Κανόνες που Πρέπει να Προστεθούν

Κανονες για Πιθανά Ζητήματα Λογικής

- array-callback-return - Βοηθά στην αποφυγή bugs
- no-compare-neg-zero - Ευτυχώς δεν το έχω δει ποτέ. Αλλά ας το επιβάλουμε 🙂
- no-cond-assign - Βοηθά στην αποφυγή Bugs.
- no-const-assign - Νομίζω υπάρχει ήδη. Σιγουρα χρήσιμο
- no-constant-binary-expression - Θα ήταν χρήσιμο σε περιπτώσεις όπου είτε ο προγραμματιστής είτε καποιο εργαλείο σαν το prettier παρεβλεπε το precedence και οδηγού στο παραπάνω.
- no-constant-condition - Βοηθά στην αποφυγή bugs
- no-dupe-else-if - Θα είχε αξία σε κάποια περίπτωση όπου ενα chain από πολλες συνθηκες δυσκολεύουν το debugging συνθηκων (πχ Αν έχω 10 else-if τότε σίγουρα θα ήταν ωφελιμος ο κανόνας) Ωστόσο ισως να μην χρειάζεται να πάμε σε review και θεσπιση κανόνων που αποτρεπουν τετοιες περιπτωσεις.
- no-dupe-args - Θα βοηθήσει στο readability κώδικα
- no-dupe-keys - Σιγουρα χρησιμο. Μαλλον υπάρχει ήδη
- no-duplicate-case - Καλό είναι να γινόταν enforce αν δεν γινεται ήδη
- no-duplicate-imports - Απαραίτητο. Νομίζω υπάρχει ήδη
- no-empty-character-class - Θεωρω θα βοηθούσε στην αποφυγη συντακτικών λαθών
- no-empty-pattern - Κανει enforce μια καλή πρακτική για μια συνθετη πραξη όπως το destructuring
- no-fallthrough - Απαραίτητος ο κανόνας γιατί κάνει enforce κατι πολύ χρήσιμο. Γενικότερα δεν εμποδίζει το μοιρασμα/κατηγοριοποίηση statements. Απλά επιβάλλει οτι κάθε case θα εχει και ένα return. Αν καποιο statement οντως δεν πρέπει να έχει καποιο return ή break, τότε θα υπάρχουν παραδείγματα σωστού κώδικα από τα οποία ο κανόνας περνάει.
- no-import-assign - Θα ήταν καλό να προστεθεί ο κανόνας αυτός.Η λογική: αν χρειάζεται modification κατι απο ενα import, πρώτα να γινεται ενα αντιγραφο και μετά οι αλλαγες (mutations) να εφαρμόζονται στο αντιγραφο. Το αρχικό import καλύτερα να θεωρείται / είναι immutable.
- no-invalid-regexp - Σιγουρα βοηθητικός κανόνας που θα ξενίσει στην αρχή αλλά θα βοηθήσει στην σωστότερη δηλωση RegExp και στον χρόνο debugging.
- no-loss-of-precision - Σπάνια περίπτωση αλλά θα ήταν πολύ χρήσιμη σε περίπτωση λάθους ή debugging.
- no-obj-calls - Θα βοηθούσε ως προειδοποιηση σε ορισμένες περιπτώσεις
- no-template-curly-in-string - Θα βοηθούσε στο readability του κώδικα
- no-undef - Πολύ χρήσιμος κανόνας που θα βοηθήσει σε mistypes. Υπάρχει ήδη
- no-unexpected-multiline - Τα ; θα εισερχονται αυτομαται από prettier αλλά και πολύ το να ειδοποιούμε ότι κάτι πάει λάθος σε περίπτωση που δεν μπει θεωρείτε χρήσιμο.

- no-unreachable - Σε περιπτωση unreachable κωδικα καλό θα ήταν να υπάρχει μια ειδοποιηση
- no-unsafe-finally - Ίσως βοηθούσε για το unexpected behavior αυτού που πραγματεύεται. Απλά ισως κάποια περίπτωση return που θα περιλαμβανε και το try/catch + καποιον αλλον υπολογισμό επιπρόσθετα να ήταν καλό να υφίσταται.
- no-unsafe-optional-chaining - Θα εκανε enforce την προσθήκη safe εναλλακτικών και θα εβαζε τον προγραμματιστη να σκεφτεί και την περίπτωση με υπαρξης του Object που θα αλλαξει.
- use-isnan - Χρήσιμο θα ήταν. Κανει enforce μια συγκεκριμένη συνάρτηση δημιουργώντας συνέπεια σε όλο τον κώδικα όπου γίνεται ο έλεγχος για NaN.
- no-useless-backreference - Χρησιμο refactor που διευκολύνει την συγγραφη RegEx
- no-unused-vars - Υπάρχει ήδη. Πολύ χρήσιμο.
- valid-typeof - Χρησιμος κανόνας για ελεγχο right hand operators ενός typeof

Προτάσεις για Consistency Γραφής μέσω Κώδικα

- arrow-body-style - Χρησιμος με την επιλογη as-needed και επιζήμιος με always, never
- block-scoped-var - Χρησιμο για συναδέλφους που δεν έχουν μεγάλη εμπειρία στην χρήση της γλώσσας. Καλό θα ήταν να γινόταν enforce το παραπάνω.
- camelcase – θα βοηθούσε το enforce ενός συγκεκριμένου τρόπου γραφής μεταβλητων και είναι κάτι που από μεγάλο μέρος συναδέλφων έχει ήδη υιοθετηθεί ήδη ατυπα.
- consistent-return - Θα ήταν χρήσιμο να γινει enforce ο κανόνας αυτός γιατί θα διευκόλυνε το readability. Ειδικά αν επιβαλλόταν και ενα return στο τελος εκτός από if else tote θα είχαμε το τελειο.
- curly - Το θεωρω χρησιμο αρκεί να μπει ενας κανόνας που θα συμφωνήσουμε όλοι οτι δεν αυξάνει τραγικά τις γραμμές κώδικα αλλά ουτε και θα χτυπάει error για τα blocks αν χρειάζονται. Προτείνεται η χρήση των keywords multi, multi-line, multi-or-nest και του consistent και αντιπροτείνεται η χρήση του all. Προτεινεται να γραψουμε με όλα τα στυλ για να βρουμε τον καλύτερο συνδυασμό πριν αποφασίσουμε για τα keywords. 🙂
- default-case - Καλό θα ηταν να υπαρχει
- default-case-last - Καλό θα ήταν να υπάρχει enforce του συγκεκριμένου κανόνα
- default-param-last - Επιβάλλει ένα ωφέλιμο pattern γραφής και αυτό που προτάσει είναι αρκετά λογικό αν καποιος δεν το σκεφτεί απο μονος του.
- eqeqeq - Πολύ σημαντικός κανόνας για enforce strict equality
- id-length - θα ήταν χρήσιμο να προστεθεί ένα max limit χαρακτηρων για ονομα μεταβλητης. Υπάρχει και η επιλογή για ελαχιστους χαρακτήρες αλλά καλύτερα να αποφευγουμε χρήση παραπανω χαρακτήρων από όσους πραγματικά χρειαζόμαστε. Κάποια μικρά ονόματα με ενα χαρακτηρα είναι ακριβέστατα βασει λοιπου context.
- init-declarations - Υποχρεώνει για ένα αρχικό declaration μεταβλητής. Χρησιμο
- logical-assignment-operators - Θα κανει πιο ευκολονοητο τον κώδικα

- max-depth - Απαραίτητο να προστεθεί. Απλά πρέπει να συμφωνήσουμε τον max callback nest allowance αριθμό. 🙂
- max-lines -Ο μέγιστος αριθμός γραμμων ενός αρχείου. Σιγουρα πρέπει να προστεθεί. Το documentation δινει ένα φυσιολογικό range μεταξύ 100-500. Προτεινεται το 300 σαν μέσος όρος.
- max-lines-per-function - Θεωρώντας οτι μια συναρτηση θα πρέπει να κάνει ένα συγκεκριμένο πράγμα θα ήταν καλό να προστεθεί και ένα ανωτατο οριο γραμμων μια συναρτησης.
- max-nested-callbacks - Οπως και τα παραπάνω θα βοηθήσει πολύ στην κατανοηση του κώδικα.
- max-params - Θα βοηθησει στο readability του κώδικα.
- new-cap - Θα είναι σπάνια η χρήση του αλλά καλό είναι να επιβάλλεται το naming convention του κανόνα για οποτε χρησιμοποιείται το new.
- no-alert - Χρησιμος κανόνας για αποφυγη προβλημάτων σε παραγωγή
- no-case-declarations - Δεν θα προσδώσει αξία στο εργο
- no-console - Υπάρχει ήδη. Χρησιμος κανόνας για την αποφυγή console στον κωδικα.
- no-continue - Υπάρχουν εναλλακτικές τόσο σε λογικό επίπεδο όσο και συντακτικά (break, return) ωστε να διακόπτεται η ροή ενός loop. Δεν χρειάζεται το continue οποτε ας γινει enforce ο κανόνας αυτός.
- prefer-spread - Και το apply και το spread το ίδιο πράγμα κάνουν αλλα το spread εχει καθιερωθεί και αναγνωρίζεται ευκολότερα αρα κανει τον κωδικα με spread πιο ευκολο να αναγνωριστεί / κατανοηθεί.
- prefer-rest-params - Οπως και το από πάνω κάνει enforce ένα νεοτερο και πιο διαδεδομένο pattern της γλώσσας το rest εναντι του arguments για να διαβαζει τις παραμέτρους μια συνάρτησης αν οι παραμετροι δεν ειναι σιγουρο το πόσοι ποιοί είναι.
- prefer-promise-reject-errors - Αρκετα σημαντικός κανόνας για normalization τον τρόπο και των μηνυμάτων λάθους. Είναι καλό τα Reject να δίνουν Error object.
- prefer-object-spread - Προτείνεται η χρήση spread εναντι του Object.assign. Χρησιμος κανόνας που ήδη ατυπα χρησιμοποιείται στην μεγαλυτερη πλειοψηφία του έργου
- prefer-const - Αρκετά χρήσιμος κανόνας που συμβαδίζει με το functional paradigm του immutability.
- prefer-arrow-callback - Είναι κάτι που γίνεται ήδη οπότε ας το κάνουμε enforce εναντι του .bind που προυπηρχε
- object-shorthand - Θα ήταν καλό να μπαίνει ένα consistent style παντού σχετικά με αυτό το κομμάτι για να βελτιωνεται το readability του κώδικα όποιο και να είναι αυτό το στυλ
- no-var - Χρησιμοποιείται ήδη νομίζω. Πολύ χρήσιμο.
- no-useless-return - Θα ενδυναμώσει το readability του κωδικα και θα επιβάλλει ένα καλύτερο τρόπο εκφρασης της εκαστοτε λογικης.
- no-useless-escape - Ενα warning θα ηταν ωφελιμο για την περιπτωση αρχειαστου escape. Θα βοηθούσε στο readability.

- no-useless-concat - Χρησιμο και σχετικά ανώδυνο refactor που θα βελτιώσει την αναγνωσιμότητα
- no-useless-catch - Χρησιμο για το performance αλλά και για την κατανόηση του κομματιού κώδικα που διαβάζει κάποιος.
- no-useless-call - Εφαρμόζει νεότερα πρότυπα περασματος παραμετρων
- no-unused-labels - Χρησιμο. Ισως υπάρχει ήδη
- no-unused-expressions - Θα βελτιώσει το χρόνο του development και του debugging.
- no-unneeded-ternary - Αφαίρεση άχρηστων ternary. θα βελτιώσει την ποιότητα κώδικα
- no-undefined - Χρησιμη προσθήκη που θα βοηθούσε στην αποφυγη bugs λόγω Undefined value.
- no-shadow-restricted-names - Θα ήταν χρήσιμο σε περίπτωση που αρχικοποιηθει μια μεταβλητή που ατυχώς έχει ένα restricted keyword να εντοπιστεί αμεσα από τον κανόνα αυτόν.
- rules/no-shadow - Χρησιμος κανόνας που διαχωρίζει τα ονοματα των μεταβλητων και ετσι βοηθά τον προγραμματιστή να κατανοήσει το προγραμμα καλύτερα.
- no-sequences - Δεν το εχω δει ποτε αλλά καλό θα ήταν να γινει enforce
- no-return-assign - Βελτιωση readability
- no-redeclare - Χρησιμο αλλά αν υπάρχει ήδη είναι αχρειαστο
- no-new-wrappers - Έχει υπάρξει η ανάγκη να τρεχει ένα effect κάθε φορά που αλλαζε η τιμή ενός string που ήταν dependency και για τον λόγο αυτό χρησιμοποιήθηκε το new keyword. Εκεί θα μπορούσαμε να κάνουμε ignore τον κανόνα αυτόν. Περαν αυτού του σεναρίου θα χρήσιμο να προστεθεί ο κανόνας αυτός για αποφυγή κακών πρακτικών του παρελθόντος.
- no-new-func - Η προσθήκη του κανόνα αυτού κρίνεται εξίσου χρήσιμη για τον ίδιο λόγο που και το no-new-wrappers θα επρεπε να μπει. Βοηθα στην αποφυγη ενός anti-pattern.
- no-multi-assign - Δεν είναι συχνό κάτι τέτοιο αλλά θα εφτιαχνε το readability και θα έδινε και ένα μήνυμα σε καποιον που πχ μπερδεύτηκε και αντί των equality operators έβαλε τον assign operator.
- no-magic-numbers - Θα ήταν αρκετά χρήσιμο να μπει και θεωρώ πως το project θα επωφελούνταν. Απλά καλό θα ήταν να το σκεφτούμε καλά γιατί το flexibility που παίρνει είναι σημαντικό.
- no-loop-func - Θα αποφεύγαμε συγγραφή anti-patterns. Θα βελτιωνόταν η δυνατοτητα κατανοησης του κώδικα
- no-lonely-if - Επιβαλει ενα καθαροτερο πρότυπο γραφής. Θα βοηθούσε στην καλυτερη κατανοηση περιεργων συνθηκων.
- no-lone-blocks - Ιδιος λόγος με τον από πάνω. Πολυ Σπανια το έχω δει βεβαια οποτε ίσως και να μην χρειαζεται καν να το υιοθετήσουμε.
- no-iterator - Επιβαλει νεοτερα πρότυπα γραφης οπότε θα ήταν καλό να μπει.
- no-inline-comments - Θα εκανε τον κωδικα πιο ευκολο να διαβάζεται αν και δεν έχω δει να γινεται εως τωρα καπου. Εχω δει ωστόσο commented code από πριν 2 χρόνια οπότε αν επιβαλλεται και εκεί ακόμα καλύτερα.

- no-extend-native - Θα ήταν καλό να γινει enforce μεσω του κανόνα η συγκεκριμένη πρακτική που πραγματεύεται. Ουτως ή αλλως ο κωδικάς στο front είναι τόσο απλός ώστε θα μπορούσε θεωρητικά να αποφευχθεί το Prototype mutation.
- no-eval - Κρινεται εποικοδομητικό καθώς στο front δυσκολα να χρειαστεί το eval και όπως αναφέρει ο κανονας εχει θεματα ασφαλειας.
- no-empty-static-block - Θα ηταν ωφελιμος και θα προσθέτει αξία σε περιπτωσεις που ο προγραμματιστης παραβλέψει καποια συνθηκη.
- no-empty-function - Ιδιο με το από πάνω
- no-empty - Αρκετα ωφελιμος για τους ίδιους λόγους με τους παραπάνω.
- no-else-return - Θα επεβαλε το pattern του ενός return στο τέλος του block κατι αρκετά ωφελιμο για το readability και την κατανόηση του κειμένου. Ίσως να μπερδευε αλλά μακροπρόθεσμα αν σκεφτόμαστε όλοι να μειώνουμε τα σημεια εξόδου των βρόχων μας θα βοηθούσε και το debugging.