

Sidebar

Programming · VS Code Prettier Cheat Sheet

The Prettier extension's settings, Prettier's own options with their defaults, config files, and ignore comments. Checked against extension 12.4.0 (bundling Prettier 3.7.4) on this machine and Prettier's documentation. Sheet 16 in the VS Code series. The option names in the cards below are the same in a config file and, with a prettier. prefix, in VS Code settings.

SETTING UP & RUNNING

- Extension ID** `esbenp.prettier-vscode` (version 12.4.0 here, bundling Prettier 3.7.4)
- editor.defaultFormatter** `"esbenp.prettier-vscode"`, set globally or inside a `[language]` block
- editor.formatOnSave** `true` to format whenever you save
- Shift+Alt+F** Format the whole document (Windows/Linux)
- Ctrl+K Ctrl+F** Format only the selection
- Format Document With...** Command Palette: choose a formatter for this one run
- Format Document (Forced)** Command from the extension: format a file that is in `.prettierignore` or in a skipped place such as `node_modules`, and ignore `requirePragma`
- Prettier: Create Configuration File** Command from the extension: writes a starter config file in the project

EXTENSION SETTINGS (PRETTIER.*)

- prettier.enable** Turn the extension on or off; reload r
- prettier.requireConfig** Format only if a Prettier con
- prettier.configPath** Path to one specific config file (c
- prettier.ignorePath** Path to the ignore file (default)
- prettier.useEditorConfig** Read `.editorconfig` a
- prettier.prettierPath** Use a Prettier install at this pa
`./node_modules/prettier`
project, else the bundled one
- prettier.resolveGlobalModules** Fall back to globally
`false`)
- prettier.withNodeModules** Also format files inside `node_modules`
`false`)
- prettier.documentSelectors** Extra glob patterns of fi
(for example for plugin
- prettier.enableDebugLogs** Verbose logging in the Pr
`false`)
- prettier.disableLanguages** / `packageManager` / `pa`

LAYOUT OPTIONS

- printWidth** Line length Prettier tries to stay within (default 80)
- tabWidth** Spaces per indent level (default 2)
- useTabs** Indent with tab characters (default `false`)
- endOfLine** `lf` (default), `crlf`, `cr`, `auto`
- proseWrap** Markdown text: `preserve` (default), `always`, `never`
- objectWrap** `preserve` (default): keep objects as multi-line if the first key is on its own line; `collapse`: fit on one line if possible
- experimentalOperatorPosition** Where to break around operators: `end` (default) or `start`
- experimentalTernaries** Try the new ternary layout (default `false`)

QUOTES & PUNCTUATION

- semi** Semicolons at the ends of statements (default `true`)
- singleQuote** Single instead of double quotes (default `false`)
- jsxSingleQuote** Single quotes in JSX attributes (default `false`)
- quoteProps** `as-needed` (default), `consistent`, `preserve`
keys
- trailingComma** `all` (default since Prettier 3), `es5`, `none`
- bracketSpacing** Spaces inside `{ }` in objects (default `true`)
- arrowParens** `always` (default) or `avoid`: `(x) =>` vs `x =>`

HTML, JSX & VUE

<code>bracketSameLine</code>	Put the closing <code>></code> of a multi-line tag on the last attribute's line (default <code>false</code>)
<code>jsxBracketSameLine</code>	Deprecated older name for <code>bracketSameLine</code>
<code>singleAttributePerLine</code>	One attribute per line in HTML, JSX, Vue and Angular (default <code>false</code>)
<code>htmlWhitespaceSensitivity</code>	<code>css</code> (default), <code>strict</code> , <code>ignore</code> : how careful to be with spaces in HTML
<code>vueIndentScriptAndStyle</code>	Indent code inside Vue <code><script></code> and <code><style></code> (default <code>false</code>)
<code>embeddedLanguageFormatting</code>	<code>auto</code> (default) or <code>off</code> : format code embedded in other code, such as CSS in JavaScript

PRAGMAS & RANGES

<code>requirePragma</code>	Format only files that start with a special pragma (default <code>false</code>)
<code>insertPragma</code>	Add that comment to files Prettier formats (default <code>false</code>)
<code>/** @format */</code>	The pragma comment; <code>@prettier</code> is preferred
<code>rangeStart</code> / <code>rangeEnd</code>	Format only part of a file (set in <code>editorconfig</code> or as a selection)

CONFIG FILES (FIRST MATCH WINS, IN THIS ORDER)

<code>"prettier"</code> key	In <code>package.json</code> or <code>package.yaml</code>
<code>.prettierrc</code>	JSON or YAML, no extension
<code>.prettierrc.json</code> / <code>.yml</code> / <code>.yaml</code> / <code>.json5</code>	The same, with an extension
<code>.prettierrc.js</code> , <code>prettier.config.js</code> , <code>.ts</code>	JavaScript or TypeScript export of the options
<code>.prettierrc.mjs</code> / <code>prettier.config.mjs</code>	ES module version
<code>.prettierrc.cjs</code> / <code>prettier.config.cjs</code>	CommonJS version
<code>.prettierrc.toml</code>	TOML version
<code>.editorconfig</code>	<code>indent_style</code> , <code>indent_size</code> / <code>tab_width</code> , <code>max_line_length</code> and <code>end_of_line</code> are mapped to Prettier options
<code>overrides</code>	<code>{ "overrides": [{ "files": "/*.md", "options": { "proseWrap": "always" } }] }</code> ; also <code>excludeFiles</code>
<code>plugins</code>	<code>"plugins": ["prettier-plugin-tailwindcss"]</code> : add formatting for more languages or rules

IGNORING CODE

<code>.prettierignore</code>	Gitignore-style patterns for files Prettier should ignore. Folders and <code>node_modules</code> are skipped by default
<code>// prettier-ignore</code>	JavaScript, TypeScript, JSX: skip the next line
<code><!-- prettier-ignore --></code>	HTML and Markdown: skip the next line
<code><!-- prettier-ignore-start --> ... <!-- prettier-ignore-end --></code>	HTML and Markdown: skip the next lines
<code><!-- prettier-ignore-attribute --></code>	HTML: leave the attribute as is
<code>/* prettier-ignore */</code>	CSS: skip the next rule
<code># prettier-ignore</code>	YAML and GraphQL
<code>{{! prettier-ignore }}</code>	Handlebars

COMMAND LINE (NPX PRETTIER ...)

<code>--write .</code>	Format every supported file under the current folder
<code>--check .</code>	Report files that are not formatted, without changing them (useful in CI)
<code>--list-different (-l)</code>	List the files that would change
<code>--config path</code>	Use a specific config file
<code>--ignore-path path</code>	Use a specific ignore file
<code>--find-config-path file</code>	Show which config file applies to a file
<code>npm i -D prettier</code>	Install Prettier in a project (recommended so everyone uses the same version)

COMMON COMPANIONS (MY KNOWLEDGE)

<code>prettier-plugin-tailwindcss</code>	Sorts Tailwind CSS classes
<code>prettier-plugin-astro</code>	Formats <code>.astro</code> files
<code>@prettier/plugin-php</code>	Formats PHP
<code>eslint-config-prettier</code>	Turns off ESLint rules that clash with Prettier

A REASONABLE STARTER SET (MY SUGGESTIONS, NOT DEFAULTS)

<code>"editor.defaultFormatter": "esbenp.prettier-vscode"</code>	Choose Prettier once, globally
<code>"editor.formatOnSave": true</code>	Format whenever you save
<code>"prettier.requireConfig": true</code>	Only format projects that opt in with a config file
<code>.prettierrc: { "singleQuote": true, "printWidth": 100 }</code>	A typical team config; commit it to the repository
<code>"[markdown]": { "editor.defaultFormatter": "esbenp.prettier-vscode" }</code>	Be explicit for Markdown, and consider <code>proseWrap</code> in an override

FORMATTING DOES NOTHING? WORK DOWN THIS LIST

1) Is Prettier the default formatter for this language (`editor.defaultFormatter`)? 2) Is `prettier.enable` on, and is the file in `.prettierignore`? 3) Do you have `prettier.requireConfig` on without a config file? 4) Open *View > Output* and choose *Prettier* in the drop-down: errors such as a syntax error in the file or in your config appear there. Turn on `prettier.enableDebugLogs` for more detail.

WHICH SETTING WINS?

The extension reads, in priority order: a Prettier config file, then `.editorconfig`, then VS Code settings, and the VS Code `prettier.*` settings are *ignored* if any local configuration is present. They are only the fallback for projects with no config. Per-language `[language]` overrides apply to that fallback. For a team, commit a config file so everyone formats the same way whatever their editor settings.