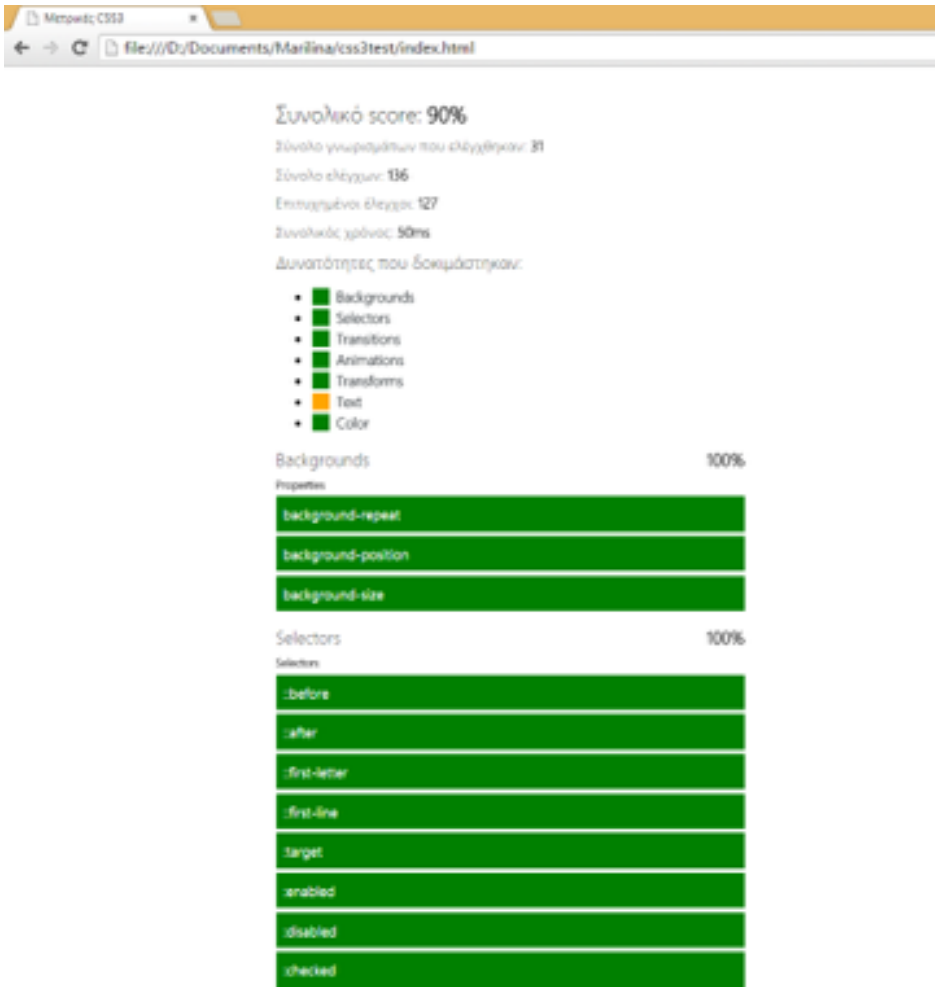
5.2.Ανάπτυξη Εργαλείου Μέτρησης Μετρικών σε CSS Κώδικα

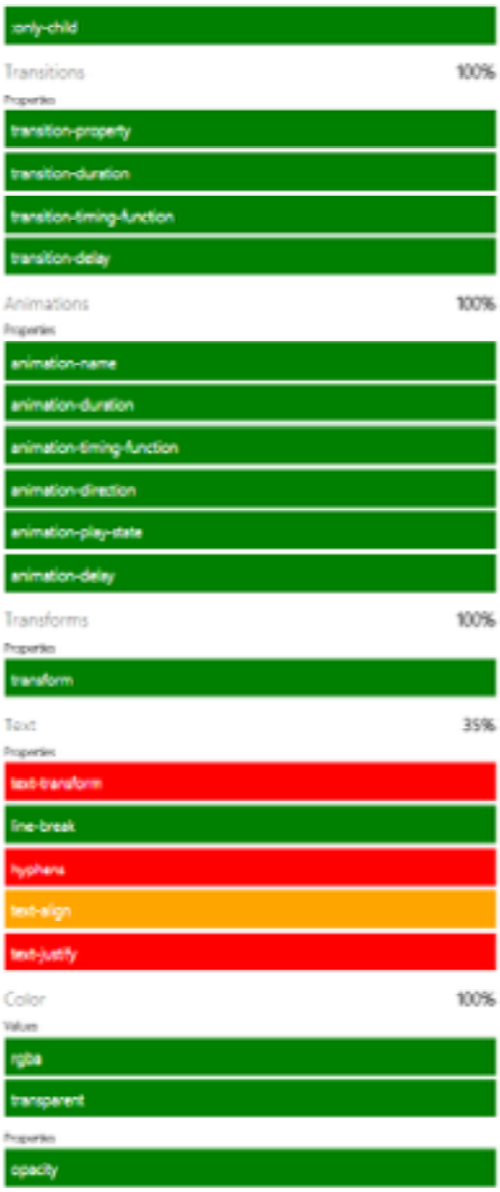
Το εργαλείο που αναπτύχθηκε με JavaScript ελέγχει την υποστήριξη των browsers για συγκεκριμένες ιδιότητες της CSS3. Οι ιδιότητες που επιλέχθηκαν, προσπαθώντας να καλύψουν όσο το δυνατό μεγαλύτερο φάσμα της γλώσσας CSS3, είναι οι ακόλουθες:

- Backgrounds
- Selectors
- Transitions
- Animations
- Transforms
- Text
- Color

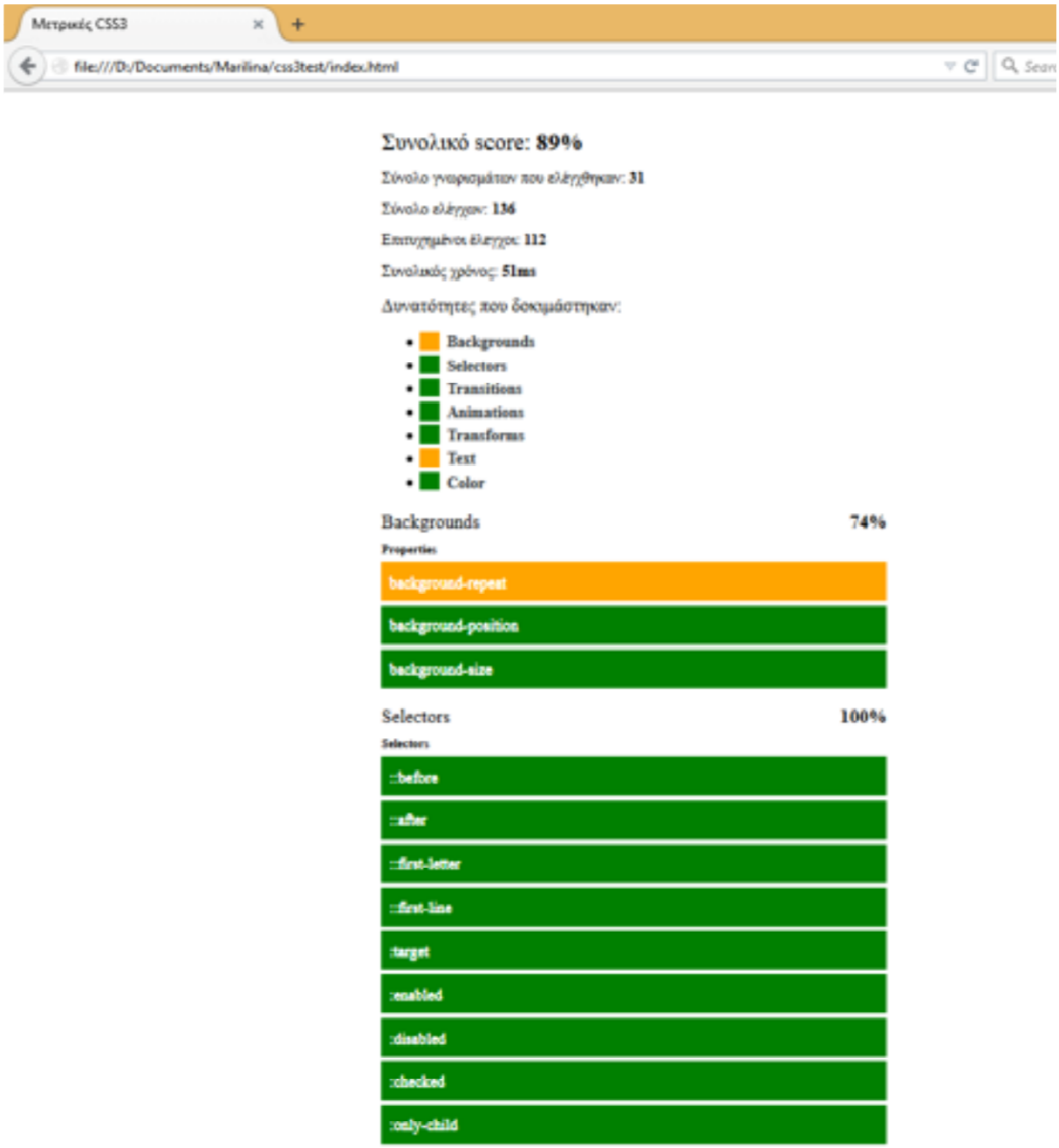
Κάθε έλεγχος συνεισφέρει με τον ίδιο συντελεστή στην τελική βαθμολογία και εμφανίζεται σαν ποσοστό. Η λειτουργία του ουσιαστικά είναι να ελέγχει εάν το συντακτικό που ορίζουμε σαν είσοδο αναγνωρίζεται από τον browser. Αυτό είναι φυσικά ανεξάρτητο από το πώς το χρησιμοποιεί ο browser, που δε μας απασχολεί στο συγκεκριμένο εργαλείο.

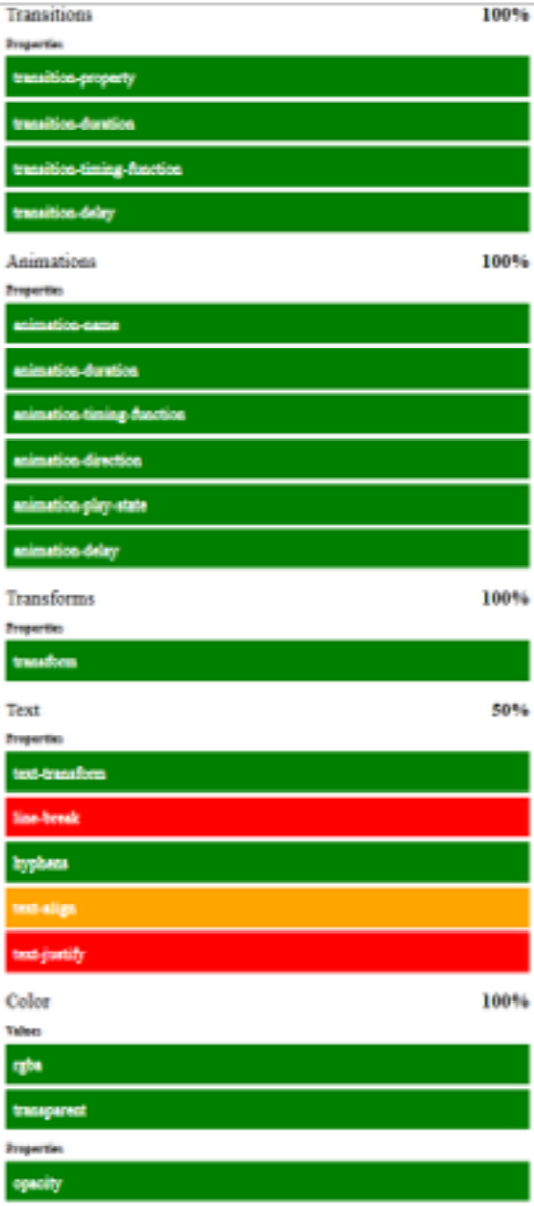
Μέτρηση στον browser Google Chrome:




:only-child
Transitions
100%
Properties
transition-property
transition-duration
transition-timing-function
transition-delay
Animations
100%
Properties
animation-name
animation-duration
animation-timing-function
animation-direction
animation-play-state
animation-delay
Transforms
100%
Properties
transform
Text
35%
Properties
text-transform
line-break
hyphens
text-align
text-justify
Color
100%
Values
rgba
transparent
Properties
opacity

Μέτρηση στον browser Mozilla Firefox:





5.2.Μετρήσεις στο stylestats.org

Total unique font families

Όταν χρησιμοποιούμε πολλές διαφορετικές γραμματοσείρες η σελίδα αργεί .Ένα τέτοιο παράδειγμα είναι το booking.com που χρησιμοποιεί 19 γραμματοσειρές σε αντίθεση με το bayer.com που χρησιμοποιεί μόνο μια.

Stylesheets

Πολλές φορές χρησιμοποιούμε διαφορετικά αρχεία css σε ένα website για να μπορεί να διαβάζεται κάθε φορά ο κώδικας που χρειάζεται και όχι όλος ο κώδικας του site. Αν μερικοί κανόνες έχουν ορισθεί με την ίδια κλάση σε διαφορετικά stylesheets , θα χρησιμοποιηθεί η κλάση από το τελευταίο stylesheet. Για παράδειγμά το http://www.-cied.teiwest.gr/ διαβάζει 22 διαφορετικά αρχεία css ενώ το http://www.realmadrid.-com/en μόνο ένα

Size

Το αρχείο css του picmonkey.com έχει μέγεθος 1228.8kb ενώ το instagram.com μόλις 5.9kb. Το instagram είναι ένα πολύ απλό εμφανησιακά website αφού βασίζεται στην φωτογραφία και δεν έχει πολλά γραφικά. Το picmonkey χρησιμοποιεί πολλούς κλανόνες css για να αλλάξει τις εικόνες, όπως opacity, shadow, cepia κλπ

Total Unique colors

Το booking.com χρησιμοποιεί 207 διαφορετικά χρώματα. Το sklavenitis.gr που είναι μια σελίδα υπό κατασκευή, χρησιμοποιεί μόνο ένα χρώμα.

Data url size

Το bbc.com έχει data url 16,7kb ενώ το ryanair.com 0,0293kb.

Rules

Το booking.com έχει 9894 κανόνες css ενώ το sklavenitis.gr μόλις 3.
Είναι καλό για την ταχύτητα του website, να χρησιμοποιούμε όσο το δυνατό λιγότερους κανόνες μπορούμε. Προκειμένου να μην κάνουμε εκπτώσεις στην εμφάνιση του web-site, πρέπει να δίνουμε προσοχή στην σωστή ομαδοποίηση των selectors ώστε να αποφεύγουμε να γράφουμε τους ίδιους κανόνες 2 φορές.

Selectors

Το picmonkey.com έχει 16859 selectors ενώ το pilavas.gr έχει 89 selectors.

Simplicity %
Το pilavas.gr έχει Simplicity 98,9% ενώ το picmonkey.com 31.4%.

Id Selectors
Το picmonkey.com έχει 6089 διαφορετικά id. Το nova.com έχει μόνο 4 διαφορετικά id

Universal Selectors
Το http://www.realmadrid.com/en έχει 1028 universal selectors ενώ το ryanair.com μόνο 1. Universal selectors είναι κάποια selectors τα οποία μπορούμε να χρησιμοποιούμε χωρίς να τα ορίζουμε σαν id στην html όπως πχ τα body, head, footer κλπ.

Unqualified Attribute Selectors
Το picmonkey.com έχει 346 ενώ το www.imdb.com μόνο 6.

Total Unique Font Sizes
Το http://www.realmadrid.com/en χρησιμοποιεί 132 διαφορετικά μεγέθη γραμματοσειρών, ενώ το www.xo.gr μόνο 4. Καλό είναι να αποφεύγουμε τις τόσες

Important Keywords
Το booking.com έχει χρησιμοποιήσει 1067 το Important ενώ το http://www.national-geographic.com/ μόνο 9. Χρησιμοποιούμε το !important σε ένα κανόνα όταν θέλουμε να "επιβληθεί" σε κάποιο κανόνα inline ή κανόνα άλλου stylesheet. Όταν ένα αρχείο έχει πολλές φορές το !important πιθανό να συμβαίνει επει ΄δη έχουν πραγματοποιηθεί πολλές αλλαγές. Καλό είναι να αποφεύγουμε να το χρησιμοποιούμε

Βιβλιογραφία

[1] Μ. Βέρου, "Εισαγωγή στο CSS."

[2] Κ. Γ, "Δημιουργία Μηχανής Αναζήτησης Προσώπων Στο Social Web," 2010.

[3] Β. Γαβράς, "Ποιότητα Λογισμικού," 2011.

[4] Ν. Πέτρου, "Σύγκριση Εργαλείων της Βιομηχανίας Λογισμικού για Στατική Ανάλυση Κώδικα," 2014.

[5] Π. Πέρδικα, "Διεξαγωγή μετρήσεων ποιότητας με στόχο τη βελτίωση της συντηρησιμότητας σε λογισμικό αλληλεπίδρασης με Βάση Δεδομένων," 2005.

[6] Σ. Κατωπόδης, "Αξιολόγηση και Διασφάλιση Ποιότητας Λογισμικού," 2009.

[7] "Ποιοτικά χαρακτηριστικά, μετρικές και εργαλεία για CSS3," 2014.

[8] Σ. Ναθαναήλ, "Πολύτροπες και Προσαρμοστικές διεπαφές για τον Παγκόσμιο Ιστό," 2005.

[9] D. S. Mcfarland, *CSS3: The missing manual*, 3rd ed. 2012.