New Rules

ESLint is all about rules. For most of the project's lifetime, we've had over 200 rules, and that list continues to grow. However, we can't just accept any proposed rule because all rules need to work cohesively together. As such, we have some guidelines around which rules can be part of the ESLint core and which are better off as custom rules and plugins.

Note: As of 2016, we accept only rules that are deemed extremely important for inclusion. We prefer that new rules be implemented in plugins.

Core Rule Guidelines

In general, ESLint core rules must be:

1. **Widely applicable.** The rules we distribute need to be of importance to a large number of developers. Individual preferences for uncommon patterns are not supported.
2. **Generic.** Rules cannot be so specific that users will have trouble understanding when to use them. A rule is typically too specific if describing what it does requires more than two "and"s (if a and b and c and d, then this rule warns).
3. **Atomic.** Rules must function completely on their own. Rules are expressly forbidden from knowing about the state or presence of other rules.
4. **Unique.** No two rules can produce the same warning. Overlapping rules confuse end users and there is an expectation that core ESLint rules do not overlap.
5. **Library agnostic.** Rules must be based solely on JavaScript runtime environments and not on specific libraries or frameworks. For example, core rules shouldn't only apply if you're using jQuery but we may have some rules that apply only if you're using Node.js (a runtime).
6. **No conflicts.** No rule must directly conflict with another rule. For example, if we have a rule requiring semicolons, we cannot also have a rule disallowing semicolons (which is why we have one rule, `semi`, that does both).

Even though these are the formal criteria for inclusion, each rule is evaluated on its own basis.

Proposing a Rule

If you want to propose a new rule, please see how tocreate a pull request or submit an issue by filling out anew rule template.

We need all of this information in order to determine whether or not the rule is a good core rule candidate.

Accepting a Rule

In order for a rule to be accepted in the ESLint core, it must:

1. Fulfill all the criteria listed in the "Core Rule Guidelines" section
2. Have an ESLint team member champion inclusion of the rule
3. Be very important for ESLint users because it either catches a serious problem or allows styling of code in accordance with a popular style guide

Keep in mind that we have over 200 rules, and that is daunting both for end users and the ESLint team (who has to maintain them). As such, any new rules must be deemed of high importance to be considered for inclusion in ESLint.

Implementation is Your Responsibility

The ESLint team doesn't implement new rules that are suggested by users because we have a limited number of people and need to focus on the overall roadmap. Once a rule is accepted, you are responsible for implementing and documenting the rule. You may, alternately, recruit another person to help you implement the rule. The ESLint team member who championed the rule is your resource to help guide you through the rest of this process.

Alternative: Creating Your Own Rules

Remember that ESLint is completely pluggable, which means you can create your own rules and distribute them using plugins. We did this on purpose because we don't want to be the gatekeepers for all possible rules. Even if we don't accept a rule into the core, that doesn't mean you can't have the exact rule that you want. See the working with rules and working with plugins documentation for more information.