ΠΤΥΧΙΑΚΗ ΜΕΛΕΤΗ

Θέμα: «ΑΝΑΛΥΣΗ ΤΗΣ ΑΠΟΔΟΣΗΣ ΠΡΟΓΡΑΜΜΑΤΩΝ JQUERY/JAVASCRIPT ΣΕ SMARTPHONES»

Ονοματεπώνυμο : Πυλαρινός Δημήτριος **Αρ. Μητρώου :** 0481

Επιβλέπων καθηγητής: Χριστοδούλου Σωτήριος

ΑΝΤΙΡΙΟ, 2017

<u>Η πτυχιακή αυτή εγκρίθηκε από την ακόλουθη τριμελή εξεταστική επιτροπή:</u>

- **Ονοματεπώνυμο:**……..

  Υπογραφή……..

- **Ονοματεπώνυμο:**……..

  Υπογραφή……..

- **Ονοματεπώνυμο:**……..

  Υπογραφή……..

ΠΕΡΙΛΗΨΗ

Η συγκεκριμένη μελέτη, παρουσιάζει αναλυτικά την απόδοση των ιστοσελίδων σε Smartphones και Tablets, αξιοποιώντας μια από τις πιο δημοφιλείς βιβλιοθήκες της JavaScript, την jQuery. Η εμπειρική ανάλυση, υλοποιήθηκε με τη βοήθεια τριών ηλεκτρονικών συσκευών, ένα Smartphone (RAM: 2GB - CPU: Quad-core 2.26 GHz Krait 400) και δυο Tablets (RAM: 1.5GB - CPU: Dual-core 1.5 GHz Cortex-A9/ RAM: 1GB - CPU: Quad-core 1.2 GHz Cortex-A9), στα οποία εφαρμόστηκαν 120 τεστ, σε τρία διαφορετικά μεγέθη DOM (Document Object Model). Τα τεστ αυτά υλοποιήθηκαν από δυο φορές σε κάθε συσκευή για την επίτευξη μεγαλύτερης ακρίβειας, με την χρήση της ιστοσελίδας jsPerf. Τα αποτελέσματα της έρευνας βασίστηκαν αποκλειστικά στον χρόνο εκτέλεσης της κάθε μεθόδου. Βάση των αρχικών αποτελεσμάτων προέκυψε ότι ο χρόνος αυτός, αντί να αυξάνεται ανάλογα με την αύξηση του μεγέθους του DOM, αντιθέτως μειωνόταν. Εξαιτίας όμως των ενδείξεων αυτών εκτελέστηκαν επιπρόσθετα οι χρόνοι των μεθόδων για κάθε selector ξεχωριστά, μετρώντας και μεμονωμένα την απόδοση του καθενός. Μέσω της προσέγγισης αυτής αποδείχθηκε ότι ο τελικός χρόνος εκτέλεσης μιας ιστοσελίδας, στην οποία εφαρμόζουμε μεθόδους jQuery επηρεάζεται σε ελάχιστο βαθμό από την εκάστοτε μέθοδο και αρκετά περισσότερο από τον selector που χρησιμοποιείται.

Λέξεις κλειδί: DOM, JavaScript, jQuery, selector.

ABSTRACT

This specific study presents the performance of web pages on smartphones and tablets, making use of the most popular Javascript library, jQuery. The empirical analysis was implemented with the assistance of three electronic devices, one Smartphone (RAM:2GB-CPU: Quad-core 2.26 GH2 Krait 400) and two tablets (RAM:1.5 GB-CPU:Dual-core 1.5 GH2 Cortex -A9/RAM: 1GB-CPU:Quad-core 1.2 GH2 Cortex-A9), in which 120 tests were applied in three different sizes DOM (Document Object Models). These specific tests were implemented twice on each device to achieve greater accuracy with the use of the website jsPerf. The results of this study were solely based on the implementation time of each method. Considering the initial results, it is indicated that this time instead of being enhanced, depending on the increase of the DOM size, on the contrary, it decreased. Owing to these indications the times of these methods were additionally conducted for each selector separately, measuring individually their performance. Through this approach, it was proved that the final implementation time of one web page in which we perform jQuery methods, it is affected to a minimum by each method and much more by the selector which is used.

Keywords: DOM, JavaScript, jQuery, selector.

ΠΕΡΙΕΧΟΜΕΝΑ

ΕΙΣΑΓΩΓΗ

Η παρούσα πτυχιακή μελέτη που εκπονείται στο πλαίσιο της πτυχιακής μου εργασίας έχει ως αντικείμενο την έρευνα και εξακρίβωση των αιτιών που προκαλούν καθυστέρηση στην εκτέλεση μιας ιστοσελίδας ή μιας εφαρμογής διαδικτύου σε ηλεκτρονικές συσκευές όπως Smartphones και Tablets. Η επιλογή του συγκεκριμένου αντικείμενου έγινε με γνώμονα την ραγδαία αύξηση της χρήσης του διαδικτύου και του αριθμού ιστοσελίδων που ενσωματώνουν γραφικά στοιχεία, βίντεο, εικόνες, διαδραστικές εφαρμογές, πεδία επικοινωνίας (chatbox) ακόμα και τεχνική νοημοσύνη (chatbots).

Η Javascript και κατά επέκταση η βιβλιοθήκη jQuery κατέχουν σημαντική θέση στις τεχνολογίες που έχουν αναπτυχθεί ώστε να είναι δυνατή η κατασκευή ιστοσελίδων εμπλουτισμένων με πολλά διαφορετικά χαρακτηριστικά. Ο μεγάλος αριθμός ιστοσελίδων και εφαρμογών που χρησιμοποιούν την συγκεκριμένη βιβλιοθήκη οδήγησε στην ανάγκη της αναζήτησης εκείνων των μεθόδων που δημιουργούν το μεγαλύτερο πρόβλημα κατά την εκτέλεση τους. Αφού εξαιτίας των περιορισμένων χαρακτηριστικά των προαναφερθέντων συσκευών, οι διάφοροι χρήστες αρκετές φορές δυσκολεύονται με την εμφάνιση των ιστοσελίδων που επιθυμούν.

Για την επίτευξη των αποτελεσμάτων, τα οποία αναδεικνύουν ποιες μέθοδοι έχουν το χειρότερο χρόνο εκτέλεσης, δημιουργήθηκαν αρχικά περιπτώσεις jQuery μεθόδων, για την κατηγορία Manipulation και στην συνέχεια με την βοήθεια ειδικών εργαλείων οι μέθοδοι αυτοί εκτελέστηκαν σε συγκεκριμένα διαφορετικά μεγέθη DOM. Συλλέγοντας τα αποτελέσματα αρχικά παρατηρήθηκε ότι υπήρχαν σημαντικές διαφορές στον συνολικό χρόνο της κάθε μεθόδου σε σημείο που σε ορισμένες περιπτώσεις φαινόταν να μην επηρεάζονται ακόμη και από το μέγεθος του DOM. Όντας στατιστικά τα οποία δεν ήταν θεωρητικά αναμενόμενα η έρευνα επεκτάθηκε στην εξακρίβωση των σημείων της κάθε μεθόδου που προξενεί αυτή την συμπεριφορά.

Βάση της μέτρησης των αρχικών χρόνων του κάθε selector δόθηκε πλέον προσοχή σε κάθε selector ξεχωριστά στον οποίο εφαρμοζόταν η κάθε μέθοδος. Με αυτή την προσέγγιση φάνηκε ότι ο μέσος όρος εκτέλεσης της μεθόδου επηρεαζόταν από συγκεκριμένους selectors που κατανάλωναν περισσότερο χρόνο. Αυτές οι παρατηρήσεις παρουσιάστηκαν αναλυτικά μέσα από δύο διαφορετικά είδη διαγραμμάτων, ένα με γνώμονα τον χρόνο εκτέλεσης ανά έναν selector και ένα για τον συνολικό αριθμό των selectors που υπάρχουν. Συνεπώς απώτερος σκοπός της μελέτης αποτέλεσε η γνωστοποίηση όλων των μεθόδων που παρουσιάζουν μεγάλη καθυστέρηση στην ιστοσελίδα ή εφαρμογή κατά την εισαγωγή τους.

1.1 Ιστορική εξέλιξη της Javascript

Το 1995, ο αμερικάνος Brendan Eich, εργαζόμενος στην Netscape Communications Corporation, δημιούργησε μια νέα γλώσσα προγραμματισμού την JavaScript, η οποία βασιζόταν στη γλώσσα προγραμματισμού C και θα απευθυνόταν κυρίως σε μη επαγγελματίες προγραμματιστές, οι οποίοι θα είχαν

την δυνατότητα να «τρέξουν» σε ένα υπολογιστικό περιβάλλον αρκετές εφαρμογές. Αν και η JavaScript αρχικά ήταν γνωστή ως Mocha και στην συνέχεια ως LiveScript τελικά εδραιώθηκε ως JavaScript για καθαρά διαφημιστικούς λόγους.

Αρκετοί θεωρούσαν ότι η JavaScript είναι παρόμοια με την Java, όμως αυτές οι δυο γλώσσες προγραμματισμού δεν μοιάζουν στο τρόπο σύνταξή τους, ούτε έχουν κοινά σημεία μεταξύ τους. Την εσφαλμένη αντίληψη ως προς την JavaScript, ενίσχυε το γεγονός ότι αρκετές κυρίαρχες εταιρείες στο χώρο της τεχνολογίας, όπως η Adobe και η Google προσπαθούσαν κατά καιρούς να την αντικαταστήσουν. Όμως η JavaScript κατάφερε να γίνει ευρέως διαδεδομένη και να εδραιωθεί ως μια από τις σημαντικότερες γλώσσες προγραμματισμού, εξαιτίας κυρίως της δημιουργίας της τεχνολογίας Ajax. Η τεχνολογία αυτή επέτρεψε στην JavaScript να επιστρέψει στο προσκήνιο και να επιφέρει μεγαλύτερη επαγγελματική προσοχή.

Η εκτενέστερη χρήση όμως (94,6% των ιστοσελίδων σήμερα), οδήγησε στην εξάπλωση των δομών (Frameworks), των βιβλιοθηκών (Libraries) όπως την Dojo, την jQuery και την React, καθώς και τη βελτιστοποίηση του τρόπου προγραμματισμού σε JavaScript. Το 2009 εμφανίστηκε η NodeJs, έως τότε η Javascript χρησιμοποιείτο μόνο για προγραμματισμό τοπικά στον φυλλομετρητή του πελάτη (client), για αυτό και θεωρείται κατά βάση ως μια client-side γλώσσα προγραμματισμού. Αυτό επεξηγεί γιατί δεν πραγματοποιούνται στον διακομιστή (server), η επεξεργασία του κώδικα Javascript και η παραγωγή του τελικού περιεχομένου HyperText Markup Language (HTML).

1.2 Ο ορισμός και η χρήση της jQuery

Το Ιανουάριο του 2006, ο John Resig επηρεασμένος από την βιβλιοθήκη cssQuery του Dean Edwards, δημιούργησε μια νέα βιβλιοθήκη, την jQuery, συμβατή με όλους τους φυλλομετρητές (browsers). Η jQuery θεωρείται μια από τις πιο δημοφιλείς βιβλιοθήκες της JavaScript, αφού τα τωρινά στατιστικά δεδομένα αναφέρουν ότι χρησιμοποιείται από το 66,3% [1] .όλων των ιστοσελίδων παγκοσμίως. Αυτή την στιγμή διατηρείται από μια ομάδα προγραμματιστών με επικεφαλή τον Timmy Willison, υπό την καθοδήγηση του Richard Gibson. Σχεδιάστηκε για να απλοποιεί το client-side scripting της HTML και να υλοποιεί οποιαδήποτε διαδικτυακή εφαρμογή, ανεξαρτήτως από το επίπεδο δυσκολίας της σε μειωμένο χρονικό διάστημα. Η jQuery δηλαδή χρησιμοποιείται αποκλειστικά για την δημιουργία ιστοσελίδων και web εφαρμογών, αφού παρέχει την δυνατότητα πρόσθεσης εφέ και κίνησης, αύξησης της διαδραστικότητας του χρήστη, καθώς και την δυνατότητα παραμονής του στην ίδια σελίδα ακόμη και όταν τροποποιείται το υπάρχον περιεχόμενό της. Επομένως σκοπός της βιβλιοθήκης αυτής, είναι να διευκολύνει την χρήση της Javascript στις ιστοσελίδες, εξαλείφοντας τυχόν προβλήματα που δημιουργούνται, όπως τα προβλήματα συμβατότητας που υπάρχουν μεταξύ των διαφόρων browsers της αγοράς.

1.3 Τα βασικά πλεονεκτήματα της jQuery

Όντας γλώσσα προγραμματισμού ανοιχτού κώδικα η jQuery υποστηρίζεται από πολλούς προγραμματιστές και ενδείκνυται στη δημιουργία πιο πολύπλοκων λειτουργιών που μπορούν να ενσωματωθούν σε μια ιστοσελίδα. Η jQuery υποστηρίζεται από μεγάλη ποικιλία plugin (πρόσθετων), όπως αλλαγή στα περιθώρια μιας ιστοσελίδας μέχρι αναπαραγωγή ήχου και εικόνας. Το θετικό της jQuery είναι ότι ο προγραμματιστής δεν χρειάζεται να γράψει διαφορετικό κώδικα για κάθε φυλλομετρητή επειδή η δυνατότητα cross-handling εξαλείφει το πρόβλημα αυτό.

Δηλαδή, η jQuery έχει την ευχέρεια να μετατρέπει τις πολύπλοκες γραμμές κώδικα Javascript σε μόνο μία γραμμή, και να απλοποιεί τον χειρισμό του DOM (Document Object Model) και την χρήση της Ajax (Asynchronous JavaScript and XML), αφού δημιουργεί αυτόματη συμπλήρωση κειμένου που αντλούν δεδομένα από μόνο μία βάση. Ένα ακόμα βασικό στοιχείο της είναι ότι λειτουργεί ακόμα και στην περίπτωση που η Javascript είναι απενεργοποιημένη στο πελάτη (client). Η jQuery είναι αρκετά φιλική και στο SEO (Search Engine Optimization), αφού ο χρόνος εκτέλεσης είναι μικρός, βασικό στοιχείο της λειτουργίας των SEO.

Ένα πολύ σημαντικό δεδομένο για τις σύγχρονες ιστοσελίδες είναι ότι η jQuery μπορεί να χρησιμοποιηθεί, σε διαφορετικό αρχείο από την ιστοσελίδα (HTML), όπως ακριβώς και η CSS (Cascading Style Sheet). Αυτό δίνει την δυνατότητα άμεσων αλλαγών χωρίς την ανάγκη αναδρομής σε όλο το περιεχόμενο της σελίδας. Επιπρόσθετα, η επιλογή να γίνεται "φόρτωμα" συγκεκριμένων div tags όπου απαιτούνται, βελτιώνει κατά πολύ τον χρόνο εκτέλεσης της σελίδας. Παρέχοντας την δυνατότητα στον προγραμματιστή να ορίσει ποια στοιχεία επιθυμεί να εμφανιστούν πρώτα με βάση την σπουδαιότητά της. Απόρροια αυτού ειδικά σε μεγάλες ιστοσελίδες είναι η εμφάνιση αρχικά των βασικών πληροφοριών της και έπειτα όποια άλλα στοιχεία περιλαμβάνονται, ώστε ο χρήστης να μην περιμένει αρκετό χρόνο μέχρι να γίνει πλήρης «φόρτωση» της ιστοσελίδας.

Σε αυτό το σημείο αξίζει να αναφερθεί ότι η jQuery είναι συμβατή με όλες τις σύγχρονες εκδόσεις φυλλομετρητών (browsers) σε φορητούς και επιτραπέζιους ηλεκτρονικούς υπολογιστές (PC και Laptop) αλλά και σε όλες τις συσκευές όπως Smartphones και Tablets. Ειδικότερα για τις δυο τελευταίες κατηγορίες συσκευών υπάρχει μεγάλη συμβατότητα ακόμα και σε απαρχαιωμένα λειτουργικά συστήματα και φυλλομετρητές όπως παράδειγμα στην έκδοση Android Eclair 2.1. (τελευταία έκδοση σήμερα 7.1.2 Nougat).

1.4 Η σύνταξη της jQuery

Ο τρόπος σύνταξης της jQuery είναι φτιαγμένος έτσι ώστε η επιλογή και η εκτέλεση στοιχείων σε κώδικα HTML να γίνεται πιο εύκολα. Συγκεκριμένα η γενική σύνταξή της έχει την ακόλουθη δομή:

```
$(selector).action();
```

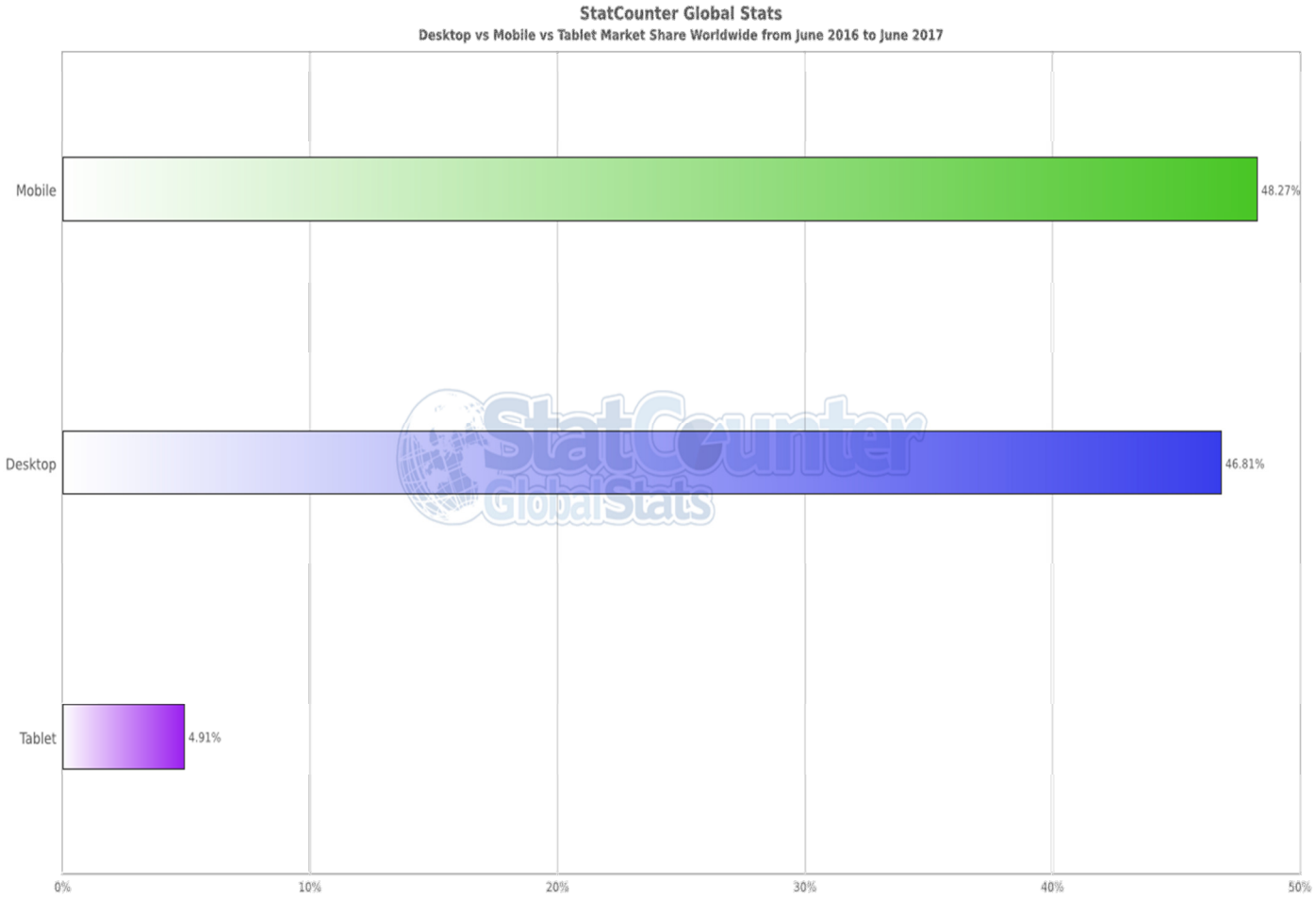
Μία τυπική «δήλωση» (statement) ξεκινάει με το σύμβολο δολάριο ($), το οποίο δηλώνει ότι ο κώδικας που ακολουθεί είναι jQuery, και ολοκληρώνεται με το ελληνικό ερωτηματικό (;), το όποιο είναι ευρύτερα γνωστό ως «semicolon». Επειδή η jQuery είναι μία βιβλιοθήκη της Javascript ο κώδικας εκτέλεσης θα πρέπει να περικλείετε μέσα σε ένα στοιχείο <script> ή να καλείτε από ένα εξωτερικό αρχείο Javascript.

Επιπρόσθετα ένα άλλο βασικό στοιχείο της jQuery είναι η ευκολία επιλογής και διαχείρισης των selectors βάση του id, της κλάσης, των χαρακτηριστικών ή του ονόματος του. Πριν την δημιουργία της, για να επιτευχθεί αυτό απαιτούνταν πολύπλοκος και μεγάλος κώδικας κάτι το οποίο πλέον έχει εξαλειφθεί.

1.5 Η σημερινή εφαρμογή της jQuery

Σε μια εποχή που όλα γίνονται μέσω ιντερνέτ παρατηρείτε μία αύξηση [2] στην δημιουργία ιστοσελίδων και εφαρμογών διαδικτύου. Νέες τεχνολογίες εμφανίζονται συνεχώς στην προσπάθεια βελτιστοποίησης της απόδοσης και εισαγωγής περισσοτέρων επιλογών στην κατασκευή τους. Απόρροια αυτών των τεχνολογιών είναι η δημιουργία όλο και μεγαλύτερων ιστοσελίδων, σε κώδικα. Αυτό έχει ως συνέπεια την αύξηση του χρόνου που χρειάζεται για να εμφανιστεί μία ιστοσελίδα στην οθόνη του χρήστη. Αν σε αυτό προστεθούν και τα χαμηλά χαρακτηριστικά της συσκευής που χρησιμοποιείτε, η καθυστέρηση φόρτωσης της σελίδας μπορεί να φτάσει σε σημείο όπου ο χρήστης θα την εγκαταλείψει.

Για παράδειγμα σε έρευνα που έχει υλοποιηθεί [3], σχετικά με την επίδοση της ιστοσελίδας ηλεκτρονικού εμπορίου παρατηρήθηκε ότι το 47% των πελατών περίμεναν να γίνει πλήρης φόρτωση της σελίδας σε λιγότερο από δύο δευτερόλεπτα. Ακόμη πιο αξιοσημείωτο είναι ότι στην ίδια έρευνα για 3 δευτερόλεπτα αναμονής μέχρι να εμφανιστεί η σελίδα, η συνολική ικανοποίηση του χρήστη μειωνόταν κατά 16%.



Εικόνα 2 : Μερίδιο αγοράς σε κινητά, επιτραπέζιους υπολογιστές και tables παγκοσμίως

Αυτό μπορεί να οδηγήσει ακόμα και σε χαμένα κέρδη για μια εταιρία αυτού του είδους καθώς ο αριθμός των πελατών της θα μειώνεται συνεχώς. Η γενική πεποίθηση ωστόσο, αναφέρει ότι κατά τη χρήση συσκευών smartphones ή tablets, η εμπειρία της πλοήγησης θα πρέπει να είναι παρόμοια με ενός σταθερού ή φορητού υπολογιστή, παρόλο που τα χαρακτηριστικά τους σε επιδόσεις είναι διαφορετικά.

Πρόσφατα στοιχεία παρουσιάζουν τους σταθερούς υπολογιστές να χάνουν συνεχώς έδαφος έναντι των smartphones ειδικά για την πλοήγηση στο ιντερνέτ, όπως ακριβώς παρουσιάζεται στην εικόνα 1. Η κατάσταση που επικρατεί στην αγορά, δημιουργεί επιτακτική την ανάγκη για την εξακρίβωση και την επίλυση όλων εκείνων των στοιχείων που δημιουργούν καθυστέρηση στην πλοήγηση των ιστοσελίδων, με απώτερο σκοπό την καλύτερη εμπειρία του χρήστη και την ταχύτερη απόκριση των ιστοσελίδων.

2.1 Βασικοί φυλλομετρητές

Ο φυλλομετρητής ιστοσελίδων (web browser), είναι λογισμικό, το οποίο επιτρέπει σε κάθε χρήστη να προβάλλει και να αλληλεπιδρά με κείμενα, εικόνες, βίντεο, μουσική, παιχνίδια και πληροφορίες, οι

οποίες είναι αναρτημένες σε μια ιστοσελίδα ή σε ένα τοπικό δίκτυο. Ο χρήστης έχει στην διάθεση του μία ευρεία γκάμα επιλογών φυλλομετρητών, όπως φυλλομετρητές που εστιάζουν στην ταχύτητα, στον περιορισμό κατανάλωσης των δεδομένων και στην ασφάλεια. Το κείμενο και οι εικόνες σε μια ιστοσελίδα, επομένως μπορεί να περιέχουν υπερσυνδέσμους προς άλλες ιστοσελίδες του ίδιου ή διαφορετικού ιστότοπου. Ο Web browser επιτρέπει στον χρήστη την γρήγορη και εύκολη πρόσβαση σε πληροφορίες που βρίσκονται σε διάφορες ιστοσελίδες και ιστότοπους, εναλλάσσοντας τες με την βοήθεια των υπερσυνδέσμων. Επιπρόσθετα, οι φυλλομετρητές χρησιμοποιούν τη γλώσσα μορφοποίησης HTML και CSS για την προβολή των ιστοσελίδων.

Ωστόσο λόγω της ραγδαίας αύξησης της χρήσης ηλεκτρονικών συσκευών, όπως Smartphones και Tablets, έγινε επιτακτική η ανάγκη της δημιουργίας νέων εκδόσεων φυλλομετρητών, οι οποίοι θα έχουν λιγότερες απαιτήσεις σε σχέση με εκείνους που εφαρμόζονται σε ηλεκτρονικούς επιτραπέζιους ή φορητούς υπολογιστές. Οι νέες αυτές εκδόσεις αφορούσαν τους ίδιους φυλλομετρητές, όμως με λιγότερες απαιτήσεις και με μεγαλύτερη λειτουργικότητα. Συγκεκριμένα η άνθηση αυτή οδήγησε στην δημιουργία νέων εκδόσεων φυλλομετρητών, ειδικούς για συστήματα android και IOS. Οι πιο διαδεδομένοι φυλλομετρητές για αυτά τα συστήματα είναι, ο Google Chrome, ο Safari (κυρίως σε συσκευές με λειτουργικό IOS), ο Mozilla Firefox και ο UC Browser.

2. 1.1 Φυλλομετρητής Google Chrome

Ο Google Chrome [4] εμφανίστηκε το 2008 και ήταν συμβατός κυρίως για εκδόσεις Android 4 (Ice Cream Sandwitch) και πάνω. Οι τελευταίες εκδόσεις επιτρέπουν τον συγχρονισμό σε όλες τις συσκευές και την αλλαγή των καρτελών με μια απλή χειρονομία. Υποστηρίζουν την λειτουργία zoom in/out, την επιτάχυνση του υλικού και την προ- απόδοση των γραφικών στοιχείων, προκειμένου να επιτευχθεί η μείωση του χρόνου εκτέλεσης. Το αρνητικό του συγκεκριμένου φυλλομετρητή είναι ότι καταναλώνει πολύ μνήμη τυχαίας προσπέλασης (RAM) και ότι ορισμένες φορές λειτουργεί τον επεξεργαστή (C PU) σε υψηλότερα επίπεδα (Spikes) από τα επιτρεπτά, με συνέπεια την κατανάλωση περισσότερης μπαταρίας σε σχέση με τους υπόλοιπους φυλλομετρητές.

2.1.2 Φυλλομετρητής Safari

Ο Safari [5] είναι ένας φυλλομετρητής, ο οποίος παρέχεται με τους υπολογιστές Macintosh. Εκδόθηκε στις 7 Ιανουαρίου του 2003 ως συνοδευτικό λειτουργικό του λειτουργικού συστήματος της εταιρείας Mac OS X. Συγκεκριμένα, ο Safari, έχει ξεκινήσει από τους υπολογιστές Mac της

εταιρείας Apple. Είναι ο κύριος browser που λειτουργεί σε όλους τους υπολογιστές της εταιρείας Apple χωρίς να υπάρχει κάποιος αντίπαλος. Παρόλο που η εταιρεία έχει σχεδιάσει έκδοση του Safari και για τα Windows το πρόγραμμα αυτό έγινε ιδιαιτέρως γνωστός και έχει αποκτήσει φανατικούς «οπαδούς» μετά την επιτυχία του iPhone και iPad.

2.1.3 Φυλλομετρητής Mozilla Firefox

Ο Mozilla Firefox [6] έκανε μια πρώιμη εμφάνιση μέσω της ελαφριάς Desktop έκδοσης, το 2009. Όμως με την έκδοση 4.0 το 2011 επιτεύχθηκε η συμβατότητα με Andoid 2.0 (Eclair) και πάνω. Από τότε μέχρι σήμερα τα χαρακτηριστικά με τα οποία λειτουργεί παραμένουν τα ίδια. Ο Mozilla Firefox, όπως ακριβώς και ο Google Chrome, μπορούν να υποστηρίξουν πολλές διαφορετικές λειτουργίες. Η μόνη διαφορά τους είναι ότι ο Mozilla Firefox δεν συλλέγει προσωπικά στοιχεία στον βαθμό που το κάνει ο Google Chrome. Υστερεί όμως στην ταχύτητα καθώς δεν χρησιμοποιεί παραπάνω υπολογιστή δύναμη (CPU cycles) όταν χρειαστεί, όπως ο άλλοι φυλλομετρητές. Στην συγκεκριμένη εργασία ασχοληθήκαμε με φυλλομετρητές μόνο για Android συσκευές.

2.1.4 Φυλλομετρητής UC Browser

Ο UC Browser [7] είναι ένας φυλλομετρητής που δημιουργήθηκε από μια κινεζική εταιρία κινητής τηλεφωνίας και διαδικτύου, την UCWeb. Πρωτοεμφανίστηκε το 2012, ως λειτουργικό των Android και έως σήμερα χρησιμοποιείτε κυρίως σε Ασιατικές χώρες όπως Κίνα, Ινδία και Ινδονησία. Ο συγκεκριμένος φυλλομετρητής δημιουργήθηκε με βασικό σκοπό την όσο μικρότερη χρήση δεδομένων και την γρηγορότερη εκτέλεση του σε κάθε συσκευή. Για αυτό ακριβώς το λόγο η αξιοποίηση του είναι κατάλληλη σε πολύ-πληθυσμιακές χώρες, όπως αυτές που προαναφέραμε.

Βασικό χαρακτηριστικό του είναι η συρρίκνωση δεδομένων (data compression) και η επιτάχυνση "νέφους" (cloud acceleration), αφού συρρικνώνει τα δεδομένα μέσω της χρήσης διακομιστών μεσολάβησης (proxy), πριν αυτά σταλούν στους χρήστες. Επιπρόσθετα, έχει την δυνατότητα να αποκλείει διάφορα διαφημιστικά μηνύματα χωρίς κανένα πρόσθετο, να επιτυγχάνει γρήγορο κατέβασμα (Speed Mode) που υποστηρίζετε για ιστοσελίδες σε HTML5 και να εφαρμόζει την λειτουργία εξοικονόμησης δεδομένων (data save). Επίσης όπως όλοι οι υπόλοιποι φυλλομετρητές επιτρέπει συγχρονισμό για όλες τις συσκευές με την δημιουργία λογαριασμού.

2.2 Εργαλείο μέτρησης απόδοσης jsPerf

Το jsPerf[8] είναι μια ιστοσελίδα που μπορεί να μετρήσει τους χρόνους εκτέλεσης του κώδικα γραμμένου σε Javascript οπότε και κατά συνέπεια η χρήση του επεκτείνεται και για την jQuery. Λειτουργεί λόγο του Benchmark.js, και έχει την δυνατότητα να επιστρέφει τα στατιστικά αποτελέσματα με μεγάλη ακρίβεια. Επιπρόσθετα, επιτρέπει την σύγκριση εκτελέσεων ανά μιλιδευτερόλεπτο σε οποιοδήποτε κώδικα HMTL επιθυμεί ο προγραμματιστής. Ο τρόπος με τον οποίο λειτουργεί το συγκεκριμένο εργαλείο είναι με το να εκτελεί συνεχώς ένα τεστ μέχρι την εξάλειψη της συνήθης απόκλισης. Ο αριθμός των επαναλήψεων ποικίλλει από τον προκαθορισμένο χρόνο εκτέλεσης που είναι στα 5 δευτερόλεπτα ή την ελάχιστη εκτέλεση των 5 επαναλήψεων. Στην συνέχεια γίνεται στατιστική ανάλυση και προκύπτει ένα αποτέλεσμα σε εκτελέσεις ανά δευτερόλεπτο.

2.3 Κώδικας DOM

Ο κώδικα DOM αποτελεί την παρουσίαση των βασικών στοιχείων που συνθέτουν μια ιστοσελίδα, όπως γραφικά κείμενα, εικόνες, με ιεραρχική δομή. Δηλαδή αποτελεί το βασικό τρόπο παρουσίαση των ιστοσελίδων από τους διάφορους browsers. Οι browsers μοντελοποιούν και χωρίζουν τις ιστοσελίδες σε μικρότερα κομμάτια. Ο διαχωρισμός αυτός επιτρέπει την πρόσβαση σε όλα τα στοιχεία που αποτελούν μια σελίδα.

Συγκεκριμένα, ο κώδικας DOM είναι αντικειμενοστραφές και μπορεί να ενσωματωθεί σε ένα μεγάλο αριθμό από προγραμματιστικά περιβάλλοντα τόσο με γλώσσες προγραμματισμού όσο και με γλώσσες script. Το DOM αποτελείται από ένα σύνολο αντικειμένων που διαθέτουν ιδιότητες και μεθόδους για επεξεργασία των διάφορων εγγράφων HTML. Δηλαδή το DOM εκτός από ανάγνωση προσφέρει και την δυνατότητα για τροποποίηση υπαρκτών αρχείων ή για δημιουργία νέων HTML εγγράφων.

2.4 Εξέλιξη των Smartphones και των Tablets

Το διαδίκτυο αποτελείται από εκατομμύρια υπολογιστές που βρίσκονται διασκορπισμένοι σε όλο τον πλανήτη και επικοινωνούν μεταξύ τους ανταλλάσοντας δεδομένα. Για την άμεση και γρήγορη επικοινωνία των ανθρώπων, οι απαρχαιωμένοι υπολογιστές μετεξελίχθηκαν σε νέας τεχνολογίας συσκευές, αυτές των έξυπνων τηλεφώνων (Smartphones) και των φορητών ταμπλέτων (Tablets). Η ανακάλυψη και η δημιουργία των tablets αποτελεί ένα από τα τελευταία τεχνολογικά επιτεύγματα της πληροφορικής.

Μέχρι το 2006 το Smartphone ήταν καθαρά επαγγελματική υπόθεση. Η έκρηξη των συσκευών αυτών υλοποιήθηκε το 2007 με το iPhone και το 2008 με την πρώτη εμφάνιση των συσκευών Android. Οι συνέπειες ήταν κοσμογονικές. Πολύ γρήγορα η εικόνα του smartphone άλλαξε και έγινε μια συσκευή στην οποία κυριαρχεί η οθόνη αφής και όχι το πληκτρολόγιο, ήρθαν τα app stores μεγάλη κλίμακας, ενώ Microsoft και BlackBerry αναγκάστηκαν να ανασυνταχθούν και να παρουσιάσουν νέα λειτουργικά συστήματα.

Συγκεκριμένα, τα πρώτα smartphones συνδύαζαν τις λειτουργίες ενός προσωπικού ψηφιακού βοηθού (PDA) και ενός κινητού τηλεφώνου. Σε μεταγενέστερα μοντέλα προστέθηκαν οι λειτουργίες των φορητών media players, low-end compact ψηφιακές φωτογραφικές μηχανές, βιντεοκάμερες τσέπης, καθώς και μονάδες πλοήγησης GPS, με αποτέλεσμα την διαμόρφωση μιας πολυχρηστικής συσκευής. Πολλά σύγχρονα smartphones περιλαμβάνουν επίσης οθόνες αφής υψηλής ανάλυσης και web browsers που εμφανίζουν τυποποιημένες ιστοσελίδες, καθώς και βελτιστοποιημένες ιστοσελίδες για κινητά. Η πρόσβαση σε δεδομένα υψηλής ταχύτητας παρέχεται μέσω Wi-Fi και μέσω κινητών ευρυζωνικών υπηρεσιών.

Τα tablets από την αντίθετη πλευρά αποτελούν την μικρογραφία ενός παραδοσιακού ηλεκτρονικού υπολογιστή. Σκοπός της δημιουργίας τους ήταν ο συνδυασμός σμίκρυνσης της τεχνολογίας με έναν ισχυρό επεξεργαστή, ο οποίος θα βελτιστοποιεί το διαθέσιμο χώρο, στον οποίο ο κάθε χρήστης θα μπορεί να αποθηκεύει φωτογραφίες, βίντεο, εφαρμογές, διάφορα αρχεία και μηνύματα. Τα tablets είναι πάρα πολύ εύχρηστα αφού παρέχουν την δυνατότητα στο κάθε χρήστη να έχει πρόσβαση στο διαδίκτυο ενώ βρίσκεται σε κίνηση. Το χαρακτηριστικό αυτό σε συνδυασμό με τις φθηνότερες εφαρμογές, τη μεγαλύτερη διάρκεια ζωής της μπαταρίας και τον λιτό σχεδιασμό, καθιστούν τα tablets την ιδανική επιλογή για mobile computing. Συνεπώς η ραγδαία εξέλιξη των smartphones και των tablets αποτέλεσε και αποτελεί προέκταση της ζωής των ανθρώπων.

3.1 Μεθοδολογία της ανάλυσης

Για την επίτευξη της εργασίας αυτής, δημιουργήθηκε ένας αριθμός από τεστ, συγκεκριμένα εφαρμόστηκαν 120 τεστ, τα οποία υλοποιήθηκαν από δυο φορές σε κάθε συσκευή για την επίτευξη μεγαλύτερης ακρίβειας. Αξιοποιήθηκε η ιστοσελίδα jsPerf, η οποία έδωσε την δυνατότητα δημιουργίας και μέτρησης του χρόνου εκτέλεσης δοκιμαστικών πεδίων γραμμένα σε κώδικα jQuery. Για την μεγαλύτερη ακρίβεια και για την καλύτερη μέτρηση των αποτελεσμάτων της κάθε περίπτωσης που εκτελέστηκε, οι μετρήσεις πραγματοποιήθηκαν σε τρία DOM διαφορετικού μεγέθους. Ένα μικρό **[βλέπε παράρτημα 1]**, το οποίο περιλαμβάνονται ένα Id tag, 8 P tags, 22 Div tags και 10 classes, ένα μεσαίο **[βλέπε παράρτημα 2]**, που

περιλαμβάνονται ένα Id tag, 136 P tags, 36 Div tags και 91 classes και ένα μεγάλο **[βλέπε παράρτημα 3]**, που περιλαμβάνονται ένα Id tag, 334 P tags, 73 Div tags και 139 classes.

Η επίτευξη της μέτρησης των χρόνων πραγματοποιήθηκε με την αλλαγή του βασικού κώδικα του DOM εφαρμόζοντας περιπτώσεις της jQuery. Επίσης μετρήθηκαν και οι selectors πριν την εφαρμογή των μεθόδων για να υπάρξει μεγαλύτερη ακρίβεια των μετρήσεων κατά την ανάλυση των αποτελεσμάτων. Τα τεστ αυτά εστίασαν στην κατηγορία «χειρισμού» (manipulation)[9], η οποία περιέχει μεθόδους κυρίως για την εισαγωγή, αλλαγή, αντιγραφή, διαγραφή και αφαίρεση στοιχείων.

3.1.1 Επιλογή Συσκευών

Η εκτέλεση όλων των τεστ έγινε σε συγκεκριμένες συσκευές οι οποίες επιλέχθηκαν κυρίως για τα τεχνικά χαρακτηριστικά που τις αντιπροσώπευαν. Δηλαδή για το λειτουργικό Android που είχαν, την μνήμη τυχαίας προσπέλασης που διέθεταν, συγκεκριμένα από 1GB έως 2GB (RAM) και έπειτα την υπολογιστική δύναμη (CPU), δηλαδή κυρίως συσκευές που είχαν επεξεργαστές τύπου Quad-core 1.5GHz.

Για την υλοποίηση την παρούσας εργασίας χρησιμοποιήθηκαν συγκεκριμένες συσκευές, οι οποίες διέθεταν τα παραπάνω τεχνικά χαρακτηριστικά. Οι συσκευές αυτές ήταν ένα smartphone LG G2- D800[10], ένα Tablet Samsung Galaxy Tab 3 8.0 3G[11] και ένα Tablet Asus Google Nexus 7(2013)[12].

3.1.2 Εκτελέσιμος κώδικα jsPerf

Για κάθε μέθοδο που χρησιμοποιήθηκε για την μελέτη αυτή, δημιουργήθηκαν 40 διαφορετικές περιπτώσεις **[Βλέπε Παράρτημα 4]** εκτελέσιμου κώδικα jsPerf, με ορισμένες εξ' αυτές να έχουν δύο διαφορετικά τεστ για την εξέταση των περιπτώσεων Get και Set. Σε αυτό το σημείο αξίζει να αναφερθεί ότι μέθοδοι όπως η .offset() μπορούν να χρησιμοποιηθούν για να μας δώσουν πληροφορίες σχετικά με την θέση ενός στοιχείου (Get) ή να μας ορίσουν μια νέα θέση του στοιχείο που λαμβάνει μέσα στο DOM.(Set). Επιπρόσθετα πρέπει να επισημάνουμε ότι σε κάθε ένα από τα τεστ που εκτελέστηκαν πραγματοποιήθηκε επιπλέον και η μέτρηση των selectors χωρίς κάποια μέθοδο να εφαρμόζετε ώστε να υπάρξει ένα μέσο όρο χρόνων εκτέλεσης των selectors.

Οι μέθοδοι που επιλέχθηκαν από την κατηγορία manipulation είναι η addClass() η οποία προσθέτει μία κλάση στον επιλεγμένο selector, η .after() που εισάγει περιεχόμενο στο τέλος του selector που εφαρμόζεται, με παρόμοια μέθοδο την insertAfter με διαφορές στον τρόπο σύνταξή τους όπως φαίνετε στο παρακάτω παράδειγμα.

Μέθοδος after:

```
$( ".inner" ).after( "<p>Test</p>" );
```

Μέθοδος insertAfter:

```
( "<p>Test</p>" ).insertAfter( ".inner" );
```

Οι μέθοδοι .append() και .appendTo() εισάγουν περιεχόμενο βάση παραμέτρων στο τέλος κάθε στοιχείου. Η .attr() μέθοδος έχοντας δύο λειτουργίες, μία την Get και μία την Set, στην πρώτη περίπτωση μας επιστρέφει τα χαρακτηριστικά του πρώτου στοιχείου μέσα από ένα σύνολο στοιχείων που υπάρχουν στον επιλεγμένο selector ενώ στην περίπτωση Set προσθέτει χαρακτηριστικά στα επιλεγμένα στοιχεία. Οι μέθοδοι .before() και insertBefore έχουν ακριβώς την ίδια λειτουργία με την .after() και .insertAfter() μεθόδους με την διαφορά ότι προσθέτει περιεχόμενο στην αρχή του κάθε selector. Με την μέθοδο clone() μπορούμε να «αντιγράψουμε» ολόκληρο τον επιλεγμένο selector και όλα τα χαρακτηριστικά που τον ακολουθούν. Η μέθοδος css() εισάγει ιδιότητες όπως χρώμα, αλλαγή μεγέθους και όλα τα χαρακτηριστικά που μπορούν να συνοδεύσουν ένα tag (π.χ. Div) χρησιμοποιώντας CSS. Οι μέθοδοι .detach() και .empty() αφαιρούν εντελώς το περιεχόμενο του επιλεγμένου selector με την διαφορά ότι με την πρώτη γίνεται αφαίρεση και του ίδιου του selector ενώ στην πρώτη δεν γίνεται και διαγραφή οπότε η μελλοντική χρήση (π.χ. με μέθοδο attach()) είναι δυνατή . Με την hasClass() μέθοδο εντοπίζουμε αν ο selector έχει κάποια κλάση ενώ με την html() μας επιστέφει το περιεχόμενο της HTML για το πρώτο στοιχείο που θα εντοπιστεί. Στην συνέχεια με τις μεθόδους .height() και width() στην περίπτωση Get μας επιστρέφετε υπολογισμένο σε pixels το ύψος και αντίστοιχα το πλάτος του συγκεκριμένου selector, ενώ στην περίπτωση Set μπορεί να δοθεί η τιμή σε pixels και να γίνει η μετακίνηση σε ύψος ή πλάτος του selector από τον κατασκευαστή της ιστοσελίδας. Το ίδιο γίνεται και για τις μεθόδους .innerHeight() και .innerWidth με την διαφορά ότι στις δύο τελευταίες ο υπολογισμός και η «τοποθέτηση» γίνεται μόνο για τα στοιχεία που υπάρχουν μέσα στον selector. Παρόμοιες μέθοδοι με την διαφορά ότι όλα τα παραπάνω γίνονται για το εξωτερικό μέρος του selector είναι η μέθοδος outerHeight() και outerWidth() με την διαφορά ότι στις δύο αυτές περιπτώσεις συνυπολογίζονται και τα όρια(borders), η απόσταση μεταξύ των περιεχομένων (padding) και προαιρετικά τα περιθώρια(margin). Στην συνέχεια χρησιμοποιήσαμε τις μεθόδους .prepend και prependTo() οι οποίες έχουν την ίδια λειτουργία με τις append() και appendTo() μεθόδους με την διαφορά ότι η εισαγωγή γίνεται στη αρχή του περιεχομένου του επιλεγμένου selector. Η μέθοδος .prop() είναι παρόμοια με την μέθοδο .attr() αλλά χρησιμοποιείτε μόνο για συγκεκριμένες περιπτώσεις όπως για παράδειγμα στις περιπτώσεις selectedIndex και tagName [1]. Η μέθοδος .position() μας επιστρέφει την τιμή σε pixels της θέσης του κάθε στοιχείου μέσα στον επιλεγμένο selector σε σχέση με

την θέση του tag στο οποίο περικλείεται. Παρόμοια λειτουργία έχει και η μέθοδος offset() με την διαφορά ότι ο συσχετισμός δεν γίνεται βάση της θέσης του tag στο οποίο περικλείεται αλλά βάση της θέσης του <body> δηλαδή όλου του DOM. Η μέθοδος offset() έχει και Get περίπτωση όπου μπορεί να γίνει εισαγωγή σε pixels τιμών. Με τις μεθόδους remove(), removeAttr() και removeProp() γίνεται αφαίρεση στοιχείων από τον selector. Συγκεκριμένα στην πρώτη περίπτωση γίνεται πλήρης αφαίρεση όλων ενώ στις υπόλοιπες μεθόδους γίνεται αφαίρεση μόνο της κλάσης και των ιδιοτήτων των selector αντίστοιχα. Η μέθοδος replaceAll() είναι παρόμοια με την replaceWith() με την διαφορά ότι αρχικά γίνεται πλήρης αλλαγή με άλλο περιεχόμενο σε όλο τον selector ενώ με την δεύτερη γίνεται επιλεγμένα η αλλαγή σε συγκεκριμένα στοιχεία. Οι μέθοδοι scrollLeft() και scrollTop() στην περίπτωση Get μας επιστρέφει την τιμή τις μπάρας που βρίσκεται αριστερά ή κάτω σε pixels και που είναι υπεύθυνη να μας δείχνει την σελίδα έξω από τα όρια της οθόνης. Με την περίπτωση Get ο κατασκευαστής ελέγχει την τιμή αυτή. Η μέθοδος .text() είναι παρόμοια με την μέθοδο html() με την διαφορά ότι μπορεί να ενσωματώσει και ιδιότητες πέρα από μόνο HTML. Η .toggleClass μέθοδος ανάλογα με το αν υπάρχει ήδη η συγκεκριμένη κλάση αφαιρεί ή προσθέτει κλάσεις στον συγκεκριμένο selector. Με την μέθοδο .val() μας επιστρέφετε η τιμή ενός πλαισίου κειμένου (textarea) ή άλλων επιλογών σε μία φόρμα. Η μέθοδος unwrap() αφαιρεί τους «γονείς» του επιλεγμένου selector. Δηλαδή στην περίπτωση που έχουμε ένα <p> μέσα σε ένα τότε το τελευταίο αφαιρείτε μετά την εφαρμογή της μεθόδου. Αντίθετο αποτέλεσμα έχουμε με την εφαρμογή της μεθόδου wrap(). Με την μέθοδο wrapAll() περικλείετε όλο το περιεχόμενο του selector στο επιλεγμένο tag (π.χ. Div) ενώ με την μέθοδο wrapInner() αυτό γίνεται μόνο για τα περιεχόμενα του selector.

3.1.3 Τρόπος εντοπισμού παραγόντων καθυστέρησης

Με την βοήθεια των εργαλείων που αναφέρθηκαν παραπάνω, η μέθοδος που ακολουθήθηκε για τον εντοπισμό της καθυστέρησης φόρτωσης μιας ιστοσελίδας είναι η εκτέλεση των τεστ σε jQuery στο DOM ώστε να βρούμε πόσες εκτελέσεις γίνονται ανά δευτερόλεπτο. Όλα τα τεστ εκτελέστηκαν δύο φορές για κάθε συσκευή που χρησιμοποιήθηκε για την μεγαλύτερη ακρίβεια των αποτελεσμάτων. Οι συσκευές ήταν χωρίς εφαρμογές πλην των προκαθορισμένων εφαρμογών Android και μετά από επαναφορά εργοστασιακών ρυθμίσεων. Πριν από κάθε εκτέλεση ανά browser γινόταν επανεκκίνηση της συσκευής και βάση της εκτέλεσης εργασιών δεν υπήρχε ανοιχτή εφαρμογή κατά την διάρκεια μέτρησης των αποτελεσμάτων εκτός των απαραιτήτων εφαρμογών που χρησιμοποιεί το λειτουργικό της κάθε συσκευής.

Η αρχική προσέγγιση είχε ως στόχο να διαπιστώσουμε ποια μέθοδος έχει την χειρότερη απόδοση όσο αφορά την πλοήγηση σε οποιαδήποτε ιστοσελίδα του διαδικτύου. Για το λόγο αυτό δημιουργήθηκε ένας συγκεντρωτικός πίνακας ανά συσκευή **[Βλέπε Παράρτηματα Πινάκων Α,Β,Γ]** στον οποίο παρουσιάζεται

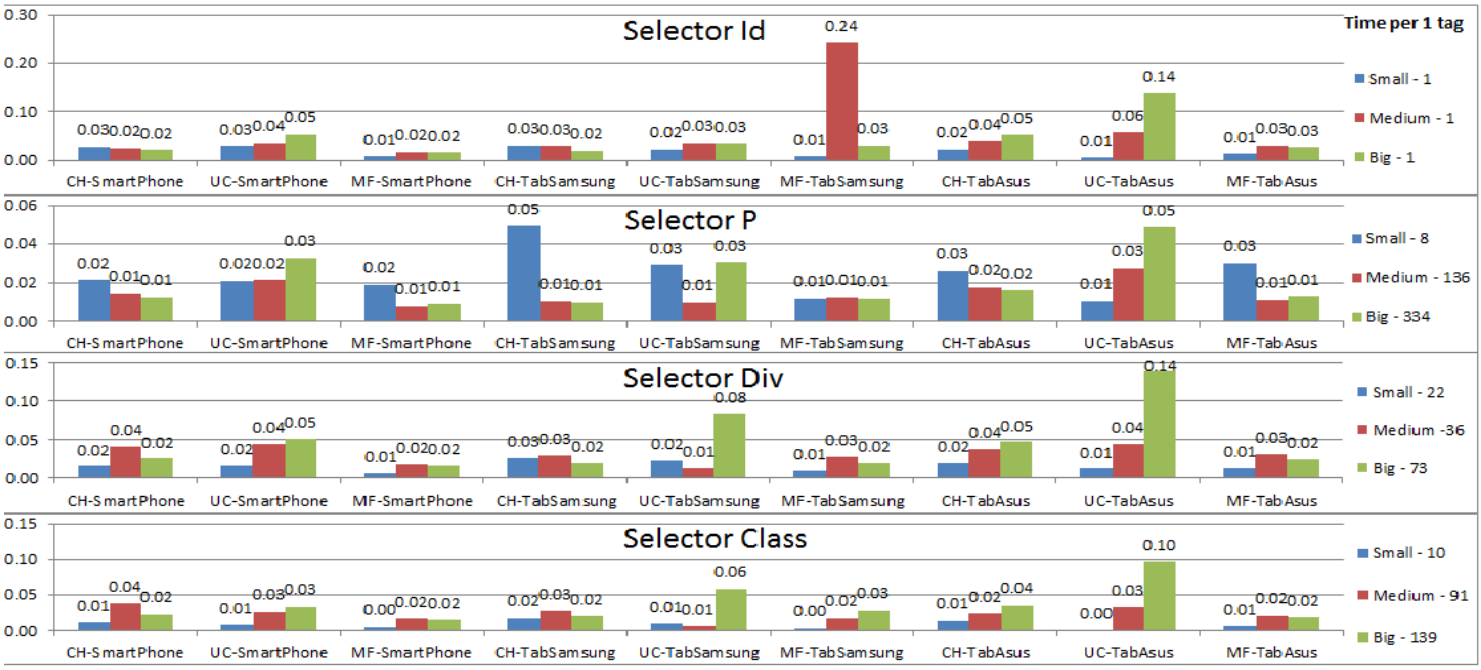
αναλυτικά ο χρόνος εκτέλεσης της κάθε μεθόδου. Μέσα από τα αποτελέσματα αυτά, παρατηρήθηκε ότι οι χρόνοι εκτέλεσης όσο μεγάλωνε το μέγεθος του DOM δεν αυξάνονταν αντίστοιχα, αλλά αντιθέτως σε μερικές περιπτώσεις μειώνονταν. Ειδικά στην περίπτωση της χρήσης του smartphone με τον browser Google Chrome οι διαφορές που υπήρχαν (εκτός συγκεκριμένων περιπτώσεων) ήταν παρεμφερείς ανεξαρτήτου του DOM, που χρησιμοποιήθηκε, δηλαδή μικρό, μεσαίο ή μεγάλο.

4.1 Ανάλυση αποτελεσμάτων

Εξαιτίας αυτής της παρατήρησης στην συνέχεια έγινε περαιτέρω ανάλυση της κάθε μεθόδου ξεχωριστά. Μετρήθηκε ο χρόνος για κάθε selector σε κάθε μέθοδο που εφαρμοζόταν ανά μία εκτέλεση του, αφαιρώντας όμως τον χρόνο που δαπανάτε για τον selector που έχει επιλεγεί. Η άμεση παρατήρηση ήταν ότι συναντάμε πολύ μικρούς χρόνους στα όρια του στατιστικού λάθους του ίδιου του εργαλείου jsPerf σε συγκεκριμένες περιπτώσεις όπως τις μεθόδους αφαίρεσης, ενώ σε άλλες καθιστάτε απαγορευτική η χρήση ειδικά όπως φαίνετε παρακάτω για τις μεθόδους που εισάγουν περιεχόμενο ή ορίζουν τις συντεταγμένες σε κάθε στοιχείο μέσα στο DOM.

Στα διαγράμματα που ακολουθούν στον αριστερό άξονα βλέπουμε την κλίμακα των χρόνων εκτέλεσης. Στον οριζόντιο άξονα περιγράφετε εν-συντομία η συσκευή και ο browser (όπου CH σημαίνει Google Chrome, UC είναι ο UC Browser και MF περιγράφει τον Mozilla Firefox). Στην δεξιά μεριά του διαγράμματος διατυπώνετε το μέγεθος του DOM, δηλαδή πόσες φορές συναντάτε ο κάθε selector που έχει μετρηθεί. Οι μπάρες περιγράφουν τους χρόνους εκτέλεσης ενώ μία διαφορά που παρατηρείτε στα δύο είδη διαγραμμάτων (για μία εκτέλεση και για την συνολική εκτέλεση) είναι το «Time per 1 Tag» το οποίο δείχνει το είδος του διαγράμματος (για μία εκτέλεση) και το «Time per Total Tags» (για συνολική εκτέλεση).

Το φάσμα των χρόνων εκτέλεσης διαμορφώνονται από 0.01 μέχρι και 600 μίλι-δευτερόλεπτα. Για παράδειγμα στο παρακάτω διάγραμμα (**Διάγραμμα 1**) παρουσιάζεται αναλυτικά ο χρόνος εκτέλεσης της μέθοδος .addClass(), ο οποίος είναι απόλυτα αποδεκτός για οποιαδήποτε συσκευή εκτελείτε.



Διάγραμμα 1 - Μέθοδος .addClass()

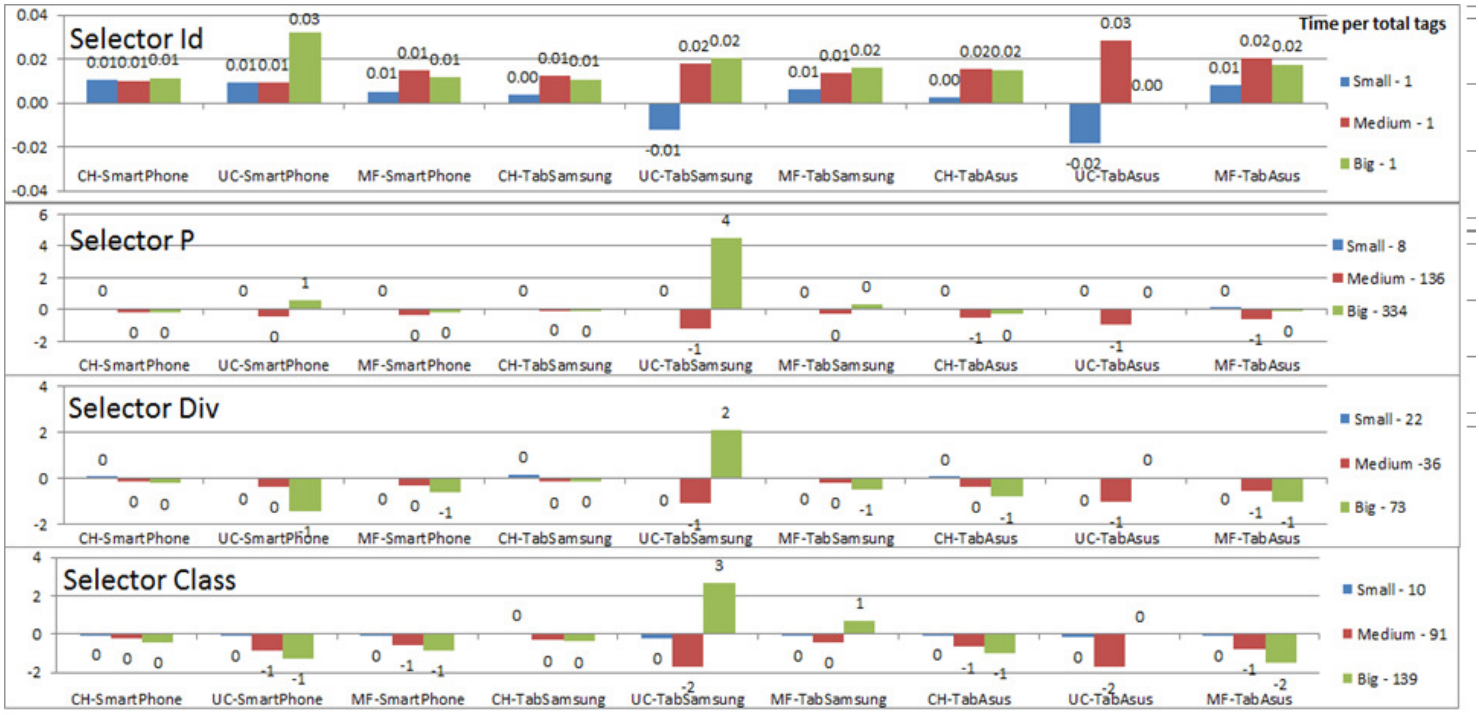
Αντίθετα συναντάμε ένα περιορισμένο αριθμό selectors οι οποίοι έχουν πολύ μεγάλες τιμές και μπορούν να δημιουργήσουν προβλήματα στην εκτέλεση της ιστοσελίδας ή ακόμα και μη εμφάνισής της. Όπως φαίνεται και στο Διάγραμμα 2 οι τιμές για την μέθοδο .wrap() έχουν ελάχιστη τιμή τα 2 και μέγιστη τα 20 μίλι-δευτερόλεπτα για μία μόνο εκτέλεση με αξιοσημείωτο γεγονός ότι και ο καλύτερος και ο χειρότερος χρόνος καταγράφηκε από την ίδια συσκευή. Συγκεκριμένα το smartphone καταγράφει χρόνους 1,24 μίλι-δευτερόλεπτα χρησιμοποιώντας Google Chrome για τον selector class, και μία εκτίναξη της τιμής στα 20.29 μίλι-δευτερόλεπτα για την ίδια συσκευή στον UC Browser.Το ίδιο παρατηρείτε και για τον Mozilla Firefox ο οποίος κυμαίνετε στα 8.14 μίλι-δευτερόλεπτα. Όλες αυτές οι παρατηρήσεις έγιναν για το μικρό DOM.

Διάγραμμα 2 - Μέθοδος .wrap()

4.1.1 Ανάλυση αποτελεσμάτων για μία εκτέλεση

Το γεγονός του μεγάλου εύρους τιμών οδήγησε στην κατηγοριοποίηση των μεθόδων ανάλογα με τους μέγιστους χρόνους εκτέλεσης που παρουσίασαν σε κάποιον browser ή συσκευή ξεχωριστά. Έτσι μπορεί να επιτευχθεί ευκολότερα η ανάλυση των αποτελεσμάτων και να φανούν κοινά χαρακτηριστικά μεταξύ των μεθόδων που να ευθύνονται για την γρήγορη ή αργή εκτέλεσή τους. Οι τέσσερις κατηγορίες που προέκυψαν περιλαμβάνουν.

Τις μεθόδους με σχεδόν μηδενικό κόστος αφού ο χρόνος εκτέλεσης είναι στην μέγιστη τιμή του, στα 0.05 μίλι-δευτερόλεπτα. Οι μέθοδοι αυτοί είναι .detach(), .empty(), .remove() και val. Στο **Διάγραμμα 3** παρουσιάζεται αναλυτικά η γρήγορη αυτή εκτέλεση για μία συγκεκριμένη μέθοδο η οποία παρουσιάζει κατά προσέγγιση όλες τις υπόλοιπες. Αναλυτικά οι χρόνοι εκτέλεσης μπορούν να φανούν στο Παράρτημα Διαγραμμάτων.



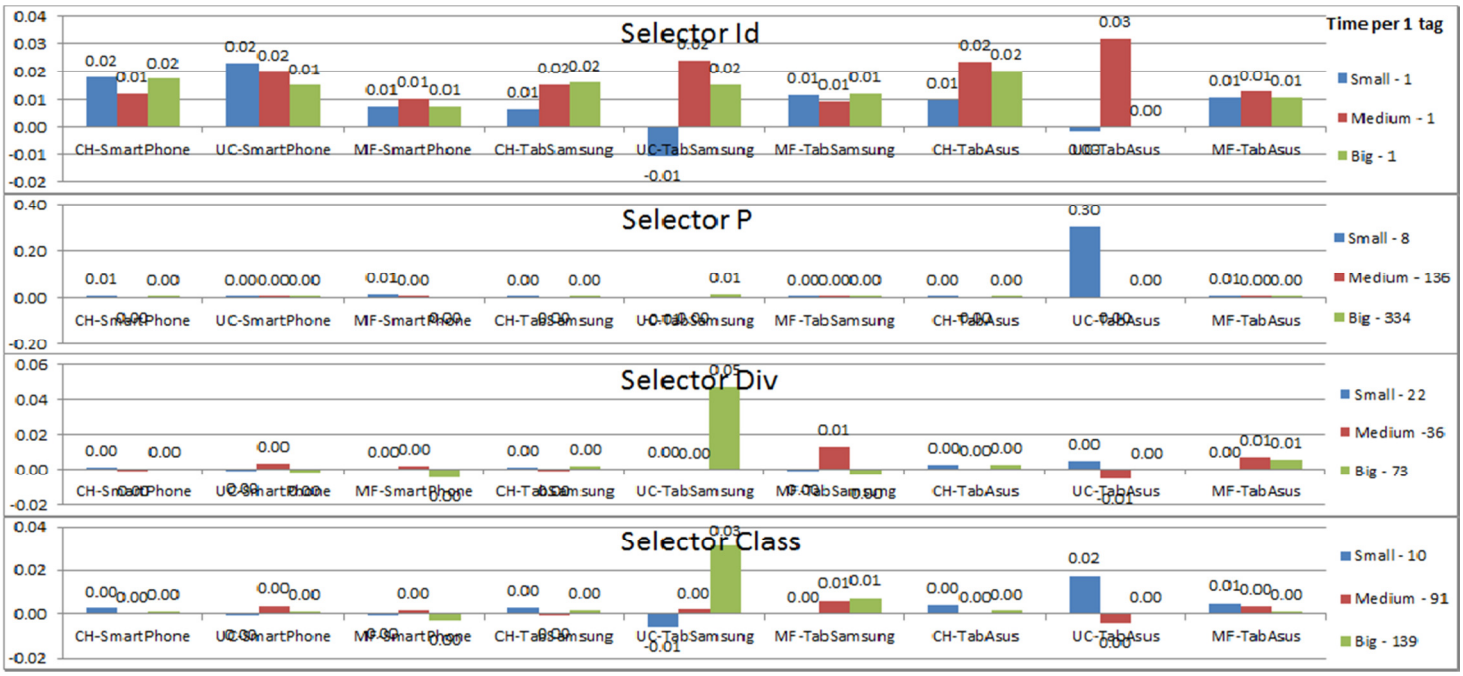
Διάγραμμα 3 – Μέθοδος .empty()

Συγκεκριμένα στο **Διάγραμμα 3** εντοπίζεται ότι εκτός της μεθόδου .val() όλες οι υπόλοιπες μέθοδοι έχουν κοινό ότι αφαιρούν ή διαγράφουν τους selectors, καθώς και όλο το περιεχόμενο που περιλαμβάνουν. Οι αρνητικές τιμές που φαίνονται στο διάγραμμα ή στους πίνακες **[Βλέπε παράρτημα πινάκων Α,Β,Γ]** δικαιολογούνται, διότι οι αρχικές τιμές των selectors έχουν προκύψει από το μέσο όρο εκτέλεσης της κάθε μεθόδου ξεχωριστά.

Στην δεύτερη κατηγορία περιλαμβάνονται οι μέθοδοι .addClass(), .css(), .hasClass(), .position(), .prop(), .removeAttr(), .removeClass(), .removeProp(), .replaceAll(), replaceWith(), scrollLeft(), scrollTop(), .text(), .toggleClass() και μόνο οι περιπτώσεις Get .height(), .offset(), .outerHeight, .outerWidth, attr() και innerWidth(). Στη συγκεκριμένη περίπτωση οι μέθοδοι αυτοί έχουν μέγιστο χρόνο κάτω των 30 μίλι-δευτερολέπτων. Παρατηρείται πάλι ότι όλες οι μέθοδοι που αφαιρούν ή διαγράφουν τον ίδιο τον selector ή/και το περιεχόμενό τους παράγουν τους καλύτερους χρόνους Στο διάγραμμα 4 φαίνετε για την μέθοδο removeProp ακριβώς η μεγάλη ταχύτητα εκτέλεσης αφού παρατηρούμε την μέγιστη τιμή στα 0,07 μίλι-δευτερόλεπτα. Αμέσως μετά βάση χρόνου εκτέλεσης διαπιστώνουμε μεθόδους οι οποίες εισάγουν ή μεταλλάσουν (στην συγκεκριμένη περίπτωση αλλάζουμε την κλάση ενός selector σε μία άλλη) το περιεχόμενο στο DOM όπως η μέθοδος .toggleClass (**Διάγραμμα 5**) ενώ φτάνοντας στα 30 μίλι-δευτερόλεπτα βλέπουμε κυρίως της περιπτώσεις Get οι οποίες ανάλογα την μέθοδο επιστρέφουν διάφορες τιμές όπως παράδειγμα στο διάγραμμα 6 η μέθοδος .attr() μας επιστρέφει τα χαρακτηριστικά του πρώτου στοιχείου από ένα σύνολο στοιχείων.

Διάγραμμα 4- Μέθοδος .removeProp

Διάγραμμα 5 – Μέθοδος .toggleClass ()



Διάγραμμα 6 – Μέθοδος .attr() - Get

Σε μία κατηγορία που μαζί με την προηγούμενη περιλαμβάνουν την πλειοψηφία των μεθόδων που ελέγξαμε, συναντάμε τις μεθόδους .after(), .append(), .appendTo(),.attr().before(), clone(), html(), .insertAfter, .insertBefore, .prepend(), .prependTo(), .width(), .wrapAll() και για περιπτώσεις Set., περιλαμβάνονται μόνο τα ακόλουθα: .height(), .innerHeight(), .outerWidth, και .innerWidth.

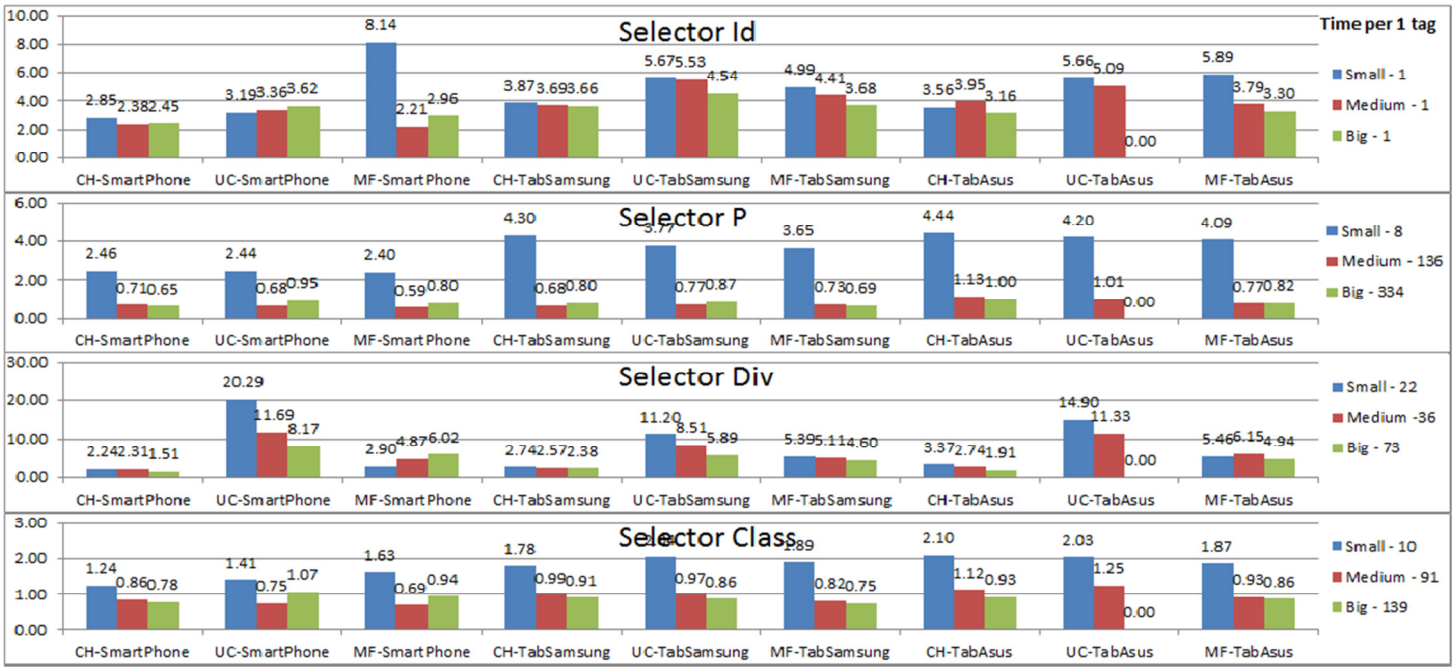
Στη συγκεκριμένη κατηγορία παρατηρούμε αισθητά αυξημένους χρόνους ειδικά στις περιπτώσεις των Set που φτάνουν μέχρι και 200 μίλι-δευτερόλεπτα, όπως στην περίπτωση atrr() για την περίπτωση Set Ταξινομώντας τις μεθόδους αυτές κατά αύξοντα αριθμό χρόνου εκτέλεσης παρατηρούμε ότι όσες αρχικά προσθέτουν περιεχόμενο (**Διάγραμμα 7**) παρουσιάζουν μειωμένο χρόνο εκτέλεσης σε σχέση με όσες αλλάζουν (Set) το DOM (**Διάγραμμα 8**)

Διάγραμμα 7 – Μέθοδος .insertAfter()

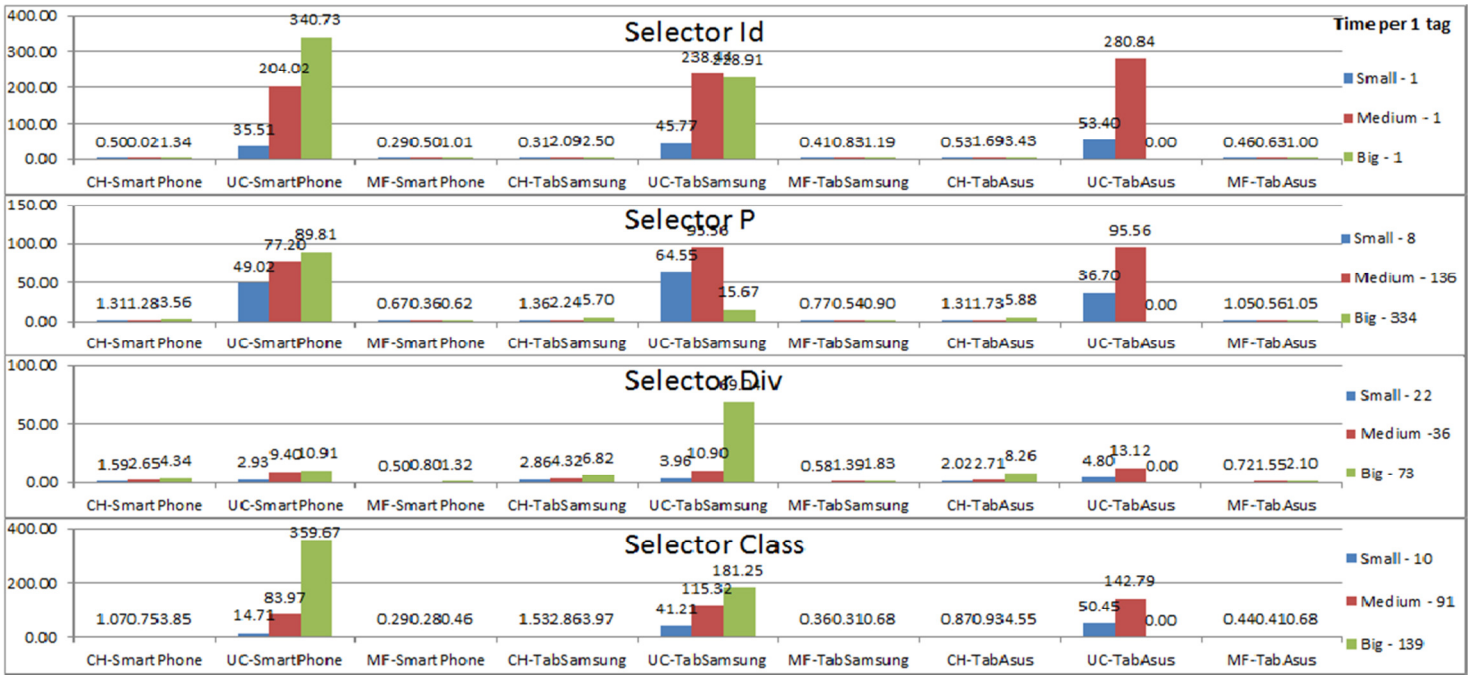
Επιπρόσθετα αξιοσημείωτο είναι το γεγονός ότι οι μέθοδοι που συναντήθηκαν στην προηγούμενη κατηγορία με την ιδιότητα του Get κατέγραψαν χαμηλούς χρόνους, ενώ εδώ στην περίπτωση Set έχουμε χρόνους που ξεπερνούν και τις 20 φορές μεγαλύτερους από τους αρχικούς όπως φαίνεται και στο **Διάγραμμα 6** και **Διάγραμμα 8**. Αυτό είναι αναμενόμενο αφού στην πρώτη περίπτωση έχουμε επιστοφή της τιμής που υπάρχει ήδη ενώ στην δεύτερη ο κατασκευαστής επιλέγει την τιμή και την εισάγει χρησιμοποιώντας την ίδια μέθοδο.

Διάγραμμα 8 - Μέθοδος .atrr() - Set

Στην τελευταία κατηγορία κατατάχτηκαν οι selectors που ο χρόνος εκτέλεσης τους ξεπερνούσε τα 200 μίλι – δευτερόλεπτα. Σε αυτή την κατηγορία δηλαδή περιλαμβάνονται οι μέθοδοι .unwrap(), .wrap(), .wrapInner() και για την περίπτωση Set το .offset(). Σημαντικό στοιχείο εδώ είναι πώς όλες οι μέθοδοι που επηρεάζουν το DOM εισάγουν πολλαπλά tags σε ήδη υπάρχων (π.χ. εισαγωγή σε <p>), με αποτέλεσμα να καταγράφουν μεγάλους χρόνους εκτέλεσης. Αν και τα αποτελέσματα από τα διαγράμματα 9 και 10 δείχνουν τους αυξημένους χρόνους εκτέλεσης η μέθοδος .offset όταν εισάγετε τιμή από τον κατασκευαστή της ιστοσελίδας βλέπουμε να εκτοξεύει τον χρόνο εκτέλεσης στην μεγαλύτερη τιμή που καταγράφηκε από όλα τα τεστ τα οποία διεξήχθησαν.



Διάγραμμα 9 - Μέθοδος .wrap()

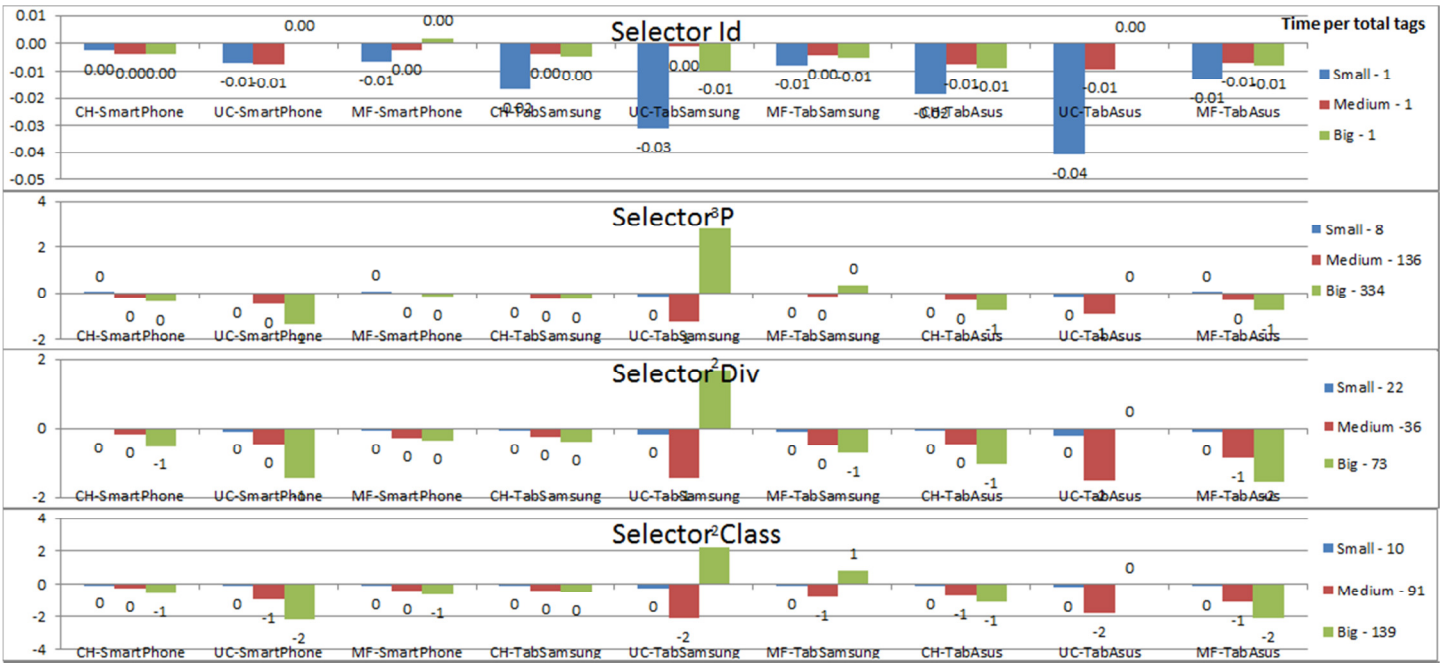


Διάγραμμα 10 - Μέθοδος .offset() – Set

Όλοι οι πίνακες αποτελεσμάτων για κάθε μέθοδο ξεχωριστά μπορούν να βρεθούν στο Παράρτημα Πινάκων. Οι συγκεκριμένες περιπτώσεις που αναφέρθηκαν δείχνουν ενδεικτικά τους χρόνους εκτέλεσης για μία εκτέλεση της κάθε μεθόδου.

4.1.2 Ανάλυση αποτελεσμάτων για συνολική εκτέλεση

Από τις μετρήσεις που πραγματοποιήθηκαν εκτός των χρόνων για μία εκτέλεση κάθε μεθόδου, έγινε μέτρηση και για το σύνολο εκτελέσεων που έγιναν στο jsPerf. Οι κατηγορίες που δημιουργήθηκαν παραμένουν ίδιες αλλά οι χρόνοι μεταλλάσσονται δίνοντας καλύτερη εικόνα για το πόσο επηρεάζονται οι χρόνοι εκτελέσεις σε συνάρτηση με το μέγεθος του DOM δείχνοντας έτσι πιο ξεκάθαρα τις επιπτώσεις που θα υπάρχουν με την χρησιμοποίηση τις κάθε μεθόδου. Παρατηρήθηκαν χρόνοι, οι οποίοι κυμαίνονται από 4 μίλι – δευτερόλεπτα (Διάγραμμα 11) έως και τα 600 μίλι-δευτερόλεπτα (Διάγραμμα 12), συναντώντας πάλι τις μεθόδους που αφαιρούν να είναι έχουν την πιο γρήγορη εκτέλεση και τις μεθόδους που εισάγουν ή μετατρέπουν το DOM να καταγράφουν τους μεγαλύτερους. Προφανώς εξαίρεση είναι ο selector Id ο οποίος βρίσκεται μία φορά σε κάθε DOM αφού η βασική ιδιότητα και χρήση ενός τέτοιου selector είναι η μοναδικότητα που δημιουργεί. Άμεσα συμπεραίνεται ότι η χρήση Id είναι η ιδανικότερη λύση όσο αναφορά τον χρόνο εκτέλεσης αφού κατά την εύρεση του δεν χρειάζεται από τον browser να εμβαθύνει σε όπως σε ένα Div όπου θα έχει να αναζητήσει μεγαλύτερο αριθμό.



Διάγραμμα 11 - Μέθοδος .remove()

Διάγραμμα 12 - Μέθοδος .unwrap()

Συμπεράσματα και μελλοντικές επεκτάσεις

Από τα αποτελέσματα που παρήχθησαν στην συγκεκριμένη εργασία παρατηρήθηκε ότι η συνεχόμενη χρήση της κατηγορίας «manipulation» της jQuery δεν ενδείκνυται. Διότι για κάθε εισαγωγή μεθόδου που πραγματοποιείτε ο συνολικός χρόνος εκτέλεσης του συνόλου της ιστοσελίδας αυξάνεται, ειδικά σε περιπτώσεις, όπως η μέθοδος .offset(), που σε μεγάλες ιστοσελίδες (εδώ θεωρείτο το μεγάλο

DOM) ο χρόνος που χρειάζεται για την εκτέλεση της είναι απαγορευτικός καθώς ξεπερνούσε και τα πέντε δευτερόλεπτα. Αυτό δεν καθιστά ανάγκη για αντικατάσταση των μεθόδων με πιο πολύπλοκο κώδικα αλλά την χρήση τους σε συγκεκριμένες λειτουργίες.

Ένα βασικό συμπέρασμα που φαίνεται άμεσα στα διαγράμματα είναι η μεγάλη απόκλιση του UC Browser σε σχέση με τον Google Chrome και Mozilla Firefox. Η απόκλιση αυτή δικαιολογείτε με την αρχιτεκτονική σχεδίασης του συγκεκριμένου φυλλομετρητή όπου ως βασικό στόχο έχει την συμπύκνωση των δεδομένων που θα μεταφέρονται στην συσκευή του χρήστη, όπως έχει προαναφερθεί στο κεφάλαιο 2.1.4.

Οι συσκευές που χρησιμοποιήθηκαν είχαν επίσης αντίκτυπο στον χρόνο εκτελέσεων των τεστ με το smartphone το οποίο έχει 2GB RAM να είναι η πιο γρήγορη συσκευή. Παρόλο που κατά την διάρκεια των εκτελέσεων ήταν η μόνη συσκευή που είχε την παροχή δικτύου ενεργή καταναλώνοντας ένα μέρος της υπολογιστικής δύναμης της. Το Tablet με 1GB μνήμης RAM ήταν αυτό με την χαμηλότερη απόδοση. Αφού όταν έγινε προσπάθεια εκτέλεσης στο μεγάλο DOM με την χρήση του UC Browser, δεν έγινε κανονική φόρτωση της ιστοσελίδας εκτός ελάχιστων τυχαίων μεθόδων αφού κατά την δεύτερη εκτέλεση των τεστ οι μέθοδοι που κατάφεραν να «φορτώσουν» και να εκτελεστούν ήταν διαφορετικοί. Με αποτέλεσμα να μην μπορεί να γίνει η άντληση δεδομένων για τις συγκεκριμένες περιπτώσεις.

Συνεπώς για την μέγιστη απόδοση μιας ιστοσελίδας και την γρήγορη ανταπόκριση της, σύμφωνα με τη συνολική μελέτη που πραγματοποιήθηκε κρίνεται σημαντικό από την πλευρά του κάθε χρήστη να χρησιμοποιεί το Google Chrome και να διαθέτει συσκευή που να έχει τουλάχιστον 2 GB μνήμη RAM, προκειμένου η φόρτωση των ιστοσελίδων να υλοποιείται σύμφωνα με τις βασικές προϋποθέσεις, δίχως ίχνος μεγάλης καθυστέρησης. Οι προγραμματιστές με τη σειρά τους θα πρέπει να αποφεύγουν μεθόδους που απαιτούν αρκετά μεγάλο χρόνο εκτέλεσης και να προτιμούν την αξιοποίηση Id selectors έναντι κλάσεων. Επιπρόσθετα θα πρέπει να εφαρμόζουν την τελευταία έκδοση της jQuery και να αποφεύγουν την επαναχρησιμοποίηση του ίδιου selector για κάθε μέθοδο. Αντιθέτως, με την ευκολία που προσφέρει η jQuery στο να γίνετε σύμπτυξη πολλών μεθόδων, γίνεται εξοικονόμηση χρόνου με την μη πολλαπλή αναζήτηση του συγκεκριμένου selector, έτσι αποφεύγεται άσκοπη αναζήτηση οπότε και μείωση του χρόνου εκτέλεσης.

Στην ανάλυση που έγινε με γνώμονα τον selector στην συγκεκριμένη εργασία φαίνεται ότι ο selector Id είναι ο πιο γρήγορος. Ακολουθεί ο selector class ο οποίος είναι πιο αργός και δείχνει χρόνους πιο κοντά στον selector Div, ενώ χειρότερους χρόνους καταγράφει ο selector P. Ειδικά στις περιπτώσεις της συνολικής εκτέλεσης κατέχει τον χειρότερο χρόνο εκτέλεσης.

Τα συνολικά αποτελέσματα και συμπεράσματα που προέκυψαν από την μελέτη αυτή αποτελούν εναρκτήρια κινητήρια δύναμη για περεταίρω έρευνα και δημιουργία μίας «πλατφόρμας». Διότι αν συνδυαστούν τα αποτελέσματα αυτά με ήδη εκπονημένη πτυχιακή εργασία, η οποία είχε ασχοληθεί με τον χρόνο εκτέλεσης αποκλειστικά των selectors, θα μπορούσαν να προσφερθούν σε κάθε προγραμματιστή αναλυτικά στοιχεία για τους χρόνους εκτέλεσης των μεθόδων που ενδιαφέρετε να συμπεριλάβει στην ιστοσελίδα του. Διότι ανάλογα με το μέγεθος του DOM που θα παραχωρεί, θα εισάγει και τον αριθμό των selectors που έχουν χρησιμοποιηθεί και θα του παρουσιάζεται το αποτέλεσμα του χρόνου εκτέλεσης της συγκεκριμένης μεθόδου και ο συνολικός χρόνος που προστίθεται στην εκτέλεση της σελίδας του.

Ένα τέτοιο εργαλείο θα μπορούσε να μειώσει σημαντικά τον χρόνο που δαπανάτε στον έλεγχο και την βελτιστοποίηση της τελικής μορφής μιας ιστοσελίδας ή web εφαρμογής. Από τον σχεδιασμό της δίνεται η δυνατότητα να γίνεται έλεγχος των μεθόδων που ενδέχεται να χρησιμοποιηθούν ώστε να γίνει καλύτερος προγραμματισμός για το τι θα συμπεριληφθεί ώστε να μην υπάρχει μεγάλο κόστος σε χρόνο εκτέλεσης της ίδιας της ιστοσελίδας με όλες τις αρνητικές συνέπειες που έχουν ήδη αναφερθεί.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. Χρήση jQuery παγκοσμίως: https://w3techs.com/technologies/details/js-jquery/all/all
2. Συνδεδεμένες συσκευές στο διαδίκτυο: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/
3. Έρευνα συμπεριφοράς χρηστών: https://www.dynatrace.com/capabilities/synthetic-monitoring/
4. Google Chrome : https://en.wikipedia.org/wiki/Google_Chrome
5. Safari Browser: https://en.wikipedia.org/wiki/Safari_(web_browser)
6. Mozilla Firefox: https://en.wikipedia.org/wiki/Firefox_for_Android
7. Uc Browser: https://en.wikipedia.org/wiki/UC_Browser
8. jsPerf: https://jsperf.com/
9. jQuery Manipulation : http://api.jquery.com/category/manipulation/
10. GSM Arena Specs Smartphone: http://www.gsmarena.com/lg_g2-5543.php
11. GSM Arena Specs Tablet 1 : http://www.gsmarena.com/samsung_galaxy_tab_3_8_0-5456.php
12. GSM Arena Specs Tablet 2 : http://www.gsmarena.com/asus_google_nexus_7-4850.php
13. jQuery manipulation Category: http://api.jquery.com/category/manipulation/

ΠΑΡΑΡΤΗΜΑ

1. Μικρού μεγέθους DOM : https://pastebin.com/ZrwJk3ds
2. Μεσαίου μεγέθους DOM : https://pastebin.com/ciGXVA1Q
3. Μεγάλου μεγέθους DOM : https://pastebin.com/Gn9LTi6Y
4. Μικρού μεγέθους τεστ για:

    4.1 Μέθοδος addClass(): https://jsperf.com/addclasscase

    4.2 Μέθοδος after(): https://jsperf.com/paperaftercases

    4.3 Μέθοδος append(): https://jsperf.com/paperappendcases2

    4.4 Μέθοδος appendTo(): https://jsperf.com/paper-append-to-cases

    4.5 Μέθοδος attr(): https://jsperf.com/paperattrcases3

    4.6 Μέθοδος before(): https://jsperf.com/paperbeforecases

    4.7 Μέθοδος clone(): https://jsperf.com/paperclonecases2

    4.8 Μέθοδος css(): https://jsperf.com/papercsscases

    4.9 Μέθοδος detach(): https://jsperf.com/paperdetachcases

    4.10 Μέθοδος empty(): https://jsperf.com/paperemptycases

    4.11 Μέθοδος hasClass(): https://jsperf.com/paperhasclasscases2

    4.12 Μέθοδος height(): https://jsperf.com/paperheightcaseget-sets

    4.13 Μέθοδος html(): https://jsperf.com/paperhtmlcases3

    4.14 Μέθοδος innerHeight(): https://jsperf.com/paperinnerheightcaseget-sets

    4.15 Μέθοδος insertAfter(): https://jsperf.com/paperinsertaftercases

    4.16 Μέθοδος insertBefore(): https://jsperf.com/paperinsertbeforecases2

    4.17 Μέθοδος offset(): https://jsperf.com/paperoffsetcaseget-sets

    4.18 Μέθοδος outerHeight(): https://jsperf.com/paperouterheightcaseget-sets

    4.19 Μέθοδος outerWidth(): https://jsperf.com/paperouterwidthcaseget-sets

    4.20 Μέθοδος position(): https://jsperf.com/paperpositioncases

    4.21 Μέθοδος prepend(): https://jsperf.com/paperprependcases

4.22 Μέθοδος prependTo(): https://jsperf.com/paperprependtocases

4.23 Μέθοδος prop(): https://jsperf.com/paperpropcases/1

4.24 Μέθοδος removeAttr(): https://jsperf.com/paperremoveatrrcases

4.25 Μέθοδος remove(): https://jsperf.com/paperremovecases/1

4.26 Μέθοδος removeClass(): https://jsperf.com/paperremoveclasscases

4.27 Μέθοδος removeProp(): https://jsperf.com/paperremovepropcases

4.28 Μέθοδος replaceAll(): https://jsperf.com/paperreplaceallcases2

4.29 Μέθοδος replaceWith(): https://jsperf.com/paperreplacewithcases2

4.30 Μέθοδος scrollLeft(): https://jsperf.com/paperscrollleftcases2

4.31 Μέθοδος scrollTop(): https://jsperf.com/paperscrolltopcases2

4.32 Μέθοδος text(): https://jsperf.com/papertextcases

4.33 Μέθοδος toggleClass(): https://jsperf.com/papertoggleclasscases

4.34 Μέθοδος unwrap(): https://jsperf.com/paperunwrapcases

4.35 Μέθοδος val(): https://jsperf.com/papervalcase4

4.36 Μέθοδος width(): https://jsperf.com/paperwidthcaseget-sets

4.37 Μέθοδος wrapAll(): https://jsperf.com/paperwrapallcases2

4.38 Μέθοδος wrap(): https://jsperf.com/paperwrapcases4

4.39 Μέθοδος wrapInner(): https://jsperf.com/paperwrapinnercases2

4.40 Μέθοδος innerWidth(): https://jsperf.com/paperinnerwidthcaseget-sets

5. Μεσαίου μεγέθους τεστ για:

5.1 Μέθοδος addClass(): https://jsperf.com/paperaddclasscasem

5.2 Μέθοδος after(): https://jsperf.com/paperaftercasem/1

5.3 Μέθοδος append(): https://jsperf.com/paperappendcasesm

5.4 Μέθοδος appendTo(): https://jsperf.com/paperappendtocasem

5.5 Μέθοδος attr(): https://jsperf.com/paperattrcasem

5.6 Μέθοδος before(): https://jsperf.com/paperbeforecasem

5.7 Μέθοδος clone(): https://jsperf.com/paperclonecasem

5.8 Μέθοδος css(): https://jsperf.com/papercsscasem

5.9 Μέθοδος detach(): https://jsperf.com/paperdetachcasem

5.10 Μέθοδος empty(): https://jsperf.com/paperemptycasem

5.11 Μέθοδος hasClass(): https://jsperf.com/paperhasclasscasem

5.12 Μέθοδος height(): https://jsperf.com/paperheightcaseget-setm

5.13 Μέθοδος html(): https://jsperf.com/paperhtmlcasem

5.14 Μέθοδος innerHeight(): https://jsperf.com/paperinnerheightcaseget-setm

5.15 Μέθοδος insertAfter(): https://jsperf.com/paperinsertaftercasem

5.16 Μέθοδος insertBefore(): https://jsperf.com/paperinsertbeforecasem

5.17 Μέθοδος offset(): https://jsperf.com/paperoffsetcaseget-setm

5.18 Μέθοδος outerHeight(): https://jsperf.com/paperouterheightcaseget-setm

5.19 Μέθοδος outerWidth(): https://jsperf.com/paperouterwidthcaseget-setm

5.20 Μέθοδος position(): https://jsperf.com/paperpositioncasem

5.21 Μέθοδος prepend(): https://jsperf.com/paperprependcasem

5.22 Μέθοδος prependTo(): https://jsperf.com/paperprependtocasem

5.23 Μέθοδος prop(): https://jsperf.com/paperpropcasem

5.24 Μέθοδος removeAttr(): https://jsperf.com/paperremoveatrrcasem

5.25 Μέθοδος remove(): https://jsperf.com/paperremovecasem

5.26 Μέθοδος removeClass(): https://jsperf.com/paperremoveclasscasem

5.27 Μέθοδος removeProp(): https://jsperf.com/paperremovepropcasem

5.28 Μέθοδος replaceAll(): https://jsperf.com/paperreplaceallcasesm

5.29 Μέθοδος replaceWith(): https://jsperf.com/paperreplacewithcasem

5.30 Μέθοδος scrollLeft(): https://jsperf.com/paperscrollleftcasem

5.31 Μέθοδος scrollTop(): https://jsperf.com/paperscrolltopcasem

5.32 Μέθοδος text(): https://jsperf.com/papertextcasem

5.33 Μέθοδος toggleClass(): https://jsperf.com/papertoggleclasscasem2

5.34 Μέθοδος unwrap(): https://jsperf.com/paperunwrapcasem

5.35 Μέθοδος val(): https://jsperf.com/papervalcasem

5.36 Μέθοδος width(): https://jsperf.com/paperwidthcaseget-setm

5.37 Μέθοδος wrapAll(): https://jsperf.com/paperwrapallcasem

5.38 Μέθοδος wrap(): https://jsperf.com/paperwrapcasem

5.39 Μέθοδος wrapInner(): https://jsperf.com/paperwrapinnercasem

5.40 Μέθοδος innerWidth(): https://jsperf.com/paperinnerwidthcaseget-setm

6. Μεγάλου μεγέθους τεστ για:

6.1 Μέθοδος addClass(): https://jsperf.com/paperaddclasscaseb

6.2 Μέθοδος after(): https://jsperf.com/paperaftercaseb

6.3 Μέθοδος append(): https://jsperf.com/paperappendcasesb

6.4 Μέθοδος appendTo(): https://jsperf.com/paperappendtocaseb

6.5 Μέθοδος attr(): https://jsperf.com/paperattrcaseb

6.6 Μέθοδος before(): https://jsperf.com/paperbeforecaseb

6.7 Μέθοδος clone(): https://jsperf.com/paperclonecaseb

6.8 Μέθοδος css(): https://jsperf.com/papercsscaseb

6.9 Μέθοδος detach(): https://jsperf.com/paperdetachcaseb

6.10 Μέθοδος empty(): https://jsperf.com/paperemptycaseb

6.11 Μέθοδος hasClass(): https://jsperf.com/paperhasclasscaseb2

6.12 Μέθοδος height(): https://jsperf.com/paperheightcaseget-setb

6.13 Μέθοδος html(): https://jsperf.com/paperhtmlcasem

6.14 Μέθοδος innerHeight(): https://jsperf.com/paperinnerheightcaseget-setb

6.15 Μέθοδος insertAfter(): https://jsperf.com/paperinsertaftercaseb

6.16 Μέθοδος insertBefore(): https://jsperf.com/paperinsertbeforecaseb

6.17 Μέθοδος offset(): https://jsperf.com/paperoffsetcaseget-setb

6.18 Μέθοδος outerHeight(): https://jsperf.com/paperouterheightcaseget-setb

6.19 Μέθοδος outerWidth(): https://jsperf.com/paperouterwidthcaseget-setb

6.20 Μέθοδος position(): https://jsperf.com/paperpositioncaseb

6.21 Μέθοδος prepend(): https://jsperf.com/paperprependcaseb

6.22 Μέθοδος prependTo(): https://jsperf.com/paperprependtocaseb

6.23 Μέθοδος prop(): https://jsperf.com/paperpropcaseb

6.24 Μέθοδος removeAttr(): https://jsperf.com/paperremoveatrrcaseb

6.25 Μέθοδος remove(): https://jsperf.com/paperremovecaseb

6.26 Μέθοδος removeClass(): https://jsperf.com/paperremoveclasscaseb

6.27 Μέθοδος removeProp(): https://jsperf.com/paperremovepropcaseb

6.28 Μέθοδος replaceAll(): https://jsperf.com/paperreplaceallcasesb

6.29 Μέθοδος replaceWith(): https://jsperf.com/paperreplacewithcaseb

6.30 Μέθοδος scrollLeft(): https://jsperf.com/paperscrollleftcaseb

6.31 Μέθοδος scrollTop(): https://jsperf.com/paperscrolltopcaseb

6.32 Μέθοδος text(): https://jsperf.com/papertextcaseb

6.33 Μέθοδος toggleClass(): https://jsperf.com/papertoggleclasscaseb

6.34 Μέθοδος unwrap(): https://jsperf.com/paperunwrapcaseb

6.35 Μέθοδος val(): https://jsperf.com/papervalcaseb

6.36 Μέθοδος width(): https://jsperf.com/paperwidthcaseget-setb

6.37 Μέθοδος wrapAll(): https://jsperf.com/paperwrapallcaseb

6.38 Μέθοδος wrap(): https://jsperf.com/paperwrapcaseb

6.39 Μέθοδος wrapInner(): https://jsperf.com/paperwrapinnercaseb

6.40 Μέθοδος innerWidth(): https://jsperf.com/paperinnerwidthcaseget-setb

ΠΑΡΑΡΤΗΜΑ ΠΙΝΑΚΩΝ

Πίνακας Α. Συνολικά αποτελέσματα του Smartphone LG

SmartPhone | RAM: 2GB - CPU: Quad-core 2.26 GHz Krait 400 | Android v 5.0.2																													
Overall DOM Testing Cases																													
Methods:	Comments:	Small Dom						Medium Dom						Big Dom															
		Chrome			UC Browser			Mozilla Firefox			Chrome			UC Browser			Mozilla Firefox			Chrome			UC Browser			Mozilla Firefox		
		1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time
.addclass()		0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.01	0.03	0.03	0.03	0.03	0.03	0.03	0.01	0.02	0.01	0.02	0.02	0.02	0.04	0.05	0.04	0.02	0.01	0.01
.after()		0.72	0.72	0.72	0.96	0.91	0.93	1.00	1.01	1.00	0.33	0.29	0.31	0.40	0.39	0.40	0.40	0.38	0.39	0.24	0.25	0.25	0.39	0.36	0.38	0.33	0.38	0.35
.append()		0.70	0.69	0.70	0.87	0.86	0.86	1.00	1.05	1.03	0.29	0.31	0.30	0.41	0.41	0.41	0.40	0.44	0.42	0.25	0.24	0.24	0.38	0.38	0.38	0.30	0.30	0.30
.appendTo()		0.70	0.61	0.66	0.96	0.92	0.94	0.91	0.84	0.87	0.34	0.35	0.35	0.50	0.52	0.51	0.29	0.29	0.29	0.32	0.30	0.31	0.52	0.44	0.48	0.27	0.32	0.29
.attr()	(Get)	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.01	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
	(Set)	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.28	0.01	0.15	0.02	0.57	0.30	0.01	0.30	0.16	0.02	0.01	0.02	0.02	0.03	0.02	0.01	0.00	0.00
.before()		1.61	0.72	1.16	0.89	0.88	0.89	1.06	1.01	1.03	0.30	0.29	0.30	0.39	0.45	0.42	0.39	0.39	0.39	0.25	0.25	0.25	0.31	0.34	0.33	0.31	0.00	0.16
.clone()		0.43	0.46	0.45	0.98	0.57	0.77	0.56	0.53	0.55	0.18	0.17	0.17	0.22	0.22	0.22	0.20	0.21	0.20	0.17	0.16	0.16	0.19	0.25	0.22	0.17	0.16	0.17
.css()		0.03	0.03	0.03	0.04	0.04	0.04	0.05	0.05	0.05	0.02	0.02	0.02	0.04	0.12	0.08	0.05	0.04	0.04	0.02	0.02	0.02	0.03	0.03	0.03	0.04	0.04	0.04
.detach()		0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	-0.01	0.00	0.00	0.00	0.00	-0.01	0.00	-0.01	0.00	-0.01	-0.01
.empty()		0.00	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	0.01	0.00	0.00	0.00	0.00
.hasClass()		0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.03	0.02	0.01	0.00	0.00
.height()	(Get)	0.07	0.07	0.07	0.08	0.10	0.09	0.08	0.08	0.08	0.07	0.06	0.07	0.07	0.11	0.09	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07	0.07
	(Set)	0.07	0.10	0.08	0.30	0.29	0.29	0.15	0.14	0.15	0.08	0.07	0.08	2.14	1.37	1.75	0.10	0.10	0.10	0.05	0.07	0.06	5.15	4.74	4.94	0.09	0.10	0.10
.html()		0.11	0.12	0.11	0.19	0.15	0.17	0.10	0.09	0.10	0.06	0.06	0.06	0.15	0.12	0.13	0.06	0.05	0.06	0.06	0.07	0.06	0.13	0.10	0.12	0.05	0.05	0.05
.innerHeight()	(Get)	0.05	0.05	0.05	0.06	0.06	0.06	0.06	0.07	0.06	0.07	0.05	0.06	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.05	0.05	0.08	0.07	0.08	0.05	0.05	0.05
	(Set)	0.13	0.13	0.13	0.40	0.41	0.41	0.25	0.27	0.26	0.19	0.14	0.17	1.36	2.39	1.88	0.17	0.18	0.17	0.14	0.15	0.14	6.25	2.27	4.26	0.16	0.16	0.16
.innerWidth()	(Get)	0.00	0.05	0.03	0.07	0.07	0.07	0.06	0.06	0.06	0.05	0.05	0.05	0.06	0.06	0.06	0.05	0.05	0.05	0.05	0.06	0.05	0.06	0.06	0.06	0.05	0.05	0.05
	(Set)	0.04	0.17	0.10	0.41	1.76	1.08	0.27	0.23	0.25	0.15	0.14	0.14	3.52	2.59	3.06	0.17	0.16	0.17	0.14	0.15	0.14	18.42	5.90	12.16	0.23	0.16	0.20
.insertAfter()		0.70	0.71	0.71	0.90	0.89	0.89	0.92	0.88	0.90	0.34	0.33	0.34	0.45	0.43	0.44	0.37	0.37	0.37	0.29	0.31	0.30	0.47	0.34	0.41	0.32	0.32	0.32
.insertBefore()		0.70	0.72	0.71	0.90	0.86	0.88	0.87	0.86	0.86	0.35	0.35	0.35	0.46	0.45	0.45	0.37	0.37	0.37	0.29	0.31	0.30	0.39	0.38	0.39	0.33	0.33	0.33
.offset()	(Get)	0.02	0.02	0.02	0.01	0.01	0.01	0.02	0.02	0.02	0.01	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01
	(Set)	1.12	1.10	1.11	29.97	21.11	25.54	0.45	0.43	0.44	1.42	1.56	1.49	53.59	133.70	93.65	0.44	0.53	0.48	2.88	3.66	3.27	213.60	186.96	200.28	0.80	0.90	0.85
.outerHeight()	(Get)	0.03	0.03	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.04	0.03	0.04	0.05	0.04	0.05	0.04	0.04	0.04	0.03	0.03	0.03	0.04	0.04	0.04	0.04	0.03	0.03
	(Set)	0.19	0.19	0.19	0.90	0.50	0.70	0.33	0.57	0.45	0.22	0.24	0.23	1.42	1.20	1.31	0.24	0.23	0.23	0.22	0.20	0.21	6.33	6.19	6.26	0.22	0.23	0.22
.outerWidth()	(Get)	0.03	0.03	0.03	0.04	0.04	0.04	0.04	0.04	0.04	0.03	0.03	0.03	0.06	0.04	0.05	0.03	0.03	0.03	0.03	0.03	0.03	0.04	0.10	0.07	0.03	0.03	0.03
	(Set)	0.27	0.30	0.29	0.55	0.51	0.53	0.34	0.32	0.33	0.22	0.21	0.21	2.84	2.83	2.83	0.24	0.23	0.24	0.20	0.20	0.20	6.98	7.19	7.09	0.23	0.22	0.23
.position()		0.15	0.15	0.15	0.20	0.21	0.21	0.19	0.19	0.19	0.13	0.12	0.12	0.16	0.17	0.16	0.23	0.23	0.23	0.12	0.12	0.12	0.15	0.15	0.15	0.21	0.21	0.21
.prepend()		0.73	0.75	0.74	0.85	0.82	0.84	1.04	1.12	1.08	0.31	0.30	0.31	0.38	0.39	0.38	0.39	0.41	0.40	0.25	0.27	0.26	0.33	0.34	0.33	0.32	0.34	0.33
.prependTo()		0.72	0.72	0.72	0.90	0.91	0.91	0.90	1.10	1.00	0.35	0.35	0.35	0.45	0.44	0.44	0.38	0.37	0.38	0.29	0.32	0.31	0.39	0.43	0.41	0.35	0.31	0.33
.prop()		0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.00	0.00
.removeAttr()		0.06	0.06	0.06	0.09	0.08	0.09	0.09	0.09	0.09	0.01	0.01	0.01	0.01	0.02	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.02	0.01	0.02	0.00	0.00	0.00
.removeCase()		0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	-0.01	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.01	-0.01	0.00	0.00
.removeClass()		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.02	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.02	0.01	0.02	0.01	0.00	0.01
.removeProp()		0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.03	0.01	0.02	0.00	0.00	0.00
.replaceAll()		0.06	0.07	0.06	0.06	0.07	0.06	0.04	0.10	0.07	0.05	0.04	0.05	0.06	0.05	0.05	0.04	0.03	0.03	0.04	0.04	0.04	0.05	0.06	0.06	0.03	0.03	0.03
.replaceWith()		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.01	-0.01
.scrollLeft()	(Get)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00	0.00
	(Set)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00
.scrollTop()	(Get)	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00
	(Set)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.06	0.03	0.01	0.00	0.01	0.02	0.01	0.02	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.00
.text()		0.09	0.10	0.09	0.12	0.11	0.11	0.08	0.08	0.08	0.01	0.01	0.01	0.04	0.04	0.04	0.02	0.01	0.02	0.01	0.02	0.01	0.03	0.06	0.05	0.02	0.01	0.01
.toggleClass()		0.08	0.08	0.08	0.07	0.08	0.07	0.08	0.11	0.09	0.06	0.06	0.06	0.10	0.14	0.12	0.07	0.07	0.07	0.05	0.06	0.06	0.11	0.13	0.12	0.07	0.06	0.07
.unwrap()		0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.01	0.01	0.02	0.02	0.02	0.03	0.03	0.03	0.83	0.73	0.78	0.02	0.02	0.02	0.04	0.06	0.05	0.02	0.49	0.25
.val()		0.02	0.01	0.02	0.03	0.04	0.03	0.03	0.03	0.03	0.02	0.02	0.02	0.03	0.03	0.03	-0.02	-0.02	-0.02	0.02	0.02	0.02	0.00	0.07	0.04	0.03	0.03	0.03
.width()	(Get)	0.07	0.07	0.07	0.08	0.09	0.09	0.08	0.12	0.10	0.07	0.06	0.06	0.07	0.08	0.08	0.07	0.07	0.07	0.06	0.06	0.06	0.10	0.12	0.11	0.07	0.07	0.07
	(Set)	0.10	0.09	0.10	0.28	0.27	0.28	0.16	0.24	0.20	0.08	0.07	0.08	1.10	1.58	1.34	0.10	0.10	0.10	0.08	0.07	0.07	5.21	0.16	2.68	0.10	0.09	0.09
.wrapAll()		2.91	2.17	2.54	2.98	3.03	3.00	2.54	3.24	2.89	0.87	0.92	0.90	1.60	1.16	1.38	1.08	1.05	1.07	0.76	0.83	0.80	0.98	1.25	1.11	0.95	0.89	0.92
.wrap()		2.12	2.28	2.20	6.93	6.74	6.83	2.87	4.66	3.77	1.59	1.53	1.56	3.92	4.32	4.12	2.04	2.14	2.09	1.32	1.37	1.35	2.90	4.01	3.45	1.95	3.41	2.68
.wrapInner()		2.49	1.70	2.10	9.50	10.29	9.90	3.28	4.24	3.76	1.11	1.16	1.13	3.33	3.54	3.44	2.43	2.31	2.37	1.44	1.07	1.25	2.98	2.37	2.68	2.11	2.03	2.07

Πίνακας Β. Συνολικά αποτελέσματα του Tablet-SAMSUNG

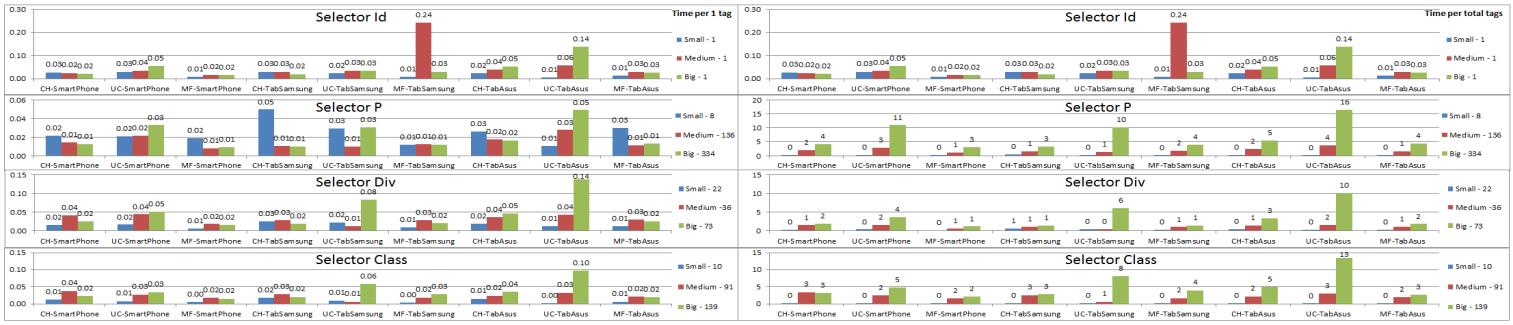
Tablet | RAM: 1.5GB - CPU: Dual-core 1.5 GHz Cortex-A9 | Android v 4.4.2																														
Overall DOM Testing Cases																														
Method:	Comments:	Small Dom									Medium Dom									Big Dom										
		Chrome			UC Browser			Mozilla Firefox			Chrome			UC Browser			Mozilla Firefox			Chrome			UC Browser			Mozilla Firefox		
		1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time
.addclass()		0.04	0.02	0.03	0.04	0.00	0.02	0.01	0.01	0.01	0.03	0.02	0.02	0.01	0.02	0.02	0.02	0.13	0.08	0.01	0.02	0.02	0.05	0.05	0.05	0.02	0.02	0.02
.after()		0.98	0.99	0.99	1.43	1.45	1.44	1.41	1.33	1.37	0.38	0.42	0.40	0.57	0.53	0.55	0.51	0.45	0.48	0.30	0.31	0.31	0.57	0.57	0.57	0.46	0.48	0.47
.append()		0.90	0.88	0.89	1.20	1.22	1.21	1.39	1.41	1.40	0.38	0.41	0.40	0.54	0.57	0.56	0.44	0.45	0.44	0.30	0.30	0.30	0.51	0.54	0.53	0.34	0.36	0.35
.appendTo()		0.90	0.93	0.91	1.20	1.16	1.18	1.04	1.05	1.05	0.39	0.43	0.41	0.60	0.62	0.61	0.30	0.34	0.32	0.36	0.42	0.39	0.63	0.58	0.60	0.31	0.36	0.33
.attr()	(Get)	0.00	0.00	0.00	0.00	-0.01	-0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.01	0.00	0.03	0.03	0.03	0.00	0.00	0.00
	(Set)	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.39	0.01	0.20	0.02	1.09	0.55	0.01	0.02	0.01	0.01	0.01	0.01	0.04	0.03	0.04	0.01	0.01	0.01
.before()		1.04	0.92	0.98	1.30	1.21	1.25	1.38	1.37	1.38	0.40	0.40	0.40	0.56	0.58	0.57	0.46	0.46	0.46	0.32	0.35	0.33	0.52	0.46	0.49	0.40	0.39	0.39
.clone()		0.55	0.56	0.55	0.71	0.76	0.74	0.75	0.75	0.75	0.19	0.21	0.20	0.25	0.25	0.25	0.25	0.30	0.27	0.16	0.19	0.17	0.24	0.23	0.23	0.22	0.22	0.22
.css()		0.03	0.02	0.02	0.08	0.07	0.07	0.07	0.05	0.06	0.02	0.02	0.02	0.07	0.06	0.07	0.05	0.05	0.05	0.02	0.03	0.02	0.08	0.06	0.07	0.04	0.04	0.04
.detach()		-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	0.00	0.00	0.00	-0.01	-0.02	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.02	0.02	-0.01	0.00	0.00
.empty()		0.01	0.00	0.00	0.00	-0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	-0.02	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.02	0.00	0.01	0.00
.hasClass()		0.01	0.00	0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.01	0.00	0.01	0.01	0.01	0.03	0.03	0.03	0.01	0.01	0.01
.height()	(Get)	0.06	0.07	0.06	0.11	0.10	0.10	0.09	0.08	0.09	0.07	0.06	0.07	0.09	0.06	0.08	0.08	0.08	0.08	0.06	0.06	0.06	0.13	0.17	0.15	0.08	0.09	0.08
	(Set)	0.09	0.09	0.09	0.40	0.42	0.41	0.17	0.15	0.16	0.08	0.07	0.07	1.62	2.19	1.90	0.11	0.11	0.11	0.07	0.16	0.12	4.74	3.02	3.88	0.11	0.10	0.10
.html()		0.16	0.17	0.16	0.24	0.31	0.28	0.10	0.10	0.10	0.08	0.09	0.08	0.16	0.13	0.14	0.05	0.07	0.06	0.07	0.08	0.08	0.21	0.11	0.16	0.06	0.06	0.06
.innerHeight()	(Get)	0.05	0.05	0.05	0.08	0.06	0.07	0.07	0.06	0.07	0.06	0.05	0.05	0.08	0.06	0.07	0.06	0.06	0.06	0.05	0.06	0.05	0.19	0.09	0.14	0.06	0.08	0.07
	(Set)	0.20	0.19	0.19	1.05	0.53	0.79	0.30	0.25	0.28	0.15	0.14	0.15	2.32	2.01	2.17	0.17	0.18	0.17	0.14	0.14	0.14	4.09	4.05	4.07	0.17	0.18	0.17
.innerWidth()	(Get)	0.05	0.06	0.05	0.06	0.09	0.08	0.08	0.06	0.07	0.05	0.05	0.05	0.06	0.08	0.07	0.05	0.07	0.06	0.05	0.06	0.05	0.03	0.08	0.06	0.06	0.07	0.06
	(Set)	0.19	0.17	0.18	0.52	0.57	0.55	0.31	0.24	0.27	0.14	0.12	0.13	2.10	2.50	2.30	0.17	0.17	0.17	0.13	0.14	0.13	0.14	3.52	1.83	0.17	0.18	0.18
.insertAfter()		0.92	0.95	0.93	1.11	1.11	1.11	1.17	1.15	1.16	0.41	0.42	0.41	0.53	0.52	0.52	0.41	0.46	0.44	0.36	0.44	0.40	0.46	0.33	0.39	0.37	0.38	0.37
.insertBefore()		0.93	0.88	0.91	1.17	1.21	1.19	1.01	1.19	1.10	0.40	0.40	0.40	0.55	0.54	0.55	0.46	0.49	0.47	0.35	0.41	0.38	0.51	0.45	0.48	0.39	0.41	0.40
.offset()	(Get)	0.01	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.03	0.00	0.01	0.01	0.02	0.02	0.02	0.02	0.02	-0.01	0.03	0.01	0.02	0.02	0.02
	(Set)	1.85	1.17	1.51	38.04	39.70	38.87	0.59	0.47	0.53	2.31	3.45	2.88	71.30	158.82	115.06	0.68	0.86	0.77	4.89	4.60	4.74	12.08	235.35	123.72	0.89	1.41	1.15
.outerHeight()	(Get)	0.03	0.03	0.03	0.05	0.06	0.06	0.05	0.04	0.04	0.04	0.03	0.03	0.46	0.05	0.25	0.04	0.05	0.04	0.03	0.04	0.04	0.08	0.10	0.09	0.04	0.05	0.04
	(Set)	0.33	0.27	0.30	0.70	0.70	0.70	0.39	0.34	0.36	0.22	0.19	0.20	2.34	1.93	2.14	0.23	0.24	0.24	0.20	0.21	0.20	4.22	3.66	3.94	0.21	0.27	0.24
.outerWidth()	(Get)	0.05	0.04	0.04	0.06	0.06	0.06	0.04	0.04	0.04	0.04	0.04	0.04	0.06	0.04	0.05	0.04	0.04	0.04	0.03	0.04	0.03	0.09	0.07	0.08	0.04	0.04	0.04
	(Set)	0.30	0.26	0.28	0.66	0.69	0.68	0.39	0.33	0.36	0.25	0.18	0.22	2.04	4.52	3.28	0.24	0.24	0.24	0.19	0.21	0.20	3.57	4.63	4.10	0.25	0.24	0.25
.position()		0.16	0.16	0.16	0.25	0.21	0.23	0.21	0.19	0.20	0.27	0.13	0.20	0.16	0.15	0.15	0.32	0.30	0.31	0.11	0.14	0.12	0.33	0.17	0.25	0.26	0.37	0.32
.prepend()		0.94	0.99	0.97	1.09	1.01	1.05	1.51	1.28	1.39	0.38	0.38	0.38	0.50	0.45	0.47	0.47	0.48	0.47	0.30	0.32	0.31	0.54	0.46	0.50	0.39	0.40	0.40
.prependTo()		0.92	0.93	0.93	1.13	1.11	1.12	1.13	1.05	1.09	0.41	0.41	0.41	0.55	0.50	0.53	0.41	0.44	0.43	0.34	0.42	0.38	0.53	0.45	0.49	0.34	0.37	0.36
.prop()		0.01	0.00	0.01	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.01	0.00	0.00	0.01	0.00	0.04	0.03	0.03	0.34	0.37	0.36
.removeAttr()		0.05	0.07	0.06	0.11	0.10	0.11	0.09	0.08	0.09	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.05	0.04	0.04	0.02	0.01	0.02
.removeCase()		-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	-0.01	0.00	0.00	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00
.removeClass()		0.00	0.00	0.00	0.00	-0.02	-0.01	0.00	0.00	0.00	0.01	0.01	0.01	0.02	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.04	0.04	0.04	0.01	0.01	0.01
.removeProp()		0.01	0.00	0.00	0.02	0.00	0.01	0.02	0.01	0.01	0.00	0.00	0.00	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.04	0.04	0.04	0.01	0.01	0.01
.replaceAll()		0.08	0.07	0.08	0.09	0.10	0.09	0.06	0.05	0.05	0.06	0.06	0.06	0.07	0.06	0.06	0.03	0.04	0.04	0.05	0.07	0.06	0.09	0.08	0.08	0.04	0.04	0.04
.replaceWith()		0.01	0.00	0.00	0.44	-0.01	0.21	0.00	-0.01	0.00	0.00	0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.02	0.02	0.02	0.00	0.00	0.00
.scrollLeft()	(Get)	0.00	0.00	0.00	0.00	-0.02	-0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.01	0.00	0.01	0.00	0.00	0.00	0.00	0.01	0.00	0.02	0.02	0.02	0.00	0.01	0.00
	(Set)	0.00	0.01	0.01	0.01	-0.01	0.00	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.00	0.00	0.01	0.01	0.01	0.03	0.03	0.03	0.00	0.01	0.01
.scrollTop()	(Get)	0.00	0.00	0.00	0.00	-0.01	-0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.03	0.03	0.03	0.00	0.01	0.01
	(Set)	0.01	0.01	0.01	0.01	-0.01	0.00	0.01	0.00	0.00	0.01	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.01	0.03	0.03	0.03	0.01	0.01	0.01
.text()		0.12	0.12	0.12	0.14	0.15	0.14	0.08	0.07	0.08	0.02	0.02	0.02	0.03	0.04	0.04	0.02	0.01	0.01	0.01	0.02	0.02	0.06	0.05	0.05	0.01	0.02	0.02
.toggleClass()		0.07	0.07	0.07	0.12	0.11	0.12	0.10	0.08	0.09	0.05	0.06	0.05	0.08	0.09	0.09	0.07	0.09	0.08	0.05	0.07	0.06	0.09	0.11	0.10	0.09	0.08	0.08
.unwrap()		0.02	0.02	0.02	0.03	0.01	0.02	0.02	0.01	0.02	0.02	0.02	0.02	0.03	0.03	0.03	1.37	1.40	1.39	0.02	0.02	0.02	0.06	0.10	0.08	0.52	0.51	0.51
.val()		0.02	0.01	0.02	0.00	0.00	0.00	0.03	0.03	0.03	0.02	0.02	0.02	0.03	0.03	0.03	-0.04	-0.04	-0.04	0.02	0.02	0.02	0.03	0.03	0.03	0.03	0.03	0.03
.width()	(Get)	0.07	0.07	0.07	0.10	0.11	0.10	0.11	0.08	0.09	0.06	0.07	0.07	0.08	0.08	0.08	0.07	0.09	0.08	0.06	0.07	0.06	0.12	0.09	0.10	0.07	0.09	0.08
	(Set)	0.10	0.09	0.09	0.45	0.46	0.46	0.20	0.14	0.17	0.07	0.07	0.07	3.93	1.60	2.77	0.10	0.10	0.10	0.07	0.07	0.07	8.50	3.10	5.80	0.11	0.12	0.11
.wrapAll()		2.64	2.74	2.69	3.15	3.27	3.21	2.71	2.55	2.63	1.12	1.10	1.11	1.68	1.55	1.61	1.40	1.38	1.39	1.14	1.04	1.09	1.40	1.28	1.34	1.36	1.30	1.33
.wrap()		3.29	3.06	3.17	5.73	5.61	5.67	3.71	4.25	3.98	2.01	1.95	1.98	4.05	3.84	3.94	2.75	2.78	2.77	1.90	1.98	1.94	3.06	3.02	3.04	2.41	2.45	2.43
.wrapInner()		2.51	2.45	2.48	8.55	8.34	8.44	4.61	4.17	4.39	1.60	1.69	1.65	3.54	1.57	2.55	2.96	3.24	3.10	1.67	1.50	1.59	2.72	2.86	2.79	2.74	2.76	2.75

Tablet | RAM: 1GB - CPU: Quad-core 1.2 GHz Cortex-A9																													
Overall DOM Testing Cases																													
Method:	Comments:	Small Dom										Medium Dom										Big Dom							
		Chrome			UC Browser			Mozilla Firefox				Chrome			UC Browser			Mozilla Firefox				Chrome			UC Browser			Mozilla Firefox		
		1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time		1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time		1st run time	2nd run time	Average time	1st run time	2nd run time	Average time	1st run time	2nd run time	Average time
.adddass()		0.02	0.02	0.02	0.00	0.01	0.01	0.01	0.02	0.02		0.03	0.03	0.03	0.03	0.05	0.04	0.02	0.02	0.02		0.02	0.05	0.04	0.08	0.13	0.11	0.02	0.02	0.02
.after()		0.96	1.45	1.20	1.74	1.39	1.56	1.28	1.28	1.28		0.64	0.42	0.53	0.63	0.70	0.67	0.53	0.53	0.53		0.34	0.05	0.19	0.66	0.55	0.60	0.58	0.50	0.54
.append()		0.95	1.08	1.01	1.26	1.22	1.24	1.22	1.39	1.30		0.62	0.40	0.51	0.68	0.62	0.65	0.48	0.49	0.48		0.36	0.35	0.35	0.59	0.77	0.68	0.39	0.38	0.38
.appendTo()		0.95	1.04	0.99	1.17	1.25	1.21	1.15	1.08	1.12		0.62	0.51	0.57	0.67	0.68	0.68	0.38	0.39	0.39		0.43	0.44	0.44	#DIV/0!	#DIV/0!	#DIV/0!	0.36	0.34	0.35
.attr()	(Get)	0.00	0.01	0.00	0.16	0.01	0.08	0.01	0.01	0.01		0.01	0.00	0.01	0.01	0.00	0.01	0.00	0.01	0.01		0.00	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.01	0.00
	(Set)	0.02	0.02	0.02	0.02	0.03	0.02	0.02	0.01	0.02		0.67	0.59	0.63	0.03	1.11	0.57	0.01	0.90	0.46		0.01	0.02	0.02	#DIV/0!	#DIV/0!	#DIV/0!	0.01	0.01	0.01
.before()		0.95	1.34	1.14	1.36	1.48	1.42	1.29	1.30	1.29		0.58	0.41	0.50	0.66	0.50	0.58	0.52	0.52	0.52		0.36	0.36	0.36	#DIV/0!	#DIV/0!	#DIV/0!	0.41	0.43	0.42
.clone()		0.59	0.69	0.64	0.58	0.79	0.69	0.79	0.79	0.79		0.31	0.25	0.28	0.31	0.33	0.32	0.29	0.30	0.30		0.22	0.23	0.22	0.40	#DIV/0!	#DIV/0!	0.26	0.26	0.26
.css()		0.03	0.03	0.03	0.10	0.11	0.10	0.08	0.07	0.07		0.04	0.03	0.03	0.08	0.09	0.08	0.05	0.06	0.05		0.02	0.03	0.03	#DIV/0!	#DIV/0!	#DIV/0!	0.05	0.05	0.05
.detach()		-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01		-0.01	-0.01	-0.01	-0.01	-0.03	-0.02	-0.01	-0.01	-0.01		-0.01	-0.01	-0.01	#DIV/0!	#DIV/0!	#DIV/0!	-0.01	-0.01	-0.01
.empty()		0.00	0.00	0.00	-0.01	0.00	-0.01	0.00	0.00	0.00		-0.01	-0.01	-0.01	-0.01	-0.01	-0.01	0.00	0.00	0.00		0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	0.00	0.00
.hasClass()		0.01	0.01	0.01	0.02	0.00	0.01	0.00	0.01	0.00		0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01		0.01	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.01	0.01	0.01
.height()	(Get)	0.09	0.11	0.10	0.11	0.12	0.12	0.12	0.12	0.12		0.11	0.09	0.10	0.10	0.10	0.10	0.01	0.01	0.01		0.08	0.07	0.08	#DIV/0!	#DIV/0!	#DIV/0!	0.10	0.10	0.10
	(Set)	0.14	0.25	0.20	0.51	0.49	0.50	0.26	0.20	0.23		0.23	0.11	0.17	2.02	4.98	3.50	0.01	0.01	0.01		0.10	0.20	0.15	#DIV/0!	#DIV/0!	#DIV/0!	0.14	0.14	0.14
.html()		0.16	0.15	0.16	0.37	0.38	0.37	0.13	0.13	0.13		0.11	0.10	0.11	0.20	0.26	0.23	0.07	0.07	0.07		0.09	0.09	0.09	#DIV/0!	#DIV/0!	#DIV/0!	0.07	0.07	0.07
.innerHeight()	(Get)	0.06	0.08	0.07	0.10	0.09	0.09	0.09	0.09	0.09		0.09	0.07	0.08	0.14	0.12	0.13	0.08	0.08	0.08		0.07	0.07	0.07	#DIV/0!	#DIV/0!	#DIV/0!	0.08	0.08	0.08
	(Set)	0.29	0.40	0.35	1.12	0.71	0.91	0.37	0.34	0.35		0.34	0.19	0.26	3.96	2.45	3.20	0.25	0.23	0.24		0.19	0.18	0.19	#DIV/0!	#DIV/0!	#DIV/0!	0.24	0.21	0.22
.innerWidth()	(Get)	0.07	0.08	0.07	0.09	0.09	0.09	0.09	0.09	0.09		0.09	0.15	0.12	0.09	0.13	0.11	0.08	0.08	0.08		0.06	0.04	0.05	#DIV/0!	#DIV/0!	#DIV/0!	0.07	0.08	0.08
	(Set)	0.27	0.38	0.32	0.64	0.74	0.69	0.35	0.39	0.37		0.32	2.15	1.24	2.58	2.82	2.70	0.24	0.24	0.24		0.18	0.12	0.15	#DIV/0!	#DIV/0!	#DIV/0!	0.25	0.22	0.23
.insertAfter()		1.02	1.06	1.04	1.21	1.24	1.22	1.13	1.15	1.14		0.57	0.50	0.53	0.66	0.58	0.62	0.49	0.49	0.49		0.43	0.45	0.44	#DIV/0!	#DIV/0!	#DIV/0!	0.43	0.42	0.43
.insert Before()		0.98	0.99	0.99	1.16	1.15	1.15	1.14	1.12	1.13		0.57	0.51	0.54	0.59	0.56	0.57	0.49	0.50	0.50		0.42	0.45	0.44	#DIV/0!	#DIV/0!	#DIV/0!	0.43	0.41	0.42
.offset()	(Get)	0.02	0.02	0.02	0.02	0.01	0.01	0.02	0.02	0.02		0.02	0.02	0.02	0.03	0.02	0.02	0.02	0.02	0.02		0.02	0.06	0.04	#DIV/0!	#DIV/0!	#DIV/0!	0.02	0.02	0.02
	(Set)	1.50	0.87	1.18	28.40	44.27	36.34	0.70	0.63	0.67		0.78	2.75	1.77	77.46	188.69	133.08	0.80	0.77	0.79		4.61	6.45	5.53	#DIV/0!	#DIV/0!	#DIV/0!	1.37	1.05	1.21
.outerHeight()	(Get)	0.04	0.06	0.05	0.08	0.13	0.10	0.05	0.05	0.05		0.07	0.05	0.06	0.08	0.08	0.08	0.05	0.05	0.05		0.04	0.05	0.05	#DIV/0!	#DIV/0!	#DIV/0!	0.05	0.05	0.05
	(Set)	0.38	0.52	0.45	0.81	1.57	1.19	0.50	0.48	0.49		0.43	0.27	0.35	3.93	2.94	3.44	0.32	0.33	0.32		0.27	0.27	0.27	#DIV/0!	#DIV/0!	#DIV/0!	0.31	0.31	0.31
.outerWidth()	(Get)	0.09	0.06	0.08	0.07	0.06	0.06	0.06	0.06	0.06		0.07	0.05	0.06	0.09	0.08	0.09	0.05	0.06	0.05		0.05	0.05	0.05	#DIV/0!	#DIV/0!	#DIV/0!	0.05	0.05	0.05
	(Set)	0.49	0.51	0.50	0.81	0.88	0.84	0.51	0.46	0.49		0.43	0.28	0.35	2.99	2.62	2.80	0.33	0.33	0.33		0.26	0.30	0.28	#DIV/0!	#DIV/0!	#DIV/0!	0.31	0.31	0.31
.position()		0.25	0.26	0.25	0.32	0.31	0.31	0.28	0.27	0.28		0.26	0.17	0.21	0.22	0.24	0.23	0.34	0.34	0.34		0.16	0.17	0.16	0.24	#DIV/0!	#DIV/0!	0.33	0.33	0.33
.prepend()		0.99	1.00	1.00	1.13	1.15	1.14	1.23	1.24	1.24		0.48	0.44	0.46	0.58	0.52	0.55	0.52	0.54	0.53		0.34	0.38	0.36	0.50	0.61	0.55	0.43	0.43	0.43
.prependTo()		0.97	1.05	1.01	1.23	1.15	1.19	1.17	1.09	1.13		0.60	0.54	0.57	0.59	0.56	0.57	0.49	0.52	0.51		0.42	0.46	0.44	#DIV/0!	#DIV/0!	#DIV/0!	0.43	0.54	0.49
.prop()		0.00	0.00	0.00	0.01	0.01	0.01	0.00	0.00	0.00		0.01	0.00	0.01	0.03	0.01	0.02	0.00	0.01	0.00		0.00	0.01	0.00	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.00	0.00
.removeAttr()		0.08	0.09	0.08	0.19	0.21	0.20	0.12	0.11	0.12		0.02	0.01	0.02	0.05	0.03	0.04	0.01	0.01	0.01		0.01	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.01	0.00	0.01
.removeCase()		-0.01	-0.01	-0.01	-0.02	-0.02	-0.02	-0.01	-0.01	-0.01		-0.01	-0.01	-0.01	-0.01	-0.03	-0.02	-0.01	-0.01	-0.01		-0.01	-0.01	-0.01	#DIV/0!	#DIV/0!	#DIV/0!	-0.01	-0.01	-0.01
.removeClass()		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00		0.02	0.01	0.02	0.03	0.03	0.03	0.01	0.01	0.01		0.01	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.01	0.01	0.01
.removeProp()		0.01	0.01	0.01	0.02	0.01	0.02	0.02	0.02	0.02		0.01	0.00	0.01	0.01	0.01	0.01	0.01	0.01	0.01		0.00	0.00	0.00	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.00	0.00
.replaceAll()		0.11	0.11	0.11	0.11	0.10	0.10	0.07	0.07	0.07		0.10	0.12	0.11	0.08	0.08	0.08	0.04	0.06	0.05		0.07	0.07	0.07	#DIV/0!	#DIV/0!	#DIV/0!	0.05	0.05	0.05
.replaceWith()		0.00	0.00	0.00	-0.01	-0.01	-0.01	0.00	0.00	0.00		0.00	-0.01	0.00	-0.01	-0.02	-0.02	-0.01	-0.01	-0.01		-0.01	0.00	0.00	#DIV/0!	#DIV/0!	#DIV/0!	-0.01	-0.01	-0.01
.scrollLeft()	(Get)	0.00	0.00	0.00	0.01	-0.01	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.04	0.00	0.02	0.00	0.00	0.00		0.00	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.00	0.00
	(Set)	0.01	0.01	0.01	0.01	0.00	0.01	0.01	0.01	0.01		0.01	0.01	0.01	0.03	0.00	0.01	0.00	0.00	0.00		0.01	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.01	0.00
.scrollTop()	(Get)	0.00	0.00	0.00	0.01	-0.01	0.00	0.00	0.00	0.00		0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00		0.00	0.01	0.00	#DIV/0!	#DIV/0!	#DIV/0!	0.00	0.00	0.00
	(Set)	0.01	0.01	0.01	0.01	0.00	0.00	0.01	0.01	0.01		0.01	0.01	0.01	0.02	0.00	0.01	0.00	0.01	0.00		0.01	0.01	0.01	#DIV/0!	#DIV/0!	#DIV/0!	0.01	0.00	0.00
.text()		0.13	0.14	0.13	0.17	0.18	0.17	0.10	0.09	0.10		0.03	0.02	0.02	0.06	0.05	0.05	0.02	0.02	0.02		0.02	0.02	0.02	0.09	0.12	0.11	0.02	0.01	0.02
.toggleClass()		0.09	0.09	0.09	0.14	0.19	0.16	0.11	0.11	0.11		0.11	0.08	0.10	0.13	0.15	0.14	0.10	0.11	0.10		0.07	0.08	0.07	#DIV/0!	#DIV/0!	#DIV/0!	0.10	0.10	0.10
.unwrap()		0.02	0.02	0.02	0.02	0.03	0.03	0.02	0.02	0.02		0.04	0.03	0.04	0.06	0.06	0.06	1.96	2.02	1.99		0.03	0.02	0.03	#DIV/0!	#DIV/0!	#DIV/0!	0.49	0.52	0.51
.val()		0.02	0.02	0.02	0.01	0.05	0.03	0.04	0.04	0.04		0.04	0.03	0.03	0.07	0.04	0.06	-0.06	-0.06	-0.06		0.03	0.03	0.03	#DIV/0!	#DIV/0!	#DIV/0!	0.04	0.04	0.04
.width()	(Get)	0.09	0.10	0.10	0.12	0.17	0.14	0.12	0.12	0.12		0.11	0.09	0.10	0.15	0.12	0.13	0.10	0.11	0.10		0.08	0.09	0.08	#DIV/0!	#DIV/0!	#DIV/0!	0.10	0.10	0.10
	(Set)	0.14	0.26	0.20	0.50	0.94	0.72	0.25	0.20	0.22		0.24	0.10	0.17	2.29	4.62	3.45	0.14	0.15	0.15		0.11	0.11	0.11	#DIV/0!	#DIV/0!	#DIV/0!	0.16	0.13	0.15
.wrapAll()		3.14	2.78	2.96	3.20	3.03	3.11	2.91	2.78	2.85		1.09	1.09	1.09	1.64	1.50	1.57	1.41	1.37	1.39		0.87	0.94	0.91	#DIV/0!	#DIV/0!	#DIV/0!	1.23	1.30	1.26
.wrap()		3.14	3.84	3.49	6.56	6.84	6.70	4.65	4.00	4.32		2.49	1.98	2.23	4.90	4.44	4.67	2.89	2.93	2.91		1.68	1.82	1.75	#DIV/0!	#DIV/0!	#DIV/0!	2.50	2.46	2.48
.wrapInner()		2.46	2.80	2.63	9.84	8.06	8.95	4.31	4.42	4.37		2.05	1.88	1.96	4.03	3.94	3.98	3.22	3.06	3.14		1.56	1.59	1.57	#DIV/0!	1.59	#DIV/0!	2.83	2.85	2.84

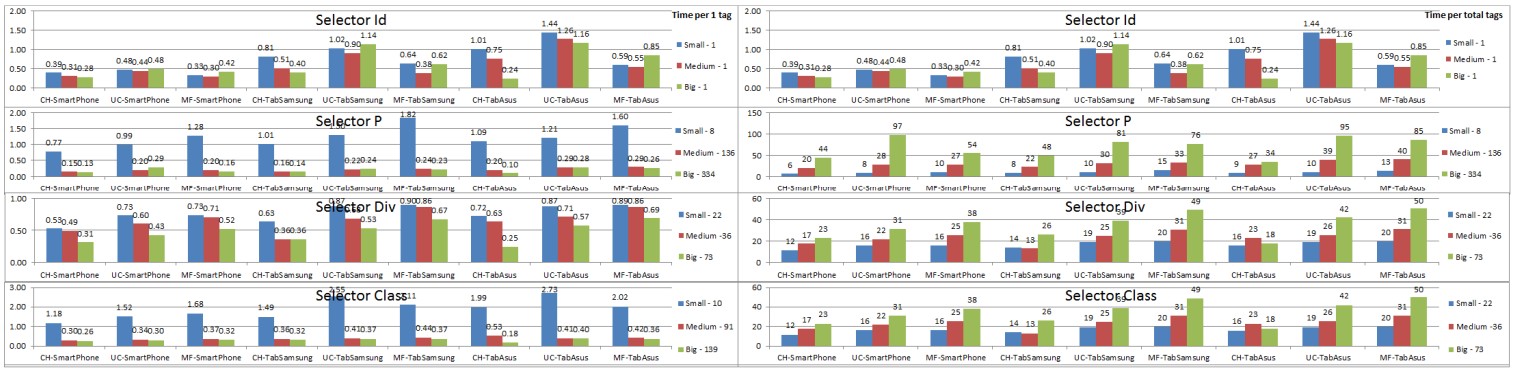
ΠΑΡΆΡΤΗΜΑ ΔΙΑΓΡΑΜΜΑΤΩΝ

Διαγράμματα εκτέλεσης μεθόδου για μία φορά ανά selector και συγκεντρωτικά:

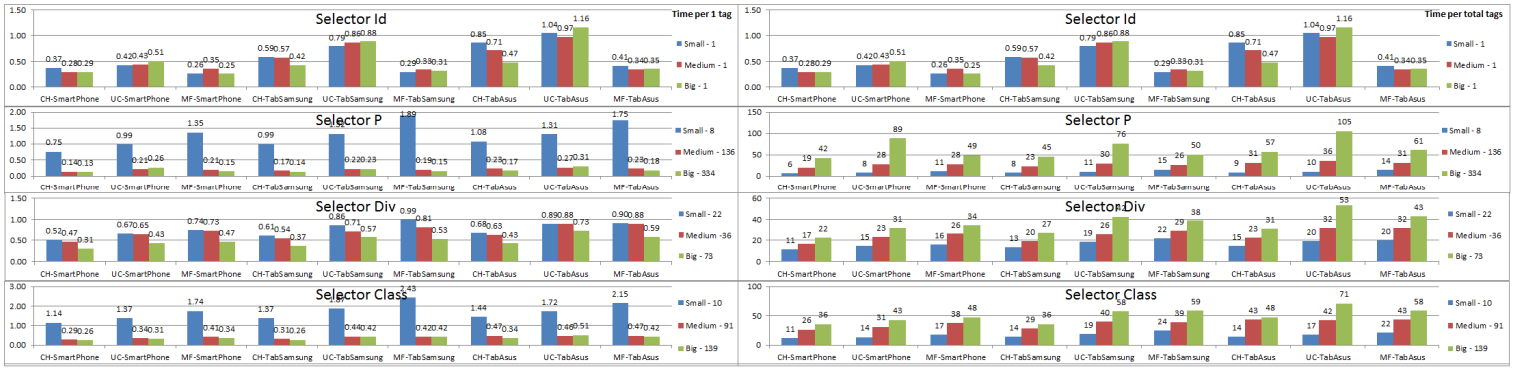
.addClass()



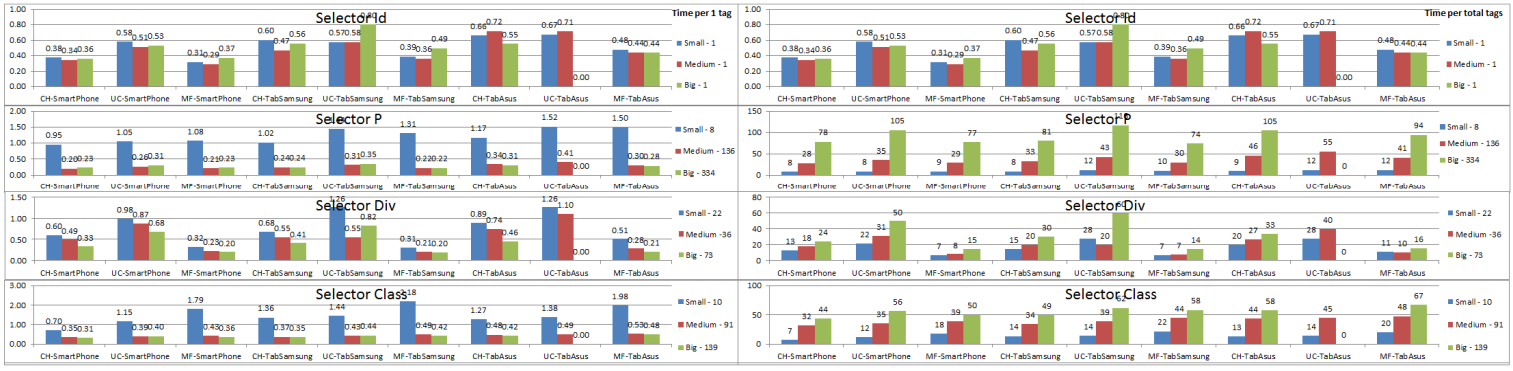
.after()



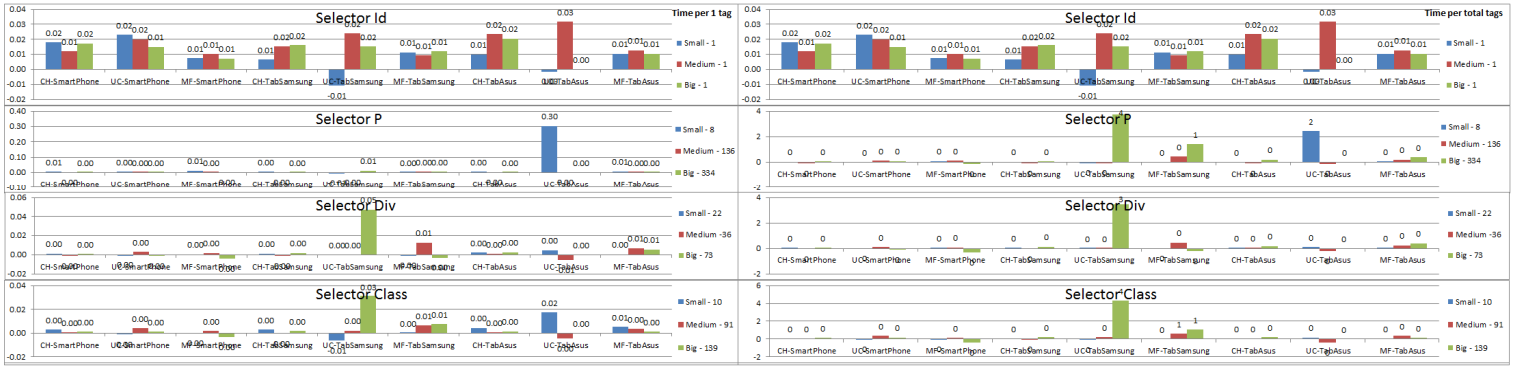
.append()



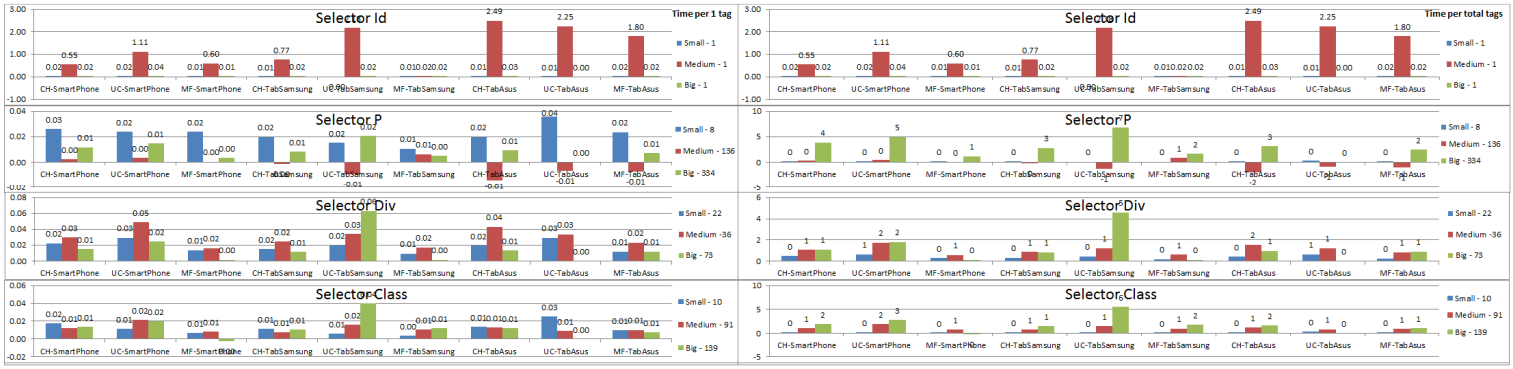
.appendTo()



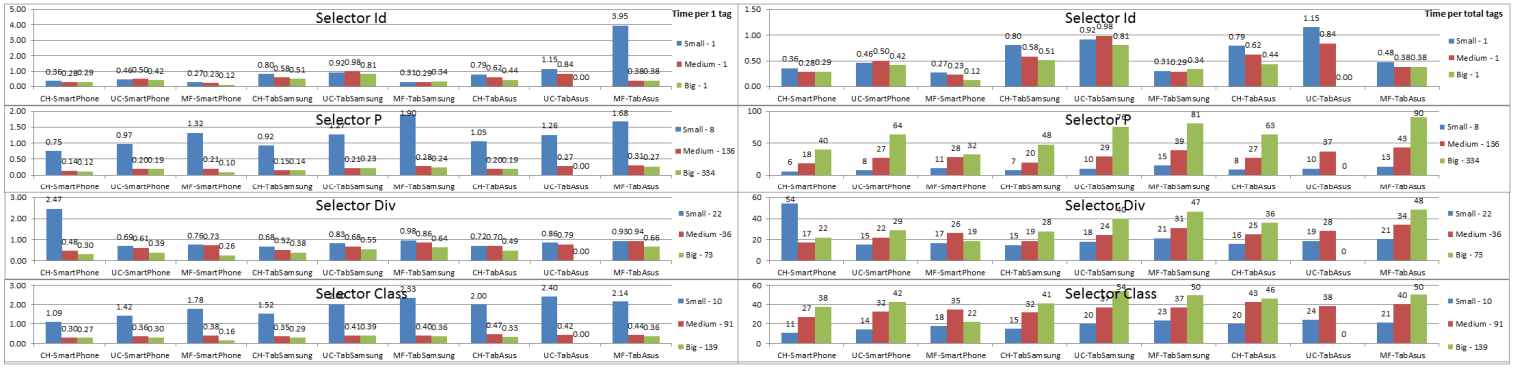
.attr() – Get



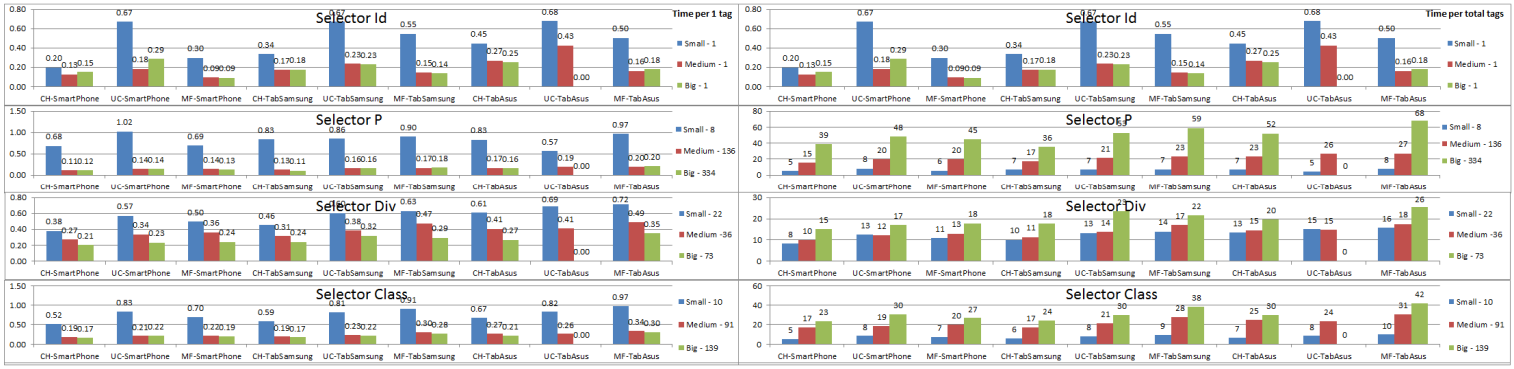
.attr() – Set



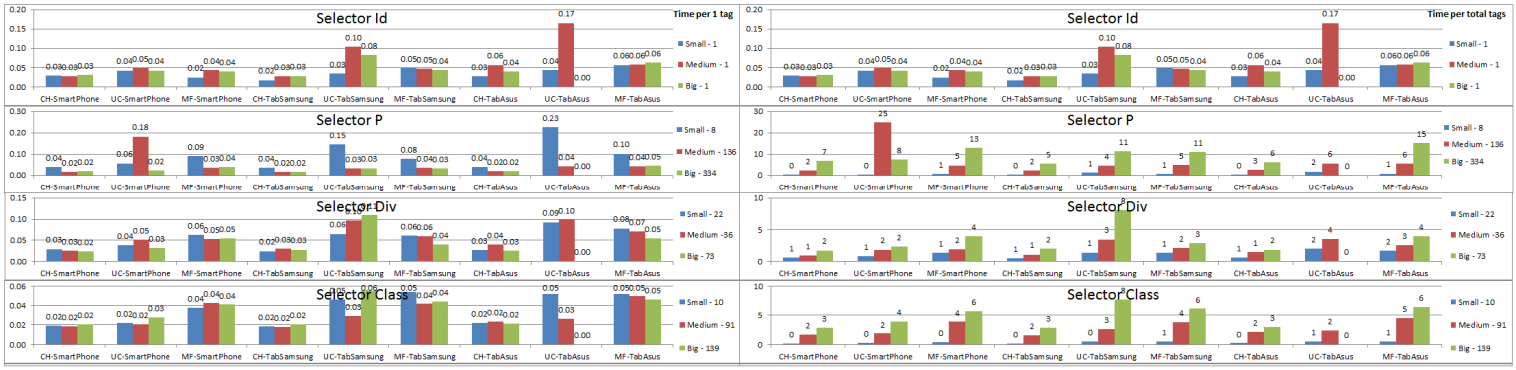
.before()



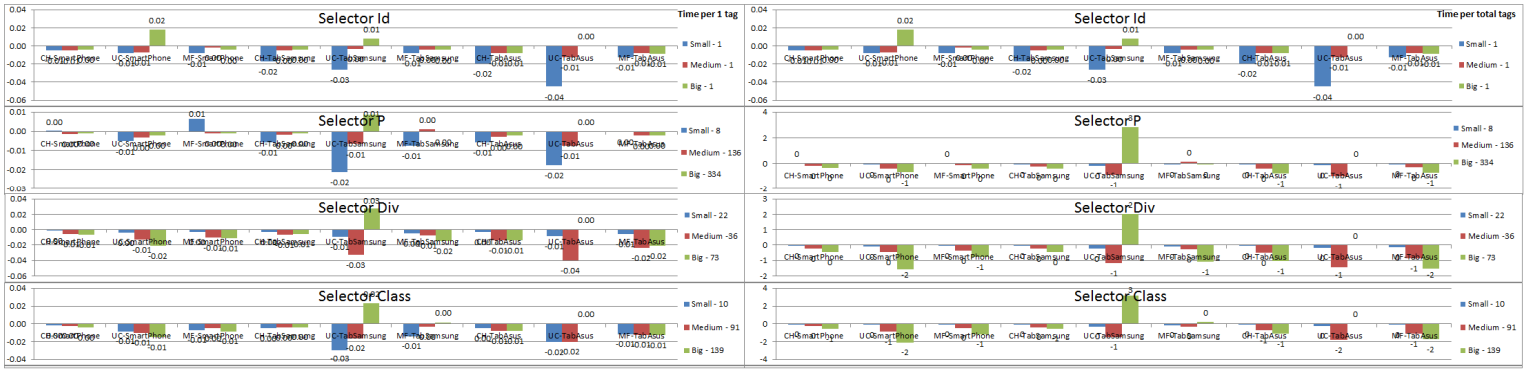
.clone()



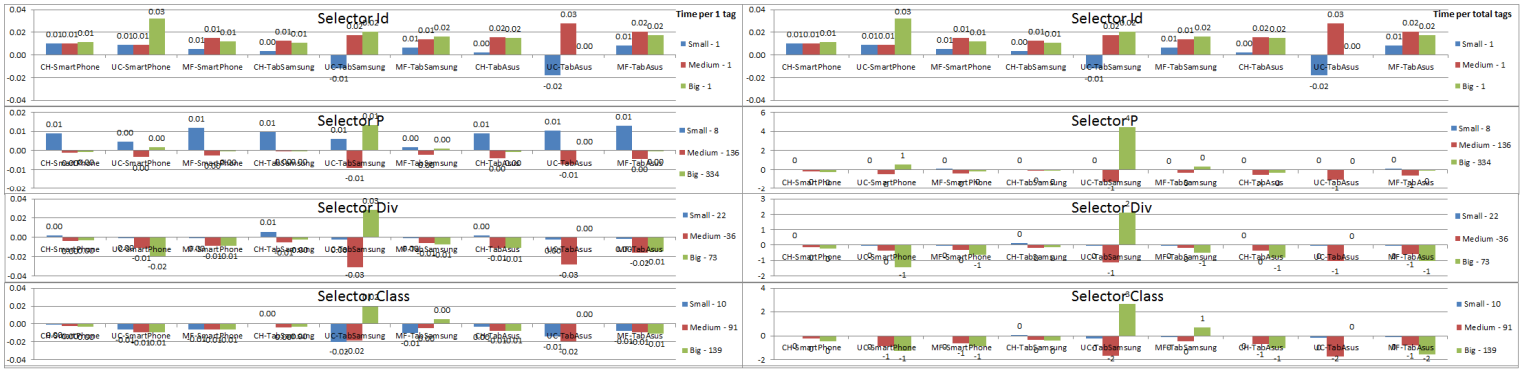
.css()



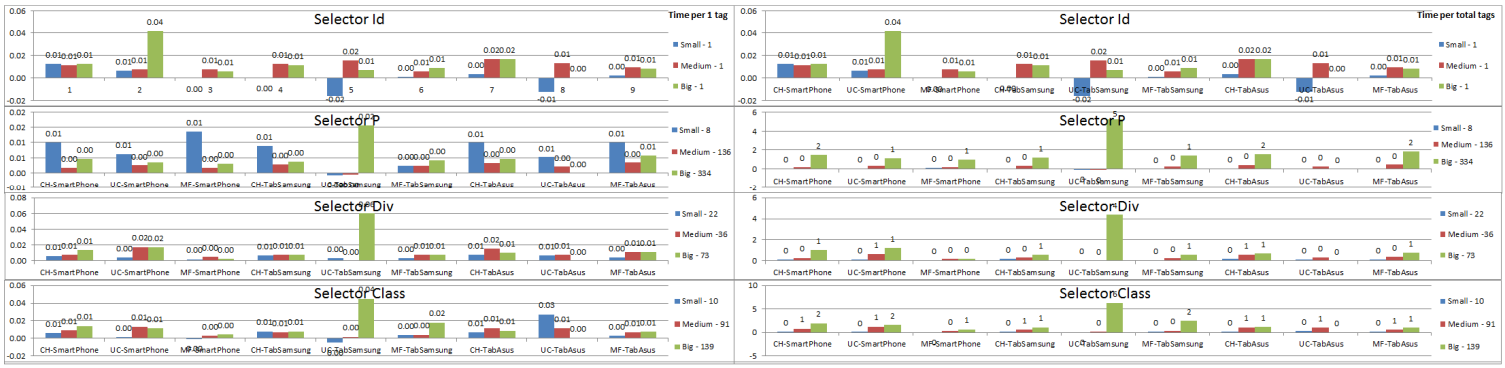
.detach()



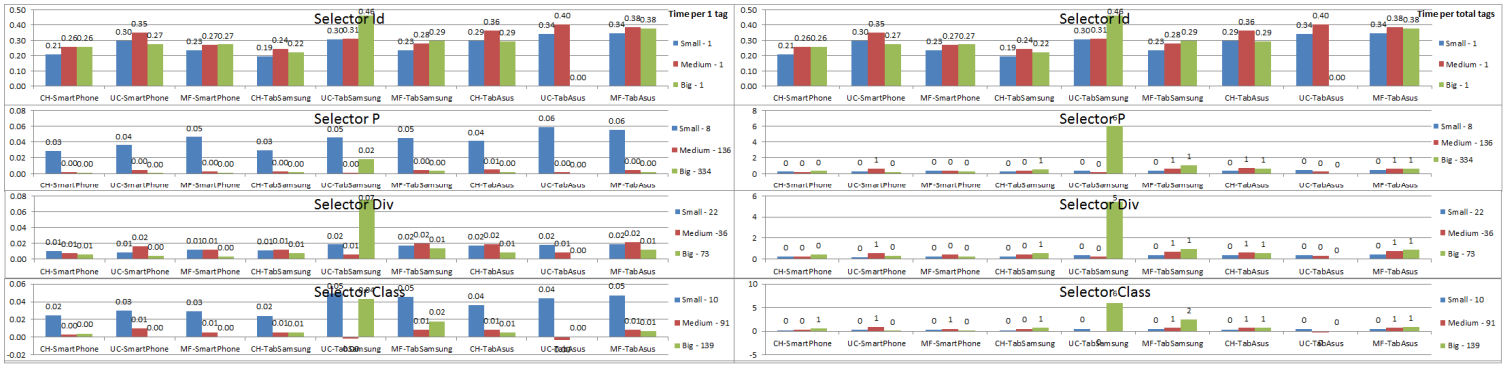
.empty()



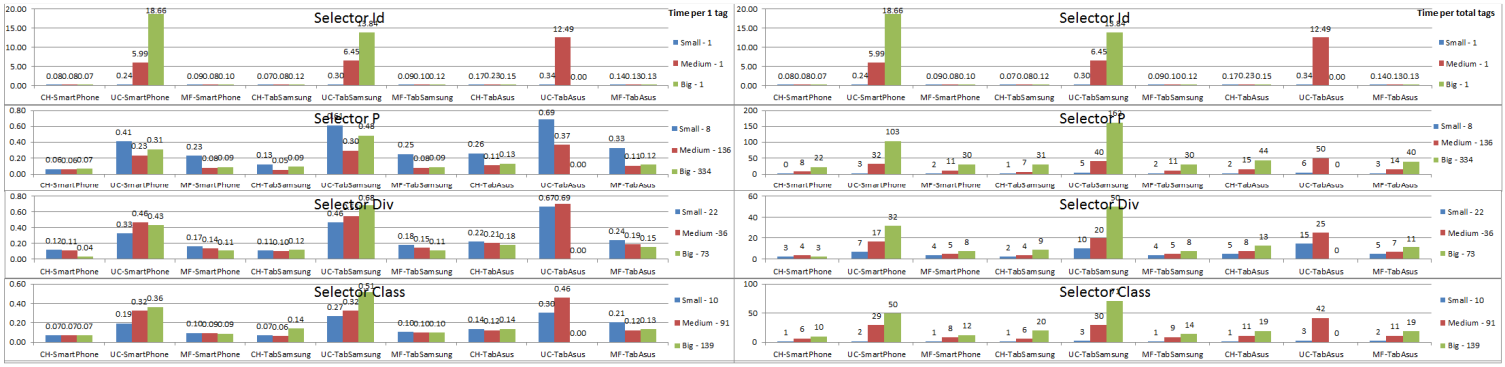
.hasClass()



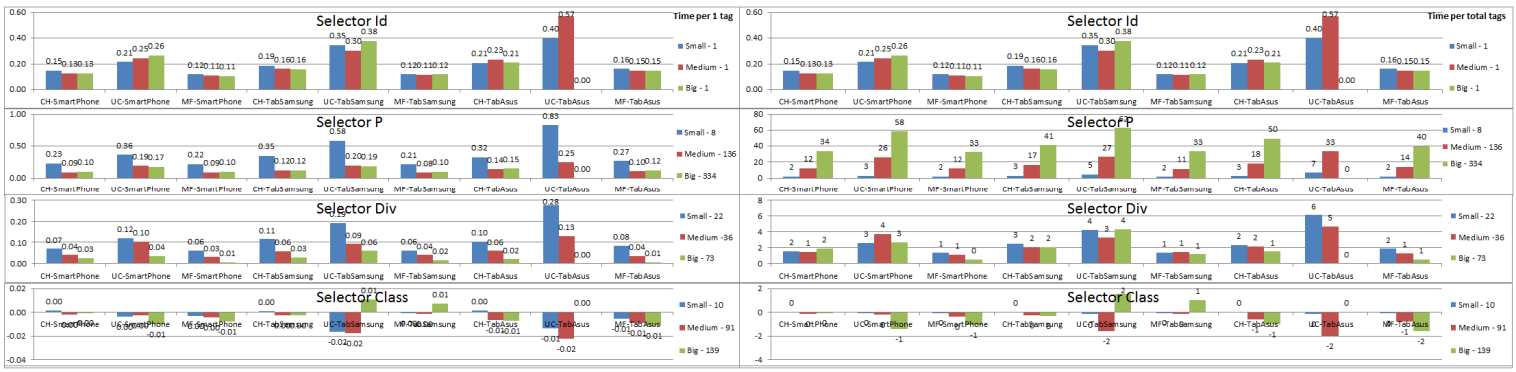
.height() - Get



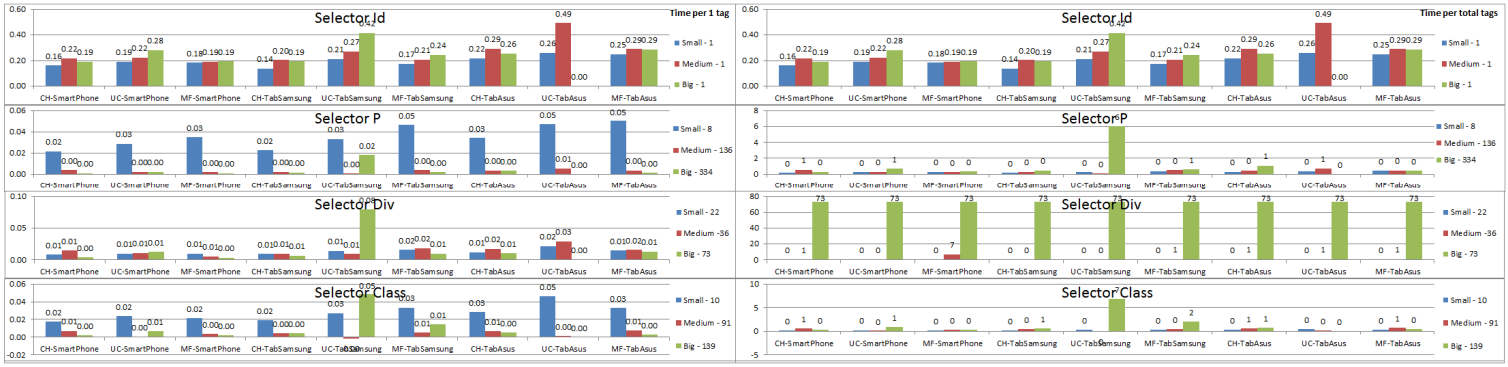
.height() – Set



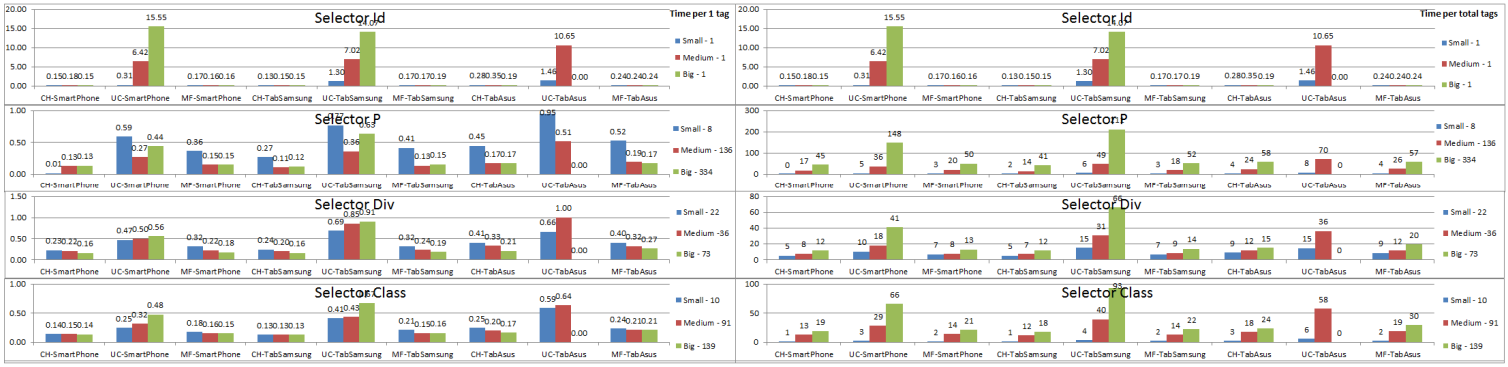
.html()



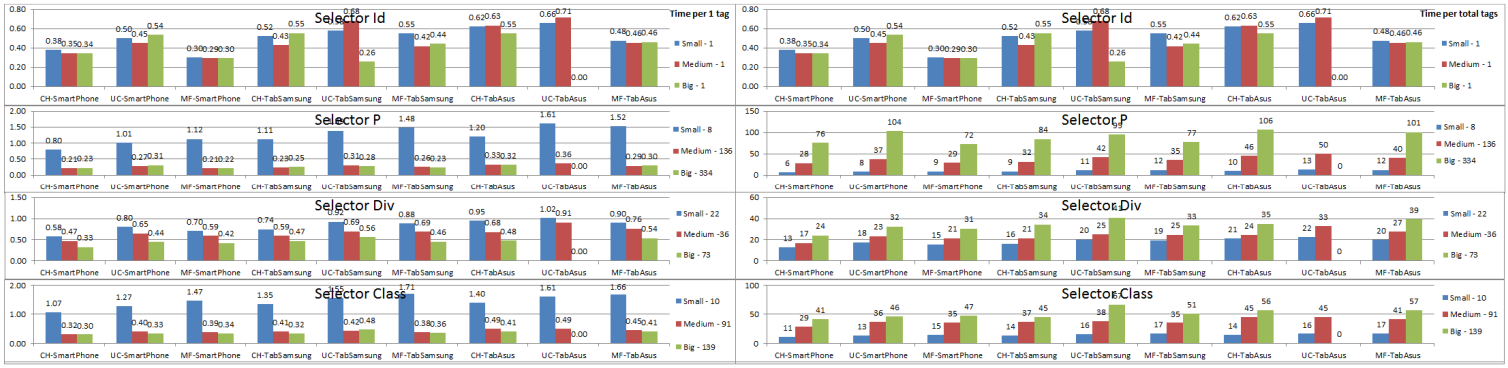
.innerHeight() - Get



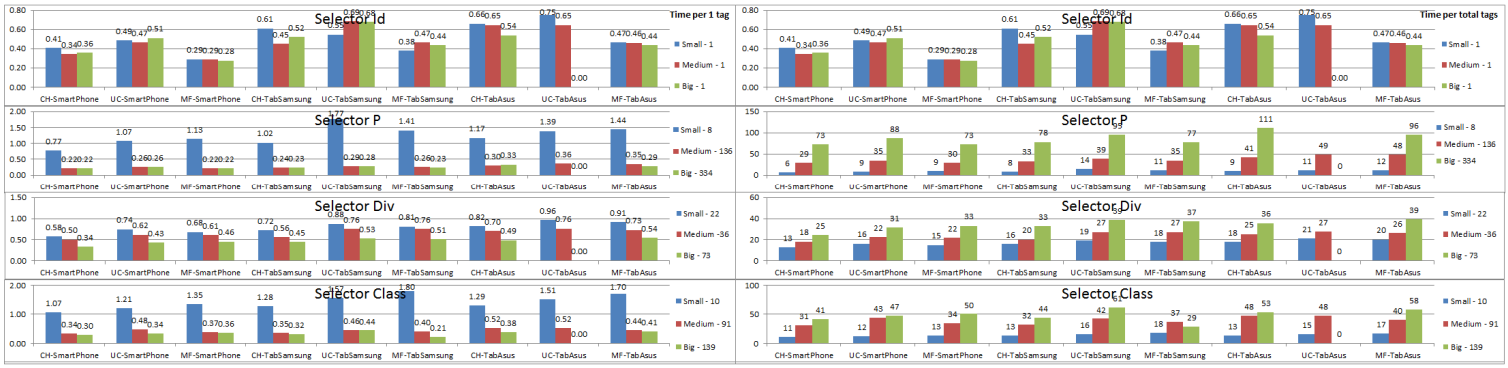
.innerHeight() - Set



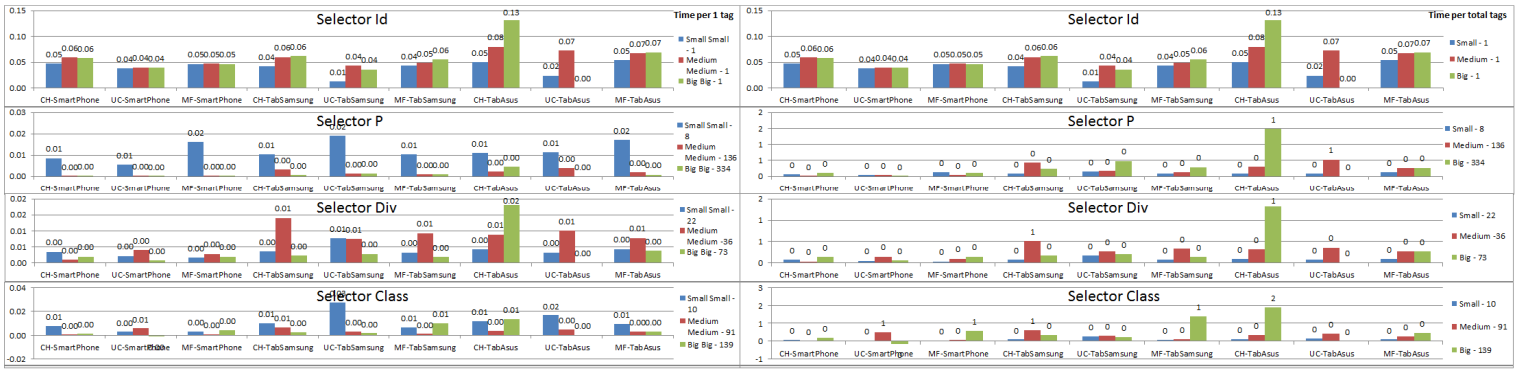
.insertAfter()



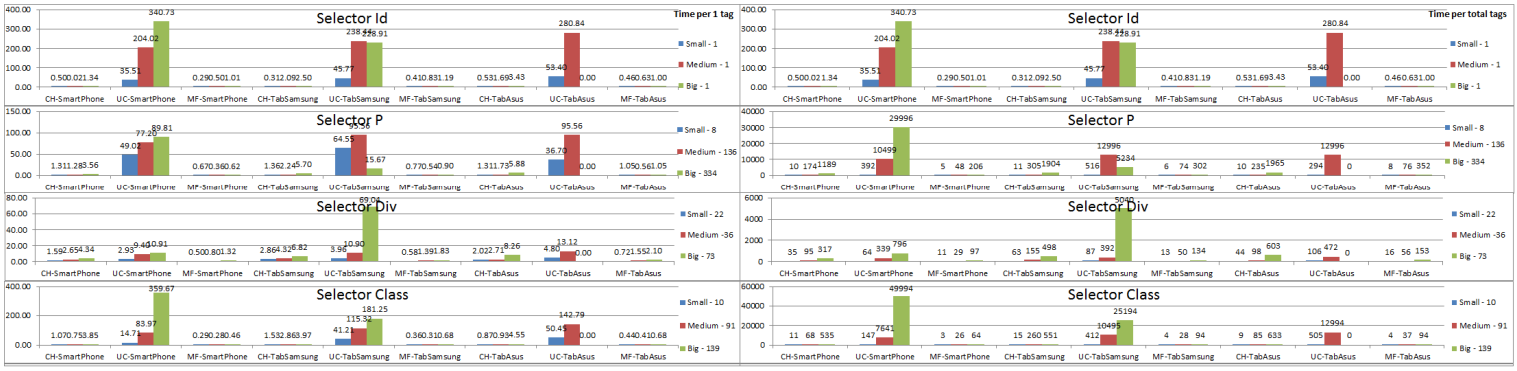
.insertBefore()



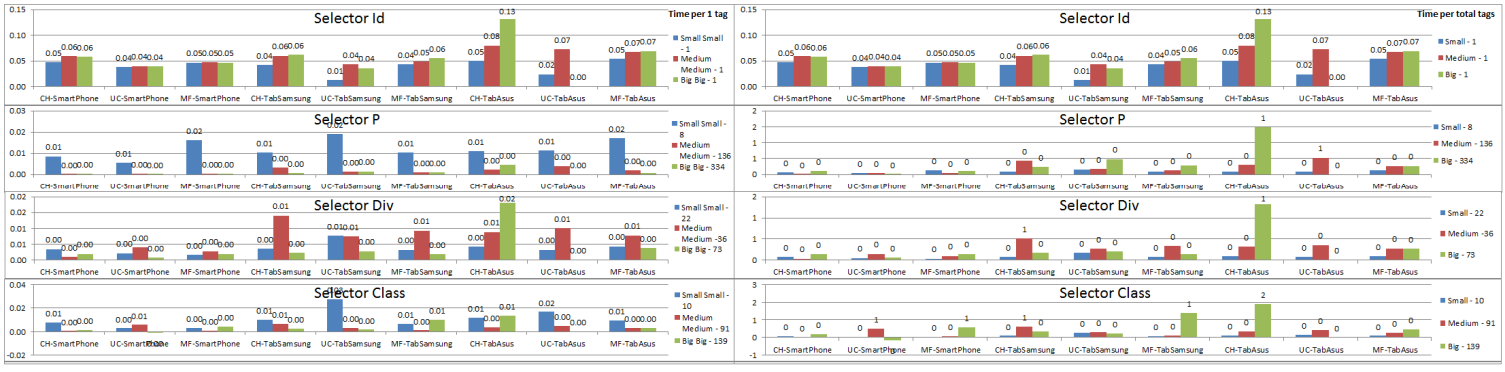
.offset() -Get



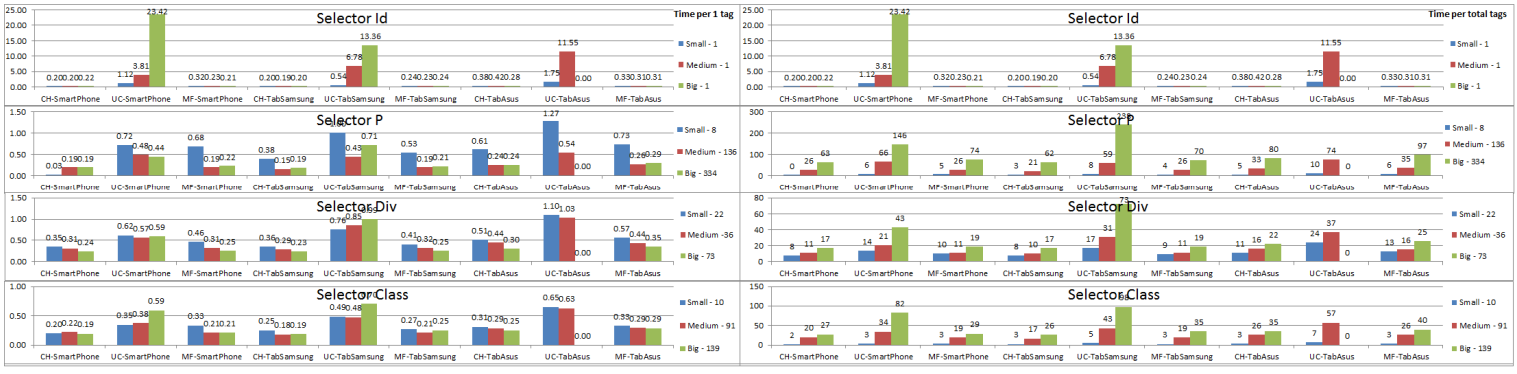
.offset() -Set



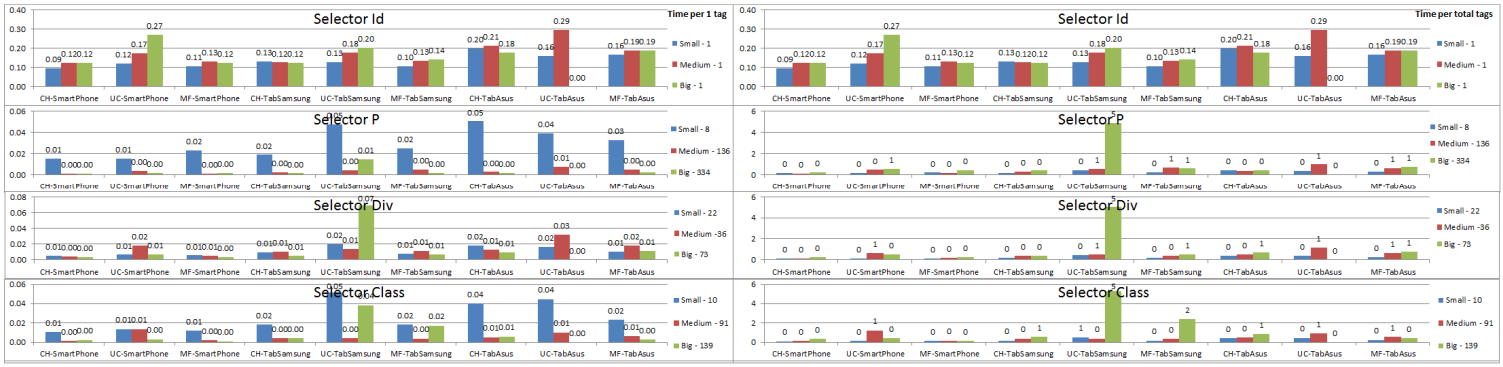
.outerHeight() - Get



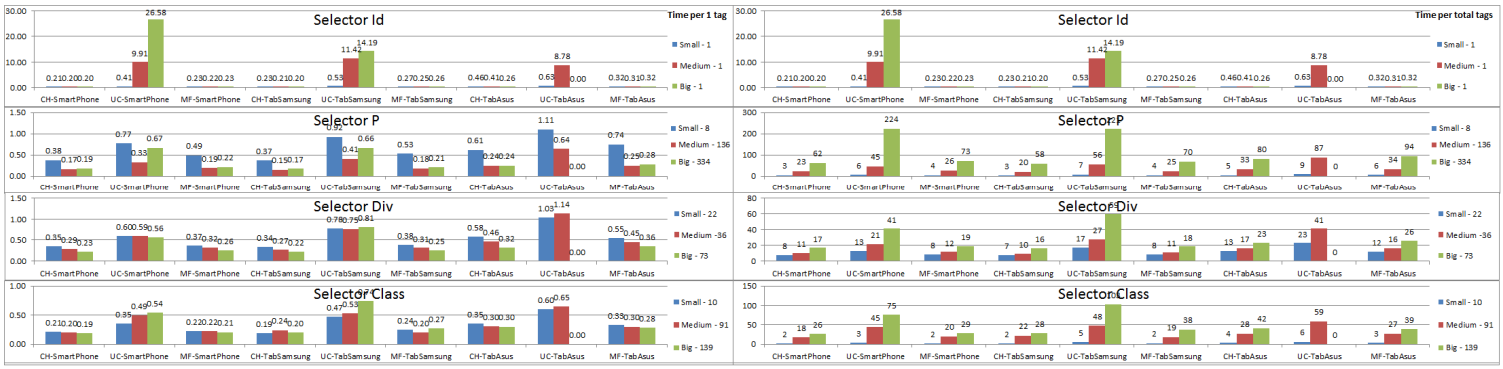
.outerHeight() - Set



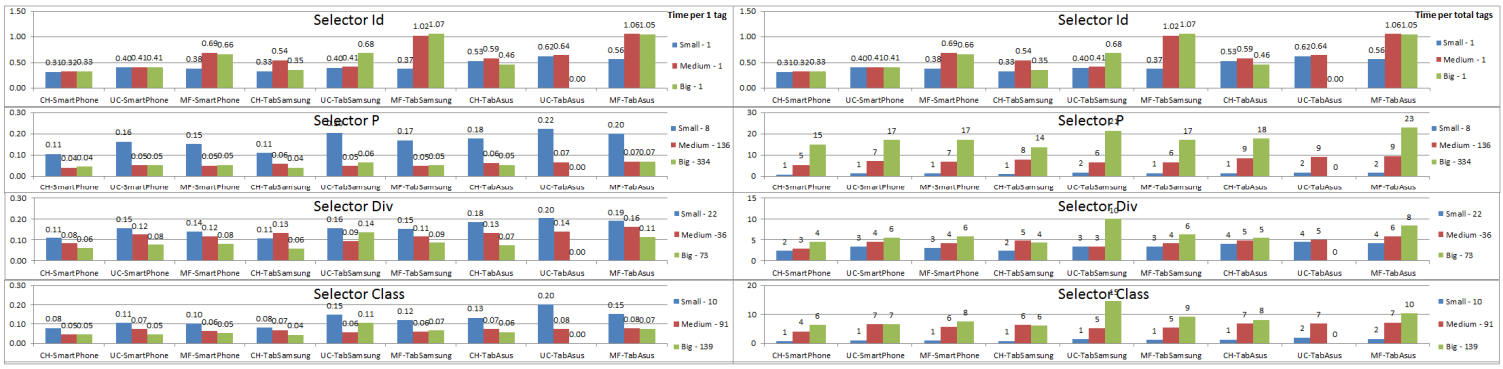
.outerWidth() – Get



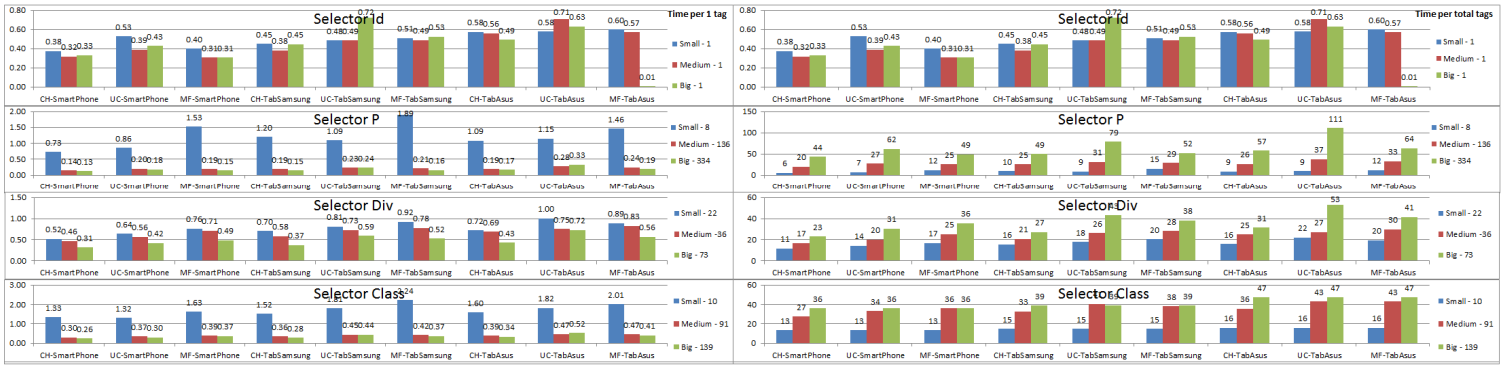
outerWidth() – Set



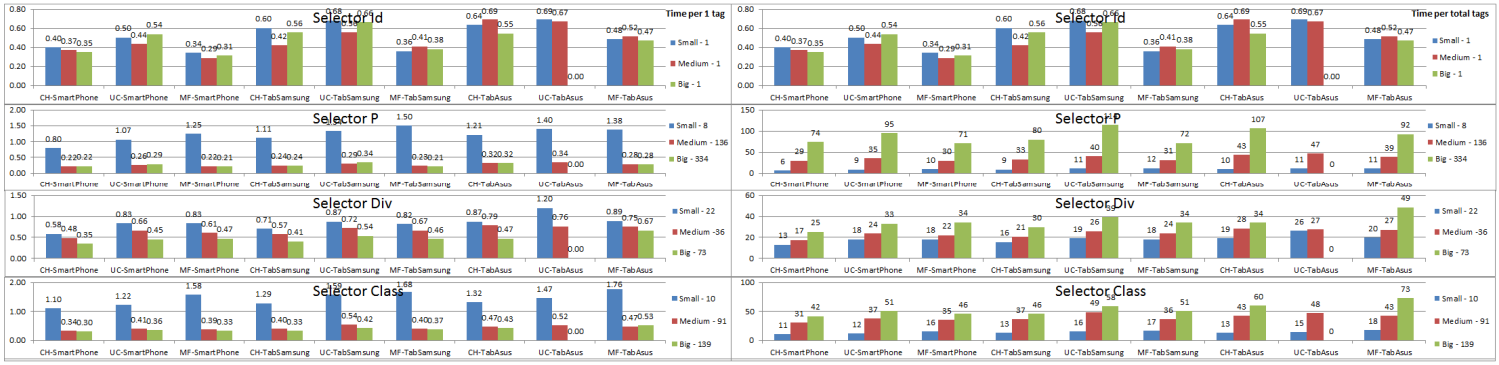
.position()



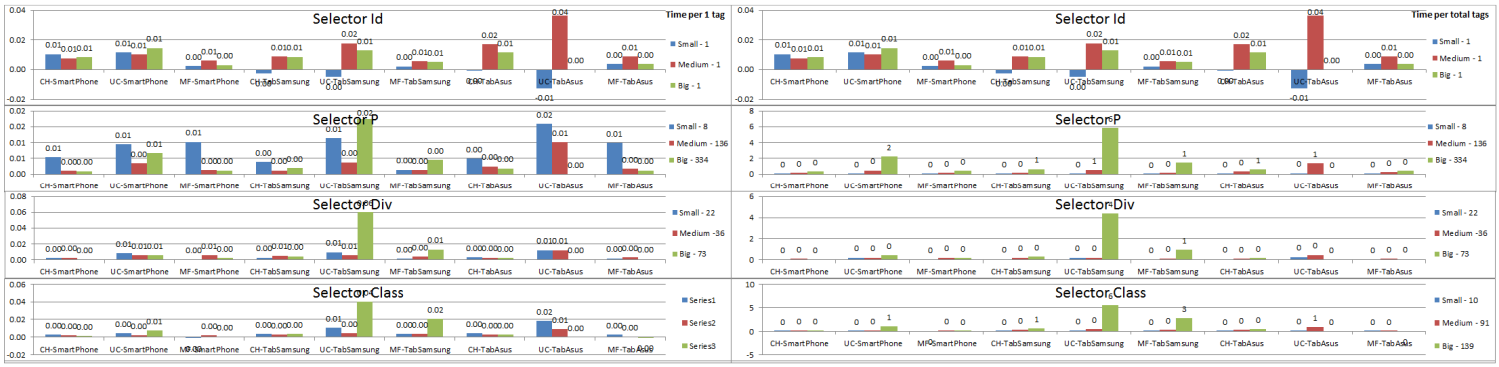
.prepend()



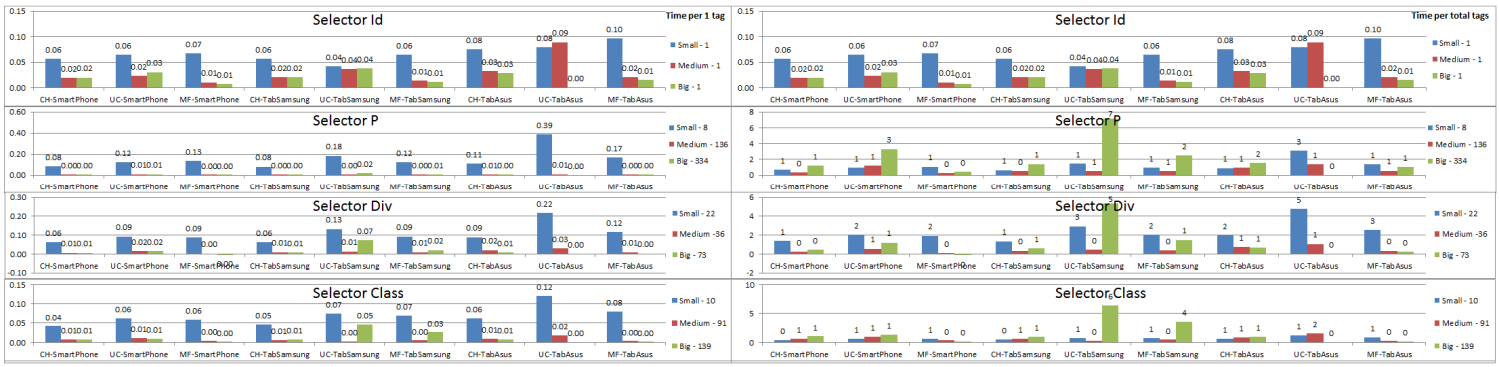
.prependTo()



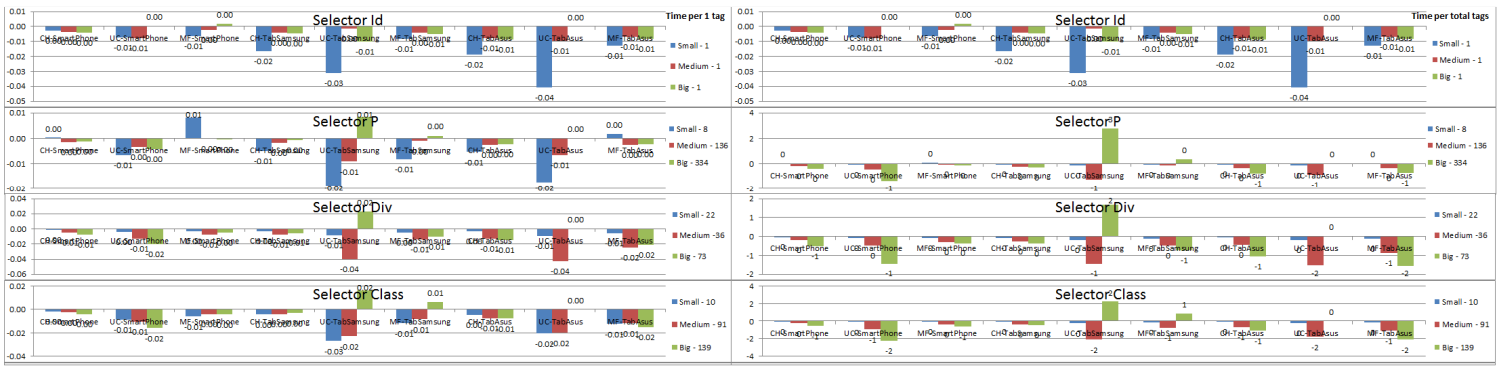
.prop()



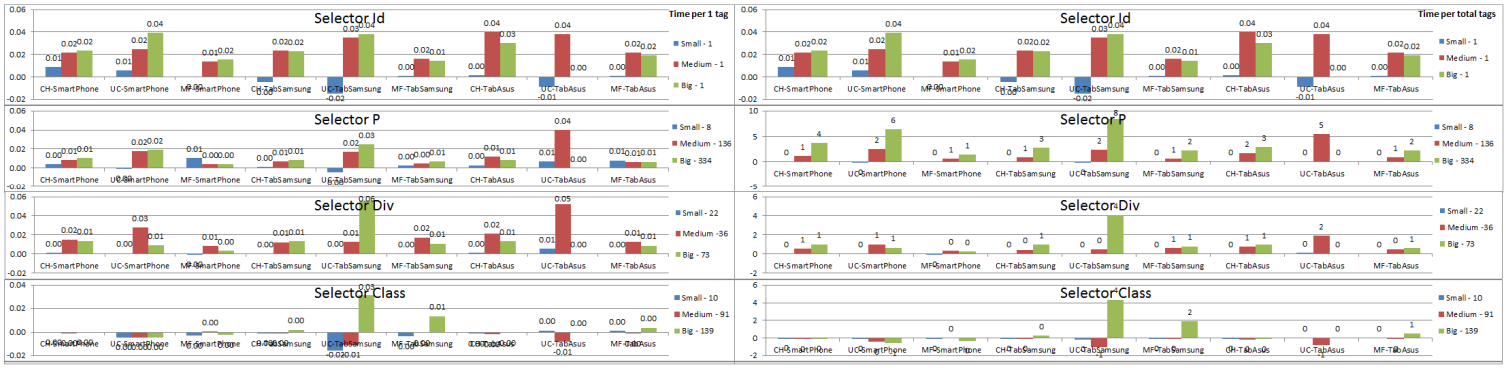
.removeAttr()



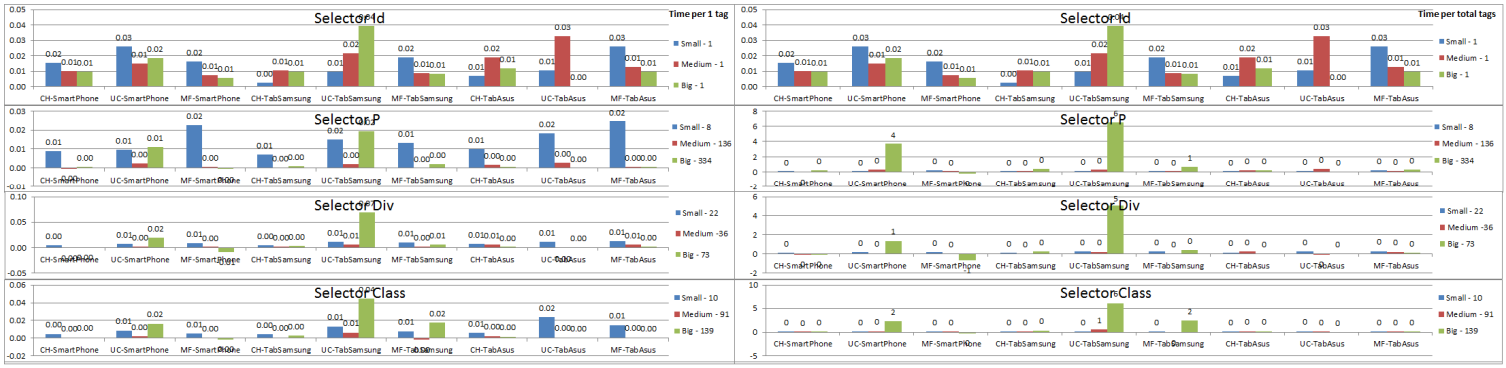
.remove()



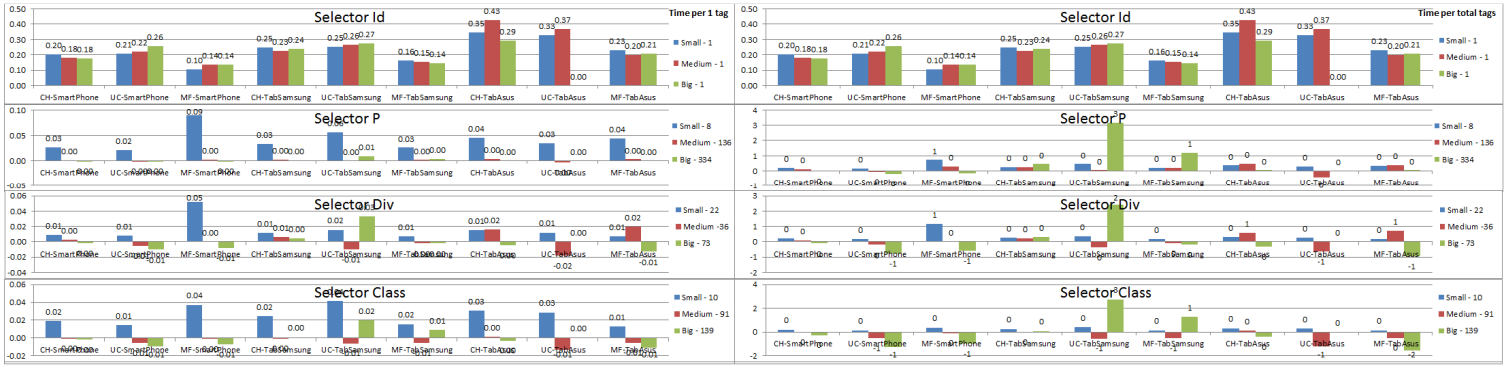
.removeClass()



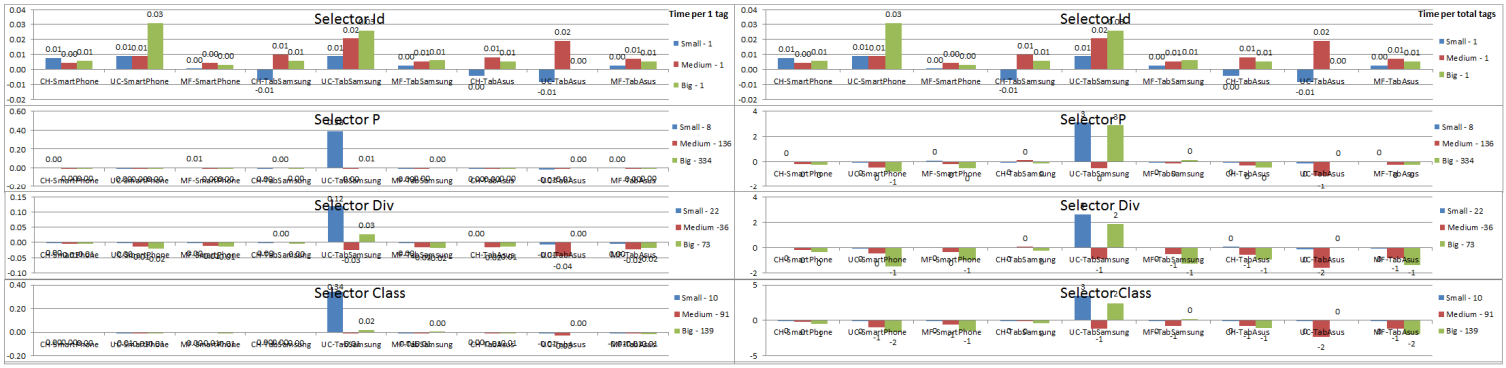
.removeProp()



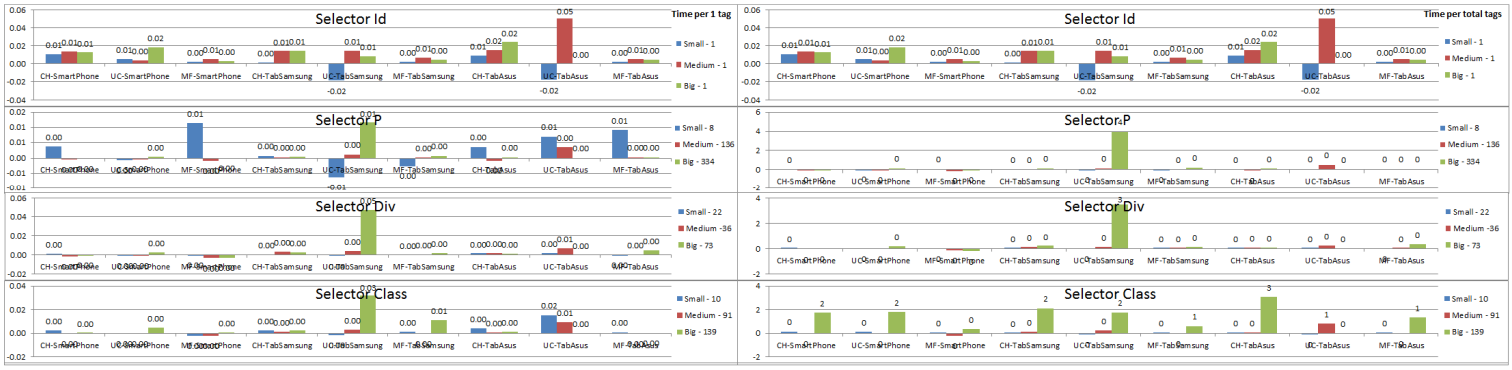
.replaceAll()



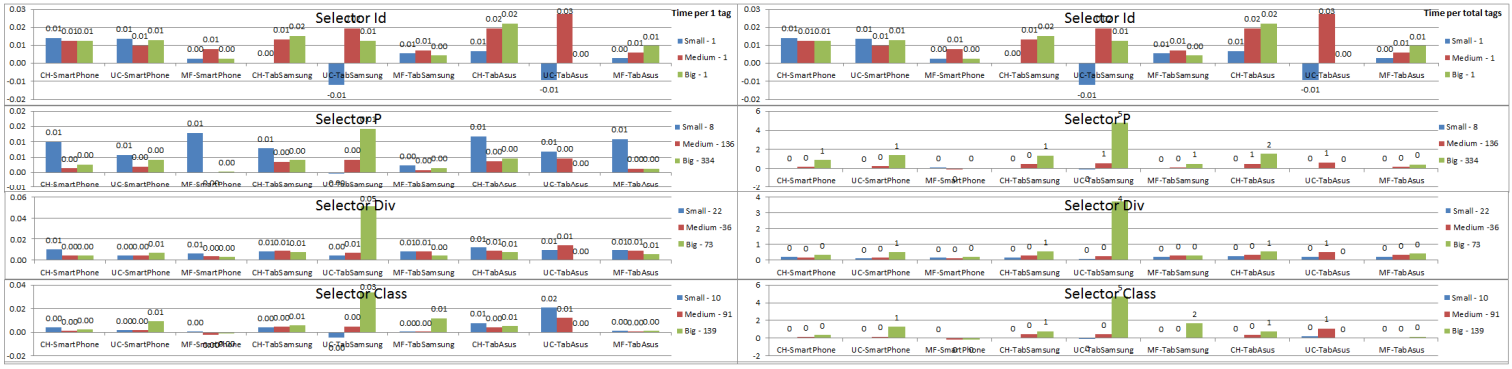
.replaceWith()



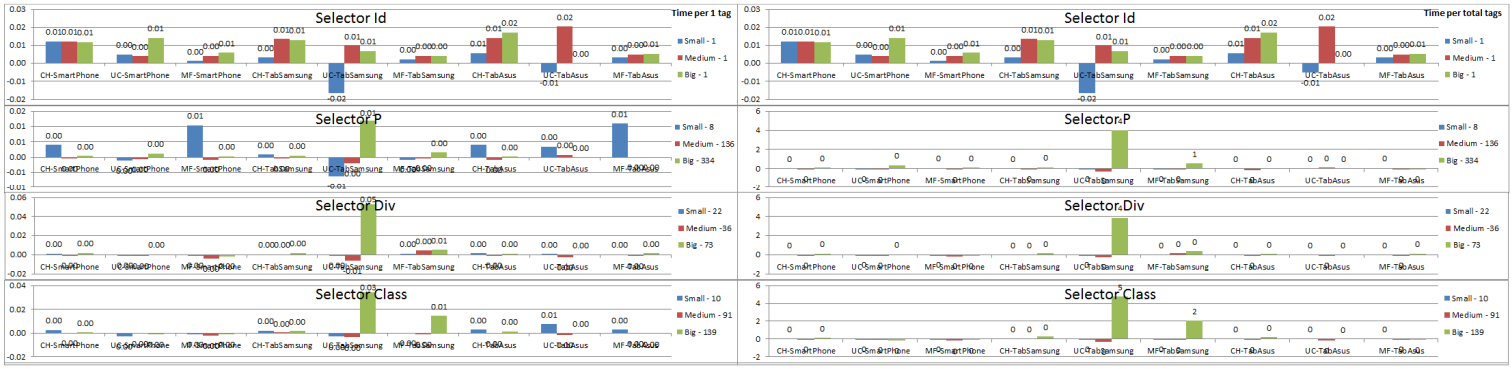
.scrollLeft() – Get



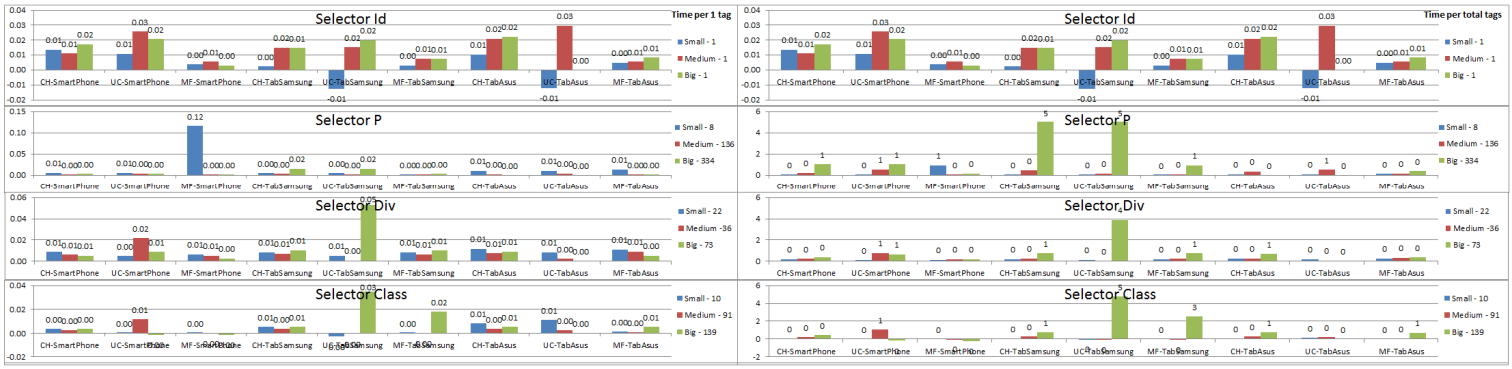
.scrollLeft() – Set



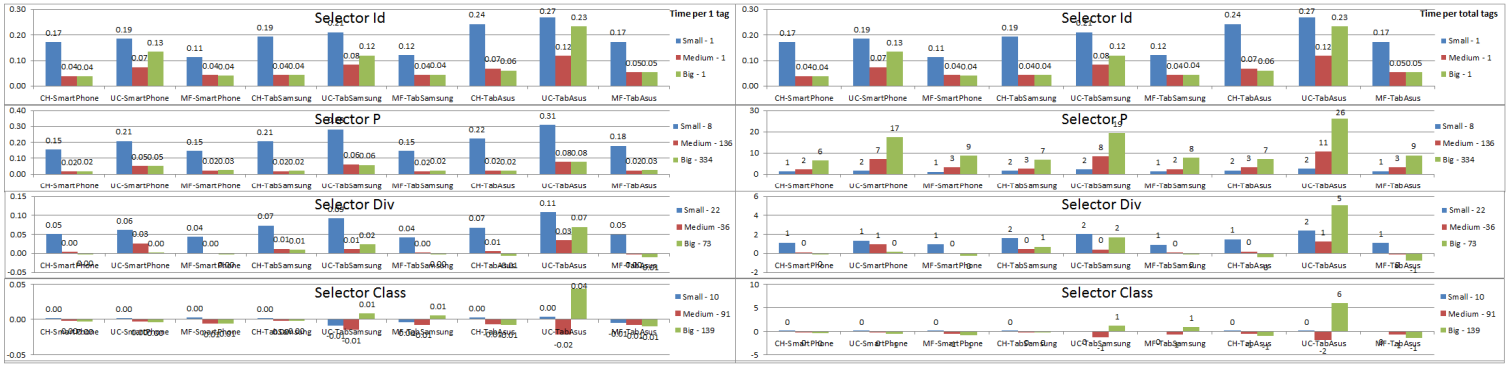
.scrollTop() – Get



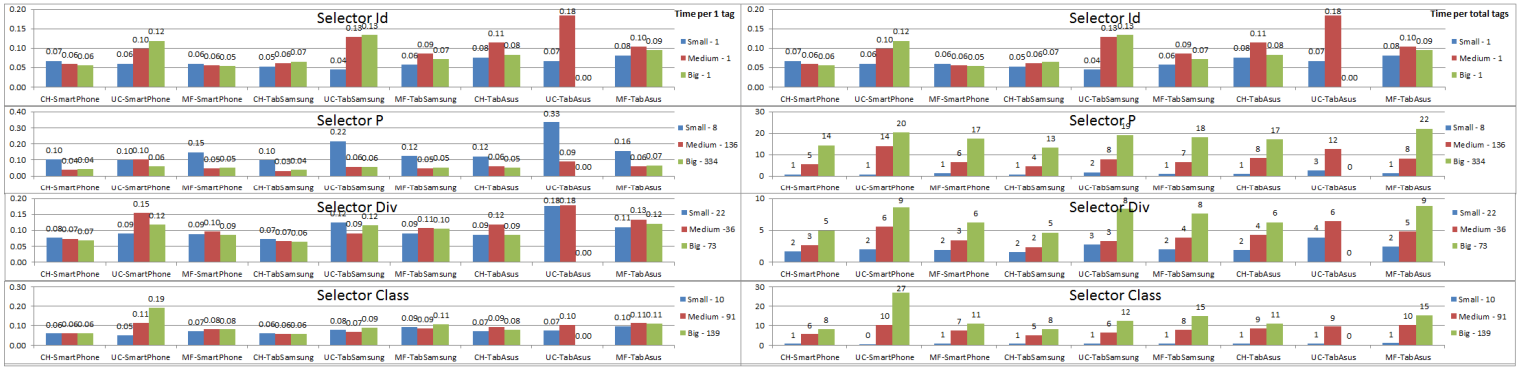
.scrollTop() – Set



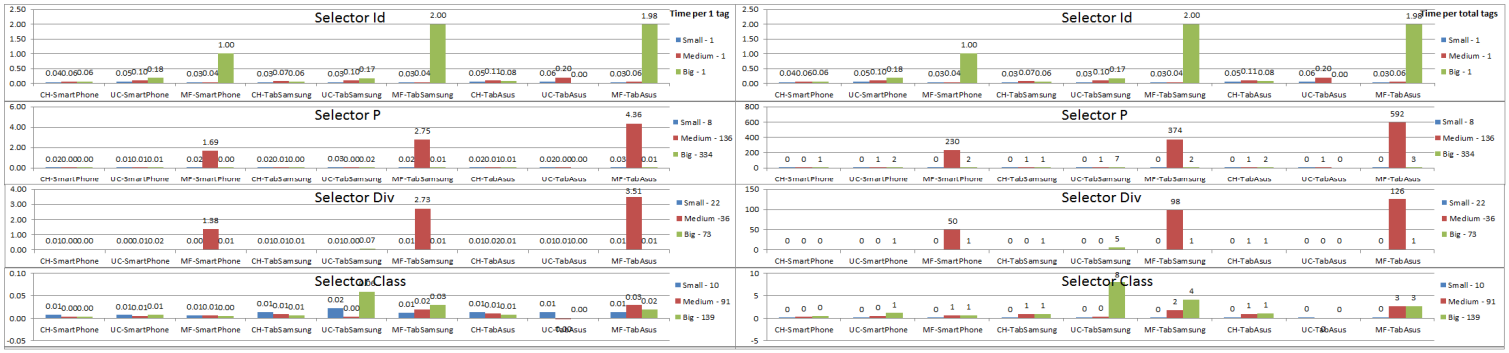
.text()



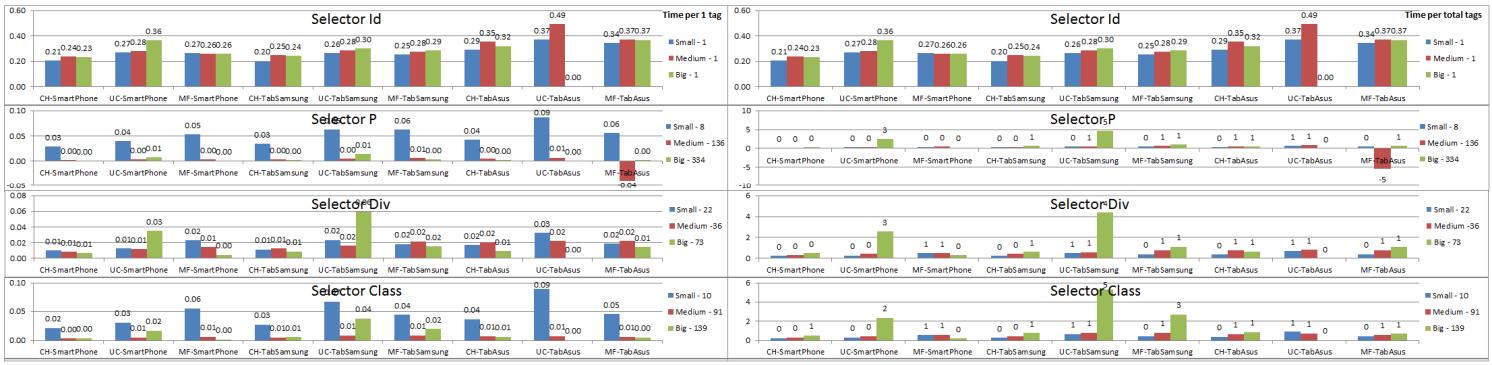
.toggleClass()



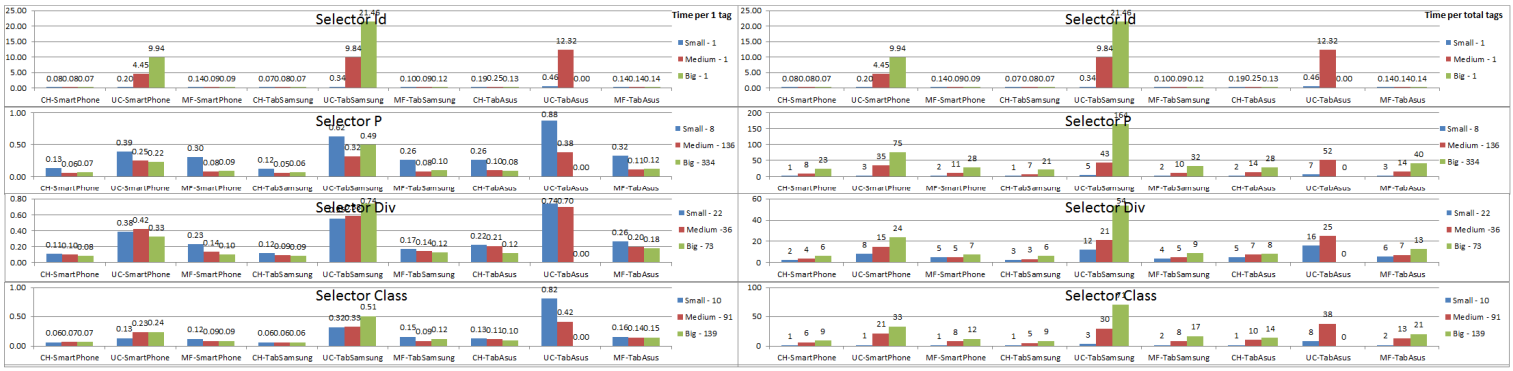
.unwrap()



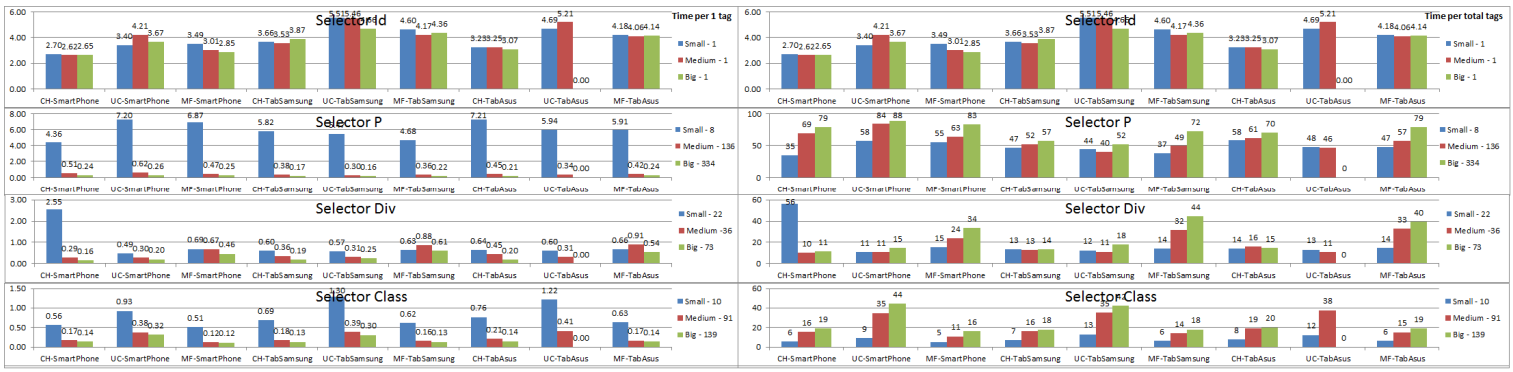
.width() - Get



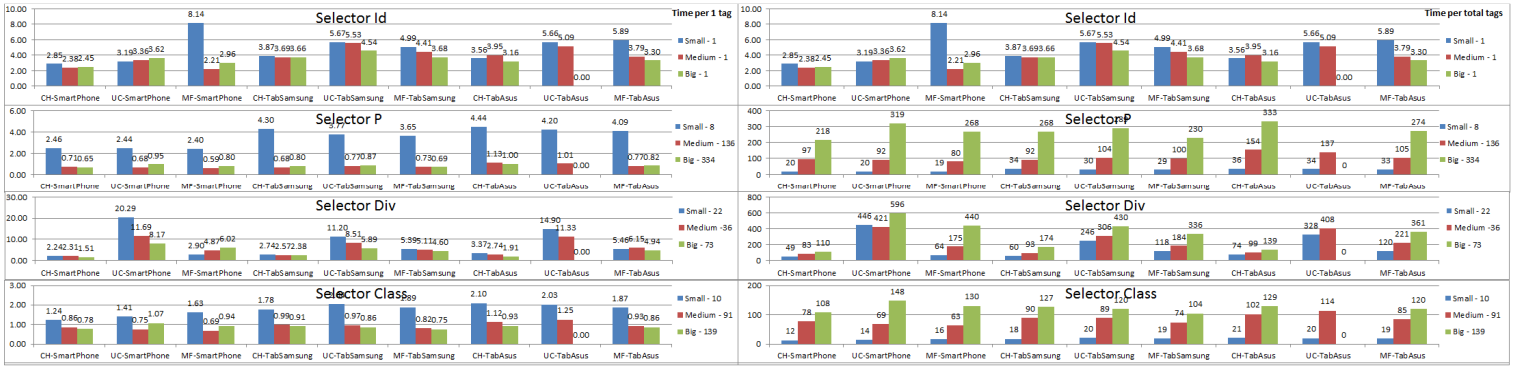
.width() - Set



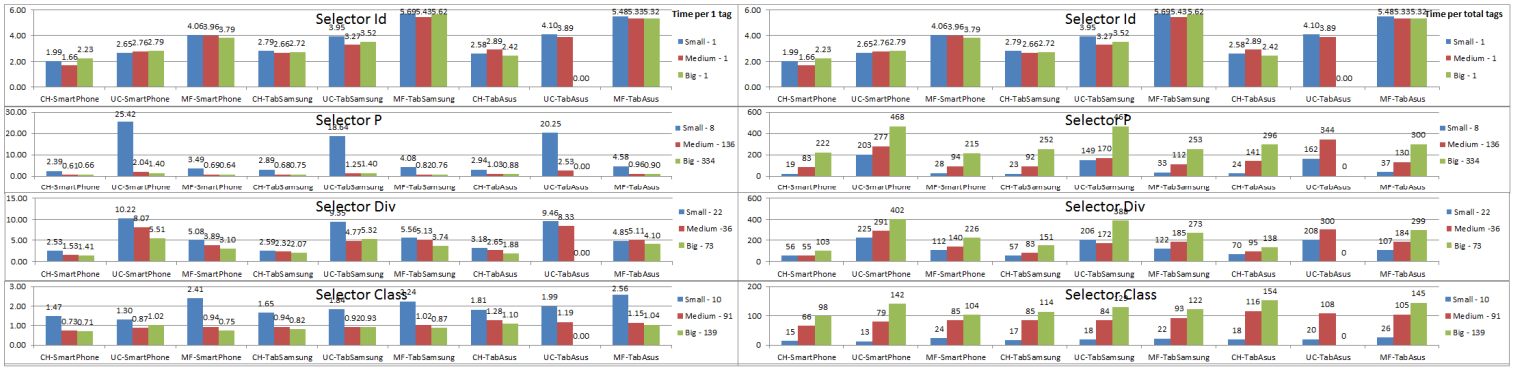
.wrapAll()



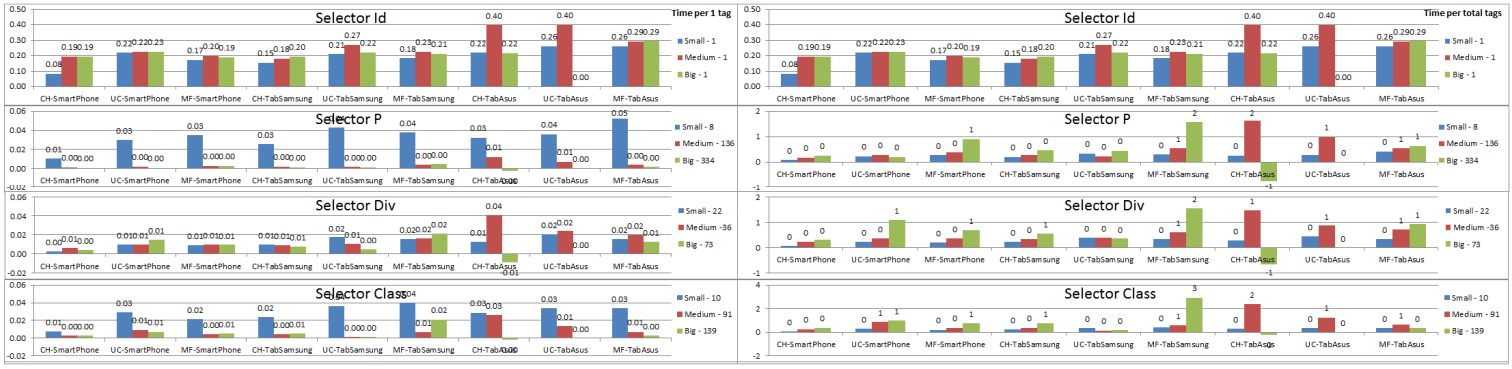
.wrap()



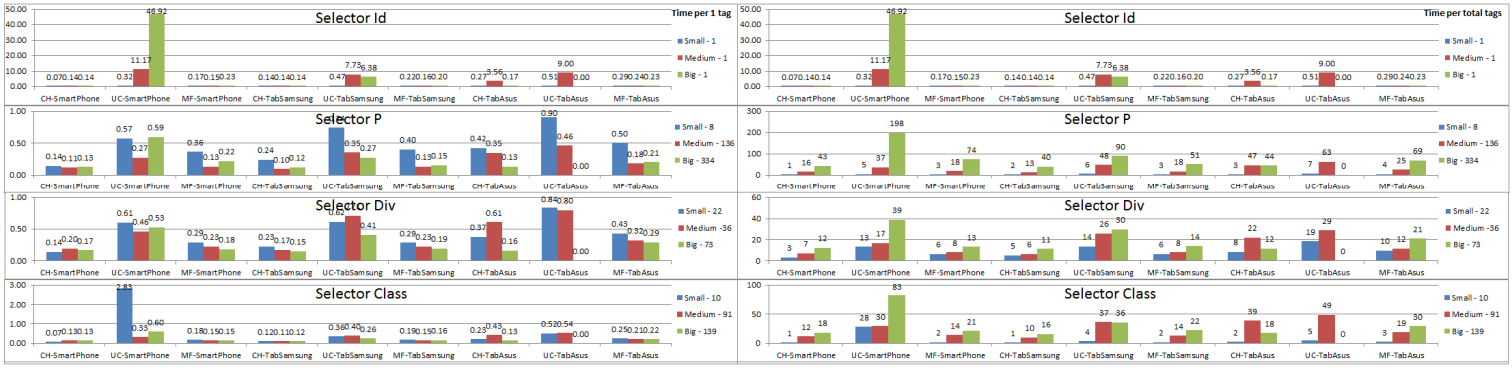
.wrapInner()



.innerWidth() - Get



.innerWidth() - Set



.val()