



Рефакторинг за 10 секунд

Александр Мирошниченко,
фронтенд-разработчик



О себе

- Работаю в платформе фронтенда
- Получаю магистерскую степень аналитика данных
- Это мой первый публичный доклад



О чём будем разговаривать

- Как часто мы в компании сталкиваемся с рефакторингом

О чём будем разговаривать

- Как часто мы в компании сталкиваемся с рефакторингом
- Как дорого он нам обходится

О чём будем разговаривать

- Как часто мы в компании сталкиваемся с рефакторингом
- Как дорого он нам обходится
- Какие инструменты помогли нам упростить обновление кодовой базы

Что для нас рефакторинг?

1. Обновление дизайн-системы (каждый квартал)

AM I TESTING THIS PROGRAM



OR IS IT TESTING ME?

Что для нас рефакторинг?

1. Обновление дизайн-системы (каждый квартал)
2. Миграция со старых версий библиотек на новые

AM I TESTING THIS PROGRAM

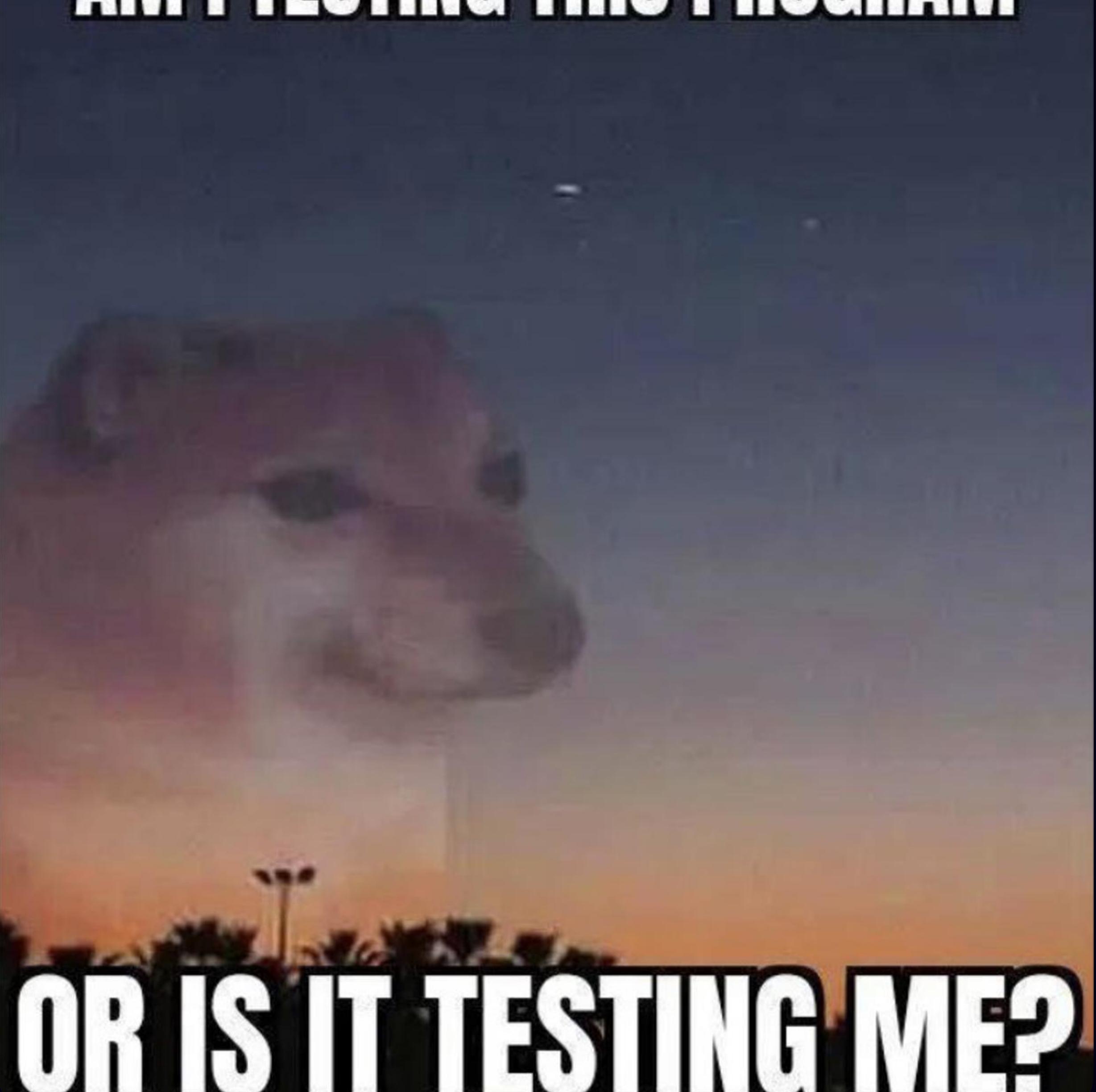


OR IS IT TESTING ME?

Что для нас рефакторинг?

1. Обновление дизайн-системы (каждый квартал)
2. Миграция со старых версий библиотек на новые
3. Внедрение новых инструментов

AM I TESTING THIS PROGRAM



OR IS IT TESTING ME?

Наши масштабы

>1 млн

строк актуального кода —
его поддерживают 30 разработчиков

80+

приложений и библиотек

Почему ручной рефакторинг — это дорого

- Много проектов



Почему ручной рефакторинг — это дорого

- Много проектов
- Большая нагрузка на продуктовые команды



Почему ручной рефакторинг — это дорого

- Много проектов
- Большая нагрузка на продуктовые команды
- Нет автоматизации



Решение есть!



Кодмоды

Утилита, изменяющая исходный код проекта
по описанным разработчиком правилам

Что мы сделали с помощью кодмодов

- Заменили дизайн-токены с JS-переменных на CSS во всех актуальных проектах

Что мы сделали с помощью кодмодов

- Заменили дизайн-токены с JS-переменных на CSS во всех актуальных проектах
- Заменили использование легаси-компонентов нашей дизайн-системы на новые

Что мы сделали с помощью кодмодов

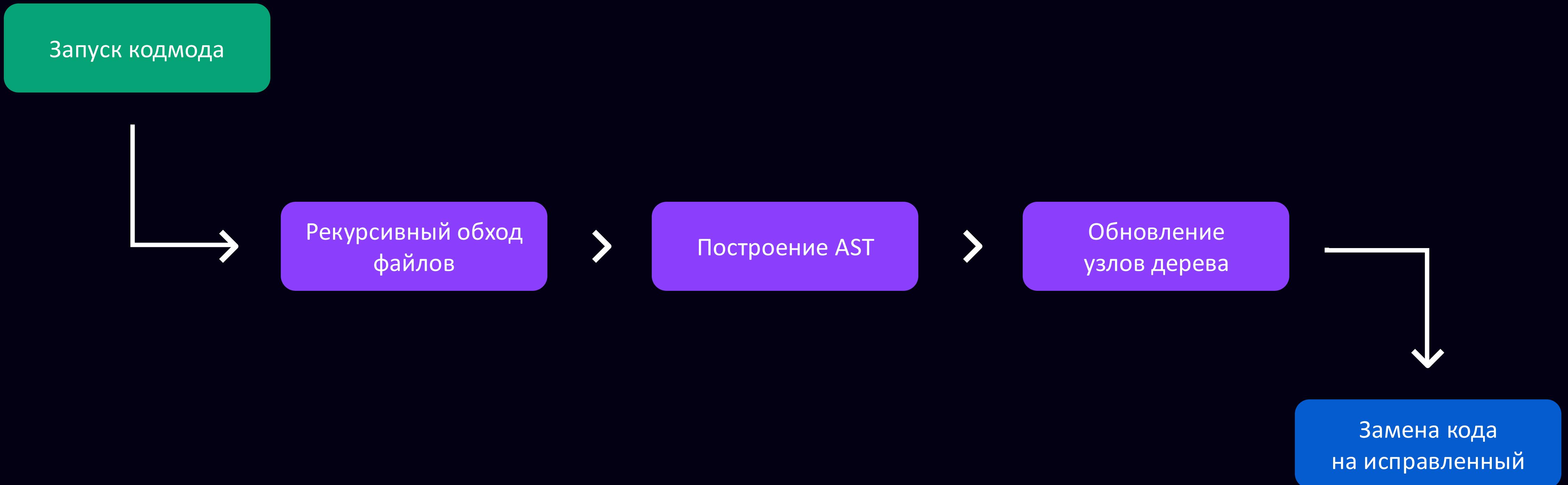
- Заменили дизайн-токены с JS-переменных на CSS во всех актуальных проектах
- Заменили использование легаси-компонентов нашей дизайн-системы на новые
- Автоматизировали актуализацию API-компонентов дизайн-системы при выпуске новых мажоров

Что мы сделали с помощью кодмодов

- Заменили дизайн-токены с JS-переменных на CSS во всех актуальных проектах
- Заменили использование легаси-компонентов нашей дизайн-системы на новые
- Автоматизировали актуализацию API-компонентов дизайн-системы при выпуске новых мажоров
- Обновили 100+ файлов-документаций в нашей альтернативе Storybook

Процесс работы кодмодов

■ Юзер ■ Ваш проект ■ Кодмод

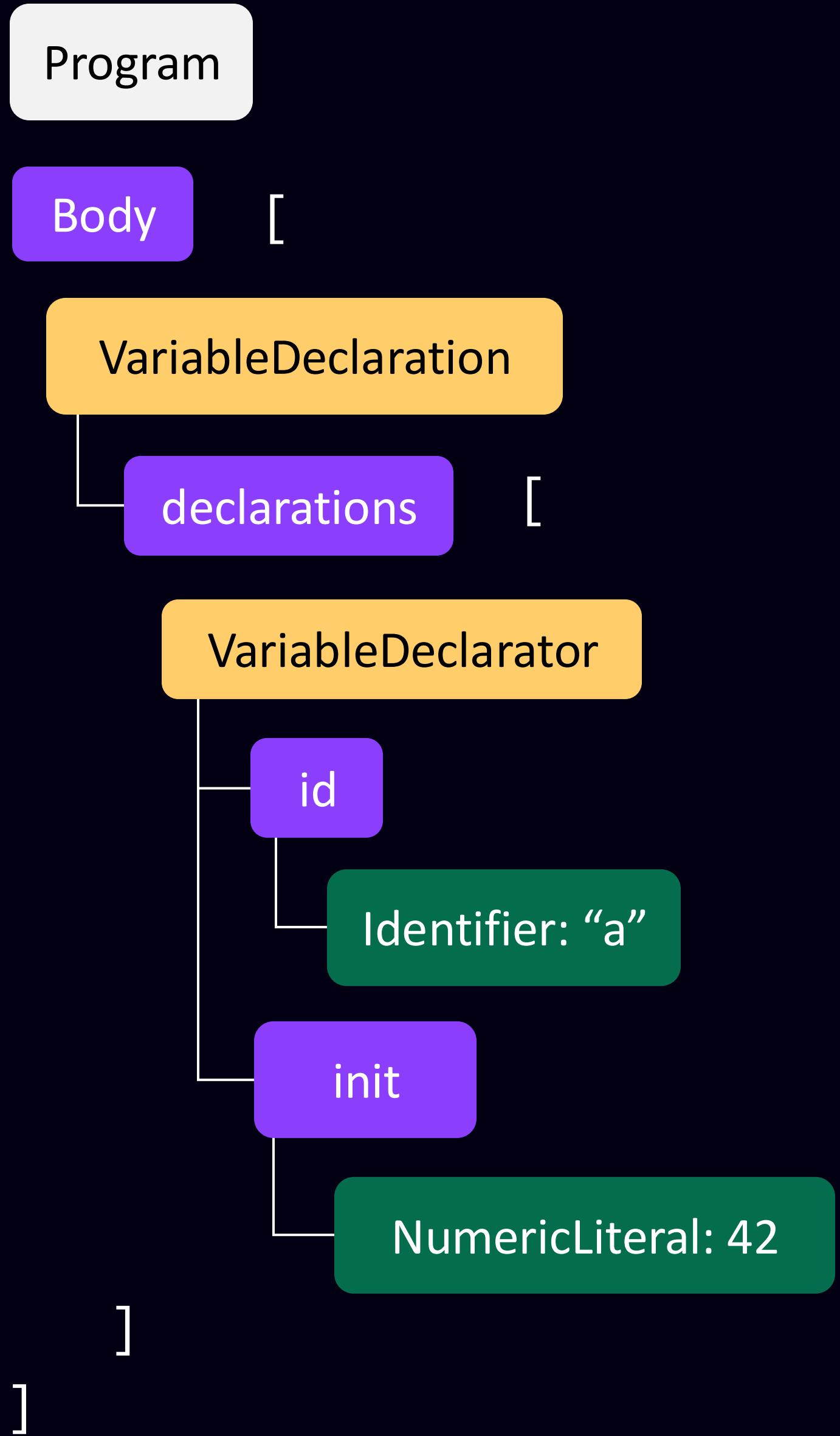


Abstract Syntax Tree

Древовидное представление исходного кода



const a = 42



Спецификация Babel AST

Какие решения
есть на рынке?

@codemod/core

- Использует экосистему **babel**
- Babel AST



BABEL

@codemod/core

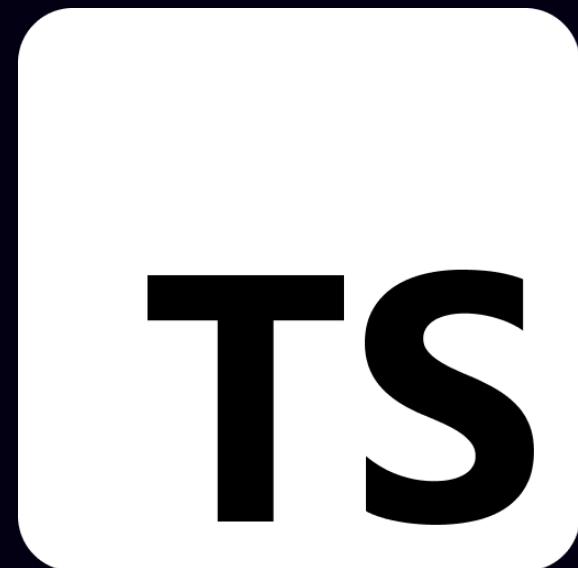
- Использует экосистему **babel**
- Babel AST

The Babel logo is displayed in a white, hand-drawn style font within a rounded rectangular frame.

BABEL

ts-morph

- Удобный поиск референсов файла в проекте
- Обертка над Typescript AST



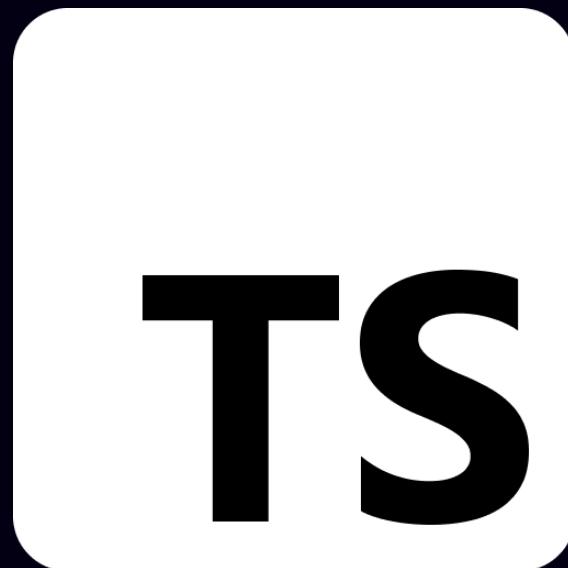
@codemod/core

- Использует экосистему **babel**
- Babel AST



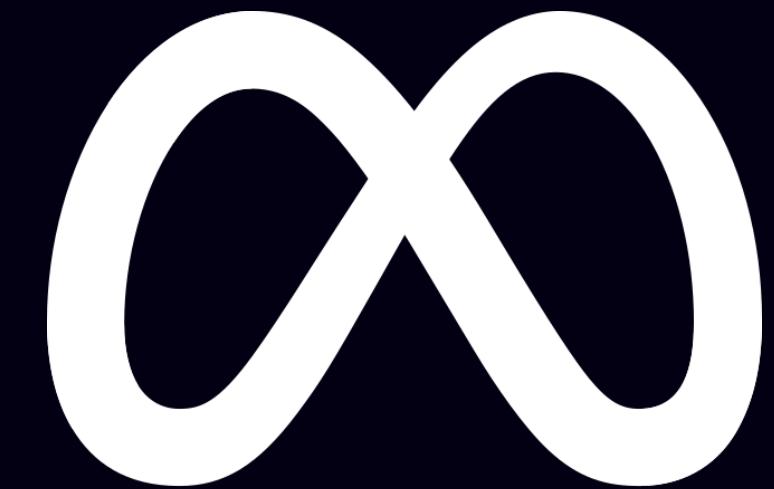
ts-morph

- Удобный поиск референсов файла в проекте
- Обертка над Typescript AST



jscodeshift

- Большие возможности у API
- Поддерживает разные парсеры
- ESTree



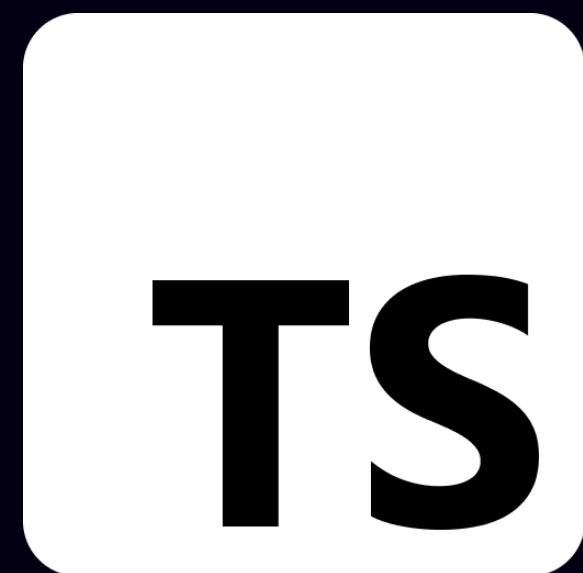
@codemod/core

- Использует экосистему **babel**
- Babel AST



ts-morph

- Удобный поиск референсов файла в проекте
- Обертка над Typescript AST



jscodeshift

- Большие возможности у API
- Поддерживает разные парсеры
- ESTree



Что дальше?

astexplorer.net

AST Explorer Snippet JavaScript @babel/parser Transform default ? Parser: [@babel/parser-7.19.0](#) 4ms

```
1 <Button
2   {...qa('request-secure-auth-button')}
3   color='active'
4   size='medium'
5   onClick={handleClick}
6   variant={isProgress ? 'progress' : 'contained'}
7   endIcon={<SvgIcon>{IconArrowRightM.component}</SvgIcon>}
8 >
9   <FormattedMessage id='secure-auth-form.get-password-button' />
10 </Button>
```

The screenshot shows the AST Explorer interface with a dark theme. On the left, there is a code editor containing JSX code. On the right, the parsed Abstract Syntax Tree (AST) is displayed in JSON format. The JSON structure represents the following tree:

```
{ "File": { "errors": [ ], "program": { "sourceType": "module", "body": [ { "ExpressionStatement": { "expression": { "JSXElement": { "openingElement": { "name": "JSXIdentifier", "value": "button" }, "attributes": [ { "JSXAttribute": { "name": "JSXIdentifier", "value": "color" }, "extra": { "rawValue": "active", "raw": "active" } }, { "JSXAttribute": { "name": "JSXIdentifier", "value": "size" }, "extra": { "rawValue": "medium", "raw": "medium" } }, { "JSXAttribute": { "name": "JSXIdentifier", "value": "onClick" }, "extra": { "rawValue": "handleClick", "raw": "handleClick" } }, { "JSXAttribute": { "name": "JSXIdentifier", "value": "variant" }, "extra": { "rawValue": "isProgress ? 'progress' : 'contained'", "raw": "isProgress ? 'progress' : 'contained'" } }, { "JSXAttribute": { "name": "JSXIdentifier", "value": "endIcon" }, "extra": { "rawValue": "<SvgIcon>{IconArrowRightM.component}</SvgIcon>", "raw": "<SvgIcon>{IconArrowRightM.component}</SvgIcon>" } } ] } } } ] } } }
```

The JSON output is heavily annotated with color-coded highlights. A yellow box surrounds the value of the 'value' attribute under the first 'JSXAttribute' node, which is 'active'. This indicates that the parser has correctly identified the string 'active' as the value of the 'color' attribute.

Сборник кодмодов от сообщества

codemod.com

- Больше 300 кодмодов в библиотеке
- Бесплатный для индивидуальных разработчиков



[← Back](#)

React/19/Migration Recipe

1.0.9 · Last update Jan 10, 2025 · by @alexbit-codemod



This recipe is a set of codemods that will fix some of React 19 breaking changes.

The recipe includes the following codemods:

- react/19/replace-reactdom-render
- react/19/replace-string-ref
- react/19/replace-act-import
- react/19/replace-use-form-state
- react/prop-types-typescript

Run

CLI

```
> codemod react/19/migration...
```



Repository

codemod-com/commons

Total runs

2,675

Version

1.0.9

migration

react

Build custom codemods

Use AI-powered codemod studio and automate undifferentiated tasks for yourself, colleagues or the community

[Get Started Now ↗](#)

Как устроены
наши кодмоды

?

Как устроены
наши кодмоды



Несколько режимов работы

Per file

- Работает в рамках одного файла
- Поиск осуществляется по glob- паттерну

Несколько режимов работы

Per file

- Работает в рамках одного файла
- Поиск осуществляется по glob- паттерну

Codebase

- Выгружает всю кодовую базу
- Можно менять несколько файлов одновременно
- Требует наличия tsconfig.json

Глоссарий

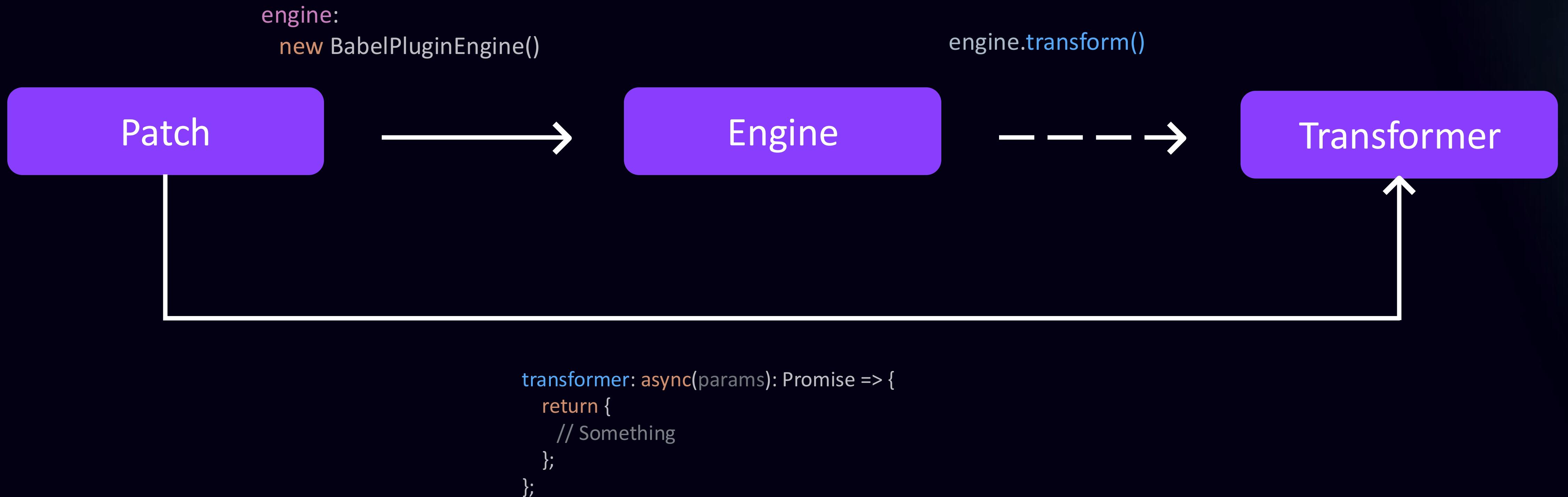
- **Transformer** — функция, изменяющая исходный код. В ней идёт обход AST или работа с файловой системой

Глоссарий

- **Transformer** — функция, изменяющая исходный код. В ней идёт обход AST или работа с файловой системой
- **Engine** — движок кодмода. Внутри него происходит запуск трансформера с нужными параметрами и настройками

Глоссарий

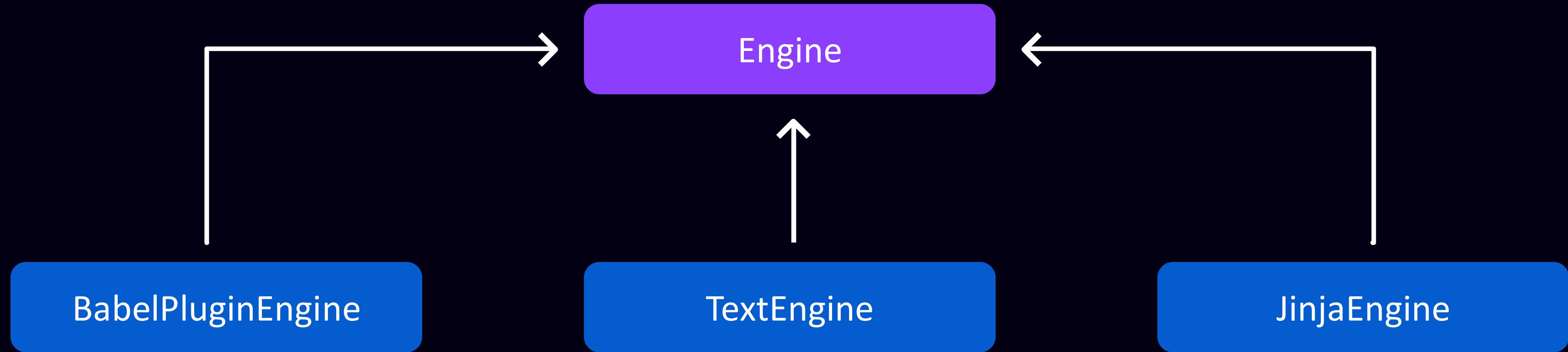
- **Transformer** — функция, изменяющая исходный код. В ней идёт обход AST или работа с файловой системой
- **Engine** — движок кодмода. Внутри него происходит запуск трансформера с нужными параметрами и настройками
- **Patch** — обертка над движком, которая добавляет логирование + собирает файлы, которые надо скормить в движок



Как всё устроено
под капотом



Диаграмма per file



API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

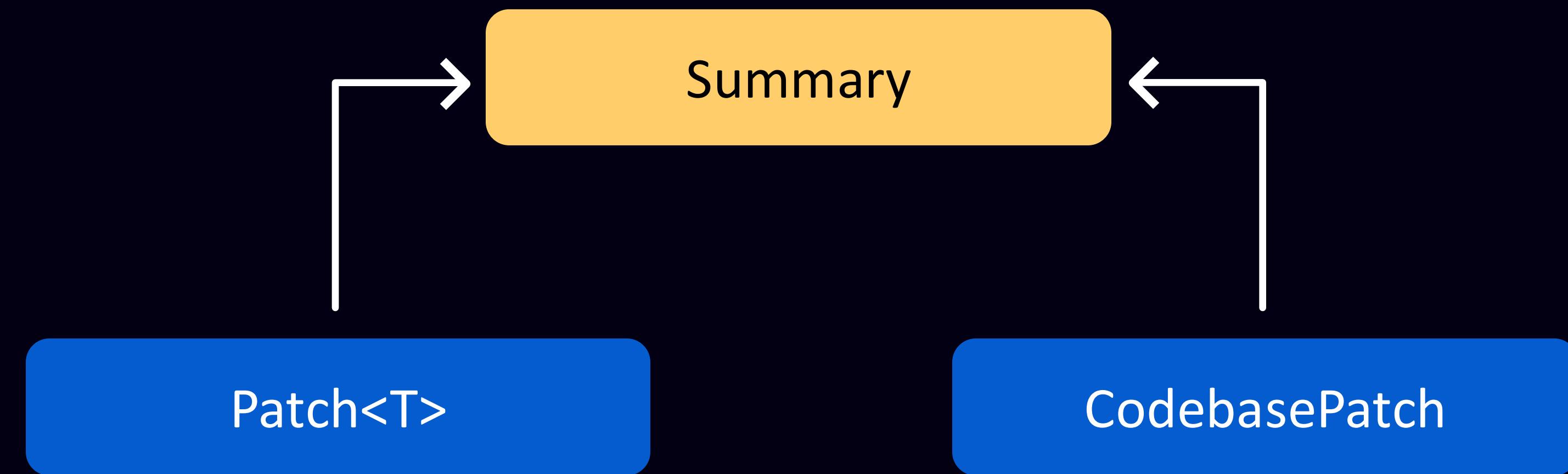
export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

API per file

```
export type EngineTransformParams<T, K> = {
    filepath: string;
    content: string;
    root: string;
    dryRun: boolean;
    transformer: (params: T) => K;
};

export interface Engine<T, K> {
    transform(params: EngineTransformParams<T, K>): void;
}
```

Диаграмма патчей



```
export class ExamplePatch extends Summary {
    constructor(
        private readonly params: PatchParams
    ) {
        super();
    }

    /**
     * Применения патча. Вызов трансформера *
     */
    private async apply() {

    }
}
```

```
type PatchParams = {
    /** Команда патча */
    command: string;
    /** Описание патча */
    description: string;
    /** Флаг включения eslint линтинга с фиксом. default: true */
    enableLinting?: boolean;
    /** Функция преобразования */
    transformer: (params: TransformerParams) => Promise<void>;
};
```

Пример патча per file

Применяем для манипуляций над AST

```
export const demonstrateCommandPatch = new Patch({
  command: 'demonstrate-command-patch',
  description: 'Делает что-то полезное',
  pattern: '/src/client/**/*.{ts|tsx}',
  engine: new BabelPluginEngine(),
  transformer: async(): Promise<PluginObj> => {
    return {
      visitor: {
        Program: (path) => {
          // Обход AST
        }
      }
    };
  }
});
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */
```

```
Program: (path) => {
```

```
}
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */
```

```
Program: (path) => {  
    // Обход JS файла  
    path.traverse({
```

```
});  
}
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */  
Program: (path) => {  
    // Обход JS файла  
    path.traverse({  
        // Находим узлы, которые являются JSX элементами  
        JSXElement: (path) => {  
  
        }  
    });  
}
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */  
Program: (path) => {  
    // Обход JS файла  
    path.traverse({  
        // Находим узлы, которые являются JSX элементами  
        JSXElement: (path) => {  
            // Игнорируем компоненты, которые не являются кнопкой.  
            // Удобно, что можно указать дополнительным параметром имя узла: это существенно упрощает код  
            if (!t.isJSXIdentifier(path.node.openingElement.name, {name: 'Button'})) {  
                return;  
            }  
        }  
    });  
}
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */  
Program: (path) => {  
    // Обход JS файла  
    path.traverse({  
        // Находим узлы, которые являются JSX элементами  
        JSXElement: (path) => {  
            // Игнорируем компоненты, которые не являются кнопкой.  
            // Удобно, что можно указать дополнительным параметром имя узла: это существенно упрощает код  
            if (!t.isJSXIdentifier(path.node.openingElement.name, {name: 'Button'})) {  
                return;  
            }  
  
            // Создадим узел для нашего пропса  
            const newAttribute = t.jsxAttribute(  
                t.jsxIdentifier("fullWidth"), // Название пропса  
                t.jsxExpressionContainer(t.booleanLiteral(true)) // Значение в фигурных скобках  
            )  
  
            //  
        }  
    });  
}
```

```
/**  
 * Задача: добавить в компонентах <Button /> атрибут "fullWidth"  
 * @example <Button /> -> <Button fullWidth={true} />  
 */  
Program: (path) => {  
    // Обход JS файла  
    path.traverse({  
        // Находим узлы, которые являются JSX элементами  
        JSXElement: (path) => {  
            // Игнорируем компоненты, которые не являются кнопкой.  
            // Удобно, что можно указать дополнительным параметром имя узла: это существенно упрощает код  
            if (!t.isJSXIdentifier(path.node.openingElement.name, {name: 'Button'})) {  
                return;  
            }  
  
            // Создадим узел для нашего пропса  
            const newAttribute = t.jsxAttribute(  
                t.jsxIdentifier("fullWidth"), // Название пропса  
                t.jsxExpressionContainer(t.booleanLiteral(true)) // Значение в фигурных скобках  
            )  
  
            // Вставим пропс в массив всех атрибутов компонента  
            path.node.openingElement.attributes.push(newAttribute);  
        }  
    });  
}
```

Пример патча codebase

Применяем, когда нужно получать ссылки на используемые сущности

```
export const exampleCodeBasePatch = new CodebasePatch({  
    command: 'example-codebase-migration',  
    description: 'Демонстрирует работу codebase-патча',  
    enableLinting: true,  
    transformer: async({project, log, rootPath}) => {  
        // project — instance клиента ts-morph  
        // log — логгер  
        // rootPath — путь до папки, в которой вызывается кодмод  
    }  
});
```



```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/
```

```
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.{class,ts}")
```

```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/
```

```
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.{class,ts}")
```

```
// Получаем все объявления классов в этих файлах  
const classDeclarations = files.map(file => file.getClasses()).flat();
```

```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/
```

```
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.class.ts")
```

```
// Получаем все объявления классов в этих файлах  
const classDeclarations = files.map(file => file.getClasses()).flat();
```

```
for (const classDeclaration of classDeclarations) {
```

```
}
```

```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/
```

```
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.class.ts")
```

```
// Получаем все объявления классов в этих файлах  
const classDeclarations = files.map(file => file.getClasses()).flat();
```

```
for (const classDeclaration of classDeclarations) {  
    const className = classDeclaration.getName();
```

```
let testCode = `describe('${className}', () => {\n`;
```

```
}
```

```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/
```

```
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.class.ts")
```

```
// Получаем все объявления классов в этих файлах  
const classDeclarations = files.map(file => file.getClasses()).flat();
```

```
for (const classDeclaration of classDeclarations) {  
    const className = classDeclaration.getName();
```

```
let testCode = `describe('${className}', () => {\n`;
```

```
// Описываем тело тест-кейсов
```

```
for (const method of classDeclaration.getMethods()) {  
    const methodName = method.getName();
```

```
    testCode += `\tit('should test ${methodName}', () => {\n`;
```

```
    testCode += `\t\texpect(new ${className}().${methodName}(...)).toBe(...);\n`;
```

```
    testCode += "\t});\n";
```

```
}
```

```
testCode += "});";
```

```
}
```

```
/**  
* Задача: для определенных классов сгенерировать тесты на методы  
*/  
  
// Находим нужные нам файлы по glob-паттерну  
const files = project.getSourceFiles("./**/*.{class,ts}")  
  
// Получаем все объявления классов в этих файлах  
const classDeclarations = files.map(file => file.getClasses()).flat();  
  
for (const classDeclaration of classDeclarations) {  
    const className = classDeclaration.getName();  
  
    let testCode = `describe('${className}', () => {\n`;  
  
    // Описываем тело тест-кейсов  
    for (const method of classDeclaration.getMethods()) {  
        const methodName = method.getName();  
  
        testCode += `\tit('should test ${methodName}', () => {\n`;  
        testCode += `\t\texpect(new ${className}().${methodName}(...)).toBe(...);\n`;  
        testCode += "\t});\n";  
    }  
  
    testCode += "});";  
  
    // Сохраняем полученный код в директорию с тестами  
    const activeDirectory = classDeclaration.getSourceFile().getDirectoryPath()  
    project.createSourceFile(`${activeDirectory}/__tests__/${className}.spec.ts`, testCode)  
}
```

```

/**
* Задача: для определенных классов сгенерировать тесты на методы
*/
// Находим нужные нам файлы по glob-паттерну
const files = project.getSourceFiles("./**/*.{class,ts}")

// Получаем все объявления классов в этих файлах
const classDeclarations = files.map(file => file.getClasses()).flat();

for (const classDeclaration of classDeclarations) {
    const className = classDeclaration.getName();

    let testCode = `describe('${className}', () => {\n`;

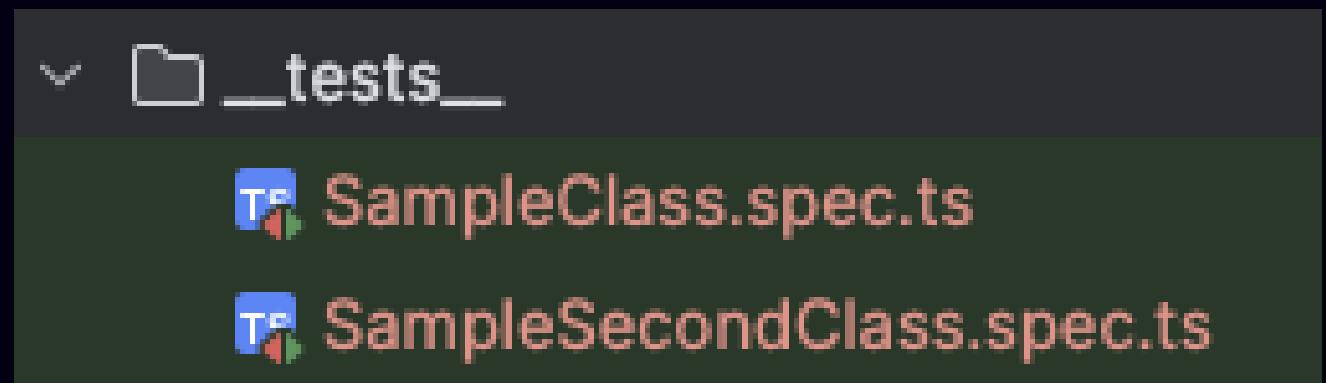
    // Описываем тело тест-кейсов
    for (const method of classDeclaration.getMethods()) {
        const methodName = method.getName();

        testCode += `\tit('should test ${methodName}', () => {\n`;
        testCode += `\t\t\txpect(new ${className}().${methodName}(...)).toBe(...);\n`;
        testCode += "\t});\n";
    }

    testCode += "});\n";

    // Сохраняем полученный код в директорию с тестами
    const activeDirectory = classDeclaration.getSourceFile().getDirectoryPath()
    project.createSourceFile(`$activeDirectory/_tests_${className}.spec.ts`, testCode)
}

```



Как их объединить?

```
public addAsCommand(program: yarg.Argv<{}>) {
  program.command({
    command: this.params.command,
    describe: this.params.description,
    handler: (argv) => this.apply({
      root: argv.root as string,
      dryRun: argv.dryRun as Boolean
    })
  })
  .option('dryRun', {
    type: 'boolean',
    description: 'Флаг запуска в режиме отладки'
  })
  .option('root', {
    type: 'string',
    description: 'Корневая директория для применения паттерна'
  });
}
```

```
export const setup = () => {
  const program = usage('Usage: $0 <command> [options]');

  patches.forEach(patch => {
    patch.addAsCommand(program);
  });

  program
    .demandCommand(1, 'Command required')
    .alias('h', 'help')
    .version()
    .argv;
};
```

```
export const setup = () => {
  const program = usage('Usage: $0 <command> [options]');

  patches.forEach(patch => {
    patch.addAsCommand(program);
  });

  program
    .demandCommand(1, 'Command required')
    .alias('h', 'help')
    .version()
    .argv;
};
```



Пример

Миграция с CSS-in-JS на CSS Modules



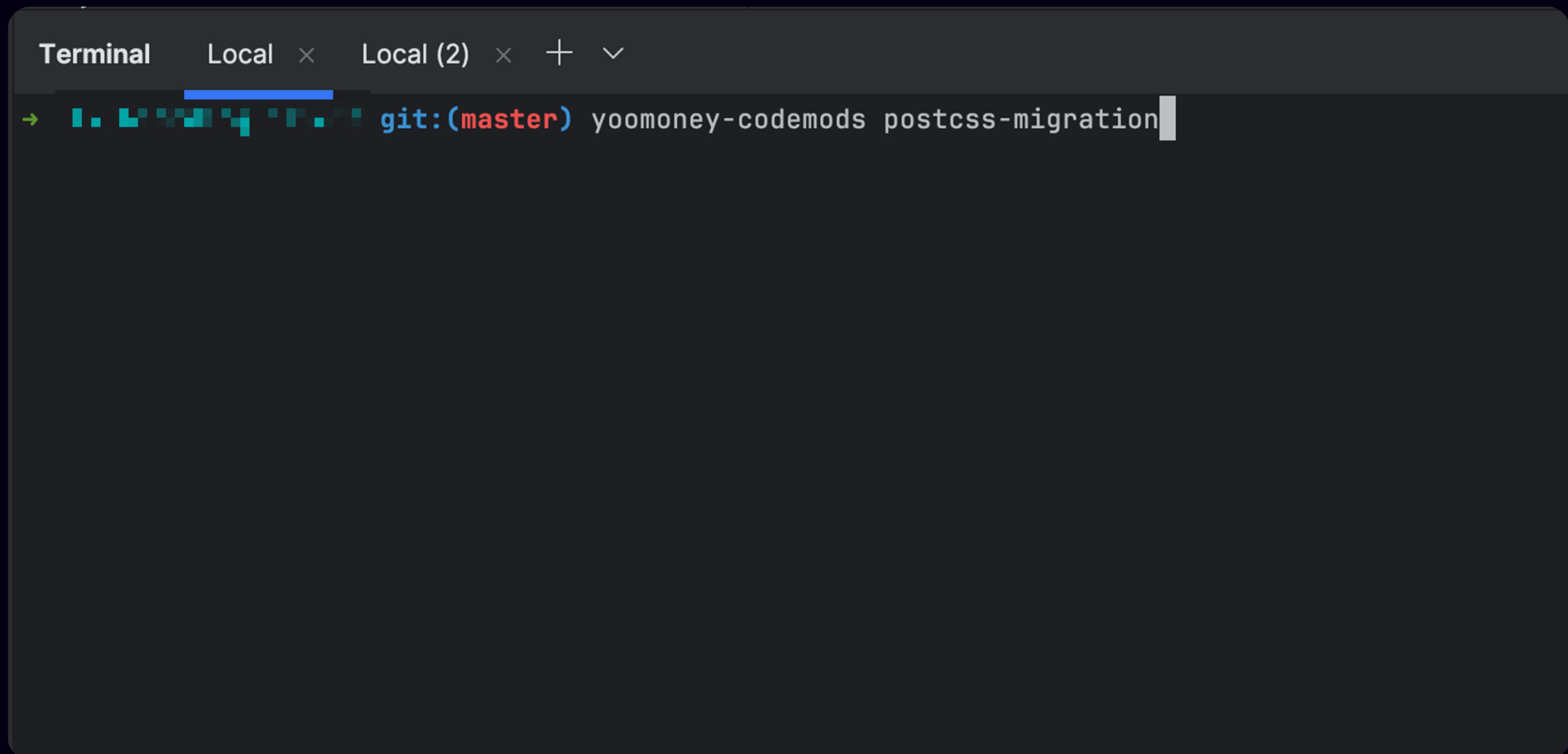
Emotion

- Крупный внутренний B2B-сервис
- 53 файла, подлежащих изменению
- 260 styled-компонентов



CSS
MODULES

Запуск кодмода в терминале



A screenshot of a dark-themed terminal window titled "Terminal". The window has tabs for "Local" and "Local (2)". The current tab shows a command line with a blue cursor. The command is:

```
→ 1. ыомонета 🎉 git:(master) yoomoney-codemods postcss-migration
```

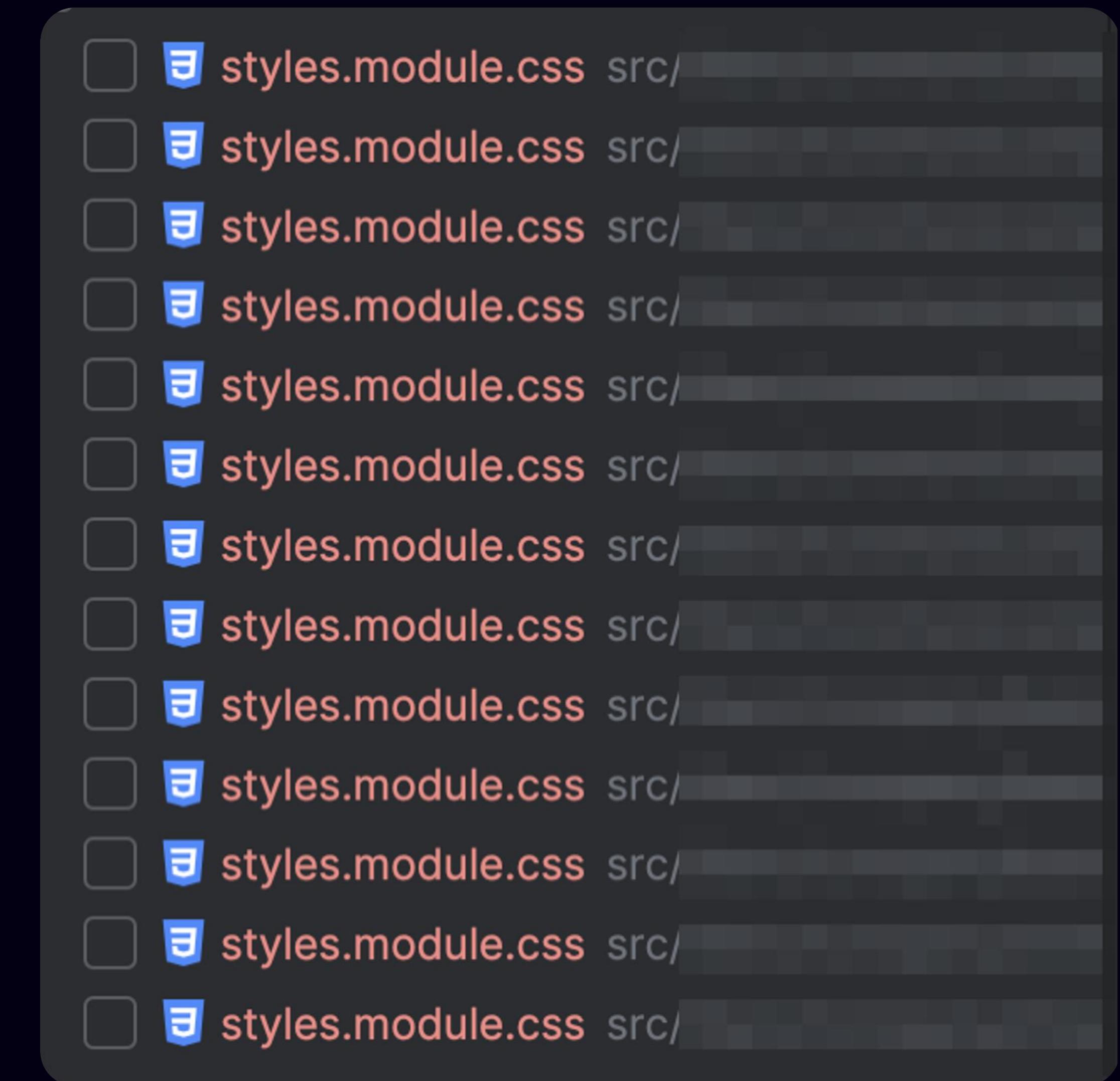
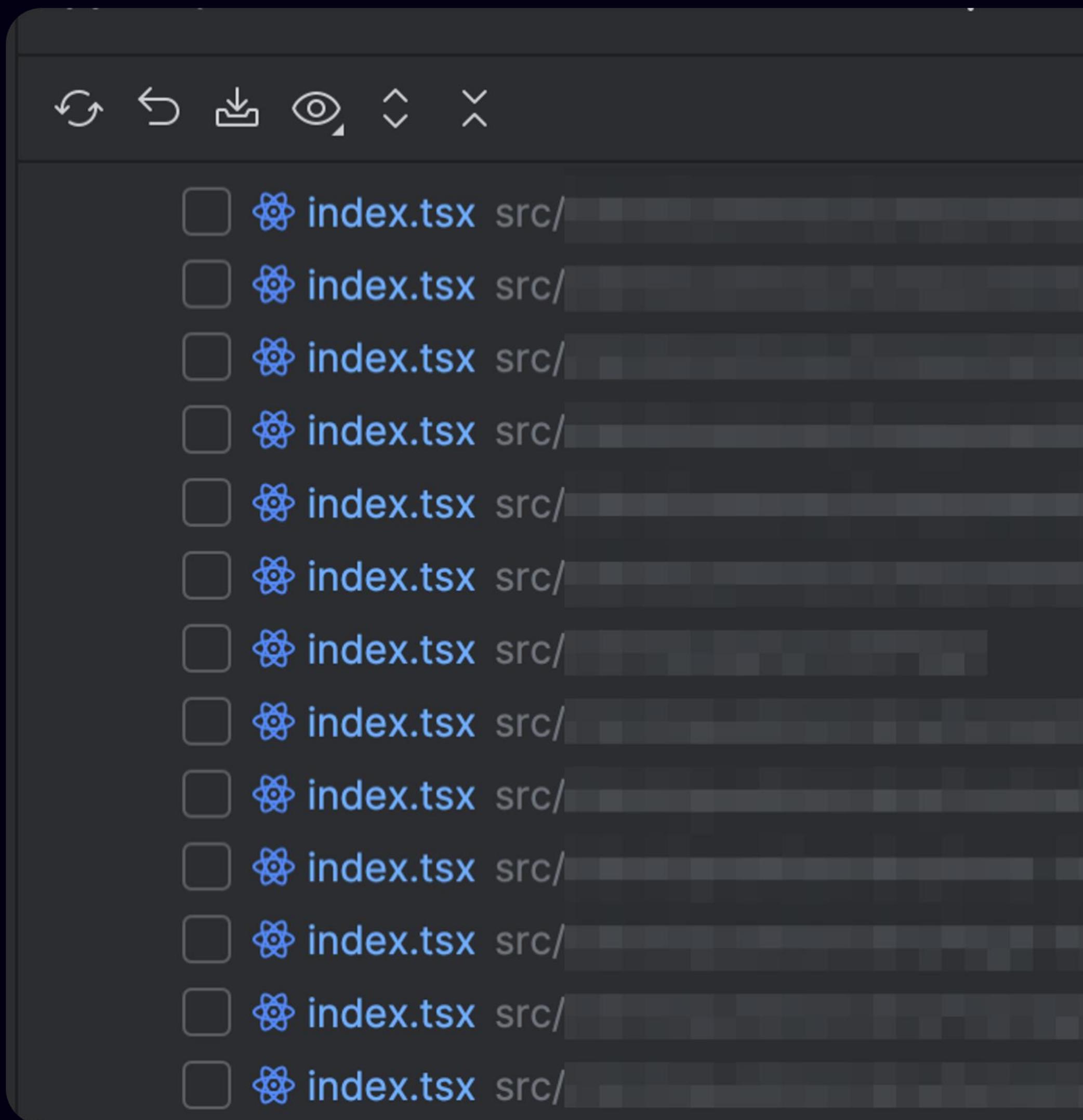
Логи

```
yoomey-codemods postcss-migration
[...]
  migration] > ... await Project parsing...
  migration] > ☒ complete Project parsed
  migration] > ... await Collecting styled components...
  migration] > ☒ complete 260 styled components found
  migration] > i info 244 styled components can be fixed automatically
  migration] > ... await 244/244 components processed
  migration] > ... await Removing unused identifiers...
  migration] > ... await Writing CSS...
  migration] > ⚠ warn 16 can't be fixed automatically. Fix it manually:
```

ЛОГИ

```
| postcss-migration] > ... await 244/244 components processed
| postcss-migration] > ... await Removing unused identifiers...
| postcss-migration] > ... await Writing CSS...
| postcss-migration] > ▲ warn 16 can't be fixed automatically. Fix it manually:
| postcss-migration] > StyledMainInfoWrapper /Users/miroshnichenkoaa/secret-path/secret-component.tsx:20
| postcss-migration] > StyledFullInfoWrapper /Users/miroshnichenkoaa/secret-path/secret-component.tsx:33
| postcss-migration] > StDate /Users/miroshnichenkoaa/secret-path/secret-component.tsx:73
| postcss-migration] > StLabel /Users/miroshnichenkoaa/secret-path/secret-component.tsx:94
| postcss-migration] > StUnit /Users/miroshnichenkoaa/secret-path/secret-component.tsx:119
| postcss-migration] > StOneLineInput /Users/miroshnichenkoaa/secret-path/secret-component.tsx:318
| postcss-migration] > StNav /Users/miroshnichenkoaa/secret-path/secret-component.tsx:49
| postcss-migration] > StNavItem /Users/miroshnichenkoaa/secret-path/secret-component.tsx:62
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:120
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:133
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:240
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:151
| postcss-migration] > StyledItem /Users/miroshnichenkoaa/secret-path/secret-component.tsx:108
| postcss-migration] > StHistoryItemsWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:5
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:121
| postcss-migration] > StWrap /Users/miroshnichenkoaa/secret-path/secret-component.tsx:169
| postcss-migration] > ✓ success Success!
```

Результат



Результат

Было

```
if (showLoader) {  
    return (  
        <StyledSecondaryContentWrapper>  
            <Loader />  
        </StyledSecondaryContentWrapper>  
    );  
  
    if (showError) {  
        return (  
            <StyledSecondaryContentWrapper>  
                <LoadPageErrorResult />  
            </StyledSecondaryContentWrapper>  
        );  
    }  
}
```

Стало

```
showLoader) {  
    return (  
        <div className={css.secondaryContentWrapper}>  
            <Loader />  
        </div>  
    );  
  
    showError) {  
        return (  
            <div className={css.secondaryContentWrapper}>  
                <LoadPageErrorResult />  
            </div>  
    );  
}
```

Есть ли минусы?

Минусы

- Сначала будет сложно

Минусы

- Сначала будет сложно
- Тестирование только ручное

Минусы

- Сначала будет сложно
- Тестирование только ручное
- Ручные правки все равно останутся

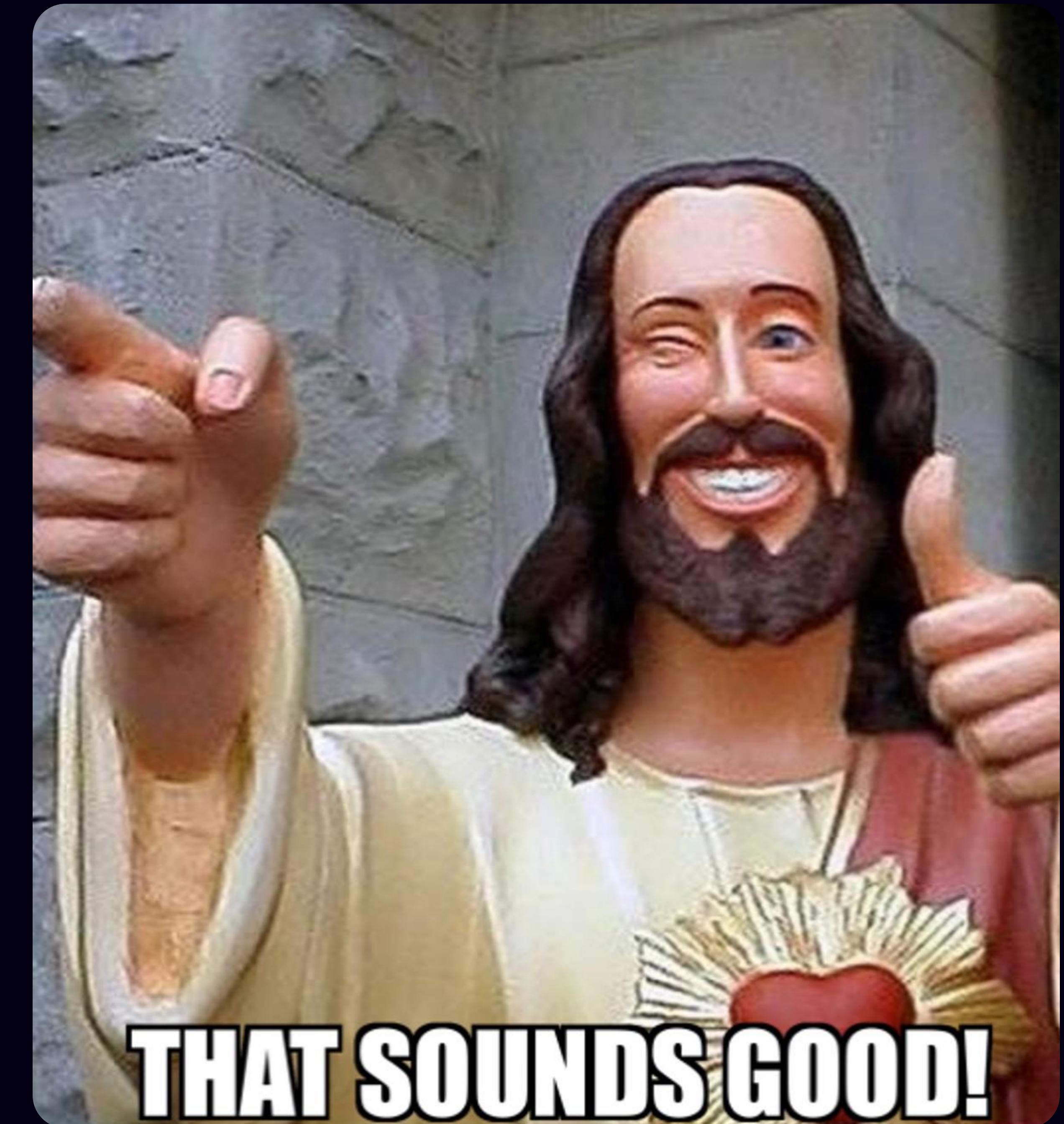
Минусы

- Сначала будет сложно
- Тестирование только ручное
- Ручные правки все равно останутся (скорее всего)

Выводы

Выводы

- Ускорение процессов в командах
- Удобно для больших проектов
- Полезные навыки по работе с AST





Спасибо
за внимание

Александр Мирошниченко
[@sashafromlibertalia](https://twitter.com/sashafromlibertalia)

