The Adoption of JavaScript Linters in Practice: A Case Study on ESLint

Kristín Fjóla Tómasdóttir, Maurício Aniche, Arie van Deursen

September 2018

Online appendix: `https://doi.org/10.5281/zenodo.1410967`.

1 Interview Questions

In the following, we present the base list of questions that we asked in each interview.

1.1 Participant Information

1. How many years/months of experience do you have as a professional developer?
2. How many years/months of experience do you have developing in JavaScript?
3. For how many years/months have you been a contributor to the X project?
4. What is your role in the X project? (*e.g.*, Founder, Lead Developer (core team), Maintainer/Developer, Tester, Documenter, Translator..)

1.2 Linter Usage

1. Why do you use a linter in your project?
2. How do you create your *.eslintrc* configuration file and maintain it? that is, how do you choose and prioritize the rules?
3. Given the rule categories from ESLint, which categories do you consider to be the most important and why?
4. Given the same categories, which categories do you consider to be the least important and why?
5. Which individual rules (within any category) do you consider to be the most important and why? (*e.g.*, top five rules)

6. Do you have any particular reasons for not choosing some of the rules for the configuration file in your project?
7. Do you use warnings and errors for different purposes?
8. Why are some files ignored in the *.eslintignore* file?
9. Are there any specific challenges about using a linter?
10. Do you experience false positives? if so, which?
11. With JavaScript being a dynamic language, do you feel that some features are missing in a static analysis tool such as ESLint?
12. If ESLint rules were to be prioritized in some manner, *e.g.*, to create a top 20 list of "must-have rules", which (if any) method(s) would you consider to be useful?
    a) Common configs from projects
    b) Developers' opinions of importance
    c) Most commonly eliminated errors
    d) The effects of the warnings/errors on change- and defect-proneness of files
13. Anything else about linters you would like to add?

2 Survey Questions

In the following, we present the main questions of the survey. The entire survey can be found in our online appendix.

1. How important do you think it is to use a linter in a JavaScript project? [Unimportant - Slightly important - Neutral/NA - Important - Very important]
2. In your previous experience with using a linter, why did you use it? Rate your level of agreement with the following reasons [Strongly disagree - Disagree - Neutral/NA - Agree - Strongly agree].
    - To avoid giving negative comments while doing code reviews and thus sparing the developers' feelings
    - To maintain code consistency
    - To avoid ambiguous or complex code
    - To catch bugs and/or typing mistakes
    - To automate parts of a code review
    - To learn about JavaScript features and/or syntax

- To save time on discussing code style

3. In your previous experience with using a linter, which method(s) did you use to select the rules that are included in the configuration file? [Check all that apply]

    - Use default configurations from the linter
    - Use a preset (a set of configurations made publicly available by someone)
    - Choose rules that fit the current style of a project
    - Have the linter automatically generate configurations that fit the project
    - Enable rules that come up in discussions on a project (*e.g.*, on a pull request or in an issue tracker)
    - Choose the most commonly used style within team
    - Try to have the configuration as minimalistic as possible
    - Choose rules that involve the least effort to follow (*e.g.*, by not having to bypass the rules too often)
    - I don't care which rules are enabled, as long as there is a set of rules present in the project
    - I have never configured a linter
    - Other: Write in

4. Which of the ESLint categories do you consider to be important to include in configurations? [Unimportant - Slightly important - Neutral/NA - Important - Very important]

    - Stylistic Issues
    - Possible Errors
    - ECMAScript 6
    - Best Practices
    - Node.js and CommonJS
    - Strict Mode
    - Variables

5. Rank the importance of each of these rules in the X category *(Stylistic Issues, Possible Errors, Best Practices, Variables)* [Unimportant - Slightly important - Neutral/NA - Important - Very important].

    - *The list of rules for each category can be found in our appendix.*

6. In your previous experience with using a linter, which challenges have you faced? [Check all that apply.]

- Creating or maintaining configurations
- Enabling rules in an existing project
- Agreeing on which rules to use within a team
- Enforcing rules in a team
- The lack of analysis for dynamic features of JavaScript
- The presence of false positives (a wrongly indicated error/warning)
- Too many warnings/errors outputted from the linter
- No challenges
- Other: Write in

7. Do you experience false positives while using a linter? A false positive is a wrongly indicated error/warning. If you intend to break a rule in a specific case, it is not considered as a false positive. [Never - Very rarely - Rarely - Not applicable - Occasionally - Frequently - Very frequently]

8. If you have any suggestions on how to improve linters for JavaScript, please add them here.