Engineer

ESLint

en regelstyrd linter för JavaScript

Vad det är

ESLint är ett verktyg med öppen källkod för statisk analys som läser källkod i JavaScript, prövar den mot en konfigurerad uppsättning regler och rapporterar varje ställe där koden bryter mot en av dem. De enklare fallen rättar verktyget själv när det ombeds. På kommandoraden är det kommandot `eslint`, som installeras från npm som ett utvecklingsberoende i det projekt som det granskar. Nicholas C. Zakas startade det i juni 2013, och projektet presenterar sig fortfarande som en konfigurerbar linter för JavaScript.

Det kör aldrig programmet. En parser gör om varje fil till ett abstrakt syntaxträd i formatet ESTree, och varje regel är en liten modul som ber om att bli anropad vid vissa slags noder: varje funktion, varje jämförelse, varje `await`. Regeln tittar på det den får och rapporterar, om något är fel, ett meddelande och en plats. Vissa regler har också en rättelse, en ändring som projektet bara godtar när den åtgärdar problemet utan att ändra vad programmet gör, och `eslint --fix` tillämpar dem på filerna. Andra regler erbjuder i stället förslag, som kan ändra beteendet och därför tillämpas ett i taget, av en person, i en editor.

Nästan allt runt den kärnan går att byta ut. Insticksmoduler är npm-paket som lägger till regler, delbara konfigurationer, processorer eller stöd för andra språk. En annan parser låter ESLint läsa syntax som standardparsern inte förstår, t.ex. TypeScript. Processorer plockar ut JavaScript ur andra filer, till exempel kodblocken i ett Markdown-dokument, och formaterare ändrar hur rapporten skrivs ut. Projektets egen beskrivning av sin filosofi är att varje regel står för sig själv, kan stängas av och kan ställas in på att varna eller att fallera på sina egna villkor.

Allt det här ställs in i en enda JavaScript-fil i projektets rot, oftast `eslint.config.js` eller varianterna `.mjs` och `.cjs`, som exporterar en array av objekt. Varje objekt anger vilka filer det gäller och vilka regler, globala variabler och parserinställningar som gäller för dem, och senare objekt åsidosätter tidigare. Den uppställningen, som kallas flat config, blev standard i ESLint v9.0.0 den 5 april 2024. ESLint v10.0.0, som släpptes den 6 februari 2026, tog helt bort det äldre formatet `.eslintrc` och filen `.eslintignore`.

Använd begreppet i ett beslut

Använd ”Vad är ESLint?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Ett team som tar en idé från en skiss till något dina kunder kan logga in i, i stället för att lämna den vidare mellan tre leverantörer.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Ledde GIS-kartappen till 10 gånger högre intäkter — 10x intäktstillväxt, med kartprodukten i spetsen

Ledde GIS-kartappen till 10 gånger högre intäkter Hör av er →

Varför det spelar roll

En linter gör sig förtjänt av sin plats i ett enda ögonblick: när den stoppar en ändring från att slås ihop. ESLint fattar det beslutet genom sin slutkod, och slutkoden följer allvarlighetsgraden. Varje regel är inställd på `"off"`, `"warn"` eller `"error"`. Ett fel får kommandot att avslutas med slutkod 1. En varning skrivs ut och kommandot avslutas ändå med 0, såvida inte `--max-warnings` sätter ett tak, och det taket är avstängt som standard. Slutkod 2 är reserverad för en trasig konfiguration eller en intern krasch.

Med ESLint handlar teamets egentliga beslut alltså om vad en varning betyder. En varning som aldrig stoppar ett bygge är ett råd, och råd i en terminal scrollas förbi så snart det finns fyrtio rader av dem. Team som vill att lintern ska hålla en gräns ställer antingen varje regel de bryr sig om på `"error"` eller kör kvalitetsgrinden med `--max-warnings 0`. I båda fallen går en frisk gren igenom lintningen ren, och en enda ny rad blir en nyhet som någon läser.

Vinsten ligger i det som JavaScript släpper igenom. Språket laddar gärna en fil som hänvisar till en variabel som ingen har deklarerat, som definierar samma nyckel två gånger i en objektliteral, som jämför ett värde med `NaN` med hjälp av `===` eller som lämnar kod efter ett `return` som aldrig kan köras. Problemen kan finnas kvar efter inläsningen och förbli obemärkta eller ge felaktigt beteende

när koden körs, även i produktion. Reglerna i den rekommenderade uppsättningen i ESLint fångar alla fyra före granskningen, och då kan granskaren koncentrera sig på utformningen.

Det finns kostnader. En konfiguration samlar på sig åsikter, och en regel som klagar på korrekt kod lär folk att sluta läsa utdatan, vilket är värre än att inte ha regeln alls. En linter är också bara så bra som det ögonblick då den körs: en som bara körs i en editor, på vissa maskiner, granskar en del av koden en del av tiden.

Så ser det ut i praktiken

ESLint var en av sex linters bakom en ribba på noll varningar i sex språk, där den granskade TypeScript vid sidan av golangci-lint för Go, SQLFluff för SQL, Ruff för Python, ShellCheck för skalskript och markdownlint för Markdown. Det som fick ribban att hålla var frågan om allvarlighetsgrad ovan, avgjord en gång för alla: varje diagnostikmeddelande från en linter är ett fel, det finns ingen ”warn”-nivå att gömma sig i, och undertryckningar direkt i koden är förbjudna. Hookar för pre-commit och pre-push kör linters och tester; fel inom den täckta regel- och testuppsättningen blockerar motsvarande commit eller push.

Samma repo behövde sedan regler som inget publicerat paket levererar, de som handlar om hur just den kodbasen är tänkt att fungera. Femton egna ESLint-regler håller fast vid husets mönster i TypeScript, och de är bara en del av repots vaktmotor: av dess 274 kontrollskript är 268 kopplade till commit-hookarna. En av validerarna kontrollerar att ingen i tysthet har lagt in en undertryckning direkt i koden. Vakterna är i sin tur täckta av 146 egna tester, för en vakt som har gått sönder låter varje fel passera utan att någon ser det, och sidan anger priset rakt ut: det går långsamt att committa.

Den här webbplatsen håller sina egna byggskript till samma standard i mindre skala. Dess `eslint.config.mjs` utgår från `js.configs.recommended` och lägger bara till regler som koden redan klarade när de mättes. Kommentarerna dokumenterar en ny mätning 2026-10-15 över alla 199 regler som inte är utfasade, och den slog på 21 regler till som handlar om korrekthet snarare än smak. Varje regel är ett fel. `reportUnusedDisableDirectives`, som är en varning som standard, är också uppgraderad till ett fel, så en kommentar som stänger av en regel men inte längre undertrycker något får körningen att misslyckas. Skripten deklarerar globala variabler för både Node och webbläsare, eftersom de funktionskroppar som de lämnar till `page.evaluate` körs inuti en webbläsare, några rader från kod som körs i Node.

Var det går fel

Det mesta besväret med ESLint är att lintern gör exakt vad konfigurationen säger, vilket inte alltid är vad teamet tror att den säger.

Bygget är grönt och terminalen är full av varningar. Varningar lämnar slutkoden på 0, så kontinuerlig integration går igenom medan listan växer med några rader i veckan, och efter ett tag ligger en verklig defekt i den, oläst. Bestäm vilka regler som spelar roll, ställ dem på `"error"`, stäng av resten och lägg till `--max-warnings 0` i kvalitetsgrinden, så att varningar inte längre kan hopa sig.

En kommentar som stänger av en regel överlever sitt skäl. En rad tystades en gång av goda skäl; koden under den ändrades, kommentaren stannade, och nu döljer den nästa problem på raden eller ingenting alls. Ställ `linterOptions.reportUnusedDisableDirectives` på `"error"` så att direktiv som inte undertrycker något får körningen att misslyckas, och kräv ett skäl efter `--` på dem som blir kvar, så att en granskare kan bedöma dem:

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

En uppgradering ändrar vilka filer som granskas. Efter flytten till v10 rapporteras plötsligt sökvägar som `.eslintignore` brukade hoppa över, eller så börjar en mapp följa andra regler. Version 10 tar inte längre hänsyn till `.eslintrc.*` eller `.eslintignore`, och den letar efter `eslint.config.*` med början i katalogen för varje fil som lintas, så en kvarglömd konfiguration i en undermapp styr nu den undermappen. Flytta undantagen till en global post för `ignores`, konvertera den gamla filen med `npx @eslint/migrate-config .eslintrc.json` och granska det som skrivs, och lista sedan varje `eslint.config.*` i trädet och ta bort dem som ingen tänkte behålla.

Kod för webbläsaren rapporteras som odefinierad. Ett Node-skript som lämnar funktioner till en webbläsare får fel från `no-undef` på `document` och `window`, och någon stänger av `no-undef`, vilket också döljer äkta stavfel. Deklarera varje miljö som koden verkligen körs i genom `languageOptions.globals`, för de filer som körs där.

En egen regel släpper igenom allt. En regel vars besökare namnger en nodtyp som aldrig förekommer, eller vars villkor är omvänt, rapporterar helt enkelt aldrig något, och tystnad ser exakt ut som regelefterlevnad. Testa varje egen regel med `RuleTester` från ESLint, med kod som måste gå igenom och kod som måste fångas, så att en regel som slutar rapportera fallerar i sitt eget test innan den sviker kodbasen.

Läs mer

- Core Concepts — regler, rättelser, insticksmoduler, parsrar och processorer, så som projektet definierar dem.
- Configuration Files — arrayen i flat config, filmönster, globala undantag och hur objekten slås samman.
- Configure Rules — allvarlighetsgrader, regelinställningar och de kommentarer som stänger av en regel för en rad.
- Command Line Interface Reference — varje flagga, bl.a. `--max-warnings`, och vad varje slutkod betyder.
- ESLint v10.0.0 released — borttagandet av eslintrc och det nya sättet att hitta konfigurationen, i versionsnyheterna.
- Custom Rules — hur en regel skrivs mot syntaxträdet och bevisas med `RuleTester`.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är ESLint?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

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

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/eslint/