Κανόνες που ίσως να προσέδιδαν αξία ίσως να ήταν άχρηστοι

Κανονες για Πιθανά Ζητήματα Λογικής

- for-direction - Ίσως χρειάζεται, ίσως απλά να μπει χωρίς να δώσει παραπάνω αξία. Τα for αλλά και γενικότερα τα loops (θα πρέπει κατά την ταπεινή μου αποψη) να μπαινουν σπάνια ή και να αποφευγονται, καθώς βαζουν στον πειρασμό τον προγραμματιστή να αλλαξει (mutate) δεδομένα κάτι που αντιβαίνει στη ιδέα του συναρτησιακού προγραμματισμού που θελει immutability του αρχικού object και οι αλλαγές να γινονται με αντίγραφα. Εφόσον όμως η JavaScript τα έχει δεν θα ηταν σωφρον να τα αγνοούμε σε περίπτωση που κάποιος θέλει να τα χρησιμοποιήσει. Ο παραπάνω κανόνας βοηθά στην αποφυγή bugs αν και δεν ξέρω κατά πόσο ένας επαγγελματίας προγραμματιστής θα "κόλλαγε" για πολύ σε έναν ατέρμονα for βρόχο
- getter-return - Δεν χρησιμοποιείται πολύ το get keyword. Κατά τα αλλα θα βοηθούσε σίγουρα στην αποφυγη bugs αν η get είχε πολυπλοκότητα
- no-setter-return - Δεν πολυ χρησιμοποιείται το set keyword. Κατά τα αλλα θα βοηθούσε σίγουρα στην αποφυγη bugs αν το set είχε πολυπλοκότητα
- no-async-promise-executor - Ισως είναι χρήσιμο να προστεθεί αν και σπάνια εχω δει async-await συναρτήσεις εντός Promise. Ωστόσο υπάρχουν οπότε ο κανόνας ίσως να βοηθούσε. Σε ένα Node.js project θα ήταν γάντι.
- no-await-in-for-loop - Ισχύει ότι και για το προηγούμενο
- no-control-regex - Δεν ξερω πόση αξία θα έδινε για να μπει σαν κανόνας που θα τηρούν όλοι. Ας το σκεφτούμε.
- no-debugger - Πολύ σπάνια χρήση αλλά ίσως να είχε αξία έτσι ώστε να μην "ξεφύγει" παραγωγή.
- no-ex-assign - Δεν ξερω πόση αξία θα είχε να προστεθεί
- no-func-assign - Ισως να ήταν καλό να προστεθεί αν αποτρεπει την επαναδηλωση της συναρτησης εντός της.
- no-inner-declarations - Ο κανόνας αφορά δηλωση συναρτησεων / μεταβλητων εντός συναρτησης. Δεν θα έκανε καλό αν οι μεταβλητές υπάγονται στον κανόνα καθώς πιθανότατα να απλοποιούν διεργασίες και λειτουργίες και να δίνουν ονοματα και περιγραφες σε περιπλοκους μηχανισμούς.. Η δήλωση συναρτήσεων όμως θα μπορούσε να γινει εκτός του εκαστοτε Component σε ένα φακελο utils επομένως ένας τέτοιος κανόνας κανόνας ίσως να εξυπηρετούσε. Θέλει περισσότερη συζήτηση.
- no-new-native-nonconstructor - Αφορά κυρίως την κλήση BigInt και Symbol οπότε ο κανόνας δεν είναι χρήσιμος για το εργο. Σπάνια περίπτωση αλλά θα ήταν πολύ χρήσιμη σε περίπτωση λάθους ή debugging. Το να μην βάζεις το new keyword σε native constructor είναι χρήσιμο ΑΝ χρησιμοποιείς αυτούς τους τύπους.
- no-self-assign - Ισως να ήταν χρήσιμο. Ισως δεν χρειάζεται

Προτάσεις για Consistency Γραφής μέσω Κώδικα

- capitalized-comments - Kanei enforce κατι που δεν είναι πρωτεύον ωστόσο θα βοηθούσε στο consistency γραφης κώδικα. Θεωρω πως το να χτυπάει ένα warning θα ήταν καλό αλλά από την άλλη πλευρά δεν ξέρω πόσο ενοχλητικό / ταίριαζε να ήταν κανόνας. Πχ εγω βαζω πάντα το πρώτο κεφάλαιο οταν γραφω περιγραφές και jsDocs αλλά όχι όταν γράφω δίπλα σε ένα line.
- complexity - Μαλλον θα επρεπε να μπει απλά θα έπρεπε να σκεφτουμε και ένα τρόπο να γραφουμε αν οι συναρτήσεις ή οι συνθήκες είναι πολλές. Επίσης χρειάζεται μια σκέψη και για το threshold.
- dot-notation - Ίσως να ήταν χρήσιμο απλά δεν ξέρω πως θα φαινόταν σε καποιον που πιθανότατα δεν ξερει καν για τον κανόνα αν κάτι που του φαίνεται valid συντακτικό ο λιντερ του το έδειχνε για λανθασμένο.
- func-name-matching - Θα ήταν καλό να γινεται μια απλή συσταση για να εξασφαλιζεται εύκολα ο εντοπισμός κατα το review παρά η εφαρμογή του κανόνα. Ωστόσο αν τεσταριστεί και δώσει καλά αποτελέσματα τότε ισως να αξιζε να προστεθεί για να διευκολυνθεί το readability και να μπορείς με μια απλή αναζήτηση να βρεις όπου χρησιμοποιείται μια συναρτηση.
- func-style - Το value που προσφερει δεν δίνει σημαντικό πλεονέκτημα για να προστεθεί. Ισως να ήταν καλό για λόγους consistency ομως.
- id-denylist - Δίνει την δυνατότητα να αποκλεισουμε μια λίστα από ονόματα μεταβλητων. Κάποια γενικά ή προβληματικά ονόματα ίσως να ήταν χρήσιμο να μπουν σε μια τετοια λίστα. Δεν ξέρω κατά πόσο θα πρόσθετε αξία ωστόσο
- id-match - Αν και φαινεται εξαιρετική η ιδέα του να καθοριστούν κανόνες ονοματοδοσίας ισως να μην είναι καλή ιδέα αν οι προγραμματιστες απλά προσπαθούν να βρουν ένα όνομα για να ξεπερασει τον λιντερ αντι να σκεφτονται το πρόβλημα που εχουν να λύσουν.
- max-statements - Ισως να βοηθουσε ο περιορισμός προτασεων σε μια συναρτηση αλλά ίσως και να μπέρδευε / δυσκολεύει την λογική του κώδικα. Θα ενθαρρύνει ωστόσο την χρήση συναρτήσεων κατι που θα βοηθησει το readability του κώδικα οπότε μάλλον θα βοηθησει.
- multiline-comment-style - Πιθανοτατα να βοηθούσε στο γραψιμο σχολίων σε πολλές γραμμές
- no-bitwise - Οι bitwise operators πράγματι χρησιμοποιούνται πολύ σπάνια αλλά διαφέρουν από τα && και ||. Δεν ξερω αν ο κανόνας αυτός έχει αξία στο εργο. Ίσως ναι ίσως όχι.
- no-delete-var - Σπάνια να χρησιμοποιηθεί αλλά καλό θα ήταν να υπάρχει. Εφόσον το delete μεταβλητης δημιουργεί πρόβλημα ίσως να αξίζει να μπει στα 'NAI'
- prefer-object-has-own - Ακολουθει το πιο μοντερνα πρότυπα της γλώσσας οποτε θα ήταν καλό να προστεθεί. Ωστόσο αν δεν χρησιμοποιείται ισως να μην χρειαζεται. Θελει σκέψη.

- prefer-destructuring - Θα είχε ενδιαφέρον να δοκιμάζαμε αυτόν τον κανόνα. Σιγουρα θα εκανε τον κωδικα σε όλο το unify consistent. Ωστόσο ισως να το παρακάναμε λίγο με αποτελεσμα περισσότερα lint-ignore μηνυματα
- operator-assignment - Θετει ενα συγκεκριμένο πρότυπο που θα ακολουθείται όταν έχουμε assign operators. Είναι καθαρά εμφανισιακό το ζήτημα οπότε ίσως να θέλαμε να το επιβάλλουμε για readability αλλά και ίσως να ήταν εξυπνότερο να αφήναμε τους προγραμματιστές να επιλέξουν οι ίδιοι τον τρόπο που θελει ο καθένας χωρίς περιορισμούς
- one-var - Είναι εμφανισιακό θεμα. Σιγουρα θα είναι χρήσιμο στο readability και στο code consistency. Ισως είναι χρήσιμο αν πχ θελω μέγιστη ταχύτητα στο τεστ μια συνάρτησης και μου κοκκινίζει λογω λάθος τρόπου.
- no-void - Καλό θα ήταν απλά ίσως να μην πρέπει να γίνει αυστηρα σαν κανόνας γιατί είναι ένα λιντερ θεμα που δεν περνάει εύκολα από το μυαλό καποιου.
- no-useless-computed-key - Ίσως να μην χρειάζεται. Από την άλλη δεν θα ήταν ασχημο να δινουμε ενα μηνυμα αν κάτι είναι αχρειαστα δυσανάγνωστο.
- no-underscore-dangle - Ισως να υπάρχει κατα καιρούς η αναγκη του convention αυτού. Ωστόσο θα ήταν καλύτερο να χρησιμοποιηθεί το πιο πρόσφατο convention.
- no-undef-init - Έχει τύχει να αρχικοποιησω την undefined τιμή έτσι οταν κανονικοποιούσα δεδομένα να τα φιλτραρω για να μην τα στέλνω καθόλου σε κλήσεις στο Backend (φιλτράρονται στα POST αν έχουν Undefined). Οπότε είναι ένα πολύ χρήσιμο trick. Από την άλλη πλευρά, οντως δεν θα πρεπει να αρχικοποιείς undefined σε τιμές. θεωρω πως δεν χρειάζεται ο συγκεκριμένος κανόνας αλλά θα ήταν καλό να γίνει μια συζήτηση.
- no-throw-literal - Ισως χρειάζεται. Πολύ σπάνια η χρήση του και ισως να μην ήταν χρήσιμο να τεθεί ο κανόνας αυτός.
- no-restricted-syntax - Αν υπάρχει καποιο feature που θεωρείται ιδιαίτερα buggy πχ το switch case θα μπορούσαμε με αυτόν τον κανονα να χτυπαει error κάθε φορά που κάποιος θα επιχειρεί να το χρησιμοποιεί.
- no-restricted-properties - Ισως χρήσιμο σε περιπτώσεις Update, ή deprecation καποιας ιδιότητας.
- no-octal-escape - Δεν ξερω αν έχει σημασία για το εργο αλλά εφόν έχει γίνει deprecated το octal escape ίσως να ήταν καλό να εμπαινε
- no-redeclare - Χρησιμο αλλά αν υπάρχει ήδη είναι αχρειαστο
- no-object-constructor - Θα είχε αξία να προστεθεί απλά ίσως να χρειαζομαστε και το flexibility του να μην έχει argument ο constructor.
- no-new - Γενικά δεν είναι σίγουρο αν θα χρειαζομασταν μια προειδοποίηση για τετοιο ζητημα. Κάποιες φορές ίσως να χρειάζεται να δημιουργηθεί και κατευθείαν. Ας το σκεφτούμε.
- no-negated-condition - Αυτός ο κανόνας υπήρχε και στον Clean Code του Μαρτιν. Προσωπικά συμφωνώ με τον κανόνα απλά ο κανόνας αυτός θα επρεπε να επιβαλλεται και σε άλλα κομμάτια για να είχε πραγματική αξία. Π.χ. ένα ονομα μεταβλητης hasNotSomeProp = …; θα επρεπε και αυτό να δημιουργούσε πρόβλημα για τον ίδιο λόγο (έχεις μια αρνητική πληροφορία η οποία κατά τον Μαρτιν είναι πιο δύσκολο να ερμηνεύσει και να κατανοήσει καποιος). Εφόσον αυτό δεν εφαρμόζεται ο

προγραμματιστης απλά θα μπορούσε να βαλει μια συνθήκη που θα περνούσε τον κανόνα σε μια μεταβλητή με “αρνητικό” όνομα και να λυνει το θεμα του linter αλλά δημιουργωντας θεμα στο readability των μεταβλητων. Ισως να ήταν καλό να είμασταν γενναίοι και να τον βαζαμε απλά τοτε θα ήθελε όλο το Unify τρελό refactor καθως καθε αρχείο μπορεί να είχε θέμα. Οπότε δεν ξερω αν τελικά θα ηταν καλο να μπει ο κανονας και να επιβάλλει ένα σωστό κατά την άποψή μου τρόπο γραφής ή θελουμε το flexibility.

- no-labels - Πολύ σπάνια περίπτωση. Αλλά σίγουρα θα βοηθούσε η επιβολή του στο readability
- no-invalid-this - Καλός κανόνας αλλά δεν ξερω αν η κατα πόσο η χρήση του θα βοηθούσε ή θα εμπόδιζε.
- no-implicit-globals - Θα χρειαζόταν λίγη ερευνα και ισως πιλοτική εφαρμογή για να βλέπαμε αν ταιριάζει στο πρότζεκτ η θα γεμίσει με warnings το project.
- no-implicit-coercion - Θα βοηθούσε αλλά κάποιες φορές ισως να ηταν συντομότερο να το κάνεις implicitly.
- no-extra-boolean-cast - Καλός κανόνας απλά ίσως να θελαμε το flexibility να εχουμε boolean cast όποτε θέλουμε και όχι βάσει κανόνα
- line-comment-position - Βελτιώνει το readability του κώδικα αλλά θα περιόριζε την ελευθερία μας. Οπότε θέλει σκέψη. Δεν βελτιώνει κάτι προγραμματιστικά απλά θέτει ένα μοναδικό τρόπο συγγραφής comments.
- symbol-description - Τα Symbol χρησιμοποιούνται σπάνια (αν χρησιμοποιούνται) οπότε ίσως να βοηθούσε αν εμπαινε αλλά ίσως όχι αν υπάρχει περίπτωση που δεν πρέπει να έχουμε descriptor.
- sort-vars - Ισως βοηθούσε αλλά θα χρειαστεί ιδιαίτερη προσπάθεια από όλους για να κατανοούν τι γινεται. Πολύ intrusive κανόνας. Ισως χρειαζεται.
- sort-keys - Ιδιος λόγος με το sort-vars.
- sort-imports - Θα αυξήσει σίγουρα τον χρόνο development αλλά ίσως να άξιζε λόγω καλύτερης αναγνωσιμότητας. Στα από πάνω θα πρέπει να συμπεριληφθεί και το ότι θα χρειαστεί καιρός ωστε τα υπάρχοντα αρχεία να μπουν σταδιακά σε αυτό το στυλ.
- require-yield - Σπάνια, ίσως και ποτε δεν έχει τύχει να χρησιμοποιησω generators. Αν υπάρχει τέτοιο case ας προστεθεί. Αν δεν υπάρχει θα ήταν κακό να μπει ένα άχρηστος (για το έργο μας) κανόνας.
- https://eslint.org/docs/latest/rules/require-unicode-regexp - Ισως προσθέσει αξία ή ίσως δυσκολέψει το εργο δημιουργίας μιας σωστής Regular Expression με συνεχόμενα λάθη ακόμα και αν είναι ορθό.