

Sidebar

Programming · VS Code ESLint Cheat Sheet

The ESLint extension's settings and commands, plus the ESLint config file, inline comments and command line. Checked against the extension (3.0.34) installed here and ESLint's documentation. Sheet 17 in the VS Code series.

SETUP & COMMANDS

- `Extension ID` `dbaeumer.vscode-eslint` (version 3.0.34 here); it uses the ESLint installed in your project
- `npm init @eslint/config` Create a starter config file and install what it needs
- `ESLint: Fix all auto-fixable Problems` Command Palette: apply every automatic fix in the current file
- `ESLint: Create ESLint configuration` Command Palette: start the config wizard
- `ESLint: Show Output Channel` Open the extension's log: the first place to look when linting fails
- `ESLint: Restart ESLint Server` Restart after installing ESLint or editing the config
- `ESLint: Revalidate all open files` Re-run the linter on everything that is open
- `ESLint: Migrate Settings` Convert older `eslint.autoFixOnSave` -style settings to the current ones
- `Ctrl+. / F8 / Ctrl+Shift+M` Quick fix / next problem / Problems panel

CORE EXTENSION SETTINGS

- `eslint.enable` Turn the extension on or off
- `eslint.run` Lint `onType` (default) or `onSave`
- `eslint.validate` Language IDs to validate
- `eslint.probe` Language IDs to probe (JavaScript, TypeScript)
- `eslint.quiet` Show errors or warnings in the console
- `eslint.ignoreUntitled` Skip linting untitled files
- `eslint.onIgnoredFiles` of `eslint.validate` to ignore files
- `eslint.problems.shortenToSize` Shorten problem messages to fit
- `eslint.packageManager` npm or yarn
- `eslint.nodePath` Folder add the node path
- `eslint.runtime / eslint.exec` Path to the ESLint CLI
- `eslint.nodeEnv` Value of `NODE_ENV` when running ESLint
- `eslint.debug / eslint.trace` Enable debug or trace logging

CONFIGURATION DISCOVERY

- `eslint.useFlatConfig` Force the flat config format on or off; `null` (default) follows your ESLint version (needs ESLint 8.57 or later)
- `eslint.experimental.useFlatConfig` Older switch for ESLint 8.21 to 8.56 only (default `false`)
- `eslint.workingDirectories` Which folders count as project roots, e.g. `[{ "mode": "auto" }]`; important in monorepos
- `eslint.options` Extra options passed to ESLint like command-line flags, e.g. `{ "overrideConfigFile": "path" }` (default `{}`)
- `eslint.useESLintClass` Use the newer ESLint class API with older ESLint versions (default `false`)
- `eslint.useRealpaths` Resolve symlinks to real paths (default `false`)

FIX ON SAVE

- `editor.codeActionsOnSave` Code actions on save
- `values` `explicit` (only when `editor.codeActionsOnSave` is `explicit`)
- `eslint.codeActionsOnSave.migrate` Migrate settings on save
- `eslint.codeActionsOnSave.rules` Rules to migrate on save
- `eslint.codeActionsOnSave.format` Format on save
- `eslint.format.enable` Use the formatter on save
- `eslint.autoFixOnSave` Old version of `eslint.autoFixOnSave` (deprecated)
- `eslint.migration.2_x` on

QUICK FIX MENU & SEVERITY

<code>eslint.codeAction.disableRuleComment</code>	Offer <i>Disable rule for this line</i> ; default <code>{ "enable": true, "location": "separateLine", "commentStyle": "line" }</code>
<code>eslint.codeAction.showDocumentation</code>	Offer <i>Show documentation for rule</i> (default <code>{ "enable": true }</code>)
<code>eslint.rules.customizations</code>	Change how severe a rule looks in the editor only, e.g. <code>{ "rule": "no-console", "severity": "warn" }</code>
<code>eslint.notebooks.rules.customizations</code>	The same, for text cells in notebooks

PERFORMANCE, TASKS & SU

<code>eslint.timeBudget.onValida</code>	
<code>eslint.timeBudget.onFixes</code>	
<code>eslint.lintTask.enable</code>	Ac
<code>eslint.lintTask.command /</code>	
<code>eslint.bulkSuppression.ena</code>	
<code>eslint.bulkSuppression.sev</code>	

CONFIG FILE (ESLINT.CONFIG.*)

File names	<code>eslint.config.js</code> , <code>.mjs</code> , <code>.cjs</code> , <code>.ts</code> , <code>.mts</code> , <code>.cts</code> in the project root (TypeScript needs <code>jiti</code> or Node 22.13+)
Structure	Export an array of config objects; <code>defineConfig([...])</code> from <code>eslint/config</code> is the helper
files / ignores	Glob patterns a config object applies to / excludes
globalIgnores([...])	A config object with only <code>ignores</code> applies to the whole project, e.g. <code>dist/</code>
extends	Configs to build on, such as a plugin's recommended set
languageOptions	<code>ecmaVersion</code> , <code>sourceType</code> , <code>globals</code> , <code>parser</code> , <code>parserOptions</code>
linterOptions	<code>reportUnusedDisableDirectives</code> , <code>noInlineConfig</code> , <code>reportUnusedInlineConfigs</code>
plugins / processor / settings	Add plugins, a processor for non-JS files, and shared settings
rules	<code>{ "semi": ["error", "never"] }</code> : a severity, optionally in an array with options
Severity values	"off" or 0, "warn" or 1, "error" or 2

INLINE COMMENTS

<code>// eslint-disable-next-lin</code>	
<code>// eslint-disable-line rul</code>	
<code>/* eslint-disable rule */</code>	
<code>/* eslint-disable */</code>	At th
<code>-- reason</code>	Add an explanatio
<code>// eslint-disa</code>	
<code>/* eslint rule: "error" */</code>	
<code>reportUnusedDisableDirecti</code>	

COMMAND LINE (NPX ESLINT ...)

<code>. --fix</code>	Lint the project and fix what can be fixed
<code>--max-warnings 0</code>	Exit with an error if there are any warnings (useful in CI)
<code>--cache</code>	Remember results so the next run is faster
<code>--format json</code>	Choose an output format (also <code>stylish</code> , <code>html</code>)
<code>--config path</code>	Use a specific config file
<code>--no-warn-ignored</code>	Do not warn about files that are ignored
<code>--debug</code>	Print detailed debugging output
<code>--print-config file</code>	Show the settings ESLint would use for one file

A REASONABLE STARTER SET

<code>"editor.codeActionsOnSave"</code>	
<code>"eslint.run": "onType"</code>	In
<code>"eslint.rules.customizatio</code>	
<code>"eslint.workingDirectories</code>	
<code>eslint-config-prettier</code>	Ac

ESLINT ISN'T DOING ANYTHING? CHECK THESE IN ORDER

1) Is ESLint installed in the project (npm ls eslint)? The extension uses the project's copy. 2) Is there a config file (eslint.config.js or similar)? 3) Run *ESLint: Show Output Channel*: errors about the config or about a missing module show up there. 4) After installing or changing config, run *ESLint: Restart ESLint Server*. 5) In a project with several packages, set `eslint.workingDirectories`. 6) Make sure the language is covered by `eslint.validate` or `eslint.probe`.

ESLINT AND PRETTIER TOGETHER

Let Prettier do the layout (spacing, line breaks) and ESLint find real problems. Use `eslint-config-prettier` to switch off the ESLint rules that would argue with Prettier, keep `eslint.format.enable` off, and use `source.fixAll.eslint` on save for ESLint's automatic fixes. With both on save, check the result once: the two should not keep changing each other's output.

