

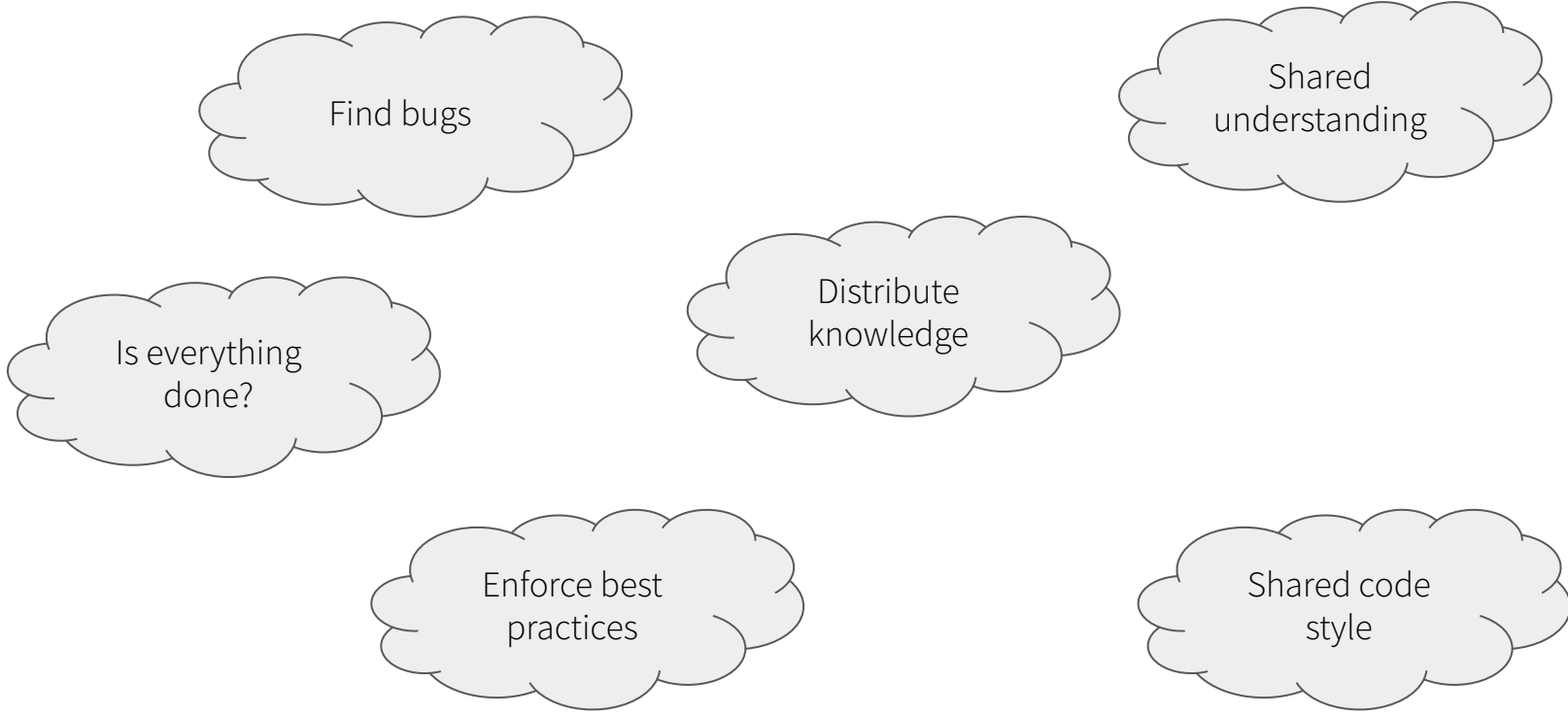
Automating Code Reviews with eslint

Rules, Lesser known rules, Custom rules

Nils Knappmeier



Why are we doing code review?



Find bugs

Shared understanding

Is everything done?

Distribute knowledge

Enforce best practices

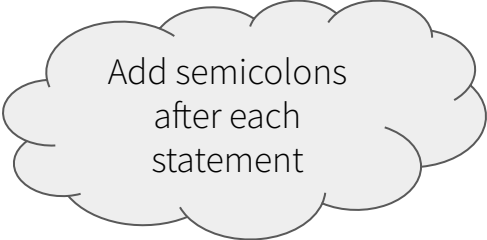
Shared code style

Checklist

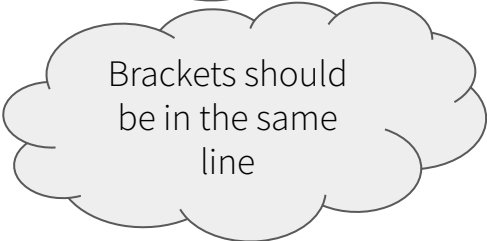
- Does the app work?
- Is the new feature complete?
- Do I understand the code?
- Are there code smells?
- Are there (good) tests?
- Is the code style ok?




Nitpicking




Add semicolons
after each
statement



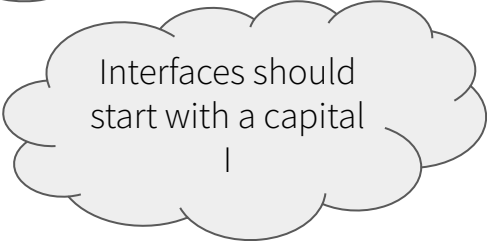
Brackets should
be in the same
line



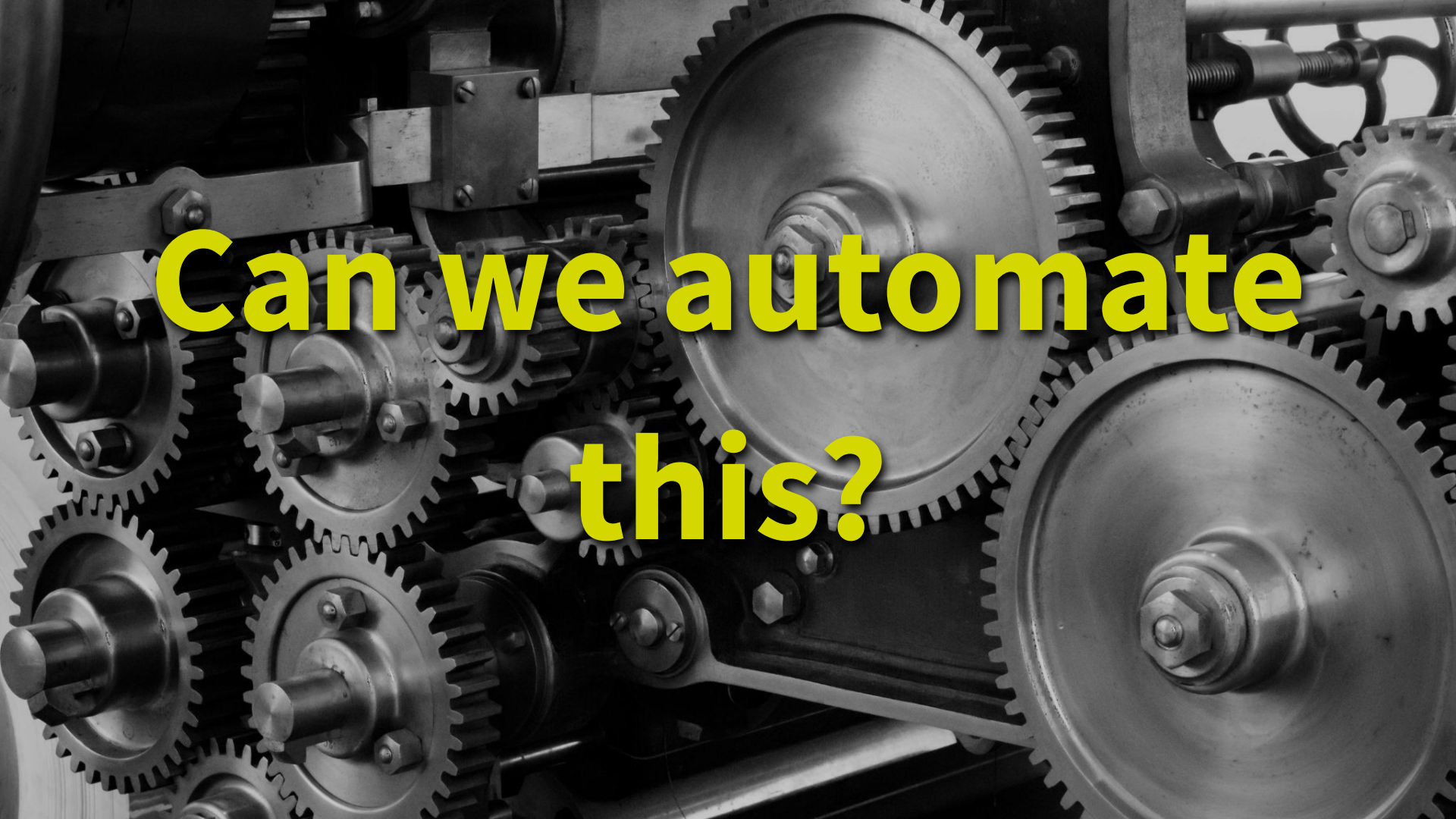
Please try to avoid
default exports



Your imports are
in the wrong
order!



Interfaces should
start with a capital
I



**Can we automate
this?**

Nitpicking



Add semicolons
after each
statement

Brackets should
be in the same
line



Please try to avoid
default exports

Your imports are
in the wrong
order!

Interfaces should
start with a capital
I

Basic setup

`npm init @eslint/config@latest`

- Plugins
- Recommended rules
- Recommended typescript rules
- Recommended rules for framework

```
→ my-vue-app git:(orm-2024) × npm init @eslint/config@latest
```

Flat config

```
import globals from "globals";
import pluginJs from "@eslint/js";
import tseslint from "typescript-eslint";
import pluginVue from "eslint-plugin-vue";
```

```
export default [
  {files: ["**/*.js,mjs,cjs,ts,vue"]}, ← Checked files
  {languageOptions: { globals: globals.browser }},
  pluginJs.configs.recommended,
  ...tseslint.configs.recommended,
  ...pluginVue.configs["flat/essential"], ← Imported config
  {files: ["**/*.vue"], languageOptions: {parserOptions: {parser: tseslint.parser}}}, ← Config for some files
];
```


Useful plugins

- `eslint-config-prettier`
disable styling rules
- `eslint-plugin-import`
(does not support eslint 9 yet)
lots of rules related to imports
- `eslint-plugin-promise`
rules related to promises

A plugin usually has “recommended” configs

```
import pluginVue from "eslint-plugin-vue";
import promisePlugin from "eslint-plugin-promise";
import eslintConfigPrettier from "eslint-config-prettier";

export default [
  { files: ["**/*.js,mjs,cjs,ts,vue"] },
  { languageOptions: { globals: globals.browser } },
  pluginJs.configs.recommended,
  ...tseslint.configs.recommended,
  ...promisePlugin.configs.recommended,
  eslintConfigPrettier,
  ...pluginVue.configs["flat/essential"],
  { files: ["**/*.vue"], languageOptions: { parserOptions: { pa
  ];
```

Adding/disabling rules

```
export default [  
  { files: ["**/*.js,mjs,cjs,ts,vue"] },  
  { languageOptions: { globals: globals.browser } },  
  pluginJs.configs.recommended,  
  ...tseslint.configs.recommended,  
  promisePlugin.configs["flat/recommended"],  
  eslintConfigPrettier,  
  ...pluginVue.configs["flat/essential"],  
  { files: ["**/*.vue"], languageOptions: { parserOptions: { parser: tseslint.parser } } },  
  {  
    rules: {  
      eqeqeq: ["error", "smart"],  
    },  
  },  
];
```

Rule **Severity** **Rule config**

no-restricted-*

- no-restricted-imports

```
import { debug } from "vitest-preview";
```

- no-restricted-properties

```
screen.debug()
```

no-restricted-syntax



```
export async function incorrect() {  
  const { largeModule } = await import("./large-module");  
  return largeModule();  
}
```

```
export async function correct() {  
  const { largeModule } = await retry(() => import("./large-module"));  
  return largeModule();  
}
```

How does eslint actually work? Parsing an AST.

AST Explorer



Snippet



JavaScript



@typescript-eslint/parser



Transform



default



Parser: [@typescript-eslint/parser-5.59.7](https://github.com/typescript-eslint/parser)

```
1 import {retry} from "@dynamic-import-wrap/retry.ts";
2
3 export async function someAsyncFunction() {
4   const { largeModule } = await retry(() => import("./large-module"));
5   return largeModule()
6 }
7
```

Tree

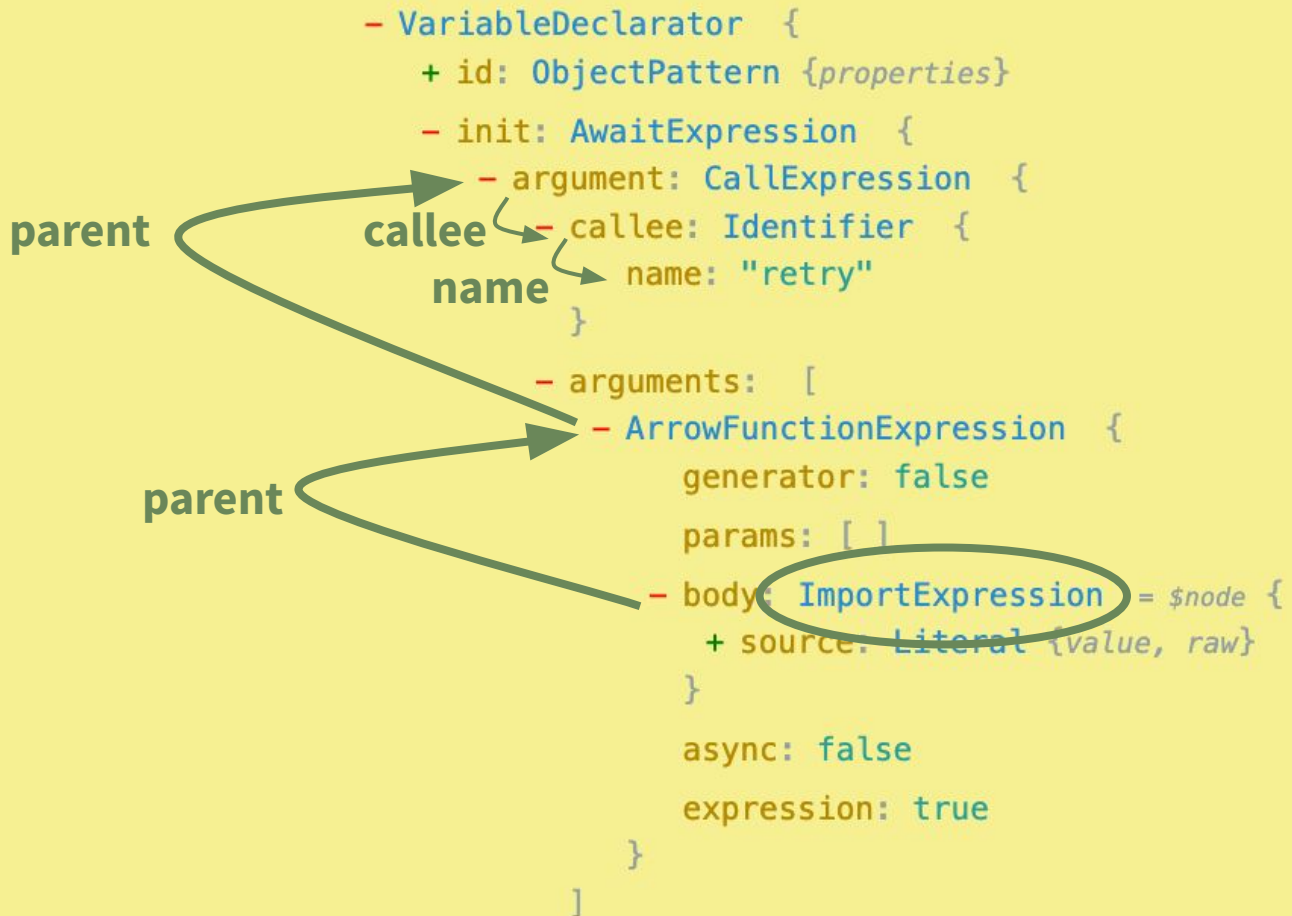
JSON

7ms

☒ Autofocus ☒ Hide methods ☒ Hide empty keys ☒ Hide location data
☐ Hide type keys

```
- ExportNamedDeclaration {
  type: "ExportNamedDeclaration"
- declaration: FunctionDeclaration {
  type: "FunctionDeclaration"
+ id: Identifier {type, name}
  generator: false
  expression: false
  async: true
  params: [ ]
- body: BlockStatement {
  type: "BlockStatement"
- body: [
-   VariableDeclaration {
    type: "VariableDeclaration"
-   declarations: [
-     VariableDeclarator {
      type: "VariableDeclarator"
+ id: ObjectPattern {type, properties}
-   init: AwaitExpression {
```

Selectors



`ImportExpression: not([parent.parent.callee.name='retry'])`

Configuring “no-restricted-syntax”

```
"no-restricted-syntax": [  
  "off",  
  {  
    selector: "ImportExpression:not([parent.parent.callee.name='retry'])",  
    message: "Import expressions should be wrapped in a retry function",  
  },  
],
```

Local Plugin

```
export const customRules = {
  rules: {
    "retry-dynamic-imports": {
      create(context) {
        return {
          "ImportExpression:not([parent.parent.callee.name='retry'])"(node) {
            context.report({
              node,
              message:
                "Import expressions should be wrapped in a retry function",
            });
          },
        };
      },
    },
  },
};
```

Selector



Adding custom rule to eslint

Local plugin namespace

Local plugin

```
import { customRules } from "../eslint-rules/index.js";
```

```
{  
  plugins: { customRules: customRules },  
  rules: {  
    "customRules/retry-dynamic-imports": "error",  
  },  
},
```



Autofix



```
"retry-dynamic-imports": {
  meta: {
    fixable: "code",
  },
  create(context) {
    const sourceCode = context.getSourceCode();
    return {
      "ImportExpression:not([parent.parent.callee.name='retry'])"(node) {
        context.report({
          node,
          message:
            "Import expressions should be wrapped in a retry function",
          fix(fixer) {
            let text = sourceCode.getText(node);
            return fixer.replaceText(node, `retry(() => ${text})`);
          },
        });
      },
    };
  },
},
```

Conclusion

- Easy to setup
- Many plugins
- Extendable

*The ESLint was
made for devs,
not devs for the
ESLint.*

Mark 2:27

**Thank you
and have fun
with ESLint!**



<https://github.com/nknapp/frontend-testing/tree/orm2024>

<https://eslint.org/docs/latest/rules/>

<https://eslint.org/docs/latest/extend/custom-rules>

<https://eslint.org/docs/latest/extend/selectors>

<https://astexplorer.net/>