Engineer

ESLint

en regeldreven linter til JavaScript

Hvad det er

ESLint er et open source-værktøj til statisk analyse, der læser kildekoden i et JavaScript-projekt, holder den op mod et sæt konfigurerede regler og melder hvert sted, hvor koden bryder en af dem, og som selv retter de enkle tilfælde, når man beder om det. På kommandolinjen hedder det `eslint`, og det installeres fra npm som en udviklingsafhængighed i det projekt, det skal tjekke. Nicholas C. Zakas startede projektet i juni 2013, og det præsenterer sig stadig som en JavaScript-linter, man kan konfigurere.

Selve programmet bliver aldrig kørt. En parser omdanner hver fil til et abstrakt syntakstræ i ESTree-formatet, og hver regel er et lille modul, der beder om at blive kaldt ved bestemte slags knuder: hver funktion, hver sammenligning, hvert `await`. Reglen ser på det, den får udleveret, og hvis noget er galt, melder den en besked og en placering. Nogle regler har en rettelse med, en ændring som projektet kun godtager, hvis den fjerner problemet uden at ændre, hvad programmet gør, og `eslint --fix` skriver dem direkte ind i filerne. Andre regler kommer i stedet med forslag. De kan ændre adfærd og bliver derfor anvendt ét ad gangen, af et menneske, i en editor.

Næsten alt omkring kernen kan skiftes ud. Plugins er npm-pakker, der tilføjer regler, delbare konfigurationer, processorer eller understøttelse af andre sprog. En anden parser lader ESLint læse syntaks, som standardparseren ikke forstår, for eksempel TypeScript. Processorer trækker JavaScript ud af andre filer, som kodeblokkene i et Markdown-dokument, og formatters bestemmer, hvordan rapporten skrives ud. I projektets egen beskrivelse af sin filosofi står hver regel for sig selv, kan slås fra og kan sættes til at advare eller fejle efter behov.

Det hele samles i én JavaScript-fil i roden af projektet, typisk `eslint.config.js` eller varianterne `.mjs` og `.cjs`, som eksporterer et array af objekter. Hvert objekt angiver, hvilke filer det gælder for, og hvilke regler, globale variabler og parserindstillinger der gælder for dem, og senere objekter overskriver tidligere. Den opbygning, kaldet flat config, blev standard med ESLint v9.0.0 den 5. april 2024. ESLint v10.0.0, udgivet den 6. februar 2026, fjernede det gamle `.eslintrc`-format og `.eslintignore`-filen helt.

Brug begrebet i en beslutning

Brug »Hvad er ESLint?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Ét team, der tager en idé fra skitse til noget, jeres kunder kan logge ind på, i stedet for at kaste den mellem tre leverandører.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har ført en GIS-kortapp til 10x omsætningsvækst — 10x omsætningsvækst, drevet af kortproduktet

Har ført en GIS-kortapp til 10x omsætningsvækst Kontakt os →

Hvorfor det betyder noget

En linter viser især sit værd, når den stopper en ændring, før den bliver merget. ESLint træffer den afgørelse gennem sin exitkode, og exitkoden følger reglernes alvorsgrad. Hver regel står til `"off"`, `"warn"` eller `"error"`. En fejl får kommandoen til at afslutte med 1. En advarsel bliver skrevet ud, og kommandoen afslutter alligevel med 0, medmindre `--max-warnings` sætter et loft, og det loft er slået fra som standard. Exitkode 2 er forbeholdt en ødelagt konfiguration eller et internt nedbrud.

Den egentlige beslutning, et team træffer med ESLint, er derfor, hvad en advarsel skal betyde. En advarsel, der aldrig får et build til at fejle, er et råd, og råd i en terminal bliver scrollet forbi, så snart der står fyrre linjer af dem. Teams, der vil have linteren til at stå fast, sætter enten hver regel, de går op i, til `"error"`, eller kører kvalitetsporten med `--max-warnings 0`. Med begge fremgangsmåder er outputtet på en sund branch blankt, og en enkelt ny linje er en nyhed, som nogen faktisk læser.

Gevinsten ligger i det, JavaScript lader slippe igennem. Sproget indlæser uden protester en fil, der bruger en variabel, ingen har erklæret, definerer den samme nøgle to gange i et objekt, sammenligner en værdi med `NaN` via `===` eller har kode efter et `return`, som aldrig kan nås. Filen kan blive indlæst med disse

problemer, som derefter kan forblive ubemærkede eller føre til forkert adfærd, når koden køres, også i produktion. Reglerne i ESLints anbefalede sæt fanger alle fire før review, så den, der læser ændringen, kan bruge tiden på designet.

Der er en pris. En konfiguration samler holdninger op, og en regel, der slår ud på korrekt kode, lærer folk at holde op med at læse outputtet, hvilket er værre end slet ingen regel. En linter er desuden kun så god som det tidspunkt, den kører på: kører den kun i en editor på nogle af maskinerne, tjekker den noget af koden noget af tiden.

Sådan ser det ud i praksis

ESLint var én af seks lintere bag en zero-warnings-grænse på tværs af seks sprog, hvor det tjekkede TypeScript ved siden af golangci-lint til Go, SQLFluff til SQL, Ruff til Python, ShellCheck til shellscripts og markdownlint til Markdown. Det, der fik grænsen til at holde, var netop spørgsmålet om alvorsgrad, afgjort én gang for alle: hver linterdiagnostik er en fejl, der findes intet »warn«-niveau at gemme sig i, og inline-undertrykkelser er forbudt. Pre-commit- og pre-push-hooks kører linterne og testene; fejl inden for det dækkede sæt af regler og test blokerer den pågældende commit eller push.

Samme repository fik siden brug for regler, som ingen udgivet pakke indeholder: dem om, hvordan netop den kodebase skal fungere. Femten egne ESLint-regler holder husets mønstre på plads i TypeScript ved siden af 274 check-scripts, hvoraf 268 er registreret i commit-hooks. Én af valideringerne tjekker, at ingen i al stilhed har tilføjet en inline-undertrykkelse. Selve guard-laget har 146 test, fordi en ødelagt guard lader alt passere, uden at nogen opdager det, og siden siger prisen ligeud: det er langsomt at committe.

Dette site holder sine egne build-scripts op mod samme standard i mindre skala. Dets `eslint.config.mjs` tager udgangspunkt i `js.configs.recommended` og tilføjer kun regler, som koden allerede bestod, da de blev målt. Kommentarerne beskriver en ny måling den 15. oktober 2026 over alle 199 regler, der ikke er udfasede, og den slog 21 regler mere til, alle om korrekthed frem for smag. Hver regel er en fejl. `reportUnusedDisableDirectives`, der som standard er en advarsel, er også hævet til fejl, så en disable-kommentar, der ikke længere undertrykker noget, får kørslen til at fejle. Scriptene erklærer både Node- og browserglobaler, fordi de funktionskroppe, de giver videre til `page.evaluate`, kører inde i en browser, få linjer fra kode, der kører i Node.

Hvor det går galt

Det meste besvær med ESLint er linteren, der gør præcis, hvad konfigurationen siger, og det er ikke altid det, teamet tror, den siger.

Buildet er grønt, og terminalen er fuld af advarsler. Advarsler lader exitkoden stå på 0, så CI går igennem, mens listen vokser med et par linjer om ugen, og til sidst står en rigtig fejl i den uden at blive læst. Beslut, hvilke regler der betyder noget, sæt dem til `"error"`, slå resten fra, og føj `--max-warnings 0` til kvalitetsporten, så advarsler holder op med at hobe sig op.

En disable-kommentar overlever sin grund. En linje blev gjort tavs af en god grund engang. Koden under den ændrede sig, kommentaren blev stående, og nu skjuler den det næste problem på linjen eller slet ingenting. Sæt `linterOptions.reportUnusedDisableDirectives` til `"error"`, så direktiver, der intet undertrykker, får kørslen til at fejle, og kræv en begrundelse efter `--` på dem, der bliver tilbage, så den, der reviewer, kan vurdere dem:

```
// eslint-disable-next-line no-console -- udskriften er CLI'ets output
console.log(report);
```

En opgradering ændrer, hvilke filer der bliver tjekket. Efter skiftet til v10 bliver stier, som `.eslintignore` før sprang over, pludselig rapporteret, eller én mappe begynder at følge andre regler. Version 10 respekterer hverken `.eslintrc.*` eller `.eslintignore`, og den leder efter `eslint.config.*` fra mappen for hver fil, der lintes, så en glemt konfiguration i en undermappe nu styrer netop den undermappe. Flyt ignoreringerne ind i et globalt `ignores`-objekt, konvertér den gamle fil med `npx @eslint/migrate-config .eslintrc.json` og gennemgå resultatet, og find derefter hver `eslint.config.*` i træet og fjern dem, ingen havde tænkt sig at beholde.

Browserkode bliver meldt som udefineret. Et Node-script, der giver funktioner videre til en browser, får `no-undef`-fejl på `document` og `window`, og nogen slår `no-undef` fra, hvilket også skjuler ægte tastefejl. Erklær hvert miljø, koden faktisk kører i, gennem `languageOptions.globals` for de filer, der kører der.

En egen regel lader alt passere. En regel, hvis visitor peger på en knudetype, der aldrig forekommer, eller hvis betingelse er vendt om, melder simpelthen aldrig noget, og tavshed ligner overholdelse til forveksling. Test hver egen

regel med ESLints `RuleTester` med kode, der skal bestå, og kode, der skal fejle, så en regel, der holder op med at melde, falder i sin egen test, før den svigter kodebasen.

Læs mere

- Core Concepts — regler, rettelser, plugins, parsere og processorer, som projektet selv definerer dem.
- Configuration Files — flat config-arrayet, filmønstre, globale ignoreringer og hvordan objekterne flettes sammen.
- Configure Rules — alvorsgrader, regelindstillinger og de kommentarer, der slår en regel fra for en enkelt linje.
- Command Line Interface Reference — hvert flag, heriblandt `--max-warnings`, og hvad hver exitkode betyder.
- ESLint v10.0.0 released — udgivelsesnoterne om fjernelsen af eslintrc og den nye måde at finde konfigurationen på.
- Custom Rules — at skrive en regel mod syntakstræet og bevise den med `RuleTester`.

Ordliste

Vælg ud fra jeres faktiske vilkår

Beskriv arbejdsgangen, de eksisterende grænseflader og de brugere, der har brug for ændringen. Vi kan definere en testbar leverance og en overdragelse, der gør teamet i stand til at vedligeholde løsningen.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er ESLint?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

welcome@engineer.company +45 50 10 61 84 København, Danmark

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/eslint/