ТИНЬКОФФ

Статические анализаторы кода

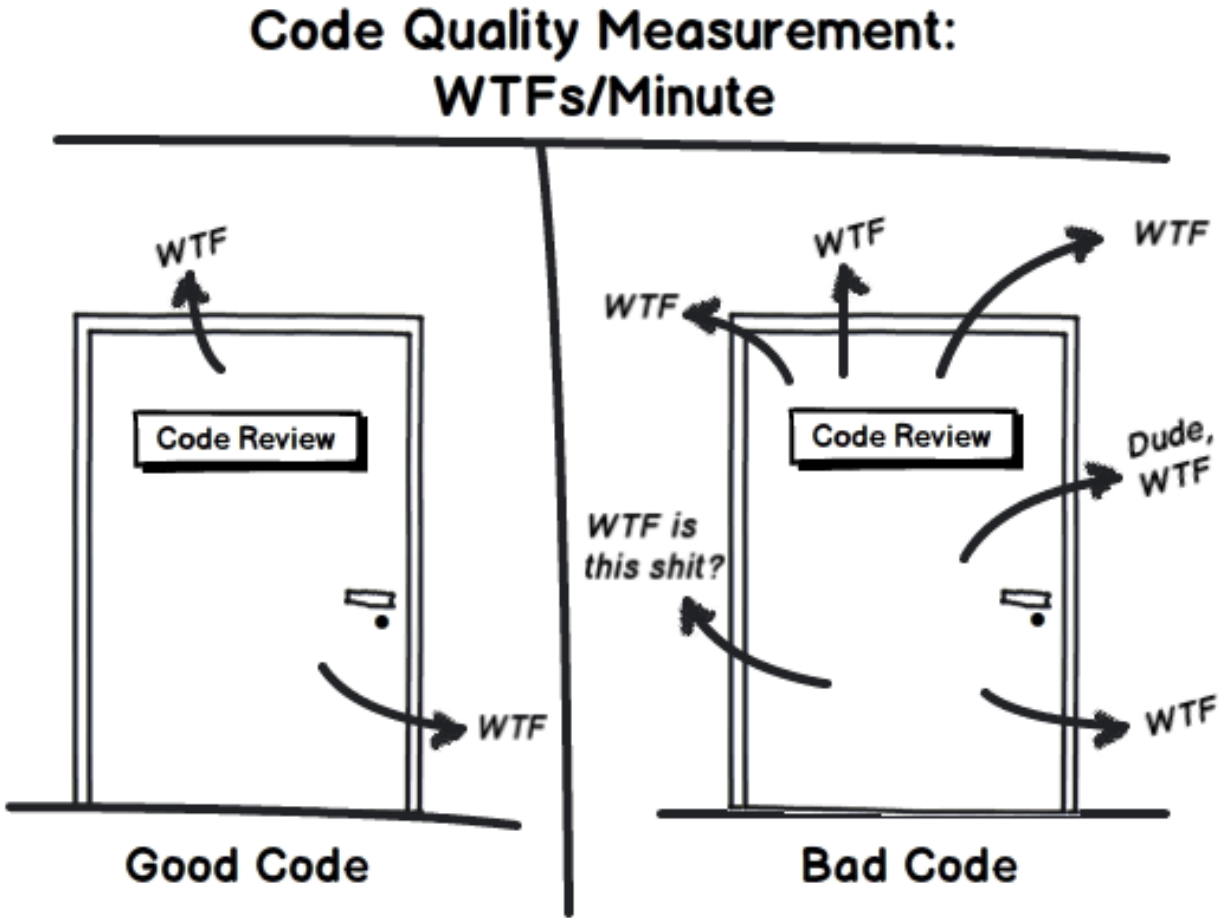
и инструменты автоматизации проверок

Сидоров Егор

06.03.2023

Программист VS Разработчик

Качество кода



Конвенция

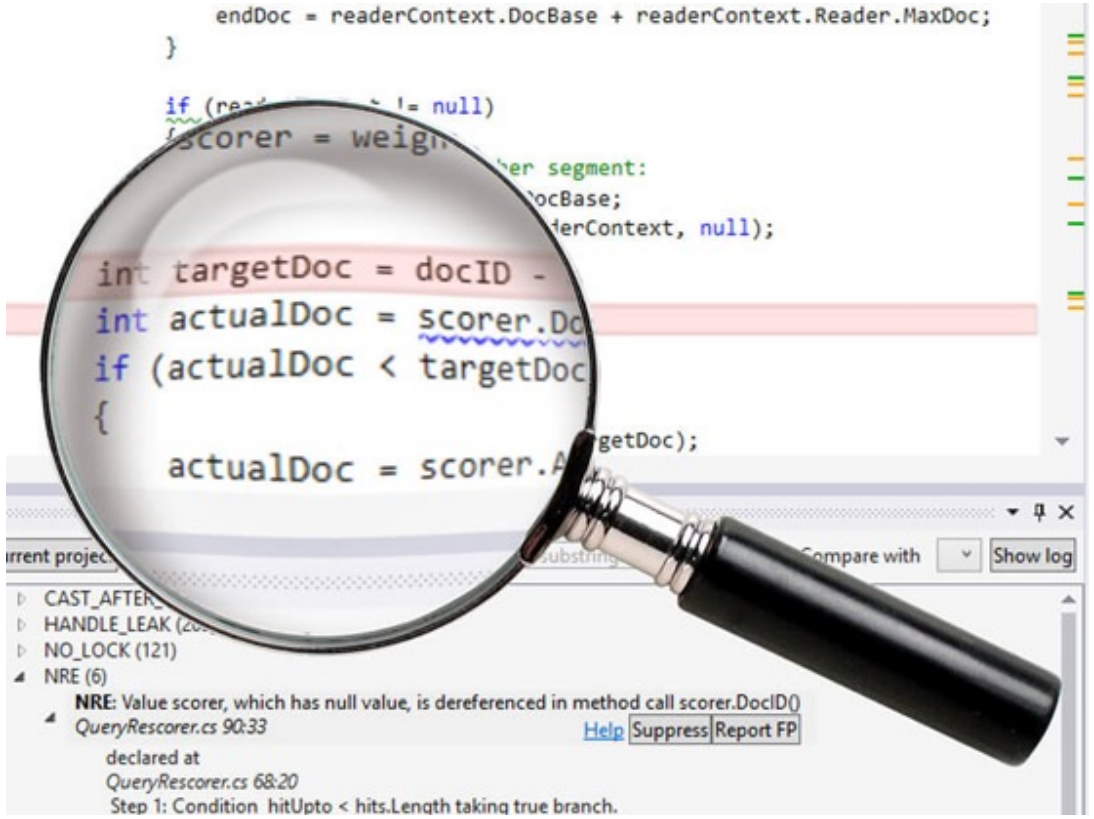
Конвенция (лат. conventio – договор, соглашение) - многостороннее соглашения о поддержке некоторых стандартов, каждым из участников.

В рамках разработки такое соглашение обычно называют **code style**.

Code style

Неужели code style необходимо заучивать?

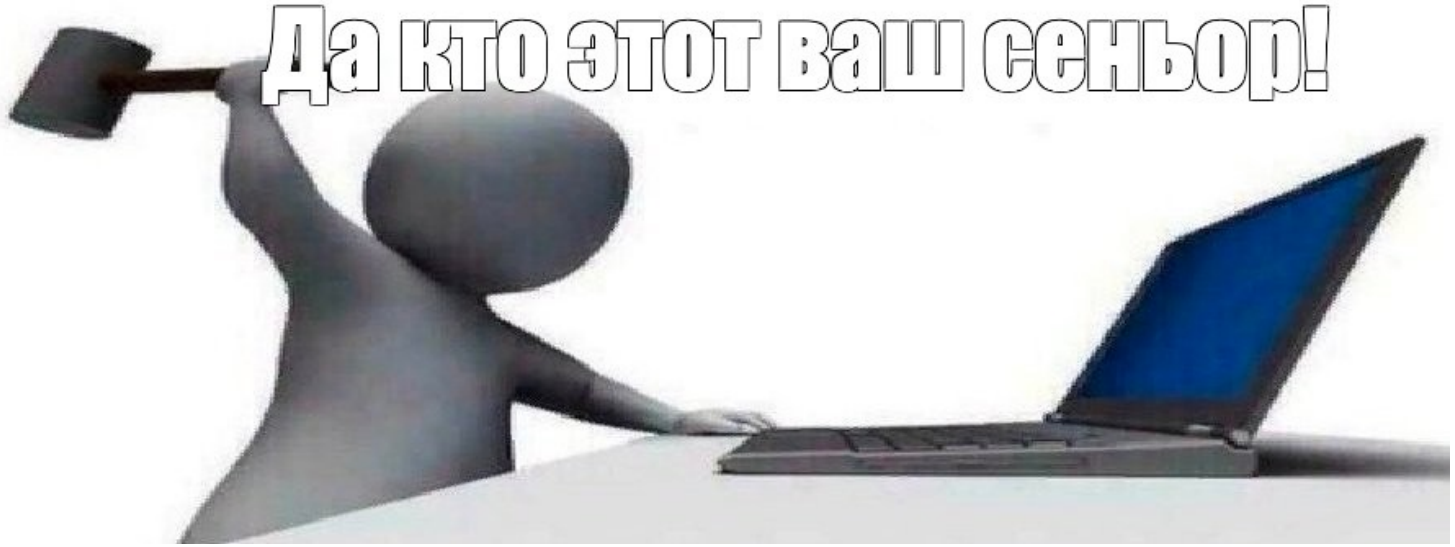
Статический анализ кода

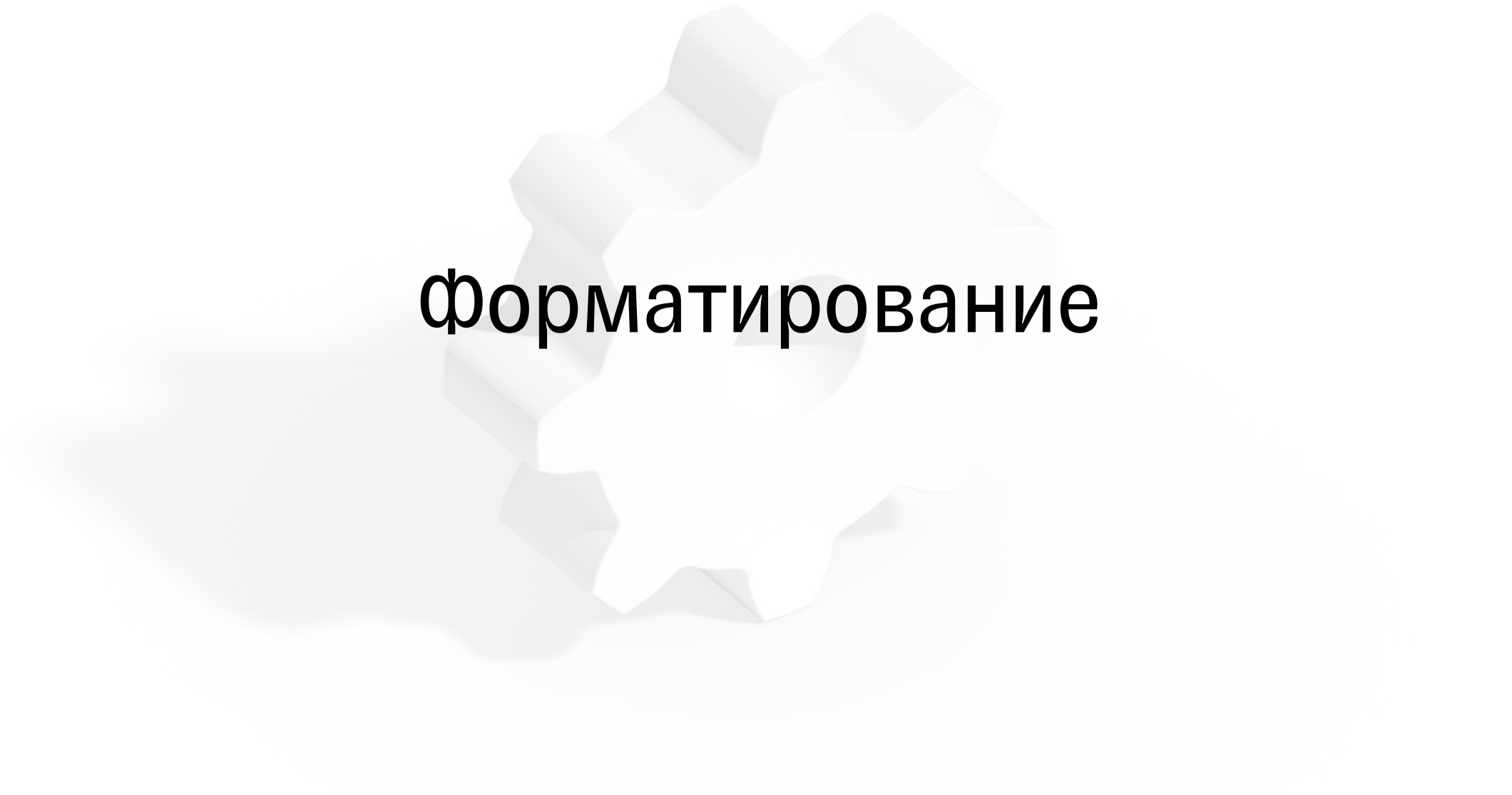


Статический анализ кода — анализ программного обеспечения, производимый без реального выполнения исследуемых программ.

Code review

Code review



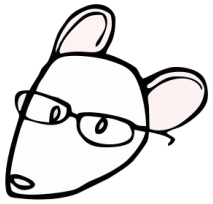


Форматирование

Форматирование

Код-форматтер - это утилита выполняющая анализ кода и его форматирование на основе установленных правил

форматирование



EditorConfig

```
# Unix-style newlines with a newline ending every file
[*]
end_of_line = lf
insert_final_newline = true

# Matches multiple files with brace expansion notation
# Set default charset
[*.{js,py}]
charset = utf-8

# 4 space indentation
[*.py]
indent_style = space
indent_size = 4

# Tab indentation (no size specified)
[Makefile]
indent_style = tab

# Indentation override for all JS under lib directory
[lib/**.js]
indent_style = space
indent_size = 2

# Matches the exact files either package.json or .travis.yml
[{package.json,.travis.yml}]
indent_style = space
indent_size = 2
```

Форматирование

Prettier

```
// prettier.config.js or .prettierrc.js
module.exports = {
  trailingComma: "es5",
  tabWidth: 4,
  semi: false,
  singleQuote: true,
};
```

Prettier

```
((count, a, b) => { new Array(count).fill(1).map((_,
number) => console.log(!(number % 3) && !(number % 5) ? (
a + b) : !(number % 3) ? a : !(number % 5) ? b : number
)) })(100, "Foo", "Bar")
```

Prettier

```
((count, a, b) => { new Array(count).fill(1).map((_,
number) => console.log(!(number % 3) && !(number % 5) ? (
a + b) : !(number % 3) ? a : !(number % 5) ? b : number
)) })(100, "Foo", "Bar")
```

```
((count, a, b) => {
  new Array(count)
    .fill(1)
    .map((_, number) =>
      console.log(
        !(number % 3) && !(number % 5)
          ? a + b
          : !(number % 3)
            ? a
            : !(number % 5)
              ? b
              : number
      )
    );
})(100, "Foo", "Bar");
```

Prettier

```
  $("#speedPercent").on("input", event ⇒ {
$(".output").value =      event.target.value + "%"    ;
})

$("#grid").on("click", event ⇒ {
if (event.target && event.target.matches("button.banana")) {

    const points =     parseInt(    event.target.dataset.points, 10     )  ;
state = {        ...state, score: state.score + points };
    setScoreInnerHTML(state);

const span    =       event.target.querySelector("span");
    span.classList.add("exit-animation");
span.on('animationend', ()   ⇒    {
event.target.parentNode.removeChild(event.target)})
    }
  })
```

```
$("#speedPercent").on("input", (event) ⇒ {
  $(".output").value = event.target.value + "%";
});

$("#grid").on("click", (event) ⇒ {
  if (event.target && event.target.matches("button.banana")) {
    const points = parseInt(event.target.dataset.points, 10);
    state = { ...state, score: state.score + points };
    setScoreInnerHTML(state);

    const span = event.target.querySelector("span");
    span.classList.add("exit-animation");
    span.on("animationend", () ⇒ {
      event.target.parentNode.removeChild(event.target);
    });
  }
});
```

Prettier

```
module.exports = {
    ...require('@tinkoff/prettier-config'),
    editorconfig: true,
}
```

Статический анализ кода

```
((count, a, b) ⇒ {
  new Array(count)
    .fill(1)
    .map((_, number) ⇒
      console.log(
        !(number % 3) && !(number % 5)
          ? a + b
          : !(number % 3)
            ? a
            : !(number % 5)
              ? b
              : number
      )
    );
})(100, "Foo", "Bar");
```

Анализ кода

Анализ кода

Линтер (от англ. слова lint) – статический анализатор кода, который указывает на “подозрительные” участки программы и тем самым помогает программисту писать более качественный код.

Анализ кода


```
{
    "root": true,
    "extends": [
        "eslint:recommended"
    ],
    "parserOptions": {
        "ecmaVersion": "latest",
        "sourceType": "module"
    },
    "rules": {
        "no-unused-vars": "off",
        "no-magic-numbers": "warn",
        "no-shadow": "error",
        "indent": ["error", "tab"],
        "comma-dangle": ["error", "always-multiline"],
        "quotes": ["error", "single"],
        "no-nested-ternary": "error",
        "no-undef": "off",
        "no-restricted-syntax": [
            "error",
            {
                "selector": "FunctionExpression",
                "message": "Function expressions are not allowed."
            },
            {
                "selector": "CallExpression[callee.name='setTimeout'][arguments.length!=2]",
                "message": "setTimeout must always be invoked with two arguments."
            },
            {
                "selector": "CallExpression[callee.name!='parseInt'] > Identifier[name='parseInt']",
                "message": "Call parseInt directly to guarantee radix param is not incorrectly provided"
            }
        ]
    }
}
```

Анализ кода


```
{
    "root": true,
    "plugins": [
        "@tinkoff/eslint-plugin"
    ],
    "extends": [
        "eslint:recommended",
        "@tinkoff/eslint-config/app"
    ],
    "rules": {}
}
```


```
const value = ((count, a, b) => {
    new Array(count)
        .fill(1)
        .map((_, number) =>
            console.log(
                !(number % 3) && ! (number % 5)
                    ? a + b
                    : !(number % 3)
                    ? a
                    : !(number % 5)
                    ? b
                    : number
            )
        )
})(100, "Foo", "Bar");
```

```
  3:1   error  Expected indentation of 1 tab but found 4 spaces    indent
  4:1   error  Expected indentation of 2 tabs but found 8 spaces   indent
  5:1   error  Expected indentation of 2 tabs but found 8 spaces   indent
  6:1   error  Expected indentation of 3 tabs but found 12 spaces  indent
  7:1   error  Expected indentation of 4 tabs but found 16 spaces  indent
  7:17  error  Do not nest ternary expressions                     no-nested-ternary
  8:3   error  Mixed spaces and tabs                               no-mixed-spaces-and-tabs
  9:1   error  Expected indentation of 5 tabs but found 20 spaces  indent
  9:23  error  Do not nest ternary expressions                     no-nested-ternary
 10:1   error  Expected indentation of 6 tabs but found 24 spaces  indent
 11:1   error  Expected indentation of 6 tabs but found 24 spaces  indent
 12:1   error  Expected indentation of 7 tabs but found 28 spaces  indent
 13:1   error  Expected indentation of 7 tabs but found 28 spaces  indent
 13:37  error  Missing trailing comma                              comma-dangle
 14:6   error  Mixed spaces and tabs                               no-mixed-spaces-and-tabs
 14:10  error  Missing trailing comma                              comma-dangle
 15:1   error  Expected indentation of 2 tabs but found 8 spaces   indent
 16:9   error  Strings must use singlequote                        quotes
 16:16  error  Strings must use singlequote                        quotes

✖ 19 problems (19 errors, 0 warnings)
  15 errors and 0 warnings potentially fixable with the `--fix` option.
```


```
const value = ((count, a, b) => {
    new Array(count).fill(1).map((_, number) => {
        if (!(number % 3) && !(number % 5)) {
            console.log(a + b);
        } else if (!(number % 3)) {
            console.log(a);
        } else if (!(number % 5)) {
            console.log(b);
        } else {
            console.log(number);
        }
    })
})(100, 'Foo', 'Bar');
```


```
const value = (
    (count, a, b) => Array
        .from({length: count})
        .forEach((_, number) => {
            const multipleOfThree = number % 3;
            const multipleOfFive = number % 5;

            if (!multipleOfThree && !multipleOfFive) {
                console.log(a + b);

                return;
            }

            if (!multipleOfThree) {
                console.log(a);

                return;
            }

            if (!multipleOfFive) {
                console.log(b);

                return;
            }

            console.log(number);
        })
)(100, 'Foo', 'Bar');
```


```
const array = new Array(1, 2, 3);
/**
 * no-array-constructor:
 * The array literal notation [] is preferable
 */
```


```
const array = new Array(1, 2, 3);
/**
 * no-array-constructor:
 * The array literal notation [] is preferable
 */
```

```
const arrayCorrect = [1, 2, 3];
```


```
const numbers = ['1', '2', '3'].map(parseInt);
/**
 * no-restricted-syntax:
 * Call parseInt directly to guarantee radix param is not incorrec
tly provided
 */

const numbersByExpression = ['1', '2', '3'].map(function (value) {
  return parseInt(value)
});
/**
 * no-restricted-syntax:
 * Function expressions are not allowed
 */
```


```
const numbers = ['1', '2', '3'].map(parseInt);
/**
 * no-restricted-syntax:
 * Call parseInt directly to guarantee radix param is not incorrec
tly provided
 */

const numbersByExpression = ['1', '2', '3'].map(function (value) {
  return parseInt(value)
});
/**
 * no-restricted-syntax:
 * Function expressions are not allowed
 */
```

```
const numbers = ['1', '2', '3'].map((value) => parseInt(value, 10));
```


```
eval('var temp = 1 + 3');
/**
 * no-implied-eval:
 * Implied eval. Consider passing a function instead of a string.
 */

setTimeout('alert("Hi!");');
/**
 * no-implied-eval:
 * Implied eval. Consider passing a function instead of a string.
 */
/**
 * no-restricted-syntax:
 * setTimeout must always be invoked with two arguments.
 */
```


```
const useMagicNumber = width * 0.5625;
/**
 * no-magic-numbers:
 * No magic number: 0.5625.
 */
```


```
const useMagicNumber = width * 0.5625;
/**
 * no-magic-numbers:
 * No magic number: 0.5625.
 */
```

```
const displayAspectRatio = 0.5625;
const useMagicNumber = width * displayAspectRatio;
```


```
const foo = i > 5 ? true : i <100 ? true : i < 1000 ? true : false;
/**
 * no-nested-ternary:
 * Do not nest ternary expressions.
 */
```


```
const foo = i > 5 ? true : i <100 ? true : i < 1000 ? true : false;
/**
 * no-nested-ternary:
 * Do not nest ternary expressions.
 */
```

```
const getFoo = () => {
  const first = i > 5;

  if (first) {
    return true;
  }

  const second = i < 100;

  if (second) {
    return true;
  }

  return i < 1000;
};

const foo = getFoo();
```


```
const variable = 'value';

function bar() {
  const variable = 'shadow value';

  return variable;
}
/**
 * no-shadow:
 * 'variable' is already declared in the upper scope on line 1 column 7.
 */
```


```
const variable = 'value';

function bar() {
  const variable = 'shadow value';

  return variable;
}
/**
 * no-shadow:
 * 'variable' is already declared in the upper scope on line 1 column 7.
 */
```

```
const variableGlobal = 'value';

function bar() {
  const variable = 'shadow value';

  return variable;
}
```

Анализ кода


```
{
  "extends": "stylelint-config-standard",
  "rules": {
    "color-no-invalid-hex": true,
    "block-no-empty": null,
    "unit-allowed-list": ["em", "rem", "%", "s"],
    "alpha-value-notation": ["percentage", { "exceptProperties": ["opacity"] }]
  }
}
```

Анализ кода

stylelint

```
body {
  margin: 0;
  padding: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
  "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
  sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}

code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
  monospace;
}

.pageContent {
  margin-left: 350px;
}

* {
  box-sizing: border-box;
}
body {
  min-height: 720px;
  max-width: 1440px;
  background-color: #1C1C1C;
  margin: 80px auto 0px auto;
}
```

Анализ кода


```
function isEmptyString(value?: string): boolean {
  return value ≠ null && value.length > 0;
}
```

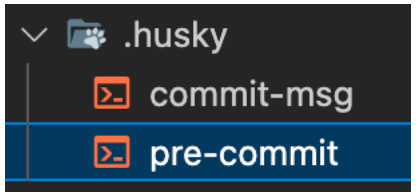
Анализ кода


```
{
    "root": true,
    "plugins": [
        "@typescript-eslint"
    ],
    "extends": [
        "plugin:@typescript-eslint/recommended",
        "eslint:recommended"
    ],
    "parser": "@typescript-eslint/parser",
    "rules": {}
}
```


Автоматизация

Автоматизация

Husky – инструмент для упрощения работы с git-хуками. Позволяет задать некоторые скрипты на различные события git в рамках конкретного проекта.




```
#!/usr/bin/env sh
. "$(dirname -- "$0")/_/husky.sh"

npx prettier --write .
npx eslint .
```

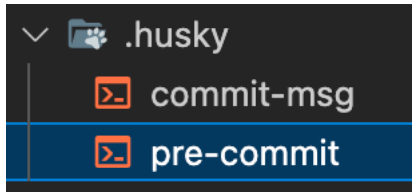
Автоматизация

Lint-staged – данный инструмент позволяет запускать некоторые скрипты только на тех файлах, что попадают в коммит.

```
{
    "*.{ts,js}": ["npx prettier --write", "git add", "npx eslint"],
    "*.html": ["npx prettier --write", "git add"],
    "*.css": ["npx npx stylelint"]
}
```

Автоматизация

Мощная комбинация для автоматизации




```
#!/usr/bin/env sh
. "$(dirname -- "$0")/_/husky.sh"

npx lint-staged
```

Автоматизация

Lefthook – другой инструмент подписки на git-хуки. Решает те же задачи, но самостоятельно. Кроме того, позволяет выполнять команды параллельно.

```yaml
pre-push:
  parallel: true
  commands:
    spelling:
      files: git diff --name-only HEAD @{push}
      glob: "*.md"
      run: npx yaspeller {files}
    check-links:
      files: git diff --name-only HEAD @{push}
      glob: "*.md"
      run: npx markdown-link-check {files}
```

Анализ коммитов

Анализ коммитов

Conventional Commits - соглашение о том, как разработчикам поддерживать историю разработки проекта в читаемом формате.

```
type(scope?): subject
body?
footer?
```

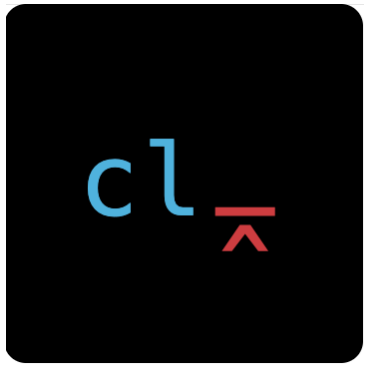
```
fix: prevent racing of requests

Introduce a request id and a reference to latest request.
Dismiss
incoming responses other than from latest request.

Remove timeouts which were used to mitigate the racing is
sue but are
obsolete now.

Reviewed-by: Z
Refs: #123
```

Анализ коммитов

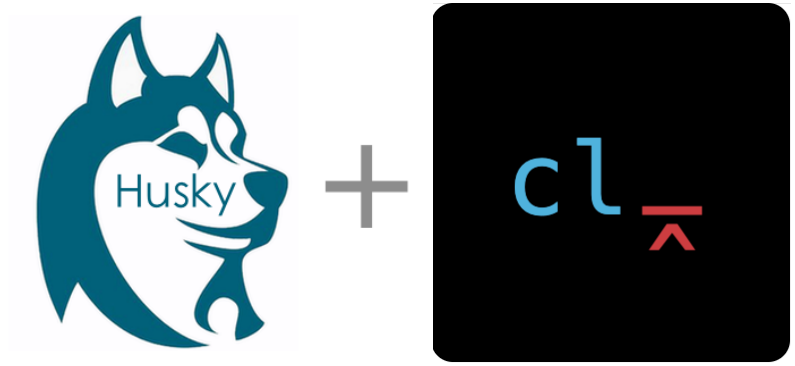


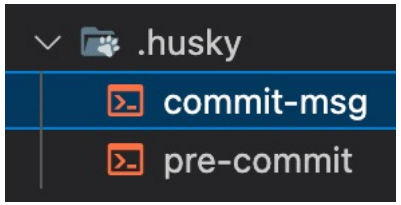
Commitlint – инструмент для проверки коммитов на соответствие соглашению.

```
module.exports = {
    extends: ['@commitlint/config-conventional'],
};
```

Автоматизация

Мощная комбинация для автоматизации






```
#!/usr/bin/env sh
. "$(dirname -- "$0")/_/husky.sh"

npx commitlint --edit ${1}
```


End

ТИНЬКОФФ

Code style не высечены в камне.

Ваша команда растёт и в неё приходят новые люди, они могу принести с собой иные взгляды на код. Возможно потребуется внести новые правила или изменить старые.

Не стоит ревностно защищать свой свод правил, если кто-то иной может **аргументированно** объяснить, почему то или иное правило нужно добавить, изменить или удалить.

Ссылки

1. Editorconfig
2. Prettier
3. Eslint
   - список правил
   - список плагинов
4. Stylelint
   - список правил
   - список плагинов
5. Husky
6. lint-staged
7. Typescript
8. Conventional Commits
9. Почему нужно использовать Conventional Commits
10. Commitlint
11. Semantic Versioning