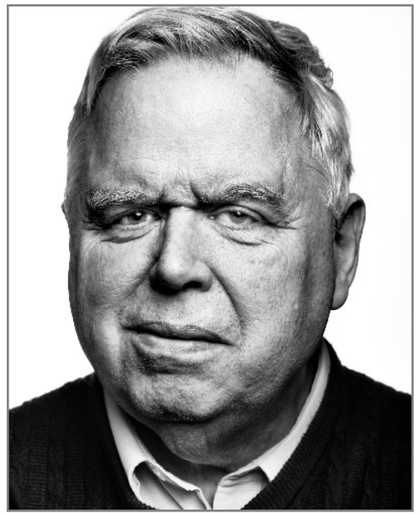
Static Analysis

Yegor Bugayenko

Lecture #23 out of 24
80 minutes



Steven Johnson

"**Lint** is a command which examines C source programs, detecting a number of bugs and obscurities. It enforces the type rules of C more strictly than the C compilers. It may also be used to enforce a number of portability restrictions involved in moving programs between different machines and/or operating systems. Another option detects a number of wasteful, or error prone, constructions which nevertheless are, strictly speaking, legal."

— Stephen C. Johnson. *Lint, a C Program Checker.* Bell Labs, 1977

Some Types of Bugs to Be Found by Static Analysis

Unreachable Code:

```
int a = 10;
if (a > 20) {
  a = a + 1;
}
```

Division by Zero:

```
int f(int x) {
  return 42 / x;
}
```

Integer Overflow:

```
var x: u8 = 142;
x = x * 2;
```

Endless Loop:

```
int x = 5;
int y = 0;
while (x > 0) {
  y = y + x;
}
```

Brian Chess

> “Beware of any tool that says something like, ‘zero defects found, your program is, rather, now secure.’ The appropriate output is, ‘sorry, couldn’t find any more bugs.”’

— Brian Chess and Gary McGraw. Static Analysis for Security. *IEEE Security & Privacy*, 2(6):76–79, 2004

Why do JavaScript developers use linters?

- Prevent Errors
- Augment Test Suites
- Avoid Ambiguous and Complex Code
- Maintain Code Consistency
- Faster Code Review
- Spare Developers' Feelings
- Save Discussion Time
- Learn About JavaScript

Source: Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie Van Deursen. Why and How JavaScript Developers Use Linters. In *Proceedings of the 32nd International Conference on Automated Software Engineering (ASE)*, pages 578–589. IEEE, 2017

My Favorite Static Analyzers

- Java: SpotBugs, Checkstyle, PMD, Qulice[Bugayenko, 2014] for Java
- C++: Clang-Tidy
- Rust: clippy

For some tools you have to pay:

- Coverity by Synopsys (US)
- Klockwork by Perforce (US)
- Fortify by Micro Focus (UK)
- Checkmarx (US)
- Veracode (US)
- Snyk (US)
- PVS-Studio (Russia)

Kristín Fjóla Tómasdóttir

"Every single interview participant mentioned that one of the reasons why they use a linter is to maintain code consistency."

— Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie Van Deursen. The Adoption of JavaScript Linters in Practice: A Case Study on ESLint. *IEEE Transactions on Software Engineering*, 46(8):863–891, 2018

Category	Description	Available rules
Possible Errors	Possible syntax or logic errors in JavaScript code	31
Best Practices	Better ways of doing things to avoid various problems	69
Strict Mode	Strict mode directives	1
Variables	Rules that relate to variable declarations	12
Node.js and CommonJS	For code running in Node.js, or in browsers with CommonJS	10
Stylistic Issues	Stylistic guidelines where rules can be subjective	81
ECMAScript 6	Rules for new features of ES6 (ES2015)	32
Total		**236**

TABLE 1: ESLint rule categories with ordering and descriptions from the ESLint documentation [28]

Source: Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie Van Deursen. The Adoption of JavaScript Linters in Practice: A Case Study on ESLint. *IEEE Transactions on Software Engineering*, 46(8):863–891, 2018

Florian Obermüller

"We introduce the concept of code perfumes as the counterpart to code smells, indicating the correct application of programming practices considered to be good. Using a catalogue of 25 code perfumes for, we empirically demonstrate that these represent frequent practices in, and we find that better programs indeed contain more code perfumes."

— Florian Obermüller, Lena Bloch, Luisa Greifenstein, Ute Heuer, and Gordon Fraser. Code Perfumes: Reporting Good Code to Encourage Learners. In *Proceedings of the 16th Workshop in Primary and Secondary Computing Education*, pages 1–10, 2021

References

Yegor Bugayenko. Strict Control of Java Code Quality. `https://www.yegor256.com/140813.html`, August 2014. [Online; accessed 26-02-2024].

Brian Chess and Gary McGraw. Static Analysis for Security. *IEEE Security & Privacy*, 2(6):76–79, 2004.

Stephen C. Johnson. *Lint, a C Program Checker*. Bell Labs, 1977.

Florian Obermüller, Lena Bloch, Luisa Greifenstein, Ute Heuer, and Gordon Fraser. Code Perfumes: Reporting Good Code to Encourage Learners. In *Proceedings of the 16th Workshop in Primary and Secondary Computing Education*, pages 1–10, 2021.

Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie Van Deursen. Why and How JavaScript Developers Use Linters. In *Proceedings of the 32nd International Conference on Automated Software Engineering (ASE)*, pages 578–589. IEEE, 2017.

Kristín Fjóla Tómasdóttir, Mauricio Aniche, and Arie Van Deursen. The Adoption of JavaScript Linters in Practice: A Case Study on ESLint. *IEEE Transactions on Software Engineering*, 46(8): 863–891, 2018.