Engineer

ESLint

en regelstyrt linter for JavaScript

Hva det er

ESLint er et verktøy med åpen kildekode for statisk analyse av JavaScript. Det leser kildekoden, tester den mot et sett regler som er satt opp for prosjektet, og melder fra om hvert sted koden bryter en av dem, og de enkleste tilfellene retter det selv når du ber om det. På kommandolinjen heter det `eslint`, og det installeres fra npm som en utviklingsavhengighet i det prosjektet det sjekker. Nicholas C. Zakas startet det i juni 2013, og prosjektet presenterer seg fortsatt som en konfigurerbar linter for JavaScript.

Selve programmet blir aldri kjørt. En parser gjør hver fil om til et abstrakt syntakstre i formatet ESTree, og hver regel er en liten modul som ber om å bli kalt ved bestemte slags noder: hver funksjon, hver sammenligning, hver `await`. Regelen ser på det den får, og er noe galt, melder den en beskjed og et sted i koden. Noen regler har også en retting (fix), en endring prosjektet bare godtar når den løser problemet uten å endre hva programmet gjør, og `eslint --fix` skriver slike rettinger inn i filene. Andre regler tilbyr forslag (suggestions) i stedet. Et forslag kan endre oppførselen, og derfor tas forslagene i bruk ett om gangen, av et menneske, i editoren.

Nesten alt rundt den kjernen kan byttes ut. Utvidelser (plugins) er programvarepakker fra npm som legger til regler, delbare konfigurasjoner, prosessorer eller støtte for andre språk. En annen parser lar ESLint lese syntaks som standardparseren ikke forstår, f.eks. TypeScript. Prosessorer henter JavaScript ut av andre filer, f.eks. kodeblokkene i et Markdown-dokument, og formaterere endrer hvordan rapporten skrives ut. Prosjektet beskriver selv filosofien sin slik: hver regel står for seg selv, kan slås av og kan settes til å advare eller til å feile, helt på egne premisser.

Alt dette settes opp i én JavaScript-fil i roten av prosjektet, som regel `eslint.config.js` eller variantene `.mjs` og `.cjs`, og filen eksporterer en liste (array) med objekter. Hvert objekt sier hvilke filer det gjelder for, og hvilke regler, globale variabler og parseropsjoner som skal gjelde for dem, og senere objekter overstyrer tidligere. Dette oppsettet, kalt flat config, ble standard i ESLint v9.0.0 den 5. april 2024. ESLint v10.0.0, utgitt 6. februar 2026, fjernet det eldre formatet `.eslintrc` og filen `.eslintignore` helt.

Bruk begrepet i en beslutning

Bruk «Hva er ESLint?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Ett team som tar en idé fra skisse til noe kundene dine kan logge inn på, i stedet for at den sendes videre mellom tre leverandører.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Ledet en GIS-kartapp til 10 ganger mer omsetning — 10x inntektsvekst, ledet fra kartproduktet

Ledet en GIS-kartapp til 10 ganger mer omsetning Ta kontakt →

Hvorfor det er viktig

En linter gjør seg fortjent til plassen sin i ett bestemt øyeblikk: når den hindrer at en endring blir slått sammen. ESLint tar den avgjørelsen gjennom avslutningskoden sin (exit code), og avslutningskoden følger alvorlighetsgraden. Hver regel er satt til `"off"`, `"warn"` eller `"error"`. En feil får kommandoen til å avslutte med 1. En advarsel skrives ut, og kommandoen avslutter likevel med 0, med mindre `--max-warnings` setter et tak, og det taket er slått av som standard. Avslutningskode 2 er satt av til en ødelagt konfigurasjon eller et internt krasj.

Den avgjørelsen et team egentlig tar med ESLint, er altså hva en advarsel skal bety. En advarsel som aldri får et bygg til å feile, er et råd, og råd i en terminal blir rullet forbi så snart det er førti linjer av dem. Team som vil at linteren skal holde en grense, setter enten hver regel de bryr seg om, til `"error"`, eller kjører kvalitetsporten med `--max-warnings 0`. Uansett hvilken vei de går, lintes en sunn gren helt ren, og én enkelt ny linje i utskriften er en nyhet som noen faktisk leser.

Gevinsten ligger i det JavaScript slipper gjennom. Språket laster uten protester en fil som viser til en variabel ingen har deklarert, definerer den samme nøkkelen to ganger i en objektliteral, sammenligner en verdi med `NaN` ved hjelp av `===` eller har kode etter en `return` som aldri kan kjøre. Slike problemer kan bestå etter innlasting og forbli ubemerket eller gi feil oppførsel når koden

kjøres, også i produksjon. Reglene i ESLints anbefalte sett fanger alle fire før kodegjennomgangen, og da kan den som går gjennom koden, vurdere utformingen.

Det har en pris. En konfigurasjon samler opp meninger, og en regel som slår ut på korrekt kode, lærer folk å slutte å lese utskriften, og det er verre enn å ikke ha regelen. En linter er dessuten bare så god som øyeblikket den kjører i: en som bare kjører i en editor, på noen av maskinene, sjekker noe av koden noe av tiden.

Slik ser det ut i praksis

ESLint var én av seks lintere bak en kvalitetsstandard med null advarsler for seks språk, der den sjekket TypeScript ved siden av golangci-lint for Go, SQLFluff for SQL, Ruff for Python, ShellCheck for skallskript og markdownlint for Markdown. Det som fikk standarden til å holde, var spørsmålet om alvorlighetsgrad ovenfor, avgjort én gang for alle: hver linterdiagnostikk er en feil, det finnes ikke noe «warn»-nivå å gjemme seg i, og inline-undertrykkelser er forbudt. Pre-commit- og pre-push-hooks kjører linterne og testene; feil innenfor det dekkede regel- og testsettet blokkerer den aktuelle committen eller pushen.

Det samme repoet trengte etter hvert regler som ingen publisert pakke leverer, de som handler om hvordan akkurat den kodebasen er ment å virke. Femten egne ESLint-regler holder husets mønstre på plass i TypeScript, ved siden av 274 kontrollskript, 268 av dem registrert i commit-hookene. Ett av kontrollskriptene sjekker at ingen i det stille har lagt inn en inline-undertrykkelse. Vaktene har 146 egne tester, fordi en ødelagt vakt slipper alt gjennom uten at noen finner det ut, og siden sier prisen rett ut: det går tregt å committe.

Dette nettstedet holder byggeskriptene sine til den samme standarden i mindre skala. Filen `eslint.config.mjs` tar utgangspunkt i `js.configs.recommended` og legger bare til regler som koden allerede besto da de ble målt. Kommentarene i filen dokumenterer en ny måling 15. oktober 2026 over alle de 199 reglene som ikke er utfaset, og den slo på 21 regler til som handler om korrekthet og ikke om smak. Hver regel er en feil. `reportUnusedDisableDirectives`, som er en advarsel som standard, er også hevet til feil, så en disable-kommentar som har sluttet å undertrykke noe, får kjøringen til å feile. Skriptene deklarerer globale variabler for både Node og nettleser, fordi funksjonskroppene de gir videre til `page.evaluate`, kjører inne i en nettleser, noen få linjer unna kode som kjører i Node.

Hvor det går galt

Det meste av trøbbelet med ESLint er at linteren gjør nøyaktig det konfigurasjonen sier, og det er ikke alltid det teamet tror den sier.

Bygget er grønt, og terminalen er full av advarsler. Advarsler lar avslutningskoden stå på 0, så CI går gjennom mens listen vokser med noen linjer i uka, og etter en stund ligger en ekte programvarefeil i den uten at noen har lest den. Bestem hvilke regler som betyr noe, sett dem til `"error"`, slå av resten, og legg `--max-warnings 0` inn i kvalitetsporten, slik at advarsler ikke lenger kan hope seg opp.

En disable-kommentar lever lenger enn grunnen sin. En linje ble en gang gjort taus av en god grunn. Koden under endret seg, kommentaren ble stående, og nå skjuler den det neste problemet på den linjen, eller ingenting i det hele tatt. Sett `linterOptions.reportUnusedDisableDirectives` til `"error"`, så direktiver som ikke undertrykker noe, får kjøringen til å feile, og krev en begrunnelse etter `--` på dem som blir igjen, slik at den som går gjennom koden, kan vurdere dem:

```
// eslint-disable-next-line no-console -- printing is this CLI's output
console.log(report);
```

En oppgradering endrer hvilke filer som sjekkes. Etter overgangen til v10 blir stier som `.eslintignore` pleide å hoppe over, plutselig rapportert, eller én mappe begynner å følge andre regler. Versjon 10 tar ikke lenger hensyn til `.eslintrc.*` eller `.eslintignore`, og den leter etter `eslint.config.*` med start i katalogen til hver fil som lintes, så en glemt konfigurasjonsfil i en undermappe styrer nå den undermappen. Flytt ignoreringene inn i en global `ignores`-oppføring, konverter den gamle filen med `npx @eslint/migrate-config .eslintrc.json` og gå gjennom det kommandoen skriver, og list så opp hver `eslint.config.*` i treet og fjern dem ingen mente å beholde.

Nettleserkode blir meldt som udefinert. Et Node-skript som gir funksjoner videre til en nettleser, får `no-undef`-feil på `document` og `window`, og så slår noen av `no-undef`, som også skjuler ekte skrivefeil. Deklarer hvert miljø koden faktisk kjører i, gjennom `languageOptions.globals`, for de filene som kjører der.

En egen regel slipper alt gjennom. En regel der visitoren navngir en nodetype som aldri forekommer, eller der betingelsen er snudd, melder rett og slett aldri noe, og stillhet ser nøyaktig ut som etterlevelse. Test hver egen regel med ESLints `RuleTester`, med kode som må bestå og kode som må feile, så en regel som slutter å melde fra, feiler sin egen test før den svikter kodebasen.

Les mer

- Core Concepts — regler, rettinger, utvidelser, parsere og prosessorer, slik prosjektet selv definerer dem.
- Configuration Files — listen i flat config, filmønstre, globale ignoreringer og hvordan objektene slås sammen.
- Configure Rules — alvorlighetsgrader, regelopsjoner og kommentarene som slår av en regel for én linje.
- Command Line Interface Reference — hvert flagg, blant dem `--max-warnings`, og hva hver avslutningskode betyr.
- ESLint v10.0.0 released — utgivelsesnotatene om fjerningen av eslintrc og det nye oppslaget etter konfigurasjon.
- Custom Rules — å skrive en regel mot syntakstreet og bevise at den virker, med `RuleTester`.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er ESLint?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/eslint/