**Engineer**

# ESLint

лінтер для JavaScript на основі правил

## Що це таке

ESLint — це відкритий інструмент статичного аналізу, який читає вихідний код JavaScript-проєкту, звіряє його з набором налаштованих правил і повідомляє про кожне місце, де код порушує якесь із них, а прості випадки на прохання виправляє сам. У командному рядку це команда `eslint`, яку встановлюють із npm як залежність для розробки того проєкту, що його перевіряють. Ніколас Закас (Nicholas C. Zakas) почав проєкт у червні 2013 року, і сам проєкт досі називає себе конфігурованим JavaScript-лінтером.

Програми, яку перевіряє, ESLint ніколи не запускає. Парсер перетворює кожен файл на абстрактне синтаксичне дерево у форматі ESTree, а кожне правило — це невеликий модуль, який просить викликати його на певних видах вузлів: на кожній функції, кожному порівнянні, кожному `await`. Правило розглядає те, що йому передали, і якщо щось не так, повідомляє текст і місце. Деякі правила мають ще й виправлення, тобто правку, яку проєкт приймає лише тоді, коли вона усуває проблему, не змінюючи того, що робить програма, і `eslint --fix` записує такі правки просто у файли. Інші правила натомість пропонують підказки. Вони можуть змінити поведінку, тож їх застосовує людина, по одній, у редакторі.

Майже все довкола цього ядра можна замінити. Плагіни — це npm-пакети, що додають правила, спільні конфігурації, процесори або підтримку інших мов. Інший парсер дає ESLint змогу читати синтаксис, якого стандартний не розуміє, наприклад TypeScript. Процесори видобувають JavaScript з інших файлів, скажімо з блоків коду в документі Markdown, а форматери визначають, у якому вигляді буде надруковано звіт. Власну філософію проєкт описує так: кожне правило стоїть окремо, його можна вимкнути, і кожному можна окремо задати, чи воно лише попереджає, чи завершує перевірку помилкою.

Усе це задають в одному JavaScript-файлі в корені проєкту, зазвичай `eslint.config.js` або його варіантах `.mjs` і `.cjs`, який експортує масив об’єктів. Кожен об’єкт називає файли, до яких застосовується, а також правила, глобальні змінні й параметри парсера для них, і пізніші об’єкти перекривають попередні. Такий устрій, який називають flat config, став типовим у версії v9.0.0, що вийшла 5 квітня 2024 року. Версія v10.0.0 від 6 лютого 2026 року остаточно прибрала старий формат `.eslintrc` і файл `.eslintignore`.

Застосуйте поняття до рішення

Застосуйте «Що таке ESLint?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Одна команда веде ідею від ескізу до продукту, у який ваші клієнти вже можуть увійти, а не перекидає роботу між трьома підрядниками.

**Переглянути наші послуги** Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Збільшила дохід GIS-застосунку в 10 разів — 10x зростання виторгу завдяки картографічному продукту

**Збільшила дохід GIS-застосунку в 10 разів** Зв'язатися з нами →

## Чому це важливо

Лінтер виправдовує себе в одну мить: коли зупиняє зміну, перш ніж її зіллють. ESLint ухвалює це рішення через код виходу, а код виходу залежить від серйозності правил. Кожне правило має значення `"off"`, `"warn"` або `"error"`. Помилка змушує команду `eslint` завершитися з кодом 1. Попередження друкується, але команда все одно завершується з кодом 0, хіба що `--max-warnings` задає межу, а ця межа типово вимкнена. Код 2 означає зламану конфігурацію або внутрішній збій.

Тож справжнє рішення, яке розробники ухвалюють щодо ESLint, — що має означати попередження. Попередження, яке ніколи не зупиняє збірку, — це порада, а поради в терміналі прогортають, щойно їх набирається сорок рядків. Ті, хто хоче, щоб лінтер тримав межу, або ставлять кожному важливому для них правилу `"error"`, або запускають перевірку з `--max-warnings 0`. Так чи так вивід на здоровій гілці порожній, і один новий рядок стає новиною, яку хтось таки прочитає.

Виграш — у тому, що JavaScript пропускає. Мова без заперечень завантажить файл, який звертається до змінної, якої ніхто не оголосив, двічі задає той самий ключ в об’єктному літералі, порівнює значення з `NaN` через `===` або має після `return` код, що ніколи не виконається. Ці проблеми можуть не

завадити завантаженню й залишитися непоміченими або спричинити неправильну поведінку під час виконання коду, зокрема й у робочому середовищі. Правила з рекомендованого набору ESLint ловлять усі чотири ще до рев'ю, тож рецензент може читати зміну заради її задуму.

У цього є ціна. Конфігурація обростає уподобаннями, і правило, що спрацьовує на правильному коді, привчає людей не читати вивід, а це гірше, ніж не мати правила взагалі. До того ж лінтер вартий рівно стільки, скільки вартий момент, коли його запускають: якщо він працює лише в редакторі й лише на частині машин, то перевіряє частину коду частину часу.

## Як це виглядає на практиці

ESLint був одним із шести лінтерів, на яких трималася планка нуля попереджень для шести мов: там він перевіряв TypeScript поряд із golangci-lint для Go, SQLFluff для SQL, Ruff для Python, ShellCheck для shell-скриптів і markdownlint для Markdown. Планку втримало саме питання серйозності, вирішене раз і для всіх: кожна діагностика лінтера — помилка, рівня «warn», де можна сховатися, немає, а inline-придушення заборонені. Хуки pre-commit і pre-push запускають лінтери й тести; невдачі в охопленому наборі правил і тестів блокують відповідний коміт або push.

Той самий репозиторій згодом потребував правил, яких не постачає жоден опублікований пакет, — правил про те, як має працювати саме ця кодова база. П'ятнадцять власних правил ESLint утримують патерни проєкту в TypeScript поряд із 274 скриптами перевірок, 268 із яких зареєстровано в commit-хуках. Один із валідаторів стежить, щоб ніхто тихцем не додав inline-придушення. Самі guard-перевірки мають 146 власних тестів, бо зламана перевірка пропускає все, і ніхто про це не дізнається, а сторінка прямо називає ціну: комітити повільно.

Цей сайт тримає власні скрипти збірки на тій самій планці, лише в меншому масштабі. Його `eslint.config.mjs` починається з `js.configs.recommended` і додає тільки ті правила, які код уже проходив на момент вимірювання. Коментарі фіксують повторне вимірювання 15 жовтня 2026 року по всіх 199 незастарілих правилах, після якого ввімкнули ще 21 правило, і всі вони стосуються коректності, а не смаку. Кожне правило — помилка. `reportUnusedDisableDirectives`, що типово є попередженням, теж піднято до помилки, тож коментар-вимикач, який уже нічого не придушує, завершує

запуск помилкою. Скрипти оголошують глобальні змінні Node і браузера разом, бо тіла функцій, які вони передають у `page.evaluate`, виконуються в браузері, за кілька рядків від коду, що працює в Node.

## Де виникають проблеми

Здебільшого клопіт з ESLint — це лінтер, який робить саме те, що каже конфігурація, а це не завжди те, що вона каже на думку розробників.

**Збірка зелена, а термінал повний попереджень.** Попередження лишають код виходу 0, тож CI проходить, а список щотижня росте на кілька рядків, і зрештою в ньому непрочитаною лежить справжня вада. Вирішіть, які правила важливі, поставте їм `"error"`, решту вимкніть і додайте `--max-warnings 0` до перевірки, щоб попередження більше не накопичувалися.

**Коментар-вимикач переживає свою причину.** Колись рядок приглушили з поважної причини; код під ним змінився, коментар лишився, і тепер він ховає наступну проблему в цьому рядку або взагалі нічого. Задайте `linterOptions.reportUnusedDisableDirectives` значення `"error"`, щоб директиви, які нічого не придушують, завершували запуск помилкою, і вимагайте пояснення після `--` для тих, що лишаються, щоб рецензент міг їх оцінити:

```
// eslint-disable-next-line no-console -- друк і є виводом цього CLI
console.log(report);
```

**Оновлення змінює, які файли перевіряються.** Після переходу на v10 шляхи, які раніше пропускав `.eslintignore`, раптом потрапляють у звіт, або одна тека починає жити за іншими правилами. Версія 10 ігнорує і `.eslintrc.*`, і `.eslintignore`, а `eslint.config.*` шукає, починаючи з теки кожного файлу, що перевіряється, тож забута конфігурація в підтеці тепер керує цією підтекою. Перенесіть ігнорування в глобальний об’єкт `ignores`, перетворіть старий файл командою `npx @eslint/migrate-config .eslintrc.json` і перегляньте те, що вона запише, а тоді знайдіть кожен `eslint.config.*` у дереві й видаліть ті, які ніхто не збирався лишати.

**Код для браузера позначено як неоголошений.** Скрипт Node, що передає функції в браузер, отримує помилки `no-undef` на `document` і `window`, і хтось вимикає `no-undef`, а разом із ним ховаються й справжні одруківки. Оголосіть

кожне середовище, у якому код справді виконується, через `languageOptions.globals` для тих файлів, що там працюють.

**Власне правило пропускає все.** Правило, чий обробник указує на тип вузла, який ніколи не трапляється, або чия умова перевернута, просто завжди мовчить, а мовчання виглядає точнісінько як дотримання правил. Тестуйте кожне власне правило за допомогою `RuleTester` з ESLint на коді, який має пройти, і на коді, який має впасти, щоб правило, яке перестало спрацьовувати, провалило власний тест раніше, ніж підведе кодову базу.

## Що почитати далі

- Core Concepts — правила, виправлення, плагіни, парсери й процесори в тому вигляді, як їх визначає сам проєкт.
- Configuration Files — масив flat config, шаблони файлів, глобальні ігнорування і те, як об’єкти зливаються.
- Configure Rules — рівні серйозності, параметри правил і коментарі, що вимикають правило для одного рядка.
- Command Line Interface Reference — кожен прапорець, зокрема `--max-warnings`, і значення кожного коду виходу.
- ESLint v10.0.0 released — нотатки до випуску про вилучення eslintrc і новий спосіб шукати конфігурацію.
- Custom Rules — як написати правило для синтаксичного дерева і довести його роботу за допомогою `RuleTester`.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

**Дивитися, скільки це коштує** Зв'язатися з нами →

# Зробіть термінологію практичною

Запитайте, як «Що таке ESLint?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

**Переглянути наші послуги**

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

**Дбайливо зроблено з любов'ю — D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.**

**Надруковано з https://engineer.company/uk/glossary/eslint/**