Автоматическое форматирование кода (Prettier)

Проблема: Грязный код и потраченное время

Знакомо?

- Где-то забыли точку с запятой.
- Где-то поставили лишний пробел.
- Отступы "пляшут".
- Кавычки то двойные, то одинарные.

Последствия:

- Тратим время на ручное выравнивание.
- Ссоры в команде из-за стиля кода.
- Сложнее читать код.

Prettier — мнение вашего кода

- **Prettier** — это "самоуверенный" форматировщик кода (opinionated code formatter).
- Он навязывает единый стиль: максимальная длина строки, отступы, пробелы, кавычки.
- **Цель:** Освободить разработчика от споров о форматировании и сэкономить время.
- Поддерживает: JS, JSX, TS, CSS, HTML, JSON, Markdown и другие.

Установка Prettier в проект

- `Prettier` устанавливается как **dev-зависимость** (инструмент, нужный только для разработки).
- Команда установки через пакетный менеджер `pnpm` (быстрый и экономит место):

```
pnpm i -D prettier
```

  - `i` — install
  - `-D` — сохранить в `devDependencies`
- После установки в `package.json` появится секция `devDependencies` с `prettier`.

Настройка и запуск Prettier

Добавим скрипт в package.json для удобного запуска:

```
json
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "prettify": "prettier --write ." // Новая команда
},
```

Запуск форматирования всего проекта:

```
pnpm prettify
```

`--write` — перезаписать файлы (без этого флага Prettier только покажет различия).

`.` — отформатировать все файлы в текущей директории.

Форматирование через VSCode (Command Palette)

Способ 1 (Ручной, для одного файла):

1. Откройте нужный файл (например, `App.jsx`).
2. Нажмите `Ctrl+Shift+P` (или `F1`), чтобы открыть **Палитру команд**.
3. Начните вводить "Format Document".
4. Выберите **"Format Document"**.

VSCode предложит выбрать форматировщик по умолчанию — выберите `Prettier`.

Магия горячих клавиш

Способ 2 (Быстрый): Используйте горячие клавиши!

- **Windows/Linux:** Shift + Alt + F
- **macOS:** Shift + Option + F

Способ 3 (Автоматический): Настройте VSCode так, чтобы он форматировал файл **автоматически при сохранении**.

- Настройки (Ctrl+,) → поиск "Format On Save" → поставить галочку.

Совет: Включите "Format On Save" и забудьте о ручном форматировании навсегда.

Выводы

1. **React — это JavaScript.** Любая логика в компонентах — это обычный JS. Массивы, объекты, функции — всё работает так же, как вы учили.
2. **Для вывода списков используется** map()**.** Это базовый паттерн в React. Не забывайте про атрибут key.
3. **Prettier автоматизирует форматирование.** Установите его в проект, настройте "Format On Save" и больше никогда не тратьте время на расстановку отступов.
4. **Горячие клавиши ускоряют работу.** Shift+Alt+F (Windows) или Shift+Option+F (macOS) должны стать вашими друзьями.

Лабораторная работа № 2_6
«Настройка Prettier»

Цель работы: Научиться устанавливать и настраивать инструмент форматирования кода Prettier в проекте на базе Vite + React + TypeScript, освоить автоматическое форматирование при сохранении файлов и выполнить практическое задание по форматированию JSX-кода.

Что такое Prettier и зачем он нужен?

Prettier — это "самоуверенный" (opinionated) форматтер кода, который автоматически приводит ваш код к единому стилю. В отличие от линтеров (ESLint), которые ищут ошибки и потенциальные проблемы, Prettier занимается исключительно внешним видом кода: расстановкой пробелов, переносов строк, кавычек и точек с запятой.

Основные преимущества Prettier:

- **Единообразие кода** — все разработчики в команде пишут в одном стиле, исчезают споры о форматировании
- **Автоматизация** — не нужно думать о том, ставить точку с запятой или нет, Prettier сделает это за вас
- **Экономия времени** — форматирование происходит автоматически при сохранении файла
- **Поддержка JSX** — отлично работает с React-компонентами

Модуль 1: Создание проекта Vite+React+TS

Шаг 1.1. Создание нового проекта

В терминале VS Code выполните команду для создания нового проекта с шаблоном React + TypeScript:

```
npm create vite@latest prettier-lab -- --template react-ts
```

Разберём команду:

- npm create vite@latest — создание нового проекта через Vite
- prettier-lab — название папки проекта (можете изменить)
- --template react-ts — использование шаблона React с TypeScript

Шаг 1.2. Переход в папку проекта и установка зависимостей

```
cd prettier-lab
npm install
```

Шаг 1.3. Запуск проекта для проверки

```
npm run dev
```

Вы должны увидеть сообщение, что сервер запущен

(обычно на `http://localhost:5173`). Откройте этот адрес в браузере — должна отобразиться стандартная страница Vite + React.

Остановите сервер комбинацией `Ctrl+C` в терминале (мы продолжим установку).

Модуль 2: Установка Prettier

Шаг 2.1. Установка Prettier как зависимости разработки

`Prettier` устанавливается как dev-зависимость, так как он нужен только во время разработки. В терминале выполните:

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

Параметры команды:

- --save-dev — добавляет пакет в раздел `devDependencies` в `package.json`
- --save-exact — фиксирует точную версию, чтобы у всех в команде была одинаковая версия Prettier

Шаг 2.2. Проверка установки

```
npx prettier --version
```

Команда npx позволяет запускать установленные пакеты. Вы должны увидеть номер версии (например, 3.5.0).

Модуль 3: Создание конфигурационных файлов

Шаг 3.1. Создание файла `.prettierrc`

Prettier настраивается через конфигурационный файл.

Создайте в корне проекта файл с именем `.prettierrc` (без расширения).

Способ 1: через терминал (выполните команду):

```
echo {} > .prettierrc
```

Способ 2: вручную — создайте файл через интерфейс VS Code (правой кнопкой мыши по панели `Explorer` → `New File`).

Шаг 3.2. Настройка правил форматирования

Откройте файл `.prettierrc` и добавьте следующие настройки

```
{
  "semi": true,
  "singleQuote": true,
  "tabWidth": 2,
  "trailingComma": "es5",
  "printWidth": 100,
  "bracketSpacing": true,
  "arrowParens": "always",
  "jsxSingleQuote": false,
  "jsxBracketSameLine": false,
  "endOfLine": "auto"
}
```

Объяснение каждой настройки:

Настройка	Значение	Описание
semi	true	Добавлять точки с запятой в конце строк
singleQuote	true	Использовать одинарные кавычки вместо двойных
tabWidth	2	Размер отступа — 2 пробела
trailingComma	"es5"	Добавлять запятые в конце списков (где допустимо по стандарту ES5)
printWidth	100	Максимальная длина строки (переносить, если больше 100 символов)
bracketSpacing	true	Добавлять пробелы внутри фигурных скобок: { foo: bar }
arrowParens	"always"	Всегда добавлять скобки вокруг параметра стрелочной функции: (x) => x
jsxSingleQuote	false	В JSX использовать двойные кавычки (независимо от singleQuote)
jsxBracketSameLine	false	> в многострочном JSX переносить на новую строку
endOfLine	"auto"	Автоматически определять стиль конца строки (для Windows/Linux/macOS)

Шаг 3.3. Создание файла .prettierignore

Файл .prettierignore указывает, какие файлы и папки не нужно форматировать. Это аналогично .gitignore.

Создайте файл .prettierignore в корне проекта и добавьте:

```
node_modules
dist
build
coverage
*.min.js
.env
```

Модуль 4: Добавление скриптов форматирования в package.json

Шаг 4.1. Добавление скриптов

Откройте файл `package.json` и найдите секцию "`scripts`". Добавьте две новые строки:

```
"scripts": {
  "dev": "vite",
  "build": "tsc && vite build",
  "lint": "eslint . --ext ts,tsx --report-unused-disable-directives --max-warnings 0",
  "preview": "vite preview",
  "format": "prettier --write \"src/**/*.{js,jsx,ts,tsx,json,css,md}\"",
  "format:check": "prettier --check \"src/**/*.{js,jsx,ts,tsx,json,css,md}\""
}
```

Объяснение скриптов:

- format — записывает (--write) отформатированный код прямо в файлы
- format:check — только проверяет, были ли файлы отформатированы (полезно для CI/CD)

Шаг 4.2. Тестирование скрипта форматирования

Выполните в терминале:

```
npm run format
```

Вы должны увидеть список обработанных файлов. `Prettier` прошёлся по всем файлам в папке `src` и применил наши правила.

Модуль 5: Настройка автоматического форматирования в VS Code

Шаг 5.1. Установка расширения Prettier

Для автоматического форматирования при сохранении необходимо установить расширение для VS Code:

1. Откройте панель расширений (иконка квадратиков слева или **Ctrl+Shift+X**)
2. В поиске введите **"Prettier - Code formatter"**
3. Найдите расширение от **esbenp.prettier-vscode** (автор: Esben Petersen)
4. Нажмите **Install**

Шаг 5.2. Настройка VS Code для автоформатирования при сохранении

Нам нужно настроить VS Code так, чтобы Prettier автоматически форматировал файлы при их сохранении.

Способ 1: Через графический интерфейс:

1. Нажмите **Ctrl+,** (запятая) для открытия настроек
2. В поиске введите **"format on save"**
3. Отметьте галочку **"Editor: Format On Save"**

Способ 2: Через редактирование `settings.json` (рекомендуется для точной настройки):

1. Откройте палитру команд: **Ctrl+Shift+P**
2. Введите **"Preferences: Open Settings (JSON)"** и выберите этот пункт
3. Добавьте следующие строки в конец файла (после последней строки, но перед закрывающей фигурной скобкой) :

```
{
  // ... другие настройки ...

  "editor.formatOnSave": true,
  "editor.defaultFormatter": "esbenp.prettier-vscode",
  "prettier.requireConfig": true
}
```

Объяснение:

- editor.formatOnSave — включает автоформатирование при сохранении
- editor.defaultFormatter — указывает, какой форматтер использовать по умолчанию (Prettier)
- prettier.requireConfig — требует наличия конфигурационного файла Prettier (.prettierrc), иначе форматирование не будет работать

Ручной запуск форматирования через терминал:

```
npx prettier --write src/components/filename.tsx
```

Или для всей папки компонентов:

```
npm run format
```

Отчёт по работе

Загрузить в moodle в виде общего .docx-файла:

Скриншоты терминала с выполненными командами:

- node --version, npm --version
- npx prettier --version
- npm run format

Файлы конфигурации (скриншоты):

- `.prettierrc` с настройками
- `.prettierignore`
- Фрагмент `package.json` со скриптами форматирования
- Настройки VS Code (`settings.json`)