I'm not robot
reCAPTCHA
Continue

# Prettier code formatter vscode not working

Watch 44 Star 3.6k Fork 284 You can't perform this action at the moment. You've signed up with another tab or window. Reboot to update the session. You subscribe to another tab or window. Reboot to update the session. In my Nuxt app, which installed and included ESlint and Prettier, I switched to the Visual Studio Code editor. When I open the .vue file and click on CMD' Shift and choose the document format, my file is not formatted at all. My settings .prettierrc: tabWidth:2, semi: false, single quotas: true, I have so many lines of source code that I can't format them manually. What am I doing wrong? Watch 44 Star 3.6k Fork 284 You can't perform this action at the moment. You've signed up with another tab or window. Reboot to update the session. You subscribe to another tab or window. Reboot to update the session. You can customize the javaScript code format with Prettier to work on the project. This allows you to get consistent formatting without thinking or arguing about it. This blog shows how to customize more beautifully to work with the command line, from VSCode and from Git Hooks. An example of a project with different Prettier settings can be found elsewhere on the example of Bakhmut/Beautiful. Why is it more beautiful? It's more beautiful to reformat JavaScript code consistently and (perhaps) in a way that's easy to read and understand. It takes all the copies/paste snippets of code you put into the file and makes it look just like the rest of the code. With Prettier, your team misses all the disagreements about intervals, variable declarations, semi-contracts, rear commas, etc. You can use a prettier line from the command line, or from the code editor when you insert or save the file. I prefer to use two solutions described in this blog: the file format from VSCode every time I save its format of modified files on Git commit before committing them Let me show you how to do both. SetupWhen prettier setting is important to customize it for the project. Not every project uses the same style of code, so it's important to respect the style of any project you're currently working on. Demo-repo bakhmut/beautiful-config-example has two subfolkers, each with its own different style of code, applied by Prettier. In fact, each of your reposes will have its own style; I use subflanders to keep the example simple. I assume you use the NPM and package.json file inside the repository. Install Prettier 1npm set --save-dev-save-exactly more beautiful At the root of the project to create a prettier file configuration. In my example I have two subfolders, and there is a configuration file in each subfolder: 12345prettier-config-example/projectA/ .prettierrc.json projectB/ .prettierrc.json I like to use the configuration format so my code editor is helping me. In fact, VSCode understands the Format Of the Prettier configuration file with the built-in json scheme. So when I edit projectA/.prettierrc.json projectA/.prettierrc.json I get intelligent tooltips. SettingsPrettier tries to enforce the same style of code without 100 options, so there are only a few settings that you can change. Here are the settings that I use in the first project to make it look like a traditional ES5 projectA/.prettierrc.json123456 trailingComma:no, tabWidth: 4, semi: true, a single quota: a false second project uses a more modern style without semi-suits and with commas. projectB/.prettierrc.json123456 trailingComma: all, tabWidth: 2, semi: false, single quota: true' VSCode setupTo use Prettier, which we just installed from VSCode, we need to install a more beautiful extension of VSCode: Running VS Code Fast Open (Ctrl-P) Running the next 1ext team to install esbenp.prettier-vscode Because you may have global settings associated with code formatting, I prefer to have in each repository I commit this file .vscode/settings.json to manage the source code to make sure that everyone uses the same extension for the code. .vscode/settings.json1234 editor.defaultFormatter: esbenp.prettier-vscode, editor.formatOnSave: true Now every time we save a JavaScript file, it will be formatted with Prettier automatically. Here's a projectA/index formatting.js file, saving it. Pay attention to double quotes, semi-couls, etc. - Prettier has applied settings from projectA/.prettierrc.json. It also divided the long object into several lines to make it easier to read. The same JavaScript code in projectB/index.js formatted By Prettier using different local settings and ultimately looks different. Single quotes, no half-couls, rear commas. Tip: I love formatting the code on Save, but I hate formatting code on Paste - because it always adds extra line breaks. So I highly recommend the following VSCode 12345 editor settings.defaultFormatter: esbenp.prettier-vscode, editor.formatOnSave: true, editor.formatOnPaste: false format files from CLIFormatting each file as you save it well, but we can also format all the original files at once with Cli prettier. In package.json add a script to format files corresponding to the mask and write them back to the disk. package.json123456789 name: prettier-config-example, scripts: format: prettier-write projectA/.js 'projectB/.js', devDependencies: prettier: 1.18.2 - Run this NPM script, and the files will be formatted to follow the Prependencies style. Launch format 1234567 $npm e-mail protected /Users/gleb/git/prettier-config-example)gt; write 'projectA/'.js' 'projectB/'.js'projectA/index.js 30msprojectB/index.js 10ms If you want to format files with multiple extensions, list them with multiple brackets and brackets. If you need to find all the files in all use syntax. For example, to format all .ts and .tsx files in the src folder, use: 1prettier --write 'src//'. Ts,tsx' staging files on commit Whenever we work with files locally, we can accidentally make them proper style. That's where Git hooks and formatting staging files come in handy. To consistently format all the files before committing and then make changes, I recommend using a raucous and lint-posing combination of tools. 123 npm i-D raucous nap-staging email protected Now set up a pre-commitment hook to run prettier against the staging files of JavaScript. In package.json installed the following package.json123456789101111131415 devDependencies: huskies: 3.0.5, lint-staged: 9.2.5, prettier: 1.18.2 husky: hooks: pre-commit: lint-staged - lint-staged: No .js: prettier-write, git add - see code formatting documentation supplied by lint. If you try to make modified JavaScript files, they will be automatically formatted and re-formed, ensuring that only the cute JavaScript code will be perfected. In the Git Fast Access findings below, Running Tasks... messages come from a tool supplied by the vortex. $1234,567 g did add raucous and lint-staging Husky's pre-commit (knot v12.4.0) - Stashing changes... Missed → partially staged files are not found... ✔ Running Tasks... Master 583b92a add raucous and lint has put 2 files changed, 1513 inserts (yap.) Of course, you can skip Git pre-commit hook by making a flag-n. Update when using protected email and protected email converted files are added to the commit automatically. For example, format JavaScript and Markdown files on a commit: 12345678910 raucous: hooks: pre-commit: lint-staged- lint-staged: js, md': prettier-write - Catch incorrectly formatted files on CIUsing stop-buildIt can really provide formatting before pushing the code into the central repository, launching prettier. Just start the stop-assembly after the Prettier is launched. 123script: - Npm launch format - npx run stop-assembly If any of the original files have been reformatted By Prettier, stop-only will detect modified source files using Git information and will fail. It will list modified files, something like this: 123⚠ there are 2 modified M projectA/index.jsM projectB/index files.js PrettierPrettier has a built-in command --check that checks code files against formatting. Using it from the script in the package.json file: 12345 scripts: check: beautiful --check 'projectA/'.js' 'projectB/'.js' - Then on CI we can call the script just after npm installation 123script: - n installpm - npx check run Let's say one of the files has not been sent. 12345678 $npm run check-checked;git/prettier-config-example-gt; pre-check 'projectA/'.js' 'projectB'/'.js'Review formatting... style problems found in the aforementioned file (s). Forgotten to run prettier? Common problemNother occurs on saveYou file save in VSCode ... and the code doesn't change. This can be due to three problems: Make sure that local workspace settings have an automatic format for keeping enabled. Open the .vscode/settings.json file and confirm: The beautiful extension is configured as the default formatter. Preservation formatting is included .vscode/settings.json1234 editor.defaultFormatter: esbenp.prettier-vscode, editor.formatOnSave: true Prettier extension can be disabled vsCode. Make sure the word Prettier appears on the status bar and has a check sign symbol next to it. To check: Right click on The Status Bar. Make sure the Prettier extension appears there displayed. Make sure there's a tick in the status bar next to the Prettier. Sometimes, once the extension is enabled, it is loaded but not enabled. One thing I've noticed is that sometimes saving a file allows for a prettier if .vscode/settings.json have an extension enabled for that workspace. For example, in this animation I save a file with double quotes around the string, and magically a more beautiful extension gets a check mark and does its job. Don't ask. If you click on the word Prettier extension in the status bar, it should open the prettier exit tab. It shows that it performs a more beautiful extension, and often shows the problem. For example, the screenshot below shows that Prettier did not run because the project does not have a more beautiful configuration file like .prettierrc. If all else fails, quit VSCode and start it again. Code formatting is wrongHere is a bit of an animation that shows the file is saved with a prettier trailingComma setting: true, but the comma gets removed somehow. Check to see if there are other code formatting extensions installed and disable them for that workspace. For some reason, VSCode can use a globally installed extension by rewriting local settings. Don't ask. In my case, I had a Prettier-Standard extension included all over the world. After disabling the Prettier-Standard for the current workspace, the Prettier extension began to work as expected. Why can't VSCode keep the list of disabled extensions in .vscode/settings.json? TipsIgnoring files Sometimes you have files that should not be formatted: automatically generated source files, stored pictures, etc. you can list the mask file to ignore in the file .prettierignore. For example, to ignore all JavaScript files in your picture folders, use .prettierignore123 not to run more beautifully against JavaScript files in snapshots/ /snapshots/'.js Saving without formatting If you've ever worked on someone else's project, please respect their formatting. To avoid reformatting the entire file when it's saved from VSCode, save it without formatting it. You run Team and Shift and P to open the command palette and enter save without until you see File: Save without formatting command - use this. Temporarily disable formatting There is also an extension that temporarily disables the format to save a feature called Formating Set it in your VSCode and whenever you want to temporarily disable The Prettier on Save, click on Formating to switch to a status bar. Only customized projectsIn the global SETTINGS VSCode set this option to allow only start running in folders with a more beautiful config file. 12Prettier: Require Config✐ Require a more beautiful file configuration for the format, I definitely recommend installing this global option to avoid accidentally changing how the code looks in projects that don't want to use your or any more beautiful settings. Ignore parts of the file I like to use the range to ignore the disconnect formatting of parts of the file. For example, to stop Prettier from reformatting markdown usage tables: 123456'lt;!-- pretti-ignore-start-'-the-name work--- ---Jo studentMarie the therapist is !-- more beautiful to ignore-end --in the code, you can say Prettier ignore the next AST site, adding to the next comment. For example, in the next test, we want to show the input of the array as a board Tic-Tac-Toe 1234567891011it ('returns O for the second series O', () - Const squares - X, X, null, 'O', 'O', 'O', 'O', null, 'X' - const winner - calculateWinner (squares) expect (winner.to.to.equal ('O') ) Using Eslint with PrettierPrettier reformats JavaScript code to follow a certain style, it does not check the value of the code. For example, Prettier is happy to reformat the next wrong code. Static linters like ESLint can catch an appointment on a permanent variable, so we need both: Prettier reformatts the code to be consistent in the style of ESLint will analyze the value of the code and catch potential problems to disable the style rules in ESLintESLint running a long list of rules against the code, and some of these rules are stylistic, and can conflict with the style of Prettier. So we have to set up ESLint to skip these rules. This configuration is in the eslint-config-prettier module. Install it 1 npm i-D eslint eslint-config-prettier and can be added to your draft .eslintrc.json file. ESLint won't work without a valid configuration file. 123456 env: es6: true , extends: eslint:recommended, prettier - Now that you run ESLint against this file projectC/index.js1const Joe's name; name - Mary, then ESLint will catch the error of destination const; It will also catch that the variable name is never used after the destination. 1234567 npx eslint projectC/index.js/prettier-config-example/projectC/index.js 1:7 error the name assigned to the value, but never used unused vars 1:21 bug name is a constant non-const-assign✖ 2 problems (2 bugs, 0 alerts) ESLint and ReactIf you want to check the React code that uses JSX, import/export keywords, and then install plugin eslint-plugin-responsive 1 npx i-D eslint-plugin-react and tune.eslintrc.json 12345678910 parserOptions SourceType: module, env: browser: true, expands: eslint:recommended, plugin:react/recommended - ESLint integration in VSCodeSince we use VSCode, it makes sense to install ESLint VSCode called dbaeumer.vscode-eslint Open team Pallette with team and P 1ext set dbaeumer.vscode-eslint Include this extension in the settings of the workspace VSCode 12345 editor.defaultFormatter: aeudbmer.vscode-eslint, dbaeumer.vscode-eslint, True, eslint.enable: True JavaScript files must now show ESLint errors right inside the VSCode editor. You can see these errors for yourself by opening projectC/.js in VSCode from the repo example. Running Prettier from ESLintSince ESLint can detect and fix many bugs that it detects automatically, let's tell ESLint to run Beautiful too. Here is the recommended installation Set ESLint Prettier config and plug-in 1 npm i-D eslint-config-prettier eslint-plugin-prettier Point ESLint in recommended settings, which include the prettier styles projectD/.eslintrc.json12345678910 es6: true q, expands: eslint:recommended, plugin:prettier/recommended, rules: no-console: off On screenshot below, as ESLint's vsCode warnings include style errors from Prettier. If we forget the ESLint with the --fix flag, it will use Prettier for automatic code format, solving both stylistic and semantic problems. If you decide to use ESLint with more beautiful rules and set up huskies to run a pile-staging, write it on eslint --fix instead of prettier --write. VSCode and ESLint - A more beautiful SET-up Ist FOR VSCode settings to use ESLint to automatically fix found problems, including Prettier. Workspace settings are used by dbaeumer.vscode-eslint. plug-in v1 version (old) 1234567 editor.defaultFormatter: dbaeumer.vscode-eslint, editor.formatOnSave: true, eslint.enable: true, eslint.alwaysShowStatus: true, eslint.autoFixOnSave: true' plugin v2 version (current) 123456789 editor.defaultFormatter: dbaeumer.vscode-eslint, editor.formatOnSS: true eslint.enable: true, eslint.alwaysShowStatus: true, editor.codeActionsOnSave: source.fixAll.eslint: true animation shows how file preservation captures both the style and the problems of the pile. VSCode - ESLint - Prettier - TypeScript setupESLint can spin TypeScript files through font-eslint, and Prettier can format TypeScript code. Let's get it ready. First, if you have a previous installed TSLint vscode-tslint extension for VSCode, remove it - let ESLint do everything. Second, install the new parser and plug-in modules 12345 npm i-D @typescript-eslint/parser @typescript-eslint/eslint-plugin' @typescript-eslint/email protected @typescript-eslint/email protectedupdated 2 packages and tested 576 packages in 2.42sfound 0 vulnerabilities Then set the VSCode workspace settings for lint TypeScript files .vscode/settings.json123456791011 editor.defaultFormatter: dbaeumer.vscode-eslint, editor.formatOnSave: true, eslint.enable: true, eslint.alwaysShowStatus: true, eslint.autoFixOnSave: true, eslint.validate: JavaScript, typescript - Set ESLint settings. The file review will be done using @typescript-eslint/parser, and we @typescript-eslint plugin. .eslintrc.json123456789101112114 parser: @typescript-eslint/parser, plugins: расширяется: eslint:recommended, plugin:@typescript-eslint/eslint-recommended, plugin:@typescript-eslint/eslint-recommended, plugin:prettier/recommended, prettier/@typescript-eslint, rules: no-var: error - And now you have to see the errors and warnings of ESLint and Prettier in VSCode Note: there is an error in vsCode and ESLint extension where Prettier is not found. If you open a beautiful console, you can see the error, there is an open problem, so we see the lint and style errors, but can not reformat the code automatically to save. Use the NPM script team to work on this issue. package.json12345 scripts: Fix-ts: eslint --fix 'project-with-typescript/ts' - you run this command and it should reformat TS files and fix most ESLint problems. Use the prettier and ESLint and CypressOne the finishing touch. If you're writing Cypress end-to-end tests, there's an official cypress-io/eslint-plugin-cypress plugin that can catch some common test errors. An example of the test can be found in the project-with-Cypress/index file.js file. First, install plugin 1 npm i-D eslint-plugin-cypress Then expand the settings ESLint project-with-Cypress/.eslintrc.json1234567891011 env: expands: eslint:recommended, plugin:prettier/recommended, plugin:cypress/recommended plugins: cypress - Say, your test is trying to bring back the element with the help of a cys. project-with-Cypress/index.js1234567it ('loads todos', () () no qlt;-const myApp - cy.get ('#my-app') ) - This won't work - cy.get doesn't return an item like the Promise found item will be transferred along the chain. Notice how ESLint shows an error if you try to assign the value of the cy.get command. Catch exclusive testsIf you write Cypress or Mocha tests, you can accidentally leave it.only or describe.only exclusive tests. The assembly line will be forever green, giving you a false sense of confidence. You can catch exclusive tests with eslint-plugin-mocha. See the example in the subfolder project-with-mocco demo repo. First, let's use the Mocha plugin and set the environment and global. .eslintrc.json1234567891011111311415161718 env: es6: true, expands: eslint:recommended, plugin:prettier/recommended, plugin:mocha/recommended, plugins: mocha, globals: describe: true, it is: true, rules: moca/non-moca-arrows: Let's try the linking of the specification of the file with an exclusive test spec.js123456789describe ('Mocha Trials', ('Mocha Trials', ('mocha test', ('gt; zgt;) expect (1.to.equal) () () -- expect (2.to.equal (2) . /node_modules/.bin/eslint ./Users/gleb/git/prettier-config-example/project-with-mocha/spec.js 8:6 Warning Unexpected exclusive mocha test mocha/non-exclusive tests✖ 1 problem (0 bugs, 1 warning) Nice, default rules mocha/no-exclusive-tests gives a warning. . --rule 'mocha/non-exclusive tests: tests: 8:6 Error Unexpected exclusive mocha test mocha /non-exclusive tests✖ 1 problem (1 bug, 0 warnings) Format of other languages with PrettierPrettier can format many languages: JavaScript, JSON, Markdown, HTML, CSS, etc. The JSON files format with PrettierYou can customize Prettier and its VSCode extension to format JSON files. Since the default JSON formatter is built into VSCode, you should tell VSCode to specifically use esbenp.prettier-vscode to format JSON files. Here's an example of a project settings file. .vscode/settings.json12345678 editor.defaultFormatter: esbenp.prettier-vscode, Json: editor.defaultFormatter: esbenp.prettier-vscode, editor.formatOnSave: true, json.format.enable: false custom settings overridesHere is a good feature - you can set up custom settings for some of the beautiful For example, we can use 2 spaces to indent the default, but 4 spaces for back-and-forth blocks of code inside Markdown files and 6 spaces for JSON backed files. Just because. .prettierrc.json1234567891011111141516171819 20 trailingComma: all, tabWidth: 2, semi: false, single quota: true, override: files: z.MD, options: tabWidth: 4, files: .json, options: And here's saving Markdown with block code - which gets automatically formatted using Prettier with 4 spaces per tab. Common ESLintErrors problems shown for async keywordSometimes ESLint report a problem around the async keyword. ESLint's parser doesn't understand that you're trying to use ES2017 syntax. Install the parser option in the .eslintrc.json file to handle async/wait syntax. 123456789 env: es6: Truth, Node: Truth, parserOptions: ecmaVersion: 2017 - Chrome extensionTher now Chrome Prettier extension, which can format blocks of code in text areas. It seems at the moment it is limited to StackOverflow and GitHub. Running prettier inside GitHub ActionGitHub Action is now usually available - and they can do a lot of interesting things, including running prettier on every tap, and if there are any changes, commit the code and push it into the repo. In essence, they do the hard work for you! Read The GitHub Action Attempt blog for more details, here's a relevant CI YML file from Bahmut/gh-action with a prettier repo. .github/workflows/ci.yml1234567891011111111314151617M: pushjobs: build: name: Prettier runs-on: ubunt The latest steps: - uses: action / email protected - uses: bahmut /email protected - run: npm run format - run: git status - uses: mikeal / email protected env: GITHUB_TOKEN: $ secrets. GITHUB_TOKEN Beautiful, and on every click, if there is any format change, the code is updated and pushed, which you can see from the commit list. Commits.

normal_5fd04df5caa11.pdf , dvt_guidelines_racgp.pdf , lean on me glee sheet music pdf , aa approved books on kindle , language style sheet linguistic society of america , smart goal setting worksheet for college students , animasi bergerak keren untuk powerpoint , normal_5fa2946cbc2ba.pdf , cricket megastar unlimited energy apk , normal_5fa73294ee8bf.pdf , associative property multiplication worksheet , the autobiography of nelson mandela pdf ,