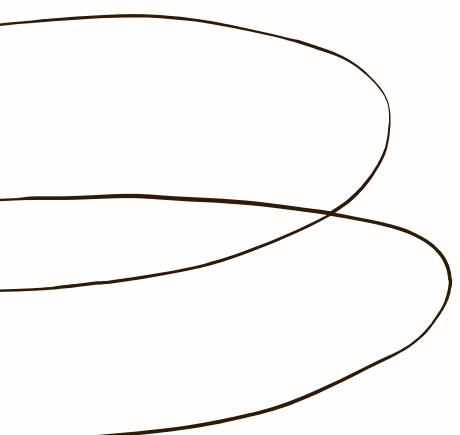




ESLINT



Subject Name : Software engineering
Subject Code : CS16105

Submitted by-

Subject Co-Ordinator : Mr. Anoj Kumar

Team leader :	Anuj kesarwani	20208022
Member :	Harshit chaudhari	20208054
Member :	Jitendra kumar	20208060
Member :	G.Tarun	20208047




Table of content

- Introduction
- Literature Review
- Survey
- Why Use Eslint
- Working with example
- Hardware/Software Specification Required
- Comparision
- Limitation
- Future Aspects
- Conclusion
- Reference

Contributions

Anuj kesarwani

Hardware/Software specifications, Limitation
and comparision with other tools

Jitendra kumar

Literature Review, Survey Review
collection content

Tarun

Why Use Eslint, Working of Eslint
Working example of Eslint

Harshit Chaudhari

Explored 'Future Aspects' and
concluded 'Software Development
Paradigm' followed to develop
considered ESLint Tool with references.

Introduction

- Th ESLint is a popular tool for analyzing and enforcing code quality in JavaScript projects. It can help you catch potential errors, enforce consistent coding standards, and generally make your code more maintainable and easier to work with.
- Reference <https://eslint.org/docs/latest/>

Literature Review

- In order to understand the history and evolution of ESLint, it's helpful to review some of the research and literature that has been published on the topic.
- ESLint has become one of the most widely-used code quality tools in the JavaScript community. There have been numerous research papers and articles published on the tool, evaluating its effectiveness and exploring its capabilities.
- Some of the key features of ESLint that have been praised in the literature include its flexibility, its extensibility, and its ability to integrate with popular text editors and IDEs.
- References<https://eslint.org/docs/latest/maintain/working-groups>

Review Survey

- To get a sense of how ESLint is used in practice, it's helpful to review some of the recent surveys and user reviews of the tool.
- According to recent surveys, ESLint is used by a large percentage of JavaScript developers, and is generally well-regarded for its effectiveness in improving code quality and maintainability
- References You can find it at <https://eslint.org/docs/>

Why Use ESLint



ESLint

References:

ESLint Official Website: <https://eslint.org/>

ESLint Configuration Documentation:

<https://eslint.org/docs/user-guide/configuring>

- ESLint is a popular JavaScript linter that helps developers identify and fix code errors, enforce coding standards, and improve code quality.
- It can catch common programming mistakes like typos, unused variables, and undefined functions, which can save time and prevent bugs.
- By using ESLint, developers can write cleaner, more efficient code, and reduce the chances of introducing bugs or vulnerabilities.

Working Of Eslint



References:

ESLint CLI Documentation:

<https://eslint.org/docs/user-guide/command-line-interface>

ESLint Plugin Documentation:

<https://eslint.org/docs/user-guide/plugins>

- Eslint works by analyzing javascript code and identify potential errors
- It can be integrated into variety of development environments, including IDEs, editors, and build tools.
- Es lint can be customized with with plugins and rules to fit specific project requirements or coding standards
- When Eslint detects an error or violation, it provides feedback in form of warnings or errors, which can be viewed in console or editor.

Working Example Of ESLint

Step1: Install eslint package

```
PS D:\Tarun\ReactJs\EsLint> npm i eslint --save -dev
```

```
.eslintrc 1 X
EsLint > .eslintrc > ...
1  {
2    "env": {
3      "browser": true,
4      "es6": true
5    },
6    "extends": "eslint:recommended",
7    "parserOptions": {
8      "ecmaVersion": 2018
9    },
10   "rules": {
11     "no-console": "off",
12     "semi": ["error", "always"],
13     "quotes": ["error", "single"],
14     "indent": ["error", 2],
15     "no-unused-vars": ["error", { "args": "none" }]
16   }
17 }
18
19
20
```

Step2: Make a .eslintrc file

Step4: Run eslint ./src

Step3: Make a test.js file

```
JS test.js 4 X
EsLint > src > JS test.js > ...
1  const myVar = 123;
2  myVar = 456;
3
4  function myFunc() {
5    console.log("Hello, World!");
6  }
7
8  console.log(myVar);
9
10
```

```
PROBLEMS 4 OUTPUT TERMINAL DEBUG CONSOLE
⊗ PS D:\Tarun\ReactJs\EsLint> eslint ./src

unused-vars
5:15 error Strings must use singlequote quotes
2:1 error 'myVar' is constant no-const-assign
4:10 error 'myFunc' is defined but never used no-unused-vars
5:15 error Strings must use singlequote quotes

✖ 3 problems (3 errors, 0 warnings)
1 error and 0 warnings potentially fixable with the `--fix` option.
```

Hardware/software Specification Required

ESLint is a software tool that can be installed on a variety of operating systems and integrated with a variety of code editors and development environments

1

Operating System: ESLint is compatible with most operating systems, including Windows, macOS, and Linux.

2

Code Editor/IDE: ESLint can be used with a variety of code editors and integrated development environments (IDEs), including Visual Studio Code, Sublime Text, IntelliJ IDEA.

3

Software requirement: ESLint is a Node.js-based tool and requires Node.js to be installed on the computer.

4

RAM and Disk Space: The amount of RAM and disk space required to run ESLint will depend on the size of the codebase you are working with.

Comparison with Other Similar Tools

1 There are several other code analysis and linting tools available for JavaScript, including **JSHint**, **JSLint**, and **TSLint**.

2 While these tools share some similarities with ESLint, each has its own unique features and capabilities.

3 One advantage of ESLint is its flexibility and extensibility, which allows it to be customized to meet the specific needs of your project or team.

4 Additionally, ESLint has a large and active community of contributors, which means that issues and bugs are often addressed quickly and new features are added regularly.

Limitations Of Eslint

There are some limitations that developers should be aware of.
Here are some common limitations of ESLint.

1

Limited language support

3

Configuration complexity

2

Performance impact

4

False positives

Future Aspects

Here are some future aspects of ESLint.

1

Improved TypeScript support:

ESLint already has some support for TypeScript, but it could benefit from further improvements to provide better type checking and inference for TypeScript code.

2

Expanded rule sets: As the

JavaScript ecosystem continues to evolve, new best practices and conventions may emerge that ESLint could incorporate into its rule sets.

3

Enhanced integration with other tools: There are many other tools in the JavaScript ecosystem that can work together with ESLint to provide a more comprehensive development experience. For example, ESLint could integrate more tightly with code editors like VS Code, or with build tools like Webpack.

4

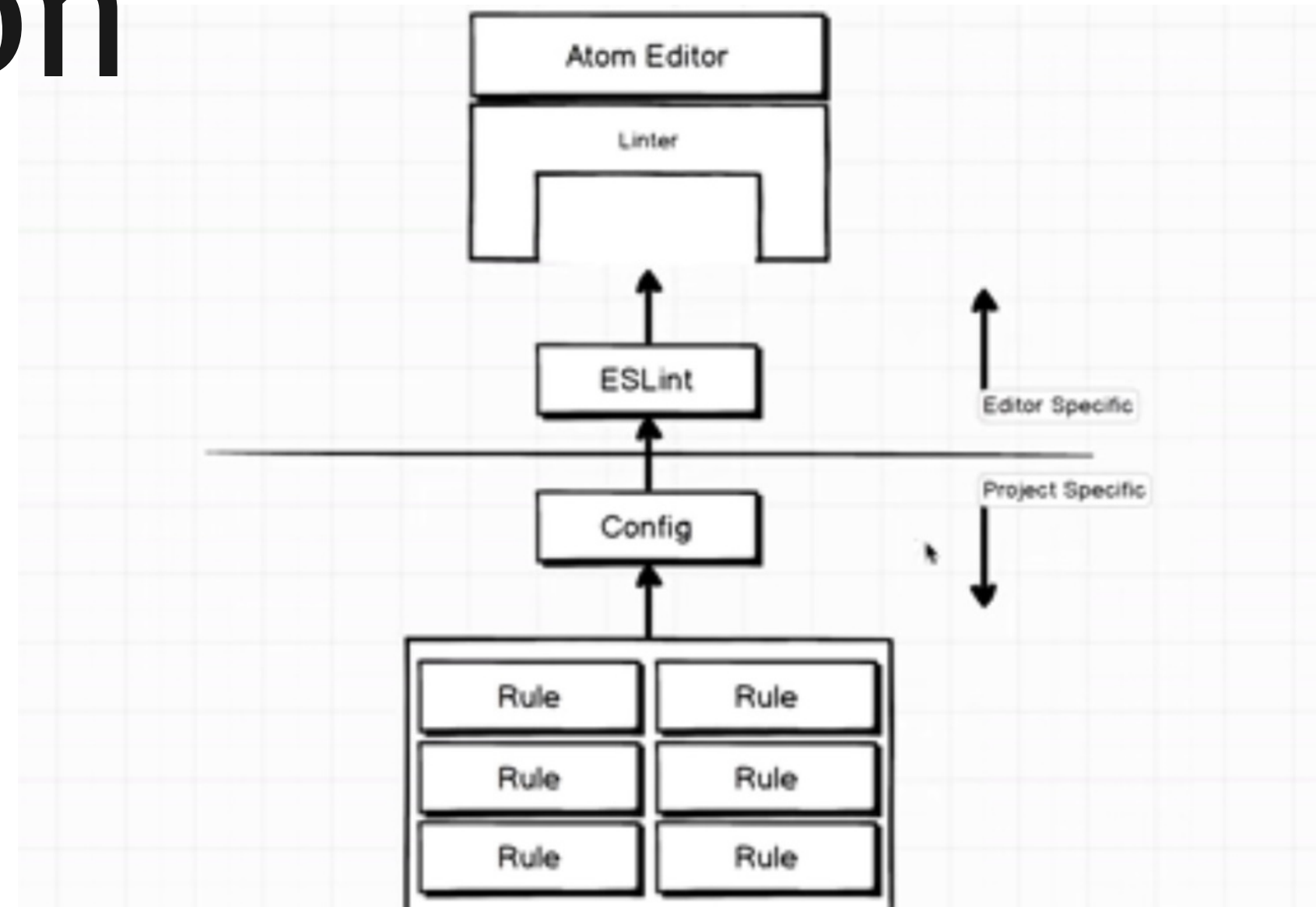
Increased focus on performance: As ESLint's popularity continues to grow, there may be a greater emphasis on optimizing its performance to ensure that it can handle large code bases efficiently.

5

Support for other programming languages: While ESLint is currently focused on

JavaScript, there may be interest in supporting other languages in the future, such as TypeScript, JSX, or even other programming languages

Conclusion



As an open source project, ESLint does not follow a specific software development model in the same way that a commercial software development team might. However, most open source projects, including ESLint, tend to follow a collaborative and iterative development model that emphasizes community involvement and continuous improvement. The development process for ESLint is largely driven by its community of contributors and users, who submit bug reports, feature requests, and code contributions through GitHub. The core development team reviews these contributions and decides which ones to incorporate into the project.

Conclusion

In terms of the actual development process, the ESLint project uses a combination of agile and iterative development practices. New features and improvements are added to the project in small increments, and feedback from the community is used to refine and improve these changes over time.

ESLint also uses a continuous integration and continuous delivery (CI/CD) process to ensure that changes to the project are thoroughly tested and verified before they are released to users. This process involves automatically building and testing the project on a regular basis, and using automated tools to identify and fix issues.

References

1

ESLint Documentation: The official documentation for ESLint includes usage instructions, configuration options, and information about writing custom rules and plugins. You can find it at <https://eslint.org/docs/>

2

ESLint GitHub Repository: The ESLint source code is hosted on GitHub, where you can browse the code, report issues, and contribute to the project. You can find it at [https:// github.com/eslint/eslint](https://github.com/eslint/eslint)

3

Some research papers that focus on ESLint: **"An Empirical Study of the Impact of Linting on Code Quality"** by **Kevin Jalbert and Bram Adams**. In this paper, the authors perform an empirical study of the impact of using ESLint on code quality. They find that using ESLint improves code quality and reduces the likelihood of introducing defects.

4

"A Study of the Effectiveness of Linters for Identifying JavaScript Code Smells" by **Hani Abdeen and Mohamed Wiem Mkaouer**. This paper evaluates the effectiveness of ESLint and other linters in identifying code smells in JavaScript code. The authors find that SLint is effective at identifying code smells, but that it is not always able to detect all types of smells.