УДК 004.4'23

ИСПОЛЬЗОВАНИЕ PRETTIER И GIT HOOKS ДЛЯ АВТОМАТИЧЕСКОГО ПОДДЕРЖАНИЯ КУЛЬТУРЫ КОДА В TYPESCRIPT-ПРОЕКТЕ

Вахрамов С.В., Мусин А.М., Ризванов Д.А., Хамитов М.А.

ФГБОУ ВО «Уфимский государственный авиационный технический университет», Уфа, e-mail: s@vakhramoff.ru

Соответствие кода общепринятым правилам достаточно тяжело поддерживать вручную без использования каких-либо утилит; здесь главную роль играет человеческий фактор: любой может пропустить ошибку, не заметив её, или под влиянием нехватки времени пропустить код без проверки. Статья рассматривает автоматические способы линтинга проектного кода с помощью утилиты ESlint и плагина Prettier для проектов, написанных на языке программирования TypeScript. Приведение в порядок существующей кодовой базы можно выполнить одной командой, а ручная обработка больших объёмов кода займёт гораздо больше времени без применения специальных утилит. В статье приведено подробное описание особенностей и возможностей Prettier и Git-хуков. Автоматическая проверка и исправление ошибок кода-стиля выполнены с помощью плагинов lint-staged и husky. Для установки универсальных параметров редактирования использован файл EditorConfig, поддерживающийся почти всеми современными IDE. Результатом применения методов, описанных в статье, является проект с настроенной системой автоматического форматирования кода. В результате внедрения описанных в статье инструментов сократилась скорость разработки проекта, а также снизился порог входа для начинающих разработчиков программного обеспечения.

Ключевые слова: TypeScript, Prettier, Git, Git hooks, lint-staged, husky, код-стиль, автоматическое форматирование кода

USE OF PRETTIER AND GIT HOOKS FOR PERFORMING AUTOMATIC CODE-STYLE EDITING IN A TYPESCRIPT PROJECT

Vakhramov S.V., Musin A.M., Rizvanov D.A., Khamitov M.A.

Ufa State Aviation Technical University, Ufa, e-mail: s@vakhramoff.ru

It's very hard to support conformity of the code to common rules without using special utilities. Human factor plays the main role here: anyone could potentially make a mistake by missing it or when there's a lack of time and you have to immediately push the code into your repository. This article review automatic linting approaches of the project's code using ESlint and Prettier plugin for projects which were written using TypeScript programming languages. Detailed review with description of Prettier and Git-hooks possibilities was provided. Automatic checking and correction of the code-style are performed with lint-staged and husky plugins. EditorConfig file was used to setting up common IDE editing style, which is supported by most of modern IDEs. The result of application of this approach is a project with fully set up automatic code-style formatting system. As a result of introducing the tools described in the article, the speed of project development has decreased and the entry threshold for beginner software developers has decreased.

Keywords: TypeScript, Prettier, Git, Git hooks, lint-staged, husky, code-style, automatic code formatting

Что такое Prettier? Prettier – это средство для форматирования кода, которое нацелено на использование жёстко заданных правил по оформлению программ [1]. Оно форматирует код автоматически, обеспечивая соблюдение предустановленных правил форматирования кода для проекта, что в свою очередь позволяет сфокусироваться на более важных вещах при написании программного кода.

Инструмент Prettier обладает следующими возможностями и особенностями (рассмотрим основные из них).

– Приведение в порядок существующей кодовой базы можно выполнить одной командой. Ручная обработка больших объёмов кода займёт гораздо больше времени: представьте себе затраты труда, необходимые для ручного форматирования 20 000 строк кода.

– Prettier легко внедряется в проект, он использует наименее спорный подход к стилю при форматировании кода. Prettier – проект с открытым исходным кодом: многие внесли вклад в его доработку и улучшение.

– Prettier позволяет сосредоточиться на написании кода. Разработчики не осознают того, как много времени и сил тратится на форматирование кода. Использование Prettier в одном из наших проектов повысило эффективность работы на 10%.

– Prettier помогает начинающим. Если вы начинающий программист, работающий в одной команде с серьёзными профессионалами, и вы хотите достойно смотреться на их фоне, в этом вам поможет Prettier.

– Затраты времени на code-review сокращаются в среднем на 15–20%: нет необходимости напоминать о забытых пробелах,

лишних запятых и других подобных ошибках – их исправит Prettier.

Для запуска локального форматирования кода в большинстве сред разработки предусмотрена возможность запуска форматирования текста прямо из контекстного меню, например как в WebStorm (рис. 1).

Что такое Git Hooks? У систем контроля версий есть механизм запуска callback-скриптов по определенному действию. Как и многие другие системы контроля версий, Git предоставляет возможность запуска пользовательских скриптов в случае возникновения определённых событий. Такие действия называются хуками и разделяются на две группы: серверные и клиентские. Если хуки на стороне клиента запускаются такими операциями, как слияние или создание коммита, то на стороне сервера они инициируются сетевыми операциями, такими как получение отправленного коммита. Хуки часто используются для широкого круга задач.

В некотором смысле хук является реализацией шаблона Observer из ООП. Хуки могут быть написаны на скриптовых языках, таких как Bash, Python, Ruby и других.

Рассмотрим виды git-hooks (рис. 2).

Для реализации нашей задачи мы будем использовать git-хук pre-commit для конфигурирования процесса отправки, проверки и форматирования изменений плагином, находящихся в коммите.

Цель исследования: исследование процесса подключения инструментов для автоматического форматирования программного кода при фиксировании изменений в git-репозитории путём коммита изменённых файлов.

Материалы и методы исследования

Далее приведено пошаговое руководство по установке и конфигурированию Prettier и локальных git-хуков в TypeScript-проекте.

1. Установка Prettier

Установить npm-пакет можно командой

```
npm install --save-dev prettier
```

2. Установка плагинов для Prettier

Установить npm-пакет плагина можно командой

```
npm install --save-dev prettier-plugin-organize-imports
```

Данный плагин необходим для автоматической сортировки импортов.

3. Установка плагинов для организации автоматической проверки и корректировки кода

Установить npm-пакеты можно командой

```
npm install --save-dev husky
```

Данный плагин позволяет настроить локальные хуки

```
npm install --save-dev lint-staged
```

Данный плагин позволяет настроить выполнение скриптов для файлов, готовых к коммиту.

4. Конфигурирование собственных правил исправления программного кода

Собственные правила в Prettier редактируются в корневом файле проекта.

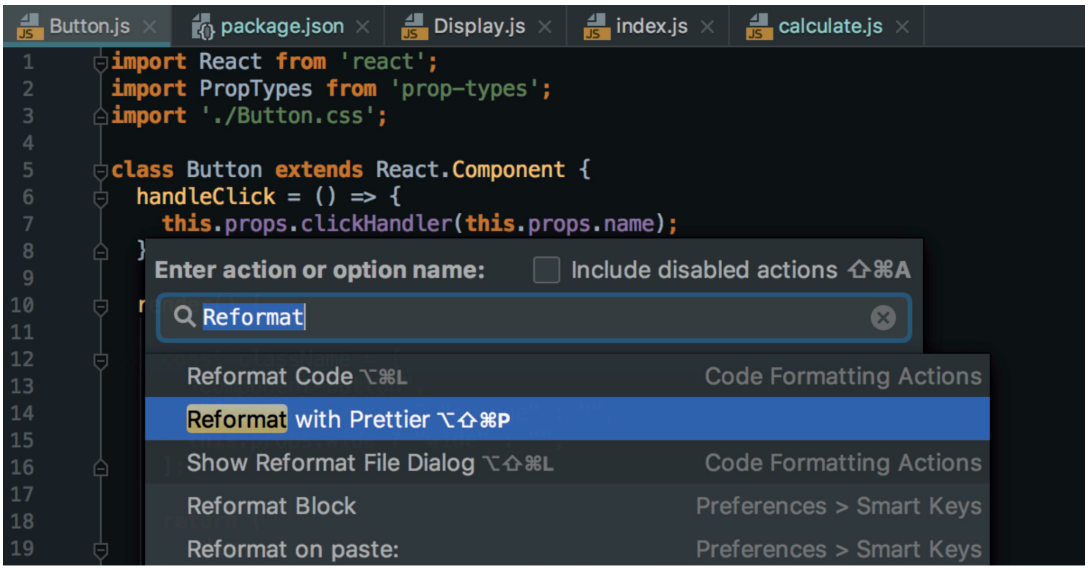

Рис. 1. Контекстное меню для запуска Prettier в IDE WebStorm

Файл имеет название «.prettierrc». Ниже приведён пример конфигурации для соответствующего файла в Angular-проекте. Подробнее о конфигурировании правил Prettier можно прочесть на официальном сайте [2].

```
{
  "bracketSpacing": true,
  "printWidth": 140,
  "semi": true,
  "singleQuote": true,
  "trailingComma": "all",
  "tabWidth": 2,
  "useTabs": false,
  "overrides": [
    {
      "files": "src/**/*.ts",
      "options": {
        "parser": "typescript"
      }
    },
    {
      "files": "*.component.html",
      "options": {
        "parser": "angular"
      }
    },
    {
      "files": [".prettierrc", "package.json"],
      "options": {
        "parser": "json"
      }
    }
  ]
}
```

Название Git hooks	Назначение хука
commit-msg	Хук вызывается перед коммитом. Он может изменять сообщение, а также использоваться, чтобы прервать выполнения коммита, если исходное сообщение не проходит вашу проверку. Хук можно обойти с помощью --no-verify
pre-push	Хук выполняется перед пушем. Может быть использован чтобы прервать его.
pre-applypatch	Хук выполняется перед применением патчка. Может быть использован чтобы прервать его.
prepare-commit-msg	Хук выполняется перед коммитом, после подготовки сообщения по умолчанию. Основная цель - подготовить сообщение.
pre-commit	Хук в осноном используется для sanity check вашего кода перед отправкой на code review.
pre-rebase	Хук вызывается перед ребейзом. Может быть использован, чтобы прервать его.
post-commit	Хук выполняется после завершения коммита. Служит главным образом для уведомлений и не может повлиять на ход коммита.

Рис. 2. Описание основных git hooks

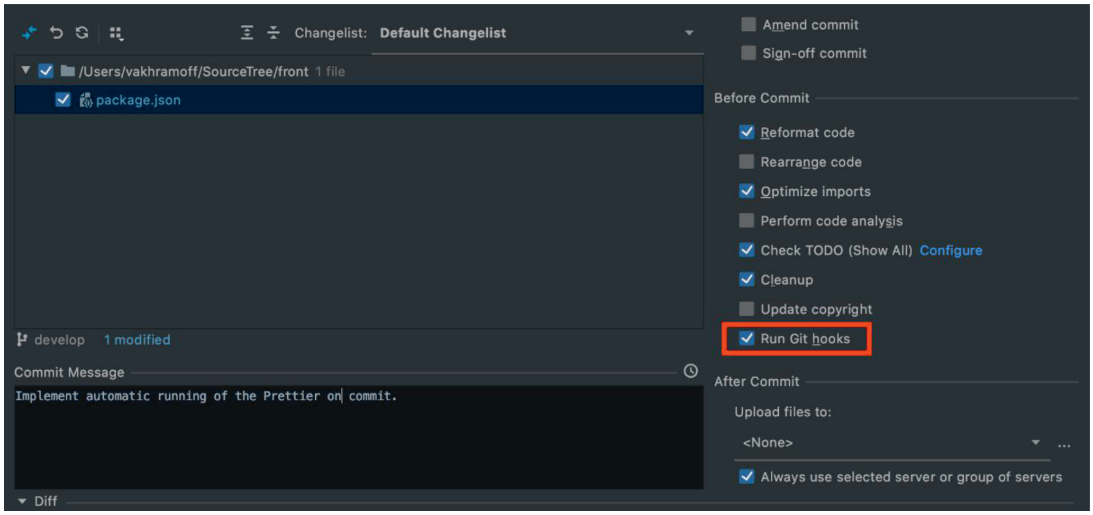

Рис. 3. Опция запуска Git-хуков в окне коммита в IDE WebStorm

5. Связывание Prettier с ESlint/TSlint

Данная статья подразумевает наличие установленного в проекте ESlint или TSlint. Для корректного подключения к проекту Prettier при наличии уже установленного инструмента проверки кода из вышеперечисленных можно воспользоваться инструкцией, расположенной на сайте [3].

6. Настройка локального pre-commit хука

Для этого в package.json необходимо в разделе scripts добавить скрипты для проверки и исправления текущего кода [4–6]:

```
"scripts": {
  "prettier": "prettier --check \"src/**/*.(ts|component.html)\"",
  "prettier:fix": "npm rum prettier -- --write"
},
Затем в секции husky описать pre-commit хук:
"husky": {
  "hooks": {
    "pre-commit": "lint-staged"
  }
}
Далее описав скрипт для lint-staged:
"lint-staged": {
  "*.(ts|component.html)": [
    "npm run prettier:fix",
    "git add"
  ]
}
```

7. Настройка универсальных параметров редактора в IDE

Для того чтобы все пользовались едиными настройками, такими как отступы или символы перевода строки, применяется файл *.editorconfig*. Он описывает единый набор правил в неоднородных командах для всех поддерживающих его IDE.

На сайте проекта [7] можно найти список редакторов, которые поддерживают этот файл. В него входят WebStorm, AppCode, Atom, Eclipse, Emacs, BBEdit и другие.

Для описания такого файла создадим в корне проекта файл «.editorconfig» и добавим в него следующий код:

```
# Файл EditorConfig верхнего уровня
root = true

[*.md]
trim_trailing_whitespace = false
```

```
[*.js]
trim_trailing_whitespace = true

# Переводы строк в стиле Unix с пустой строкой в конце файла
[*]
indent_style = space
indent_size = 2
end_of_line = lf
charset = utf-8
insert_final_newline = true
max_line_length = 100
```

8. Настройка автоматического исправления файлов, попавших в коммит

С этого момента IDE будет автоматически применять данный pre-commit хук. Главное – не снимать галочку, которая отвечает за выполнение подобных локальных хуков (рис. 2).

Результаты исследования и их обсуждение

В результате мы получили проект с настроенной системой автоматического форматирования кода. В проекте автоматически соблюдаются заранее предустановленные правила форматирования, что позволяет существенно экономить время на написание и ревю кода.

Выводы

Внедрение Prettier сократило время разработки проекта, так как не требовалось тратить дополнительное время на соблюдение кода-стиля для проекта. Автоматизация процесса приведения кода к стандартизированному форматированию через хуки позволила забыть про ручной контроль каждого измененного файла. Увеличилась читаемость кода: это стало заметно на программистах с опытом разработки менее 1 года, которые только пришли в проект.

Список литературы

1. What is Prettier? [Электронный ресурс]. URL: https://prettier.io/docs/en/index.html (дата обращения: 10.07.2020).

2. Options – Prettier. [Электронный ресурс]. URL: https://prettier.io/docs/en/options.html (дата обращения: 10.07.2020).

3. Integrating with Linters. [Электронный ресурс]. URL: https://prettier.io/docs/en/integrating-with-linters.html (дата обращения: 10.07.2020).

4. Customizing Git – Git Hooks. [Электронный ресурс]. URL: https://git-scm.com/book/en/v2/Customizing-Git-Git-Hooks (дата обращения: 10.07.2020).

5. Черная магия Git hook. Как не пустить джуниоров в мастер-ветку и вообще все автоматизировать. [Электронный ресурс]. URL:https://xakep.ru/2016/02/11/git-hook-magic/ (дата обращения: 10.07.2020).

6. Prettier, ESLint, Husky, Lint-Staged и EditorConfig: инструменты для написания аккуратного кода. [Электронный ресурс]. URL: https://habr.com/en/company/ruvds/blog/428173/ (дата обращения: 10.07.2020).

7. EditorConfig. [Электронный ресурс]. URL:https://editorconfig.org (дата обращения: 10.07.2020).