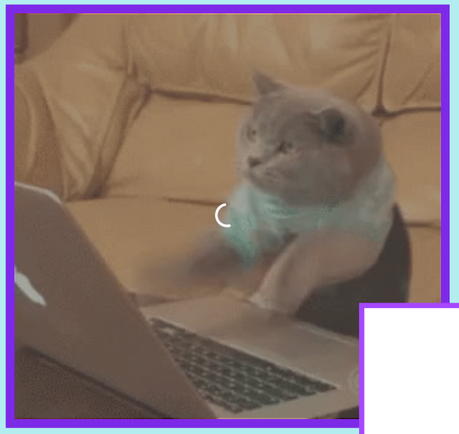
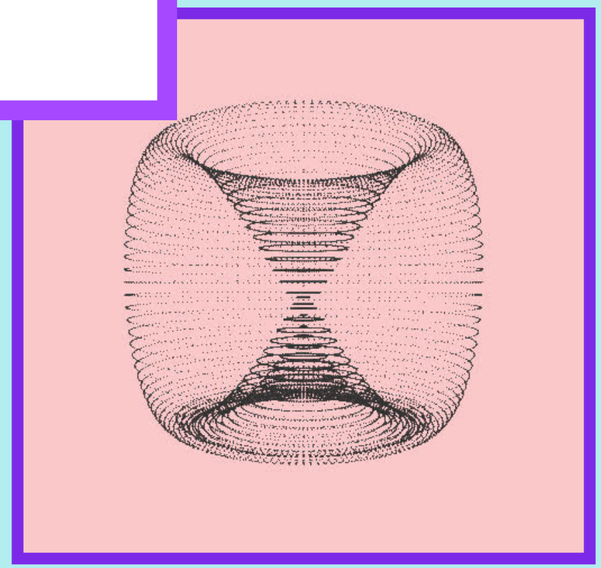


Linting



Presented by
Tom Isles

Canva

What is Linting?

Linting is a method of static code analysis.

It takes the code you write as an input, and analyses it in place to find errors.

It does not attempt to compile or run your code.

What is Linting?

It allows you to avoid common code mistakes or deviations from best practice - things that you might otherwise forget about.

It's fast. If properly integrated into your editor, it can report errors as soon as they are made.

It boosts productivity by providing a guardrail against human error.

Linting holds your hand while you code, and lets you focus on the big picture.

Linting is Not

Linting is not a substitute for good testing.

It is not a way to ensure your design is correct.

It is not a way to enforce a good architecture or system.

It is not a magic 'make my code work' tool.

It is not a system for enforcing the correct types.

It is not the only method of static code analysis (it is one of the most prevalent however)

A Bit of History...

Javascript is a language with a lot of rough edges. The first implementation was completed in 10 days for Netscape Navigator.

Initially, Javascript's creator (Brendan Eich) had a totally different language in mind (Scheme) but higher-ups at Netscape wanted a language that looked more like Java.

Over the years, Javascript has accumulated a lot of features that you should probably never use - in the early days, people created unmaintainable, buggy code with these features.

Douglas Crockford - Javascript: The Good Parts (2008)

A Bit of History...

In 2008, Douglas Crockford released *Javascript: The Good Parts*, which detailed the parts of Javascript we should be using, and the parts we should throw away.

Most idiomatic Javascript code can trace its foundations back to this book.

6 years before the release of *Javascript: The Good Parts, Douglas Crockford released JSLint, the first javascript linter.*

JS Linting

Extra benefits of linting in Javascript:

- There is no compiler in JS. Linting allows us to find and fix common mistakes before deploying.
- Javascript has many *bad parts*. Linting helps us ensure we are only using the *good* parts.

ESLint is now the Javascript industry standard.

How it Works

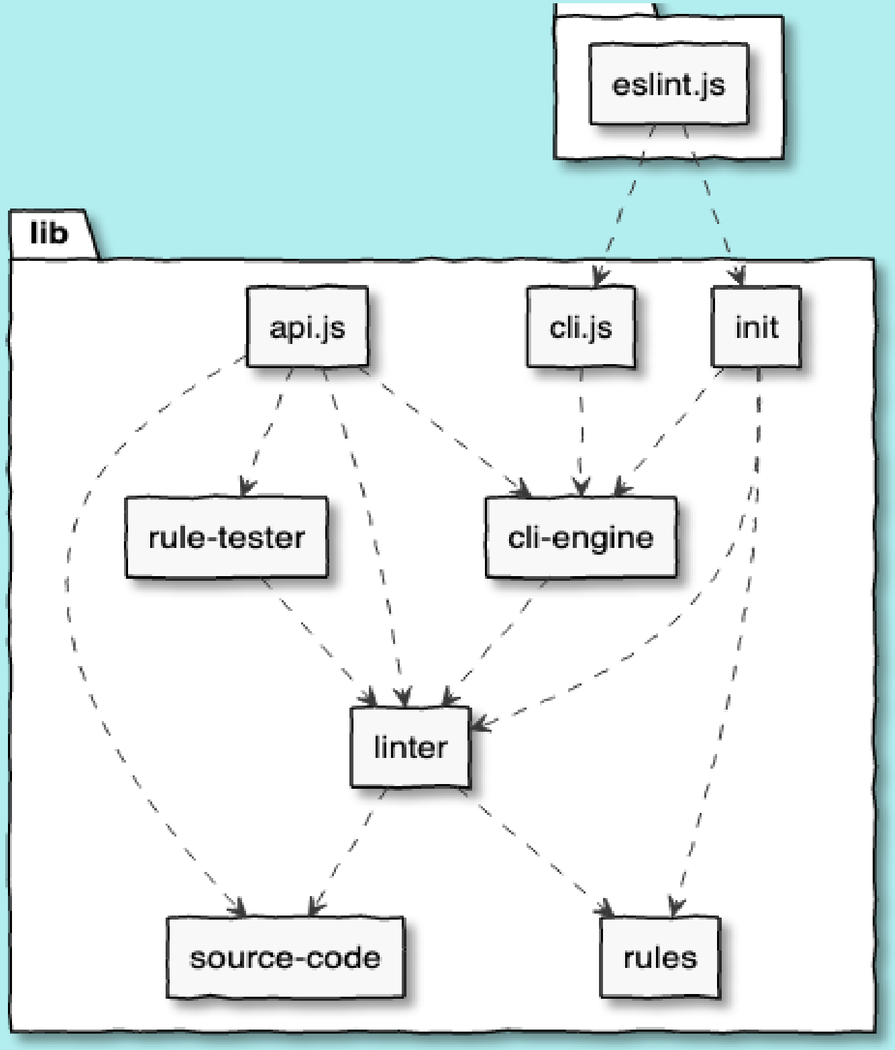
- Runs against your code.
- Has a configuration, which contains a set of rules.
- Detects and reports cases where your code violates one of the rules in the set.

How it Works

- Many rules exist but fall into two general categories:
  - Possible errors
  - Best practices + styling enforcement.
- Rules can be configured with different parameters and can be set to report as a warning or error.
- Runs as a command, or can be set to 'watch' changes in your code. When a change is made on the file system, it will automatically lint results for you.

How it Works (Really)

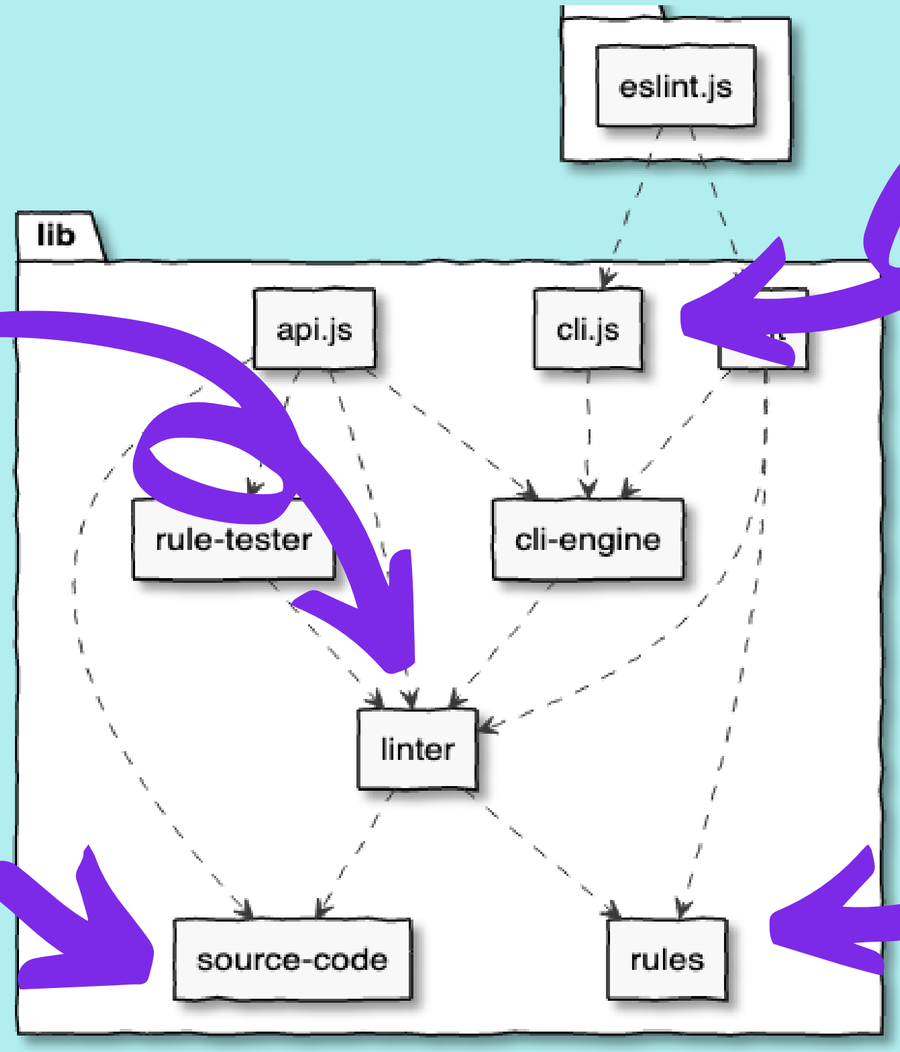
1. The linter is provided code as a raw string.
2. The linter parses the code and generates an AST (*Abstract Syntax Tree*) which represents the program's structure.
3. Each node in the AST represents a block of code, and is appended with the line number and column number the code starts and ends.
4. The linting configuration defines a ruleset, which is a list of rules.
5. Each rule is a function that acts on the AST. It inspects the AST for patterns that 'break' the rule. If it finds any, it reports them.



https://eslint.org/docs/developer-guide/architecture

User interface

Handles the execution of rules



Defines the rules

Represents the parsed source code

https://eslint.org/docs/developer-guide/architecture

Example Lint Rules

No Unreachable: https://eslint.org/docs/rules/no-unreachable
Stops you from writing code that cannot be reached, ie code after a return statement.

No Unused Variables: https://eslint.org/docs/rules/no-unused-vars
Stops you from defining a variable and then not using it.

No Use Before Define: https://eslint.org/docs/rules/no-use-before-define
Stops you from referencing a variable before it is defined.

Example Lint Rules

Indent: https://eslint.org/docs/rules/no-unreachable
Enforces consistent indentation.

No Var: https://eslint.org/docs/rules/no-var
Stops you from using the var keyword, enforces use of let and const.

No Console: https://eslint.org/docs/rules/no-console
Stops you from using methods on console.

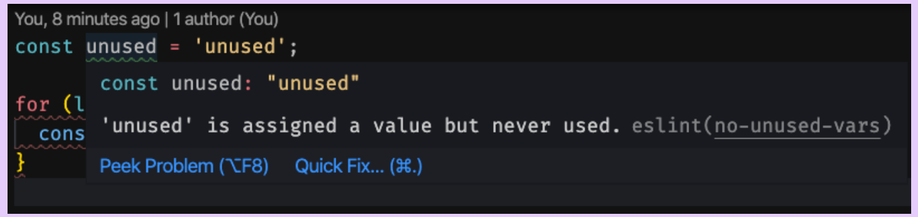
Getting started with linting

Q

How do I get started with linting?

A

You already have! **Create-React-App comes bundled with a pre-configured installation of ESLint.**



Demo

Good Work!