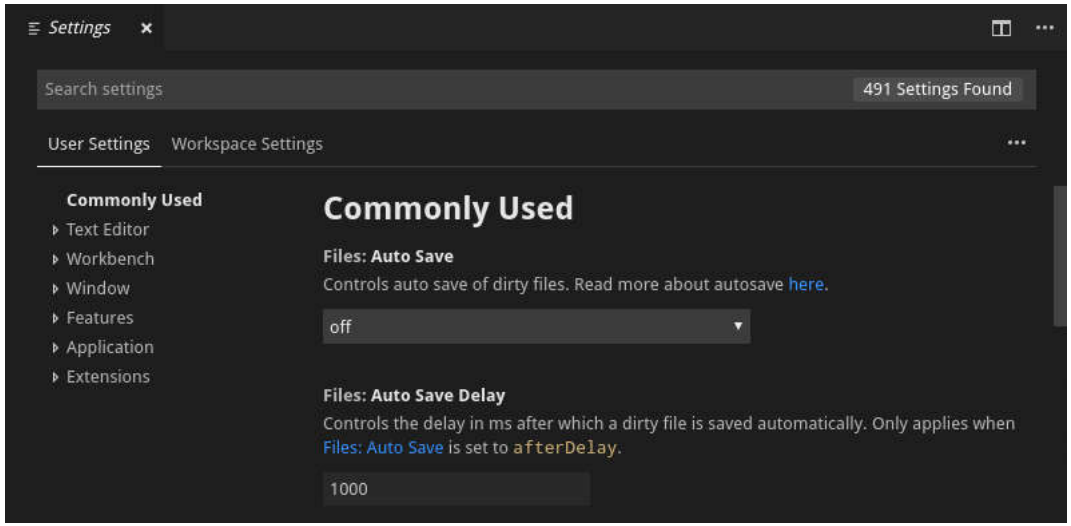
TOPICS Settings

User and Workspace Settings

(https://github.com/Microsoft/vscode-docs/blob/master/docs/getstarted/settings.md)

It is easy to configure Visual Studio Code to your liking through its various settings. Nearly every part of VS Code's editor, user interface, and functional behavior has options you can modify.



VS Code provides two different scopes for settings:

- **User Settings -** Settings that apply globally to any instance of VS Code you open.
- **Workspace Settings -** Settings stored inside your workspace and only apply when the workspace is opened.

Workspace settings override user settings.

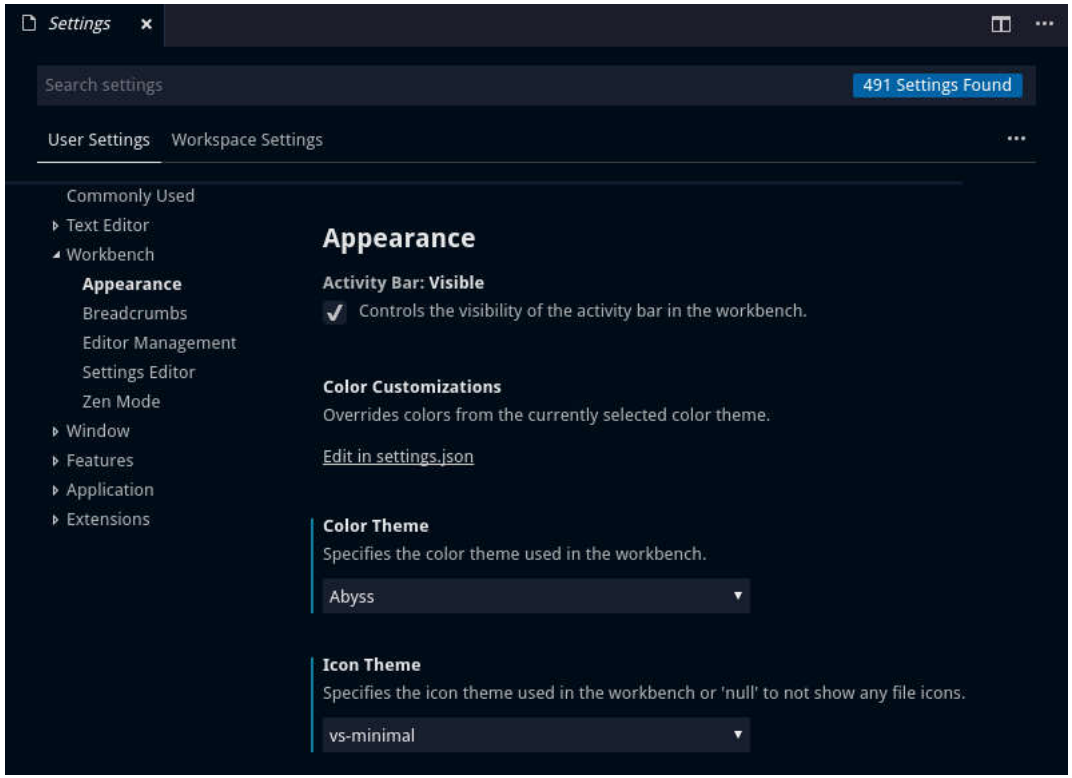
Creating User and Workspace Settings

To open your user and workspace settings, use the following VS Code menu command:

- On Windows/Linux **- File** > **Preferences** > **Settings**
- On macOS **- Code** > **Preferences** > **Settings**

You can also open the Settings editor from the **Command Palette** (`Ctrl+Shift+P`) with **Preferences: Open Settings** or use the keyboard shortcut (`Ctrl+,`).

In the example below, the color theme and the icon theme have been changed.

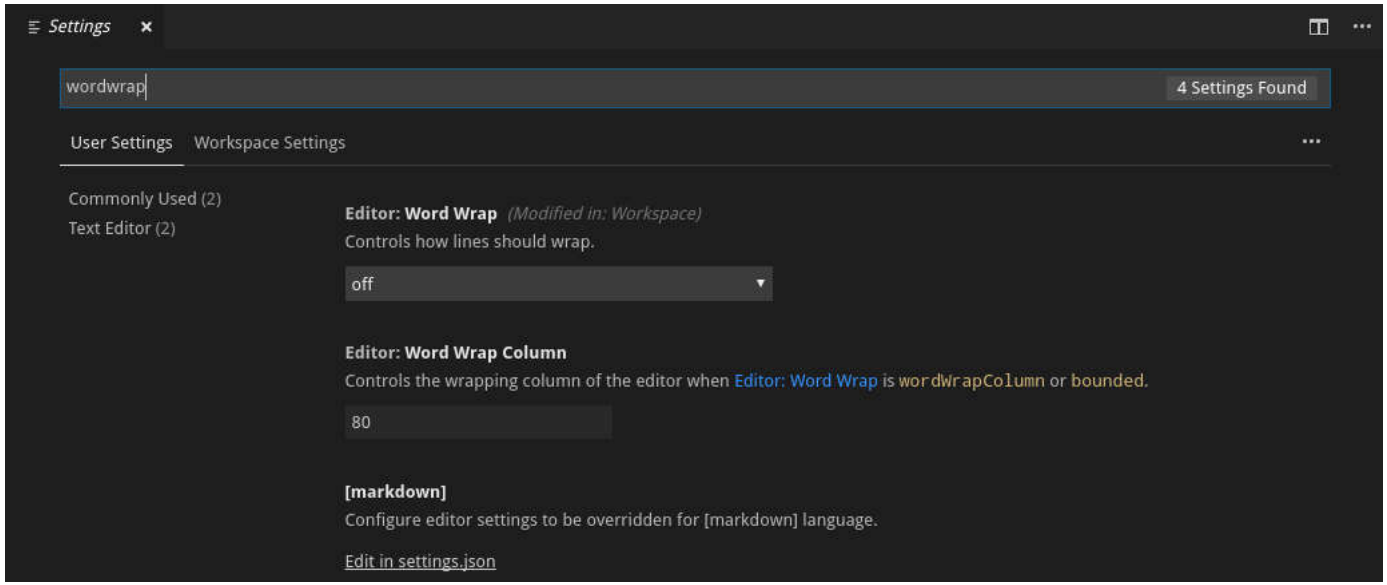


Changes to settings are reloaded by VS Code as you change them. Modified settings are now indicated with a *blue line* similar to modified lines in the editor. The gear icon opens a context menu with options to reset the setting to its default value as well as copy setting as JSON.

> **Note:** Workspace settings are useful for sharing project specific settings across a team.

Settings editor

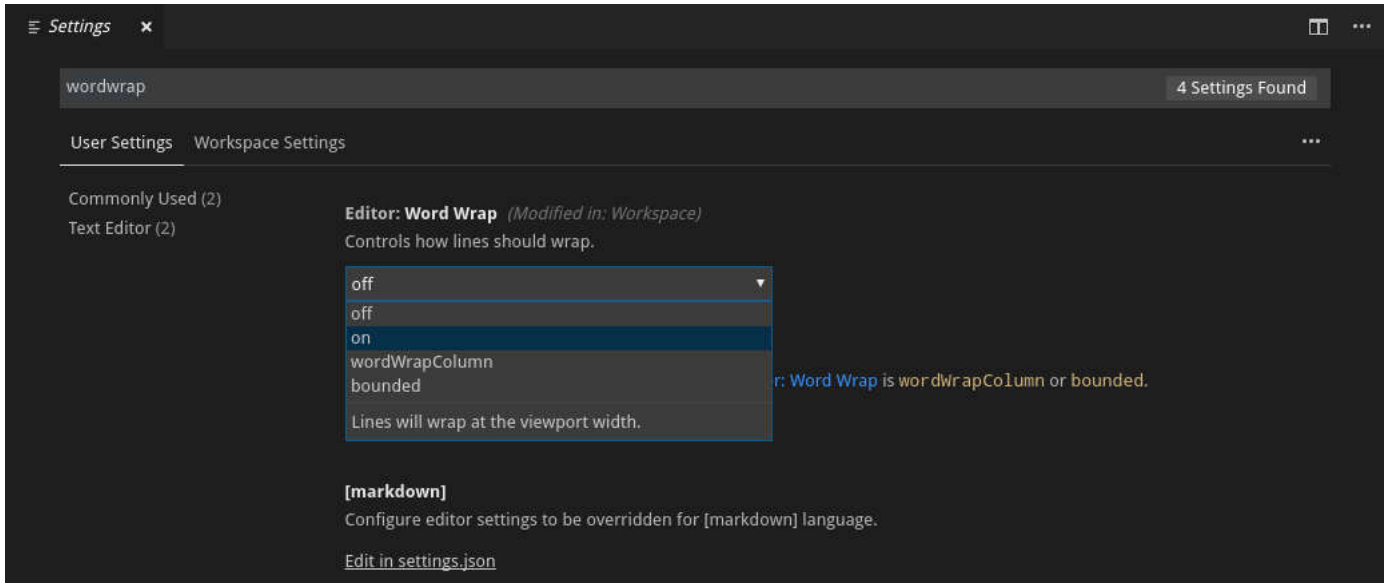
When you open the settings editor, you can search and discover settings you are looking for. When you search using the Search bar, it will not only show and highlight the settings matching your criteria, but also filter out those which are not matching. This makes finding settings quick and easy.



Note: VS Code extensions can also add their own custom settings and they will be visible under an **Extensions** section.

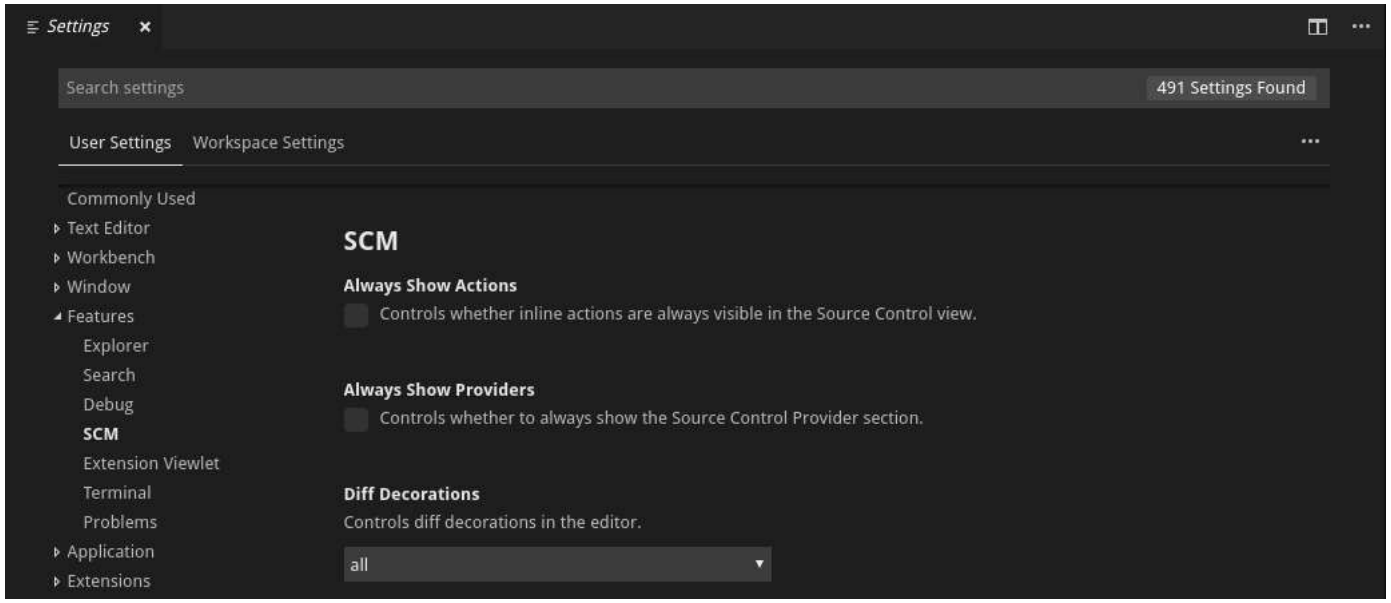
Edit settings

Each setting can be edited by either a **checkbox**, an **input** or by a **drop-down**. Simply edit the text or select the option you want to change to the desired settings.



Settings groups

Default settings are represented in groups so that you can navigate them easily. It has a **Commonly Used** group at the top which shows popular customizations.



Below is a copy of the default settings (/docs/getstarted/settings#_default-settings) that come with VS Code.

Settings file locations

By default VS Code shows the Settings editor, but you can still edit the underlying `settings.json` file by using the **Open Settings (JSON)** command or by changing your default settings editor with the `workbench.settings.editor` setting.

Depending on your platform, the user settings file is located here:

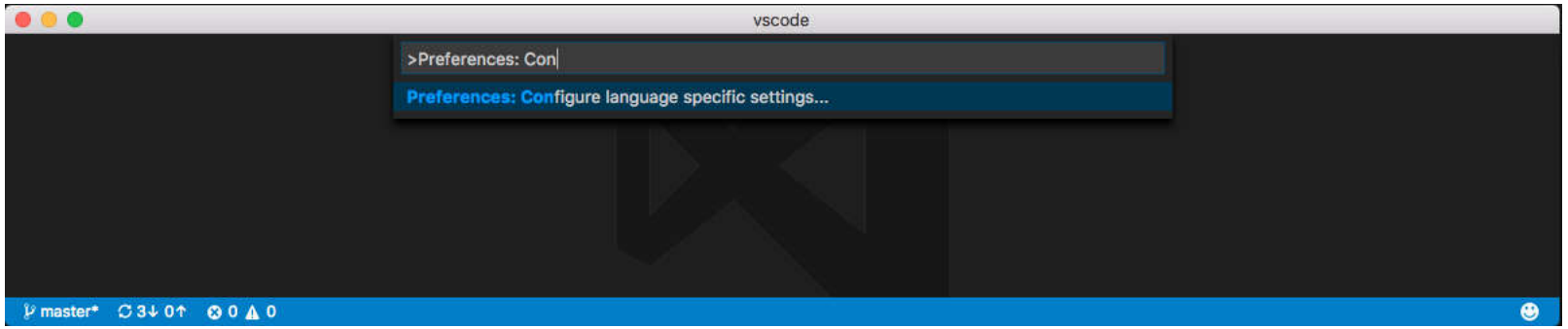
- **Windows** `%APPDATA%\Code\User\settings.json`
- **macOS** `$HOME/Library/Application Support/Code/User/settings.json`
- **Linux** `$HOME/.config/Code/User/settings.json`

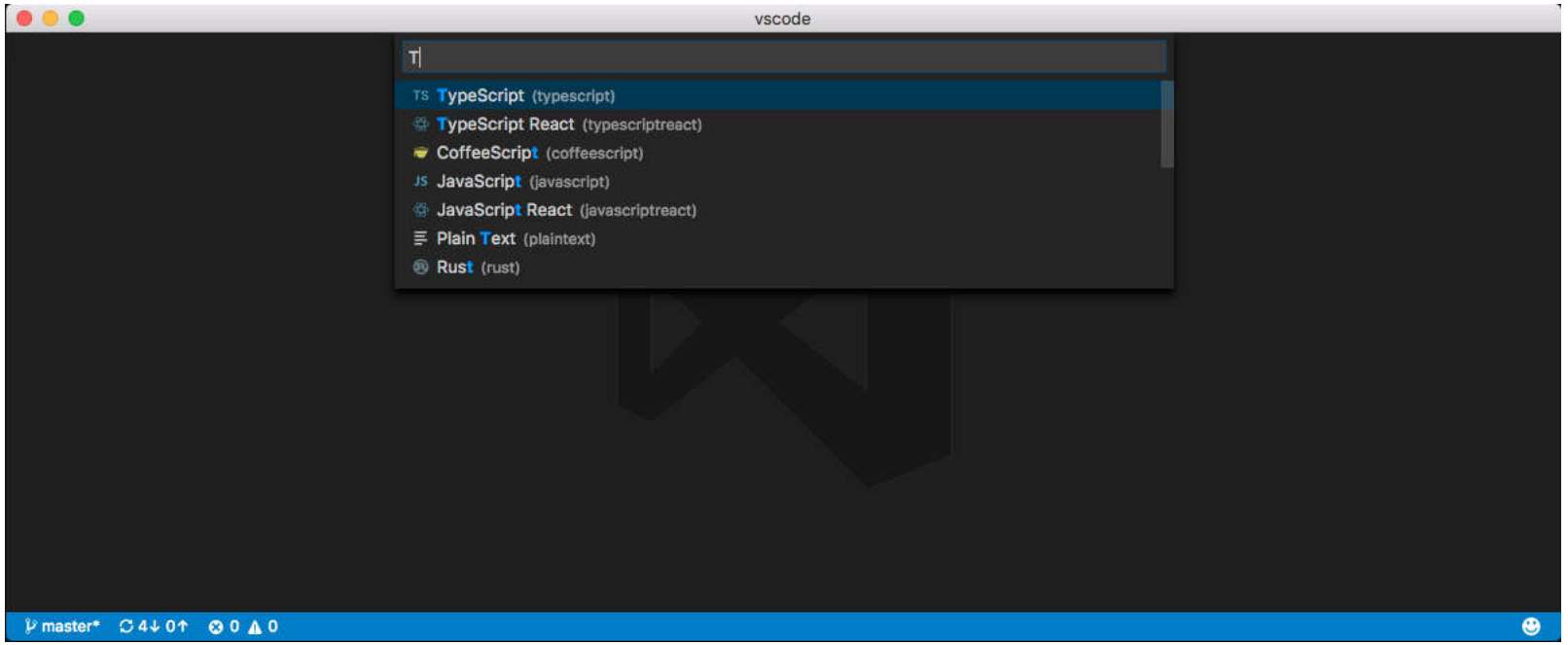
The workspace settings file is located under the `.vscode` folder in your root folder.

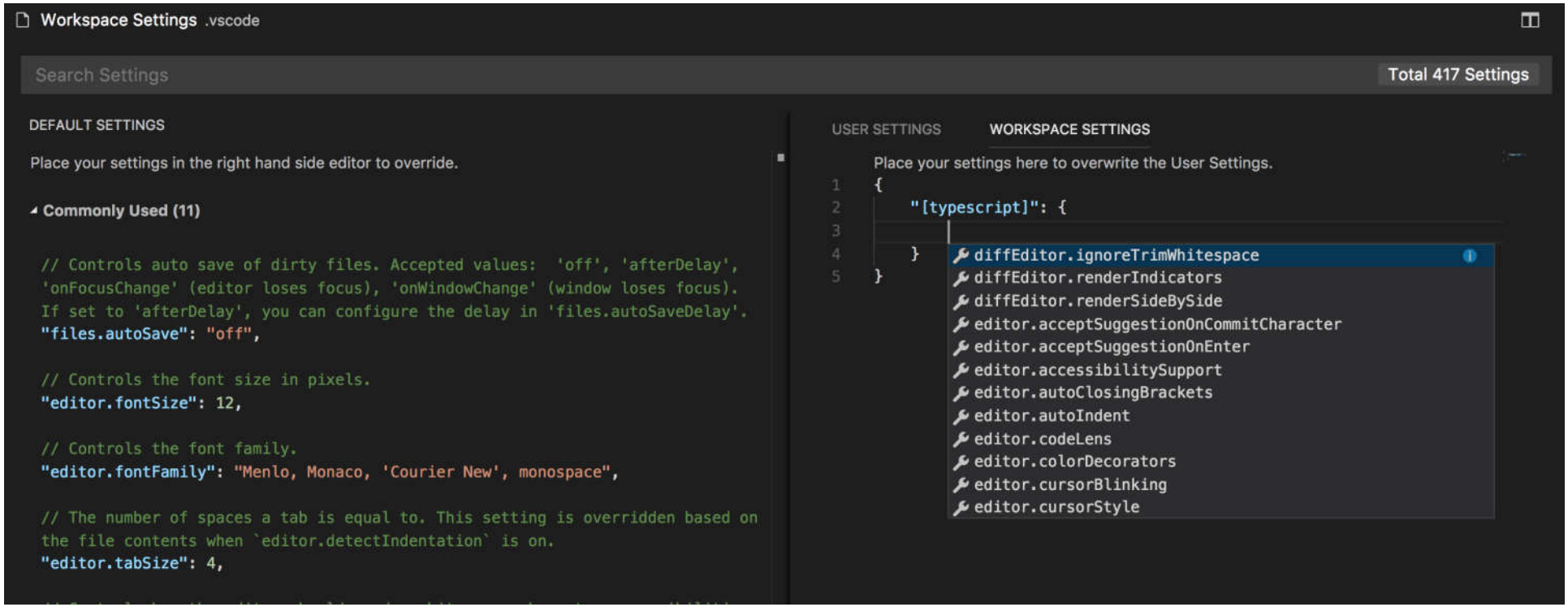
> **Note:** In case of a Multi-root Workspace (/docs/editor/multi-root-workspaces#_settings), workspace settings are located inside the workspace configuration file.

Language specific editor settings

To customize your editor by language, run the global command **Preferences: Configure Language Specific Settings** (command id: `workbench.action.configureLanguageBasedSettings`) from the **Command Palette** (`Ctrl+Shift+P`) which opens the language picker. Selecting the language you want, opens the Settings editor with the language entry where you can add applicable settings.


vscode
>Preferences: Con
Preferences: Configure language specific settings...
master* 3↓ 0↑ 0 0


vscode
T
TypeScript (typescript)
TypeScript React (typescriptreact)
CoffeeScript (coffeescript)
JavaScript (javascript)
JavaScript React (javascriptreact)
Plain Text (plaintext)
Rust (rust)
master* 4↓ 0↑ 0 0



If you have a file open and you want to customize the editor for this file type, click on the Language Mode in the Status Bar to the bottom-right of the VS Code window. This opens the Language Mode picker with an option **Configure 'language_name' language based settings**. Selecting this opens the Settings editor with the language entry where you can add applicable settings.

You can also configure language based settings by directly opening `settings.json`. You can scope them to the workspace by placing them in the workspace settings just like other settings. If you have settings defined for a language in both user and workspace scopes, then they are merged by giving precedence to the ones defined in the workspace.

The following examples customize editor settings for language modes `typescript` and `markdown`.

```
{
  "[typescript]": {
    "editor.formatOnSave": true,
    "editor.formatOnPaste": true
  },
  "[markdown]": {
    "editor.formatOnSave": true,
    "editor.wordWrap": "on",
    "editor.renderWhitespace": "all",
    "editor.acceptSuggestionOnEnter": "off"
  }
}
```

You can use IntelliSense in Settings editor to help you find allowed language based settings. All editor settings and some non-editor settings are supported.

Settings and security

Some settings allow you to specify an executable that VS Code will run to perform certain operations. For example, you can choose which shell the Integrated Terminal should use. For enhanced security, such settings can only be defined in user settings and not at workspace scope.

Here is the list of settings not supported in workspace settings:

- `git.path`
- `terminal.integrated.shell.linux`
- `terminal.integrated.shellArgs.linux`
- `terminal.integrated.shell.osx`
- `terminal.integrated.shellArgs.osx`
- `terminal.integrated.shell.windows`
- `terminal.integrated.shellArgs.windows`
- `terminal.external.windowsExec`
- `terminal.external.osxExec`
- `terminal.external.linuxExec`

The first time you open a workspace which defines any of these settings, VS Code will warn you and subsequently always ignore the values after that.

Default settings

Below are the Visual Studio Code default settings and their values. You can also view the default values in the Settings editor.

```
{

// Editor

  // Controls whether the diff editor shows changes in leading or trailing whitespace as diffs.
  "diffEditor.ignoreTrimWhitespace": true,

  // Controls whether the diff editor shows +/- indicators for added/removed changes.
  "diffEditor.renderIndicators": true,

  // Controls whether the diff editor shows the diff side by side or inline.
  "diffEditor.renderSideBySide": true,

  // Controls whether suggestions should be accepted on commit characters. For example, in JavaScript, the semi-colon (`;`) can be a commit character that accepts a suggestion and types that charact
er.
  "editor.acceptSuggestionOnCommitCharacter": true,

  // Controls whether suggestions should be accepted on `Enter`, in addition to `Tab`. Helps to avoid ambiguity between inserting new lines or accepting suggestions.
  //  - on
  //  - smart: Only accept a suggestion with `Enter` when it makes a textual change.
  //  - off
  "editor.acceptSuggestionOnEnter": "on",

  // Controls whether the editor should run in a mode where it is optimized for screen readers.
  //  - auto: The editor will use platform APIs to detect when a Screen Reader is attached.
  //  - on: The editor will be permanently optimized for usage with a Screen Reader.
  //  - off: The editor will never be optimized for usage with a Screen Reader.
  "editor.accessibilitySupport": "auto",

  // Controls whether the editor should automatically close brackets after the user adds an opening bracket.
  //  - always
  //  - languageDefined: Use language configurations to determine when to autoclose brackets.
  //  - beforeWhitespace: Autoclose brackets only when the cursor is to the left of whitespace.
  //  - never
  "editor.autoClosingBrackets": "languageDefined",

  // Controls whether the editor should automatically close quotes after the user adds an opening quote.
  //  - always
  //  - languageDefined: Use language configurations to determine when to autoclose quotes.
  //  - beforeWhitespace: Autoclose quotes only when the cursor is to the left of whitespace.
  //  - never
  "editor.autoClosingQuotes": "languageDefined",

  // Controls whether the editor should automatically adjust the indentation when users type, paste or move lines. Extensions with indentation rules of the language must be available.
  "editor.autoIndent": true,

  // Controls whether the editor should automatically surround selections.
  //  - languageDefined: Use language configurations to determine when to automatically surround selections.
  //  - brackets: Surround with brackets but not quotes.
  //  - quotes: Surround with quotes but not brackets.
  //  - never
  "editor.autoSurround": "languageDefined",

  // Code action kinds to be run on save.
  "editor.codeActionsOnSave": {},

  // Timeout in milliseconds after which the code actions that are run on save are cancelled.
  "editor.codeActionsOnSaveTimeout": 750,

  // Controls whether the editor shows CodeLens.
  "editor.codeLens": true,
```

```
// Controls whether the editor should render the inline color decorators and color picker.
"editor.colorDecorators": true,

// Controls whether syntax highlighting should be copied into the clipboard.
"editor.copyWithSyntaxHighlighting": true,

// Control the cursor animation style.
"editor.cursorBlinking": "blink",

// Controls whether the smooth caret animation should be enabled.
"editor.cursorSmoothCaretAnimation": false,

// Controls the cursor style.
"editor.cursorStyle": "line",

// Controls the width of the cursor when `editor.cursorStyle` is set to `line`.
"editor.cursorWidth": 0,

// Defines a default formatter which takes precedence over all other formatter settings. Must be the identifier of an extension contributing a formatter.
"editor.defaultFormatter": null,

// Controls whether `editor.tabSize#` and `#editor.insertSpaces` will be automatically detected when a file is opened based on the file contents.
"editor.detectIndentation": true,

// Controls whether the editor should allow moving selections via drag and drop.
"editor.dragAndDrop": true,

// Controls whether copying without a selection copies the current line.
"editor.emptySelectionClipboard": true,

// Scrolling speed mulitiplier when pressing `Alt`.
"editor.fastScrollSensitivity": 5,

// Controls whether the Find Widget should add extra lines on top of the editor. When true, you can scroll beyond the first line when the Find Widget is visible.
"editor.find.addExtraSpaceOnTop": true,

// Controls whether the find operation is carried out on selected text or the entire file in the editor.
"editor.find.autoFindInSelection": false,

// Controls whether the Find Widget should read or modify the shared find clipboard on macOS.
"editor.find.globalFindClipboard": true,

// Controls whether the search string in the Find Widget is seeded from the editor selection.
"editor.find.seedSearchStringFromSelection": true,

// Controls whether the editor has code folding enabled.
"editor.folding": true,

// Controls the strategy for computing folding ranges. `auto` uses a language specific folding strategy, if available. `indentation` uses the indentation based folding strategy.
"editor.foldingStrategy": "auto",

// Controls the font family.
"editor.fontFamily": "Consolas, 'Courier New', monospace",

// Enables/Disables font ligatures.
"editor.fontLigatures": false,

// Controls the font size in pixels.
"editor.fontSize": 14,
```

```
// Controls the font weight.
"editor.fontWeight": "normal",

// Controls whether the editor should automatically format the pasted content. A formatter must be available and the formatter should be able to format a range in a document.
"editor.formatOnPaste": false,

// Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down.
"editor.formatOnSave": false,

// Timeout in milliseconds after which the formatting that is run on file save is cancelled.
"editor.formatOnSaveTimeout": 750,

// Controls whether the editor should automatically format the line after typing.
"editor.formatOnType": false,

// Controls whether the editor should render the vertical glyph margin. Glyph margin is mostly used for debugging.
"editor.glyphMargin": true,

// Controls the behavior of 'Go To' commands, like Go To Definition, when multiple target locations exist.
//  - peek: Show peek view of the results (default)
//  - gotoAndPeek: Go to the primary result and show a peek view
//  - goto: Go to the primary result and enable peek-less navigation to others
"editor.gotoLocation.multiple": "peek",

// Controls whether the cursor should be hidden in the overview ruler.
"editor.hideCursorInOverviewRuler": false,

// Controls whether the editor should highlight the active indent guide.
"editor.highlightActiveIndentGuide": true,

// Controls the delay in milliseconds after which the hover is shown.
"editor.hover.delay": 300,

// Controls whether the hover is shown.
"editor.hover.enabled": true,

// Controls whether the hover should remain visible when mouse is moved over it.
"editor.hover.sticky": true,

// Insert spaces when pressing `Tab`. This setting is overridden based on the file contents when `editor.detectIndentation` is on.
"editor.insertSpaces": true,

// Controls the letter spacing in pixels.
"editor.letterSpacing": 0,

// Enables the code action lightbulb in the editor.
"editor.lightbulb.enabled": true,

// Controls the line height. Use 0 to compute the line height from the font size.
"editor.lineHeight": 0,

// Controls the display of line numbers.
//  - off: Line numbers are not rendered.
//  - on: Line numbers are rendered as absolute number.
//  - relative: Line numbers are rendered as distance in lines to cursor position.
//  - interval: Line numbers are rendered every 10 lines.
"editor.lineNumbers": "on",

// Controls whether the editor should detect links and make them clickable.
"editor.links": true,
```

```
  // Highlight matching brackets when one of them is selected.
  "editor.matchBrackets": true,

  // Controls whether the minimap is shown.
  "editor.minimap.enabled": true,

  // Limit the width of the minimap to render at most a certain number of columns.
  "editor.minimap.maxColumn": 120,

  // Render the actual characters on a line as opposed to color blocks.
  "editor.minimap.renderCharacters": true,

  // Controls whether the minimap slider is automatically hidden.
  "editor.minimap.showSlider": "mouseover",

  // Controls the side where to render the minimap.
  "editor.minimap.side": "right",

  // A multiplier to be used on the `deltaX` and `deltaY` of mouse wheel scroll events.
  "editor.mouseWheelScrollSensitivity": 1,

  // Zoom the font of the editor when using mouse wheel and holding `Ctrl`.
  "editor.mouseWheelZoom": false,

  // The modifier to be used to add multiple cursors with the mouse. The Go To Definition and Open Link mouse gestures will adapt such that they do not conflict with the multicursor modifier. [Read
more](https://code.visualstudio.com/docs/editor/codebasics#_multicursor-modifier).
  //  - ctrlCmd: Maps to `Control` on Windows and Linux and to `Command` on macOS.
  //  - alt: Maps to `Alt` on Windows and Linux and to `Option` on macOS.
  "editor.multiCursorModifier": "alt",

  // Controls whether the editor should highlight semantic symbol occurrences.
  "editor.occurrencesHighlight": true,

  // Controls whether a border should be drawn around the overview ruler.
  "editor.overviewRulerBorder": true,

  // Controls the number of decorations that can show up at the same position in the overview ruler.
  "editor.overviewRulerLanes": 3,

  // Controls whether the parameter hints menu cycles or closes when reaching the end of the list.
  "editor.parameterHints.cycle": false,

  // Enables a pop-up that shows parameter documentation and type information as you type.
  "editor.parameterHints.enabled": true,

  // Controls whether suggestions should automatically show up while typing.
  "editor.quickSuggestions": {
    "other": true,
    "comments": false,
    "strings": false
  },

  // Controls the delay in milliseconds after which quick suggestions will show up.
  "editor.quickSuggestionsDelay": 10,

  // Controls whether the editor should render control characters.
  "editor.renderControlCharacters": false,

  // Render last line number when the file ends with a newline.
  "editor.renderFinalNewline": true,
```

```
  // Controls whether the editor should render indent guides.
  "editor.renderIndentGuides": true,

  // Controls how the editor should render the current line highlight.
  //  - none
  //  - gutter
  //  - line
  //  - all: Highlights both the gutter and the current line.
  "editor.renderLineHighlight": "line",

  // Controls how the editor should render whitespace characters.
  //  - none
  //  - boundary: Render whitespace characters except for single spaces between words.
  //  - all
  "editor.renderWhitespace": "none",

  // Controls whether selections should have rounded corners.
  "editor.roundedSelection": true,

  // Render vertical rulers after a certain number of monospace characters. Use multiple values for multiple rulers. No rulers are drawn if array is empty.
  "editor.rulers": [],

  // Controls the number of extra characters beyond which the editor will scroll horizontally.
  "editor.scrollBeyondLastColumn": 5,

  // Controls whether the editor will scroll beyond the last line.
  "editor.scrollBeyondLastLine": true,

  // Controls whether the Linux primary clipboard should be supported.
  "editor.selectionClipboard": true,

  // Controls whether the editor should highlight matches similar to the selection.
  "editor.selectionHighlight": true,

  // Controls whether the fold controls on the gutter are automatically hidden.
  "editor.showFoldingControls": "mouseover",

  // Controls fading out of unused code.
  "editor.showUnused": true,

  // Controls whether the editor will scroll using an animation.
  "editor.smoothScrolling": false,

  // Controls whether snippets are shown with other suggestions and how they are sorted.
  //  - top: Show snippet suggestions on top of other suggestions.
  //  - bottom: Show snippet suggestions below other suggestions.
  //  - inline: Show snippets suggestions with other suggestions.
  //  - none: Do not show snippet suggestions.
  "editor.snippetSuggestions": "inline",

  // Keep peek editors open even when double clicking their content or when hitting `Escape`.
  "editor.stablePeek": false,

  // Controls whether some suggestion types should be filtered from IntelliSense. A list of suggestion types can be found here: https://code.visualstudio.com/docs/editor/intellisense#_types-of-compl
etions.
  "editor.suggest.filteredTypes": {
    "keyword": true
  },

  // Controls whether filtering and sorting suggestions accounts for small typos.
  "editor.suggest.filterGraceful": true,
```

```
// Controls whether sorting favours words that appear close to the cursor.
"editor.suggest.localityBonus": false,

// Controls how many suggestions IntelliSense will show before showing a scrollbar (maximum 15).
"editor.suggest.maxVisibleSuggestions": 12,

// Controls whether remembered suggestion selections are shared between multiple workspaces and windows (needs `editor.suggestSelection`).
"editor.suggest.shareSuggestSelections": false,

// Controls whether to show or hide icons in suggestions.
"editor.suggest.showIcons": true,

// Control whether an active snippet prevents quick suggestions.
"editor.suggest.snippetsPreventQuickSuggestions": true,

// Font size for the suggest widget. When set to `0`, the value of `editor.fontSize` is used.
"editor.suggestFontSize": 0,

// Line height for the suggest widget. When set to `0`, the value of `editor.lineHeight` is used.
"editor.suggestLineHeight": 0,

// Controls whether suggestions should automatically show up when typing trigger characters.
"editor.suggestOnTriggerCharacters": true,

// Controls how suggestions are pre-selected when showing the suggest list.
//  - first: Always select the first suggestion.
//  - recentlyUsed: Select recent suggestions unless further typing selects one, e.g. `console.| -> console.log` because `log` has been completed recently.
//  - recentlyUsedByPrefix: Select suggestions based on previous prefixes that have completed those suggestions, e.g. `co -> console` and `con -> const`.
"editor.suggestSelection": "recentlyUsed",

// Enables tab completions.
//  - on: Tab complete will insert the best matching suggestion when pressing tab.
//  - off: Disable tab completions.
//  - onlySnippets: Tab complete snippets when their prefix match. Works best when 'quickSuggestions' aren't enabled.
"editor.tabCompletion": "off",

// The number of spaces a tab is equal to. This setting is overridden based on the file contents when `editor.detectIndentation` is on.
"editor.tabSize": 4,

// Overrides editor colors and font style from the currently selected color theme.
"editor.tokenColorCustomizations": {},

// Remove trailing auto inserted whitespace.
"editor.trimAutoWhitespace": true,

// Inserting and deleting whitespace follows tab stops.
"editor.useTabStops": true,

// Controls whether completions should be computed based on words in the document.
"editor.wordBasedSuggestions": true,

// Characters that will be used as word separators when doing word related navigations or operations.
"editor.wordSeparators": "`~!@#$%^&*()-=+[{]}\\|;:'\",.<>/?",

// Controls how lines should wrap.
//  - off: Lines will never wrap.
//  - on: Lines will wrap at the viewport width.
//  - wordWrapColumn: Lines will wrap at `editor.wordWrapColumn`.
//  - bounded: Lines will wrap at the minimum of viewport and `editor.wordWrapColumn`.
"editor.wordWrap": "off",
```

```
  // Controls the wrapping column of the editor when `editor.wordWrap` is `wordWrapColumn` or `bounded`.
  "editor.wordWrapColumn": 80,

  // Controls the indentation of wrapped lines.
  //  - none: No indentation. Wrapped lines begin at column 1.
  //  - same: Wrapped lines get the same indentation as the parent.
  //  - indent: Wrapped lines get +1 indentation toward the parent.
  //  - deepIndent: Wrapped lines get +2 indentation toward the parent.
  "editor.wrappingIndent": "same",

// SCM

  // Controls whether inline actions are always visible in the Source Control view.
  "scm.alwaysShowActions": false,

  // Controls whether to always show the Source Control Provider section.
  "scm.alwaysShowProviders": false,

  // Controls diff decorations in the editor.
  "scm.diffDecorations": "all",

  // Controls the width(px) of diff decorations in gutter (added & modified).
  "scm.diffDecorationsGutterWidth": 3,

  // Controls how many providers are visible in the Source Control Provider section. Set to `0` to be able to manually resize the view.
  "scm.providers.visible": 10,

// Workbench

  // Controls the visibility of the activity bar in the workbench.
  "workbench.activityBar.visible": true,

  // Overrides colors from the currently selected color theme.
  "workbench.colorCustomizations": {},

  // Specifies the color theme used in the workbench.
  "workbench.colorTheme": "Default Dark+",

  // Controls the number of recently used commands to keep in history for the command palette. Set to 0 to disable command history.
  "workbench.commandPalette.history": 50,

  // Controls whether the last typed input to the command palette should be restored when opening it the next time.
  "workbench.commandPalette.preserveInput": false,

  // Controls if the centered layout should automatically resize to maximum width when more than one group is open. Once only one group is open it will resize back to the original centered width.
  "workbench.editor.centeredLayoutAutoResize": true,

  // Controls the behavior of empty editor groups when the last tab in the group is closed. When enabled, empty groups will automatically close. When disabled, empty groups will remain part of the g
rid.
  "workbench.editor.closeEmptyGroups": true,

  // Controls whether editors showing a file that was opened during the session should close automatically when getting deleted or renamed by some other process. Disabling this will keep the editor
 open  on such an event. Note that deleting from within the application will always close the editor and that dirty files will never close to preserve your data.
  "workbench.editor.closeOnFileDelete": false,

  // Controls whether opened editors show as preview. Preview editors are reused until they are pinned (e.g. via double click or editing) and show up with an italic font style.
  "workbench.editor.enablePreview": true,

  // Controls whether opened editors from Quick Open show as preview. Preview editors are reused until they are pinned (e.g. via double click or editing).
  "workbench.editor.enablePreviewFromQuickOpen": true,
```

```
  // Controls whether tabs are closed in most recently used order or from left to right.
  "workbench.editor.focusRecentEditorAfterClose": true,

  // Controls whether a top border is drawn on modified (dirty) editor tabs or not.
  "workbench.editor.highlightModifiedTabs": false,

  // Controls the format of the label for an editor.
  //  - default: Show the name of the file. When tabs are enabled and two files have the same name in one group the distinguishing sections of each file's path are added. When tabs are disabled, the
path relative to the workspace folder is shown if the editor is active.
  //  - short: Show the name of the file followed by its directory name.
  //  - medium: Show the name of the file followed by its path relative to the workspace folder.
  //  - long: Show the name of the file followed by its absolute path.
  "workbench.editor.labelFormat": "default",

  // Controls where editors open. Select `left` or `right` to open editors to the left or right of the currently active one. Select `first` or `last` to open editors independently from the currently
active one.
  "workbench.editor.openPositioning": "right",

  // Controls the default direction of editors that are opened side by side (e.g. from the explorer). By default, editors will open on the right hand side of the currently active one. If changed to
 `down`, the editors will open below the currently active one.
  "workbench.editor.openSideBySideDirection": "right",

  // Restores the last view state (e.g. scroll position) when re-opening files after they have been closed.
  "workbench.editor.restoreViewState": true,

  // Controls whether an editor is revealed in any of the visible groups if opened. If disabled, an editor will prefer to open in the currently active editor group. If enabled, an already opened edi
tor will be revealed instead of opened again in the currently active editor group. Note that there are some cases where this setting is ignored, e.g. when forcing an editor to open in a specific gro
up or to the side of the currently active group.
  "workbench.editor.revealIfOpen": false,

  // Controls whether opened editors should show with an icon or not. This requires an icon theme to be enabled as well.
  "workbench.editor.showIcons": true,

  // Controls whether opened editors should show in tabs or not.
  "workbench.editor.showTabs": true,

  // Navigate between open files using three-finger swipe horizontally.
  "workbench.editor.swipeToNavigate": false,

  // Controls the position of the editor's tabs close buttons, or disables them when set to 'off'.
  "workbench.editor.tabCloseButton": "right",

  // Controls the sizing of editor tabs.
  //  - fit: Always keep tabs large enough to show the full editor label.
  //  - shrink: Allow tabs to get smaller when the available space is not enough to show all tabs at once.
  "workbench.editor.tabSizing": "fit",

  // Controls font aliasing method in the workbench.
  //  - default: Sub-pixel font smoothing. On most non-retina displays this will give the sharpest text.
  //  - antialiased: Smooth the font on the level of the pixel, as opposed to the subpixel. Can make the font appear lighter overall.
  //  - none: Disables font smoothing. Text will show with jagged sharp edges.
  //  - auto: Applies `default` or `antialiased` automatically based on the DPI of displays.
  "workbench.fontAliasing": "default",

  // Specifies the icon theme used in the workbench or 'null' to not show any file icons.
  //  - null: No file icons
  //  - vs-minimal
  //  - vs-seti
  "workbench.iconTheme": "vs-seti",
```

```
  // Controls whether keyboard navigation in lists and trees is automatically triggered simply by typing. If set to `false`, keyboard navigation is only triggered when executing the `list.toggleKeyb
oardNavigation` command, for which you can assign a keyboard shortcut.
  "workbench.list.automaticKeyboardNavigation": true,

  // Controls whether lists and trees support horizontal scrolling in the workbench.
  "workbench.list.horizontalScrolling": false,

  // Controls the keyboard navigation style for lists and trees in the workbench. Can be simple, highlight and filter.
  //  - simple: Simple keyboard navigation focuses elements which match the keyboard input. Matching is done only on prefixes.
  //  - highlight: Highlight keyboard navigation highlights elements which match the keyboard input. Further up and down navigation will traverse only the highlighted elements.
  //  - filter: Filter keyboard navigation will filter out and hide all the elements which do not match the keyboard input.
  "workbench.list.keyboardNavigation": "highlight",

  // The modifier to be used to add an item in trees and lists to a multi-selection with the mouse (for example in the explorer, open editors and scm view). The 'Open to Side' mouse gestures - if su
pported - will adapt such that they do not conflict with the multiselect modifier.
  //  - ctrlCmd: Maps to `Control` on Windows and Linux and to `Command` on macOS.
  //  - alt: Maps to `Alt` on Windows and Linux and to `Option` on macOS.
  "workbench.list.multiSelectModifier": "ctrlCmd",

  // Controls how to open items in trees and lists using the mouse (if supported). For parents with children in trees, this setting will control if a single click expands the parent or a double clic
k. Note that some trees and lists might choose to ignore this setting if it is not applicable.
  "workbench.list.openMode": "singleClick",

  // Controls the default location of the panel (terminal, debug console, output, problems). It can either show at the bottom or on the right of the workbench.
  "workbench.panel.defaultLocation": "bottom",

  // Controls whether Quick Open should close automatically once it loses focus.
  "workbench.quickOpen.closeOnFocusLost": true,

  // Controls whether the last typed input to Quick Open should be restored when opening it the next time.
  "workbench.quickOpen.preserveInput": false,

  // Determines which settings editor to use by default.
  //  - ui: Use the settings UI editor.
  //  - json: Use the JSON file editor.
  "workbench.settings.editor": "ui",

  // Controls whether to enable the natural language search mode for settings. The natural language search is provided by a Microsoft online service.
  "workbench.settings.enableNaturalLanguageSearch": true,

  // Controls whether opening keybinding settings also opens an editor showing all default keybindings.
  "workbench.settings.openDefaultKeybindings": false,

  // Controls whether opening settings also opens an editor showing all default settings.
  "workbench.settings.openDefaultSettings": false,

  // Controls the behavior of the settings editor Table of Contents while searching.
  //  - hide: Hide the Table of Contents while searching.
  //  - filter: Filter the Table of Contents to just categories that have matching settings. Clicking a category will filter the results to that category.
  "workbench.settings.settingsSearchTocBehavior": "filter",

  // Controls whether to use the split JSON editor when editing settings as JSON.
  "workbench.settings.useSplitJSON": false,

  // Controls the location of the sidebar. It can either show on the left or right of the workbench.
  "workbench.sideBar.location": "left",

  // Controls which editor is shown at startup, if none are restored from the previous session.
  //  - none: Start without an editor.
  //  - welcomePage: Open the Welcome page (default).
  //  - readme: Open the README when opening a folder that contains one, fallback to 'welcomePage' otherwise.
```

```
  //  - newUntitledFile: Open a new untitled file (only applies when opening an empty workspace).
  //  - welcomePageInEmptyWorkbench: Open the Welcome page when opening an empty workbench.
  "workbench.startupEditor": "welcomePage",

  // Controls the visibility of the status bar at the bottom of the workbench.
  "workbench.statusBar.visible": true,

  // When enabled, will show the watermark tips when no editor is open.
  "workbench.tips.enabled": true,

  // Controls tree indentation in pixels.
  "workbench.tree.indent": 8,

  // Controls whether the tree should render indent guides.
  "workbench.tree.renderIndentGuides": "onHover",

  // Controls the visibility of view header actions. View header actions may either be always visible, or only visible when that view is focused or hovered over.
  "workbench.view.alwaysShowHeaderActions": false,

// Window

  // If enabled, will automatically change to high contrast theme if Windows is using a high contrast theme, and to dark theme when switching away from a Windows high contrast theme.
  "window.autoDetectHighContrast": true,

  // If enabled, clicking on an inactive window will both activate the window and trigger the element under the mouse if it is clickable. If disabled, clicking anywhere on an inactive window will ac
tivate it only and a second click is required on the element.
  "window.clickThroughInactive": true,

  // Controls whether closing the last editor should also close the window. This setting only applies for windows that do not show folders.
  "window.closeWhenEmpty": false,

  // Controls whether the menu bar will be focused by pressing the Alt-key. This setting has no effect on toggling the menu bar with the Alt-key.
  "window.customMenuBarAltFocus": true,

  // If enabled, double clicking the application icon in the title bar will close the window and the window cannot be dragged by the icon. This setting only has an effect when `window.titleBarStyle`
is set to `custom`.
  "window.doubleClickIconToClose": false,

  // Enables macOS Sierra window tabs. Note that changes require a full restart to apply and that native tabs will disable a custom title bar style if configured.
  "window.nativeTabs": false,

  // If enabled, the main menus can be opened via Alt-key shortcuts. Disabling mnemonics allows to bind these Alt-key shortcuts to editor commands instead.
  "window.enableMenuBarMnemonics": true,

  // Control the visibility of the menu bar. A setting of 'toggle' means that the menu bar is hidden and a single press of the Alt key will show it. By default, the menu bar will be visible, unless
 the window is full screen.
  //  - default: Menu is only hidden in full screen mode.
  //  - visible: Menu is always visible even in full screen mode.
  //  - toggle: Menu is hidden but can be displayed via Alt key.
  //  - hidden: Menu is always hidden.
  "window.menuBarVisibility": "default",

  // Controls the dimensions of opening a new window when at least one window is already opened. Note that this setting does not have an impact on the first window that is opened. The first window w
ill always restore the size and location as you left it before closing.
  //  - default: Open new windows in the center of the screen.
  //  - inherit: Open new windows with same dimension as last active one.
  //  - maximized: Open new windows maximized.
  //  - fullscreen: Open new windows in full screen mode.
  "window.newWindowDimensions": "default",

  // Controls whether files should open in a new window.
```

```
  // Note that there can still be cases where this setting is ignored (e.g. when using the `--new-window` or `--reuse-window` command line option).
  //  - on: Files will open in a new window.
  //  - off: Files will open in the window with the files' folder open or the last active window.
  //  - default: Files will open in a new window unless picked from within the application (e.g. via the File menu).
  "window.openFilesInNewWindow": "off",

  // Controls whether folders should open in a new window or replace the last active window.
  // Note that there can still be cases where this setting is ignored (e.g. when using the `--new-window` or `--reuse-window` command line option).
  //  - on: Folders will open in a new window.
  //  - off: Folders will replace the last active window.
  //  - default: Folders will open in a new window unless a folder is picked from within the application (e.g. via the File menu).
  "window.openFoldersInNewWindow": "default",

  // Controls whether a new empty window should open when starting a second instance without arguments or if the last running instance should get focus.
  // Note that there can still be cases where this setting is ignored (e.g. when using the `--new-window` or `--reuse-window` command line option).
  //  - on: Open a new empty window.
  //  - off: Focus the last active running instance.
  "window.openWithoutArgumentsInNewWindow": "on",

  // Controls whether a window should restore to full screen mode if it was exited in full screen mode.
  "window.restoreFullscreen": false,

  // Controls how windows are being reopened after a restart.
  //  - all: Reopen all windows.
  //  - folders: Reopen all folders. Empty workspaces will not be restored.
  //  - one: Reopen the last active window.
  //  - none: Never reopen a window. Always start with an empty one.
  "window.restoreWindows": "one",

  // Controls the window title based on the active editor. Variables are substituted based on the context:`${activeEditorShort}`: the file name (e.g. myFile.txt).
  // - `${activeEditorMedium}`: the path of the file relative to the workspace folder (e.g. myFolder/myFileFolder/myFile.txt).
  // - `${activeEditorLong}`: the full path of the file (e.g. /Users/Development/myFolder/myFileFolder/myFile.txt).
  // - `${activeFolderShort}`: the name of the folder the file is contained in (e.g. myFileFolder).
  // - `${activeFolderMedium}`: the path of the folder the file is contained in, relative to the workspace folder (e.g. myFolder/myFileFolder).
  // - `${activeFolderLong}`: the full path of the folder the file is contained in (e.g. /Users/Development/myFolder/myFileFolder).
  // - `${folderName}`: name of the workspace folder the file is contained in (e.g. myFolder).
  // - `${folderPath}`: file path of the workspace folder the file is contained in (e.g. /Users/Development/myFolder).
  // - `${rootName}`: name of the workspace (e.g. myFolder or myWorkspace).
  // - `${rootPath}`: file path of the workspace (e.g. /Users/Development/myWorkspace).
  // - `${appName}`: e.g. VS Code.
  // - `${dirty}`: a dirty indicator if the active editor is dirty.
  // - `${separator}`: a conditional separator (" - ") that only shows when surrounded by variables with values or static text.
  "window.title": "${dirty}${activeEditorShort}${separator}${rootName}${separator}${appName}",

  // Adjust the appearance of the window title bar. On Linux and Windows, this setting also affects the application and context menu appearances. Changes require a full restart to apply.
  "window.titleBarStyle": "custom",

  // Adjust the zoom level of the window. The original size is 0 and each increment above (e.g. 1) or below (e.g. -1) represents zooming 20% larger or smaller. You can also enter decimals to adjust
 the zoom level with a finer granularity.
  "window.zoomLevel": 0,

// Files

  // Configure file associations to languages (e.g. `"*.extension": "html"`). These have precedence over the default associations of the languages installed.
  "files.associations": {},

  // When enabled, the editor will attempt to guess the character set encoding when opening files. This setting can also be configured per language.
  "files.autoGuessEncoding": false,

  // Controls auto save of dirty files. Read more about autosave [here](https://code.visualstudio.com/docs/editor/codebasics#_save-auto-save).
  //  - off: A dirty file is never automatically saved.
```

```
  //  - afterDelay: A dirty file is automatically saved after the configured `files.autoSaveDelay`.
  //  - onFocusChange: A dirty file is automatically saved when the editor loses focus.
  //  - onWindowChange: A dirty file is automatically saved when the window loses focus.
  "files.autoSave": "off",

  // Controls the delay in ms after which a dirty file is saved automatically. Only applies when `files.autoSave` is set to `afterDelay`.
  "files.autoSaveDelay": 1000,

  // The default language mode that is assigned to new files.
  "files.defaultLanguage": "",

  // Moves files/folders to the OS trash (recycle bin on Windows) when deleting. Disabling this will delete files/folders permanently.
  "files.enableTrash": true,

  // The default character set encoding to use when reading and writing files. This setting can also be configured per language.
  "files.encoding": "utf8",

  // The default end of line character.
  //  - \n: LF
  //  - \r\n: CRLF
  //  - auto: Uses operating system specific end of line character.
  "files.eol": "auto",

  // Configure glob patterns for excluding files and folders. For example, the files explorer decides which files and folders to show or hide based on this setting. Read more about glob patterns [he
re](https://code.visualstudio.com/docs/editor/codebasics#_advanced-search-options).
  "files.exclude": {
    "**/.git": true,
    "**/.svn": true,
    "**/.hg": true,
    "**/CVS": true,
    "**/.DS_Store": true
  },

  // Controls whether unsaved files are remembered between sessions, allowing the save prompt when exiting the editor to be skipped.
  //  - off: Disable hot exit.
  //  - onExit: Hot exit will be triggered when the last window is closed on Windows/Linux or when the `workbench.action.quit` command is triggered (command palette, keybinding, menu). All windows w
ith backups will be restored upon next launch.
  //  - onExitAndWindowClose: Hot exit will be triggered when the last window is closed on Windows/Linux or when the `workbench.action.quit` command is triggered (command palette, keybinding, menu),
and also for any window with a folder opened regardless of whether it's the last window. All windows without folders opened will be restored upon next launch. To restore folder windows as they were
 before shutdown set `window.restoreWindows` to `all`.
  "files.hotExit": "onExit",

  // When enabled, insert a final new line at the end of the file when saving it.
  "files.insertFinalNewline": false,

  // Controls the memory available to VS Code after restart when trying to open large files. Same effect as specifying `--max-memory=NEWSIZE` on the command line.
  "files.maxMemoryForLargeFilesMB": 4096,

  // When enabled, will trim all new lines after the final new line at the end of the file when saving it.
  "files.trimFinalNewlines": false,

  // When enabled, will trim trailing whitespace when saving a file.
  "files.trimTrailingWhitespace": false,

  // Configure glob patterns of file paths to exclude from file watching. Patterns must match on absolute paths (i.e. prefix with ** or the full path to match properly). Changing this setting requir
es a restart. When you experience Code consuming lots of cpu time on startup, you can exclude large folders to reduce the initial load.
  "files.watcherExclude": {
    "**/.git/objects/**": true,
    "**/.git/subtree-cache/**": true,
    "**/node_modules/*/**": true
  },
```

```
// Zen Mode

  // Controls whether turning on Zen Mode also centers the layout.
  "zenMode.centerLayout": true,

  // Controls whether turning on Zen Mode also puts the workbench into full screen mode.
  "zenMode.fullScreen": true,

  // Controls whether turning on Zen Mode also hides the activity bar at the left of the workbench.
  "zenMode.hideActivityBar": true,

  // Controls whether turning on Zen Mode also hides the editor line numbers.
  "zenMode.hideLineNumbers": true,

  // Controls whether turning on Zen Mode also hides the status bar at the bottom of the workbench.
  "zenMode.hideStatusBar": true,

  // Controls whether turning on Zen Mode also hides workbench tabs.
  "zenMode.hideTabs": true,

  // Controls whether a window should restore to zen mode if it was exited in zen mode.
  "zenMode.restore": false,

// File Explorer

  // Controls whether the explorer should automatically reveal and select files when opening them.
  "explorer.autoReveal": true,

  // Controls whether the explorer should ask for confirmation when deleting a file via the trash.
  "explorer.confirmDelete": true,

  // Controls whether the explorer should ask for confirmation to move files and folders via drag and drop.
  "explorer.confirmDragAndDrop": true,

  // Controls whether file decorations should use badges.
  "explorer.decorations.badges": true,

  // Controls whether file decorations should use colors.
  "explorer.decorations.colors": true,

  // Controls whether the explorer should allow to move files and folders via drag and drop.
  "explorer.enableDragAndDrop": true,

  // Number of editors shown in the Open Editors pane.
  "explorer.openEditors.visible": 9,

  // Controls sorting order of files and folders in the explorer.
  //  - default: Files and folders are sorted by their names, in alphabetical order. Folders are displayed before files.
  //  - mixed: Files and folders are sorted by their names, in alphabetical order. Files are interwoven with folders.
  //  - filesFirst: Files and folders are sorted by their names, in alphabetical order. Files are displayed before folders.
  //  - type: Files and folders are sorted by their extensions, in alphabetical order. Folders are displayed before files.
  //  - modified: Files and folders are sorted by last modified date, in descending order. Folders are displayed before files.
  "explorer.sortOrder": "default",

// Search

  // Controls the positioning of the actionbar on rows in the search view.
  //  - auto: Position the actionbar to the right when the search view is narrow, and immediately after the content when the search view is wide.
  //  - right: Always position the actionbar to the right.
  "search.actionsPosition": "auto",
```

```
  // Controls whether the search results will be collapsed or expanded.
  //  - auto: Files with less than 10 results are expanded. Others are collapsed.
  //  - alwaysCollapse
  //  - alwaysExpand
  "search.collapseResults": "auto",

  // Configure glob patterns for excluding files and folders in searches. Inherits all glob patterns from the `files.exclude` setting. Read more about glob patterns [here](https://code.visualstudio.
com/docs/editor/codebasics#_advanced-search-options).
  "search.exclude": {
    "**/node_modules": true,
    "**/bower_components": true
  },

  // Controls whether to follow symlinks while searching.
  "search.followSymlinks": true,

  // Controls whether the search view should read or modify the shared find clipboard on macOS.
  "search.globalFindClipboard": false,

  // Controls whether the search will be shown as a view in the sidebar or as a panel in the panel area for more horizontal space.
  "search.location": "sidebar",

  // Whether to include results from recently opened files in the file results for Quick Open.
  "search.quickOpen.includeHistory": true,

  // Whether to include results from a global symbol search in the file results for Quick Open.
  "search.quickOpen.includeSymbols": false,

  // Controls whether to show line numbers for search results.
  "search.showLineNumbers": false,

  // Search case-insensitively if the pattern is all lowercase, otherwise, search case-sensitively.
  "search.smartCase": false,

  // Controls whether to use global `.gitignore` and `.ignore` files when searching for files.
  "search.useGlobalIgnoreFiles": false,

  // Controls whether to use `.gitignore` and `.ignore` files when searching for files.
  "search.useIgnoreFiles": true,

  // Whether to use the PCRE2 regex engine in text search. This enables using some advanced regex features like lookahead and backreferences. However, not all PCRE2 features are supported - only fea
tures that are also supported by JavaScript.
  "search.usePCRE2": false,

  // Controls whether to open Replace Preview when selecting or replacing a match.
  "search.useReplacePreview": true,

// HTTP

  // The proxy setting to use. If not set will be taken from the http_proxy and https_proxy environment variables.
  "http.proxy": "",

  // The value to send as the 'Proxy-Authorization' header for every network request.
  "http.proxyAuthorization": null,

  // Controls whether the proxy server certificate should be verified against the list of supplied CAs.
  "http.proxyStrictSSL": true,

  // Use the proxy support for extensions.
  //  - off: Disable proxy support for extensions.
```

```
  //  - on: Enable proxy support for extensions.
  //  - override: Enable proxy support for extensions, override request options.
  "http.proxySupport": "override",

  // Controls whether CA certificates should be loaded from the OS. (On Windows and macOS a reload of the window is required after turning this off.)
  "http.systemCertificates": true,

// Keyboard

  // Controls the dispatching logic for key presses to use either `code` (recommended) or `keyCode`.
  "keyboard.dispatch": "code",

  // Enables the macOS touchbar buttons on the keyboard if available.
  "keyboard.touchbar.enabled": true,

// Update

  // Enable to download and install new VS Code Versions in the background on Windows
  "update.enableWindowsBackgroundUpdates": true,

  // Configure whether you receive automatic updates. Requires a restart after change. The updates are fetched from a Microsoft online service.
  //  - none: Disable updates.
  //  - manual: Disable automatic background update checks. Updates will be available if you manually check for updates.
  //  - default: Enable automatic update checks. Code will check for updates automatically and periodically.
  "update.mode": "default",

  // Show Release Notes after an update. The Release Notes are fetched from a Microsoft online service.
  "update.showReleaseNotes": true,

// Debug

  // Allow setting breakpoints in any file.
  "debug.allowBreakpointsEverywhere": false,

  // Controls the font family in the debug console.
  "debug.console.fontFamily": "default",

  // Controls the font size in pixels in the debug console.
  "debug.console.fontSize": 14,

  // Controls the line height in pixels in the debug console. Use 0 to compute the line height from the font size.
  "debug.console.lineHeight": 0,

  // Controls if the lines should wrap in the debug console.
  "debug.console.wordWrap": true,

  // Controls whether the non-debug hovers should be enabled while debugging. When enabled the hover providers will be called to provide a hover. Regular hovers will not be shown even if this settin
g is enabled.
  "debug.enableAllHovers": false,

  // Show variable values inline in editor while debugging.
  "debug.inlineValues": false,

  // Controls when the internal debug console should open.
  "debug.internalConsoleOptions": "openOnFirstSessionStart",

  // Controls when the debug view should open.
  "debug.openDebug": "openOnSessionStart",

  // Automatically open the explorer view at the end of a debug session.
  "debug.openExplorerOnEnd": false,
```

```
  // Controls when the debug status bar should be visible.
  //  - never: Never show debug in status bar
  //  - always: Always show debug in status bar
  //  - onFirstSessionStart: Show debug in status bar only after debug was started for the first time
  "debug.showInStatusBar": "onFirstSessionStart",

  // Controls whether the debug sub-sessions are shown in the debug tool bar. When this setting is false the stop command on a sub-session will also stop the parent session.
  "debug.showSubSessionsInToolBar": false,

  // Controls the location of the debug toolbar. Either `floating` in all views, `docked` in the debug view, or `hidden`.
  "debug.toolBarLocation": "floating",

  // Global debug launch configuration. Should be used as an alternative to 'launch.json' that is shared across workspaces.
  "launch": {
    "configurations": [],
    "compounds": []
  },

// HTML

  // Enable/disable autoclosing of HTML tags.
  "html.autoClosingTags": true,

  // List of tags, comma separated, where the content shouldn't be reformatted. `null` defaults to the `pre` tag.
  "html.format.contentUnformatted": "pre,code,textarea",

  // Enable/disable default HTML formatter.
  "html.format.enable": true,

  // End with a newline.
  "html.format.endWithNewline": false,

  // List of tags, comma separated, that should have an extra newline before them. `null` defaults to `"head, body, /html"`.
  "html.format.extraLiners": "head, body, /html",

  // Format and indent ``.
  "html.format.indentHandlebars": false,

  // Indent `<head>` and `<body>` sections.
  "html.format.indentInnerHtml": false,

  // Maximum number of line breaks to be preserved in one chunk. Use `null` for unlimited.
  "html.format.maxPreserveNewLines": null,

  // Controls whether existing line breaks before elements should be preserved. Only works before elements, not inside tags or for text.
  "html.format.preserveNewLines": true,

  // List of tags, comma separated, that shouldn't be reformatted. `null` defaults to all tags listed at https://www.w3.org/TR/html5/dom.html#phrasing-content.
  "html.format.unformatted": "wbr",

  // Wrap attributes.
  //  - auto: Wrap attributes only when line length is exceeded.
  //  - force: Wrap each attribute except first.
  //  - force-aligned: Wrap each attribute except first and keep aligned.
  //  - force-expand-multiline: Wrap each attribute.
  //  - aligned-multiple: Wrap when line length is exceeded, align attributes vertically.
  //  - preserve: Preserve wrapping of attributes
  //  - preserve-aligned: Preserve wrapping of attributes but align.
  "html.format.wrapAttributes": "auto",
```

```
  // Maximum amount of characters per line (0 = disable).
  "html.format.wrapLineLength": 120,

  // Controls whether the built-in HTML language support suggests HTML5 tags, properties and values.
  "html.suggest.html5": true,

  // Traces the communication between VS Code and the HTML language server.
  "html.trace.server": "off",

  // Controls whether the built-in HTML language support validates embedded scripts.
  "html.validate.scripts": true,

  // Controls whether the built-in HTML language support validates embedded styles.
  "html.validate.styles": true,

// JSON

  // Enable/disable default JSON formatter
  "json.format.enable": true,

  // Associate schemas to JSON files in the current project
  "json.schemas": [],

  // Traces the communication between VS Code and the JSON language server.
  "json.trace.server": "off",

// Markdown

  // Sets how line-breaks are rendered in the markdown preview. Setting it to 'true' creates a <br> for every newline.
  "markdown.preview.breaks": false,

  // Double click in the markdown preview to switch to the editor.
  "markdown.preview.doubleClickToSwitchToEditor": true,

  // Controls the font family used in the markdown preview.
  "markdown.preview.fontFamily": "-apple-system, BlinkMacSystemFont, 'Segoe WPC', 'Segoe UI', 'Ubuntu', 'Droid Sans', sans-serif",

  // Controls the font size in pixels used in the markdown preview.
  "markdown.preview.fontSize": 14,

  // Controls the line height used in the markdown preview. This number is relative to the font size.
  "markdown.preview.lineHeight": 1.6,

  // Enable or disable conversion of URL-like text to links in the markdown preview.
  "markdown.preview.linkify": true,

  // Mark the current editor selection in the markdown preview.
  "markdown.preview.markEditorSelection": true,

  // How should clicking on links to markdown files be handled in the preview.
  //  - inPreview: Try to open links in the markdown preview
  //  - inEditor: Try to open links in the editor
  "markdown.preview.openMarkdownLinks": "inPreview",

  // When a markdown preview is scrolled, update the view of the editor.
  "markdown.preview.scrollEditorWithPreview": true,

  // When a markdown editor is scrolled, update the view of the preview.
  "markdown.preview.scrollPreviewWithEditor": true,

  // A list of URLs or local paths to CSS style sheets to use from the markdown preview. Relative paths are interpreted relative to the folder open in the explorer. If there is no open folder, they
```

```
 are interpreted relative to the location of the markdown file. All '\' need to be written as '\\'.
  "markdown.styles": [],

  // Enable debug logging for the markdown extension.
  "markdown.trace": "off",

// PHP

  // Controls whether the built-in PHP language suggestions are enabled. The support suggests PHP globals and variables.
  "php.suggest.basic": true,

  // Enable/disable built-in PHP validation.
  "php.validate.enable": true,

  // Points to the PHP executable.
  "php.validate.executablePath": null,

  // Whether the linter is run on save or on type.
  "php.validate.run": "onSave",

// TypeScript

  // Enable/disable automatic closing of JSX tags. Requires using TypeScript 3.0 or newer in the workspace.
  "javascript.autoClosingTags": true,

  // Enable/disable default JavaScript formatter.
  "javascript.format.enable": true,

  // Defines space handling after a comma delimiter.
  "javascript.format.insertSpaceAfterCommaDelimiter": true,

  // Defines space handling after the constructor keyword. Requires using TypeScript 2.3.0 or newer in the workspace.
  "javascript.format.insertSpaceAfterConstructor": false,

  // Defines space handling after function keyword for anonymous functions.
  "javascript.format.insertSpaceAfterFunctionKeywordForAnonymousFunctions": true,

  // Defines space handling after keywords in a control flow statement.
  "javascript.format.insertSpaceAfterKeywordsInControlFlowStatements": true,

  // Defines space handling after opening and before closing JSX expression braces.
  "javascript.format.insertSpaceAfterOpeningAndBeforeClosingJsxExpressionBraces": false,

  // Defines space handling after opening and before closing non-empty braces. Requires using TypeScript 2.3.0 or newer in the workspace.
  "javascript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyBraces": true,

  // Defines space handling after opening and before closing non-empty brackets.
  "javascript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyBrackets": false,

  // Defines space handling after opening and before closing non-empty parenthesis.
  "javascript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyParenthesis": false,

  // Defines space handling after opening and before closing template string braces.
  "javascript.format.insertSpaceAfterOpeningAndBeforeClosingTemplateStringBraces": false,

  // Defines space handling after a semicolon in a for statement.
  "javascript.format.insertSpaceAfterSemicolonInForStatements": true,

  // Defines space handling after a binary operator.
  "javascript.format.insertSpaceBeforeAndAfterBinaryOperators": true,
```

```
// Defines space handling before function argument parentheses.
"javascript.format.insertSpaceBeforeFunctionParenthesis": false,

// Defines whether an open brace is put onto a new line for control blocks or not.
"javascript.format.placeOpenBraceOnNewLineForControlBlocks": false,

// Defines whether an open brace is put onto a new line for functions or not.
"javascript.format.placeOpenBraceOnNewLineForFunctions": false,

// Enable/disable semantic checking of JavaScript files. Existing jsconfig.json or tsconfig.json files override this setting. Requires using TypeScript 2.3.1 or newer in the workspace.
"javascript.implicitProjectConfig.checkJs": false,

// Preferred path style for auto imports.
//  - auto: Automatically select import path style. Prefers using a relative import if `baseUrl` is configured and the relative path has fewer segments than the non-relative import.
//  - relative: Relative to the file location.
//  - non-relative: Based on the `baseUrl` configured in your `jsconfig.json` / `tsconfig.json`.
"javascript.preferences.importModuleSpecifier": "auto",

// Preferred quote style to use for quick fixes: `single` quotes, `double` quotes, or `auto` infer quote type from existing imports. Requires using TypeScript 2.9 or newer in the workspace.
"javascript.preferences.quoteStyle": "auto",

// Enable/disable introducing aliases for object shorthand properties during renames. Requires using TypeScript 3.4 or newer in the workspace.
"javascript.preferences.renameShorthandProperties": true,

// Enable/disable references CodeLens in JavaScript files.
"javascript.referencesCodeLens.enabled": false,

// Enable/disable auto import suggestions. Requires using TypeScript 2.6.1 or newer in the workspace.
"javascript.suggest.autoImports": true,

// Complete functions with their parameter signature.
"javascript.suggest.completeFunctionCalls": false,

// Enable/disable suggestion to complete JSDoc comments.
"javascript.suggest.completeJSDocs": true,

// Enabled/disable autocomplete suggestions.
"javascript.suggest.enabled": true,

// Enable/disable including unique names from the file in JavaScript suggestions.
"javascript.suggest.names": true,

// Enable/disable suggestions for paths in import statements and require calls.
"javascript.suggest.paths": true,

// Enable/disable suggestion diagnostics for JavaScript files in the editor. Requires using TypeScript 2.8 or newer in the workspace.
"javascript.suggestionActions.enabled": true,

// Enable/disable automatic updating of import paths when you rename or move a file in VS Code. Requires using TypeScript 2.9 or newer in the workspace.
//  - prompt: Prompt on each rename.
//  - always: Always update paths automatically.
//  - never: Never rename paths and don't prompt.
"javascript.updateImportsOnFileMove.enabled": "prompt",

// Enable/disable JavaScript validation.
"javascript.validate.enable": true,

// Enable/disable automatic closing of JSX tags. Requires using TypeScript 3.0 or newer in the workspace.
"typescript.autoClosingTags": true,

// Check if npm is installed for Automatic Type Acquisition.
```

```
"typescript.check.npmIsInstalled": true,

// Disables automatic type acquisition. Automatic type acquisition fetches `@types` packages from npm to improve IntelliSense for external libraries.
"typescript.disableAutomaticTypeAcquisition": false,

// Enable/disable default TypeScript formatter.
"typescript.format.enable": true,

// Defines space handling after a comma delimiter.
"typescript.format.insertSpaceAfterCommaDelimiter": true,

// Defines space handling after the constructor keyword. Requires using TypeScript 2.3.0 or newer in the workspace.
"typescript.format.insertSpaceAfterConstructor": false,

// Defines space handling after function keyword for anonymous functions.
"typescript.format.insertSpaceAfterFunctionKeywordForAnonymousFunctions": true,

// Defines space handling after keywords in a control flow statement.
"typescript.format.insertSpaceAfterKeywordsInControlFlowStatements": true,

// Defines space handling after opening and before closing JSX expression braces.
"typescript.format.insertSpaceAfterOpeningAndBeforeClosingJsxExpressionBraces": false,

// Defines space handling after opening and before closing non-empty braces. Requires using TypeScript 2.3.0 or newer in the workspace.
"typescript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyBraces": true,

// Defines space handling after opening and before closing non-empty brackets.
"typescript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyBrackets": false,

// Defines space handling after opening and before closing non-empty parenthesis.
"typescript.format.insertSpaceAfterOpeningAndBeforeClosingNonemptyParenthesis": false,

// Defines space handling after opening and before closing template string braces.
"typescript.format.insertSpaceAfterOpeningAndBeforeClosingTemplateStringBraces": false,

// Defines space handling after a semicolon in a for statement.
"typescript.format.insertSpaceAfterSemicolonInForStatements": true,

// Defines space handling after type assertions in TypeScript. Requires using TypeScript 2.4 or newer in the workspace.
"typescript.format.insertSpaceAfterTypeAssertion": false,

// Defines space handling after a binary operator.
"typescript.format.insertSpaceBeforeAndAfterBinaryOperators": true,

// Defines space handling before function argument parentheses.
"typescript.format.insertSpaceBeforeFunctionParenthesis": false,

// Defines whether an open brace is put onto a new line for control blocks or not.
"typescript.format.placeOpenBraceOnNewLineForControlBlocks": false,

// Defines whether an open brace is put onto a new line for functions or not.
"typescript.format.placeOpenBraceOnNewLineForFunctions": false,

// Enable/disable implementations CodeLens. This CodeLens shows the implementers of an interface.
"typescript.implementationsCodeLens.enabled": false,

// Sets the locale used to report JavaScript and TypeScript errors. Requires using TypeScript 2.6.0 or newer in the workspace. Default of `null` uses VS Code's locale.
"typescript.locale": null,

// Specifies the path to the npm executable used for Automatic Type Acquisition. Requires using TypeScript 2.3.4 or newer in the workspace.
"typescript.npm": null,
```

```
  // Preferred path style for auto imports.
  //  - auto: Automatically select import path style. Prefers using a relative import if `baseUrl` is configured and the relative path has fewer segments than the non-relative import.
  //  - relative: Relative to the file location.
  //  - non-relative: Based on the `baseUrl` configured in your `jsconfig.json` / `tsconfig.json`.
  "typescript.preferences.importModuleSpecifier": "auto",

  // Preferred quote style to use for quick fixes: `single` quotes, `double` quotes, or `auto` infer quote type from existing imports. Requires using TypeScript 2.9 or newer in the workspace.
  "typescript.preferences.quoteStyle": "auto",

  // Enable/disable introducing aliases for object shorthand properties during renames. Requires using TypeScript 3.4 or newer in the workspace.
  "typescript.preferences.renameShorthandProperties": true,

  // Enable/disable references CodeLens in TypeScript files.
  "typescript.referencesCodeLens.enabled": false,

  // Report style checks as warnings.
  "typescript.reportStyleChecksAsWarnings": true,

  // Enable/disable auto import suggestions. Requires using TypeScript 2.6.1 or newer in the workspace.
  "typescript.suggest.autoImports": true,

  // Complete functions with their parameter signature.
  "typescript.suggest.completeFunctionCalls": false,

  // Enable/disable suggestion to complete JSDoc comments.
  "typescript.suggest.completeJSDocs": true,

  // Enabled/disable autocomplete suggestions.
  "typescript.suggest.enabled": true,

  // Enable/disable suggestions for paths in import statements and require calls.
  "typescript.suggest.paths": true,

  // Enable/disable suggestion diagnostics for TypeScript files in the editor. Requires using TypeScript 2.8 or newer in the workspace.
  "typescript.suggestionActions.enabled": true,

  // Controls auto detection of tsc tasks.
  //  - on: Create both build and watch tasks.
  //  - off: Disable this feature.
  //  - build: Only create single run compile tasks.
  //  - watch: Only create compile and watch tasks.
  "typescript.tsc.autoDetect": "on",

  // Specifies the folder path containing the tsserver and lib*.d.ts files to use.
  "typescript.tsdk": null,

  // Enables logging of the TS server to a file. This log can be used to diagnose TS Server issues. The log may contain file paths, source code, and other potentially sensitive information from your
project.
  "typescript.tsserver.log": "off",

  // Additional paths to discover Typescript Language Service plugins. Requires using TypeScript 2.3.0 or newer in the workspace.
  "typescript.tsserver.pluginPaths": [],

  // Enables tracing of messages sent to the TS server. This trace can be used to diagnose TS Server issues. The trace may contain file paths, source code, and other potentially sensitive informatio
n from your project.
  "typescript.tsserver.trace": "off",

  // Enable/disable automatic updating of import paths when you rename or move a file in VS Code. Requires using TypeScript 2.9 or newer in the workspace.
  //  - prompt: Prompt on each rename.
  //  - always: Always update paths automatically.
```

```
  //  - never: Never rename paths and don't prompt.
  "typescript.updateImportsOnFileMove.enabled": "prompt",

  // Enable/disable TypeScript validation.
  "typescript.validate.enable": true,

// CSS

  // By default, VS Code triggers property value completion after selecting a CSS property. Use this setting to disable this behavior.
  "css.completion.triggerPropertyValueCompletion": true,

  // Invalid number of parameters.
  "css.lint.argumentsInColorFunction": "error",

  // Do not use `width` or `height` when using `padding` or `border`.
  "css.lint.boxModel": "ignore",

  // When using a vendor-specific prefix make sure to also include all other vendor-specific properties.
  "css.lint.compatibleVendorPrefixes": "ignore",

  // Do not use duplicate style definitions.
  "css.lint.duplicateProperties": "ignore",

  // Do not use empty rulesets.
  "css.lint.emptyRules": "warning",

  // Avoid using `float`. Floats lead to fragile CSS that is easy to break if one aspect of the layout changes.
  "css.lint.float": "ignore",

  // `@font-face` rule must define `src` and `font-family` properties.
  "css.lint.fontFaceProperties": "warning",

  // Hex colors must consist of three or six hex numbers.
  "css.lint.hexColorLength": "error",

  // Selectors should not contain IDs because these rules are too tightly coupled with the HTML.
  "css.lint.idSelector": "ignore",

  // IE hacks are only necessary when supporting IE7 and older.
  "css.lint.ieHack": "ignore",

  // Avoid using `!important`. It is an indication that the specificity of the entire CSS has gotten out of control and needs to be refactored.
  "css.lint.important": "ignore",

  // Import statements do not load in parallel.
  "css.lint.importStatement": "ignore",

  // Property is ignored due to the display. E.g. with `display: inline`, the `width`, `height`, `margin-top`, `margin-bottom`, and `float` properties have no effect.
  "css.lint.propertyIgnoredDueToDisplay": "warning",

  // The universal selector (`*`) is known to be slow.
  "css.lint.universalSelector": "ignore",

  // Unknown at-rule.
  "css.lint.unknownAtRules": "warning",

  // Unknown property.
  "css.lint.unknownProperties": "warning",

  // Unknown vendor specific property.
  "css.lint.unknownVendorSpecificProperties": "ignore",
```

```
  // A list of properties that are not validated against the `unknownProperties` rule.
  "css.lint.validProperties": [],

  // When using a vendor-specific prefix, also include the standard property.
  "css.lint.vendorPrefix": "warning",

  // No unit for zero needed.
  "css.lint.zeroUnits": "ignore",

  // Traces the communication between VS Code and the CSS language server.
  "css.trace.server": "off",

  // Enables or disables all validations.
  "css.validate": true,

// LESS

  // Invalid number of parameters.
  "less.lint.argumentsInColorFunction": "error",

  // Do not use `width` or `height` when using `padding` or `border`.
  "less.lint.boxModel": "ignore",

  // When using a vendor-specific prefix make sure to also include all other vendor-specific properties.
  "less.lint.compatibleVendorPrefixes": "ignore",

  // Do not use duplicate style definitions.
  "less.lint.duplicateProperties": "ignore",

  // Do not use empty rulesets.
  "less.lint.emptyRules": "warning",

  // Avoid using `float`. Floats lead to fragile CSS that is easy to break if one aspect of the layout changes.
  "less.lint.float": "ignore",

  // `@font-face` rule must define `src` and `font-family` properties.
  "less.lint.fontFaceProperties": "warning",

  // Hex colors must consist of three or six hex numbers.
  "less.lint.hexColorLength": "error",

  // Selectors should not contain IDs because these rules are too tightly coupled with the HTML.
  "less.lint.idSelector": "ignore",

  // IE hacks are only necessary when supporting IE7 and older.
  "less.lint.ieHack": "ignore",

  // Avoid using `!important`. It is an indication that the specificity of the entire CSS has gotten out of control and needs to be refactored.
  "less.lint.important": "ignore",

  // Import statements do not load in parallel.
  "less.lint.importStatement": "ignore",

  // Property is ignored due to the display. E.g. with `display: inline`, the `width`, `height`, `margin-top`, `margin-bottom`, and `float` properties have no effect.
  "less.lint.propertyIgnoredDueToDisplay": "warning",

  // The universal selector (`*`) is known to be slow.
  "less.lint.universalSelector": "ignore",

  // Unknown property.
```

```
  "less.lint.unknownProperties": "warning",

  // Unknown vendor specific property.
  "less.lint.unknownVendorSpecificProperties": "ignore",

  // A list of properties that are not validated against the `unknownProperties` rule.
  "less.lint.validProperties": [],

  // When using a vendor-specific prefix, also include the standard property.
  "less.lint.vendorPrefix": "warning",

  // No unit for zero needed.
  "less.lint.zeroUnits": "ignore",

  // Enables or disables all validations.
  "less.validate": true,

// SCSS (Sass)

  // Invalid number of parameters.
  "scss.lint.argumentsInColorFunction": "error",

  // Do not use `width` or `height` when using `padding` or `border`.
  "scss.lint.boxModel": "ignore",

  // When using a vendor-specific prefix make sure to also include all other vendor-specific properties.
  "scss.lint.compatibleVendorPrefixes": "ignore",

  // Do not use duplicate style definitions.
  "scss.lint.duplicateProperties": "ignore",

  // Do not use empty rulesets.
  "scss.lint.emptyRules": "warning",

  // Avoid using `float`. Floats lead to fragile CSS that is easy to break if one aspect of the layout changes.
  "scss.lint.float": "ignore",

  // `@font-face` rule must define `src` and `font-family` properties.
  "scss.lint.fontFaceProperties": "warning",

  // Hex colors must consist of three or six hex numbers.
  "scss.lint.hexColorLength": "error",

  // Selectors should not contain IDs because these rules are too tightly coupled with the HTML.
  "scss.lint.idSelector": "ignore",

  // IE hacks are only necessary when supporting IE7 and older.
  "scss.lint.ieHack": "ignore",

  // Avoid using `!important`. It is an indication that the specificity of the entire CSS has gotten out of control and needs to be refactored.
  "scss.lint.important": "ignore",

  // Import statements do not load in parallel.
  "scss.lint.importStatement": "ignore",

  // Property is ignored due to the display. E.g. with `display: inline`, the `width`, `height`, `margin-top`, `margin-bottom`, and `float` properties have no effect.
  "scss.lint.propertyIgnoredDueToDisplay": "warning",

  // The universal selector (`*`) is known to be slow.
  "scss.lint.universalSelector": "ignore",
```

```
  // Unknown property.
  "scss.lint.unknownProperties": "warning",

  // Unknown vendor specific property.
  "scss.lint.unknownVendorSpecificProperties": "ignore",

  // A list of properties that are not validated against the `unknownProperties` rule.
  "scss.lint.validProperties": [],

  // When using a vendor-specific prefix, also include the standard property.
  "scss.lint.vendorPrefix": "warning",

  // No unit for zero needed.
  "scss.lint.zeroUnits": "ignore",

  // Enables or disables all validations.
  "scss.validate": true,

// Extensions

  // When enabled, automatically checks extensions for updates. If an extension has an update, it is marked as outdated in the Extensions view. The updates are fetched from a Microsoft online servic
e.
  "extensions.autoCheckUpdates": true,

  // When enabled, automatically installs updates for extensions. The updates are fetched from a Microsoft online service.
  "extensions.autoUpdate": true,

  // When enabled, editors with extension details will be automatically closed upon navigating away from the Extensions View.
  "extensions.closeExtensionDetailsOnViewChange": false,

  // When enabled, the notifications for extension recommendations will not be shown.
  "extensions.ignoreRecommendations": false,

  // When enabled, recommendations will not be fetched or shown unless specifically requested by the user. Some recommendations are fetched from a Microsoft online service.
  "extensions.showRecommendationsOnlyOnDemand": false,

// External Terminal

  // Customizes what kind of terminal to launch.
  //  - integrated: Use VS Code's integrated terminal.
  //  - external: Use the configured external terminal.
  "terminal.explorerKind": "integrated",

  // Customizes which terminal to run on Linux.
  "terminal.external.linuxExec": "xterm",

  // Customizes which terminal application to run on macOS.
  "terminal.external.osxExec": "Terminal.app",

  // Customizes which terminal to run on Windows.
  "terminal.external.windowsExec": "C:\\WINDOWS\\System32\\cmd.exe",

// Integrated Terminal

  // A set of command IDs whose keybindings will not be sent to the shell and instead always be handled by Code. This allows the use of keybindings that would normally be consumed by the shell to ac
t the same as when the terminal is not focused, for example ctrl+p to launch Quick Open.
  "terminal.integrated.commandsToSkipShell": [],

  // Controls whether to confirm on exit if there are active terminal sessions.
  "terminal.integrated.confirmOnExit": false,
```

```
  // Controls whether text selected in the terminal will be copied to the clipboard.
  "terminal.integrated.copyOnSelection": false,

  // Controls whether the terminal cursor blinks.
  "terminal.integrated.cursorBlinking": false,

  // Controls the style of terminal cursor.
  "terminal.integrated.cursorStyle": "block",

  // An explicit start path where the terminal will be launched, this is used as the current working directory (cwd) for the shell process. This may be particularly useful in workspace settings if t
he root directory is not a convenient cwd.
  "terminal.integrated.cwd": "",

  // Controls whether bold text in the terminal will always use the "bright" ANSI color variant.
  "terminal.integrated.drawBoldTextInBrightColors": true,

  // Controls whether the terminal bell is enabled.
  "terminal.integrated.enableBell": false,

  // Object with environment variables that will be added to the VS Code process to be used by the terminal on Linux. Set to `null` to delete the environment variable.
  "terminal.integrated.env.linux": {},

  // Object with environment variables that will be added to the VS Code process to be used by the terminal on macOS. Set to `null` to delete the environment variable.
  "terminal.integrated.env.osx": {},

  // Object with environment variables that will be added to the VS Code process to be used by the terminal on Windows. Set to `null` to delete the environment variable.
  "terminal.integrated.env.windows": {},

  // Controls the font family of the terminal, this defaults to `editor.fontFamily`'s value.
  "terminal.integrated.fontFamily": "",

  // Controls the font size in pixels of the terminal.
  "terminal.integrated.fontSize": 14,

  // The font weight to use within the terminal for non-bold text.
  "terminal.integrated.fontWeight": "normal",

  // The font weight to use within the terminal for bold text.
  "terminal.integrated.fontWeightBold": "bold",

  // Whether new shells should inherit their environment from VS Code. This is not supported on Windows.
  "terminal.integrated.inheritEnv": true,

  // Controls the letter spacing of the terminal, this is an integer value which represents the amount of additional pixels to add between characters.
  "terminal.integrated.letterSpacing": 0,

  // Controls the line height of the terminal, this number is multiplied by the terminal font size to get the actual line-height in pixels.
  "terminal.integrated.lineHeight": 1,

  // Controls whether to force selection when using Option+click on macOS. This will force a regular (line) selection and disallow the use of column selection mode. This enables copying and pasting
 using the regular terminal selection, for example, when mouse mode is enabled in tmux.
  "terminal.integrated.macOptionClickForcesSelection": false,

  // Controls whether to treat the option key as the meta key in the terminal on macOS.
  "terminal.integrated.macOptionIsMeta": false,

  // Controls how the terminal is rendered.
  //  - auto: Let VS Code guess which renderer to use.
  //  - canvas: Use the standard GPU/canvas-based renderer
  //  - dom: Use the fallback DOM-based renderer.
  "terminal.integrated.rendererType": "auto",
```

```
  // Controls how terminal reacts to right click.
  //  - default: Show the context menu.
  //  - copyPaste: Copy when there is a selection, otherwise paste.
  //  - selectWord: Select the word under the cursor and show the context menu.
  "terminal.integrated.rightClickBehavior": "copyPaste",

  // Controls the maximum amount of lines the terminal keeps in its buffer.
  "terminal.integrated.scrollback": 1000,

  // Controls whether locale variables are set at startup of the terminal.
  "terminal.integrated.setLocaleVariables": true,

  // The path of the shell that the terminal uses on Linux (default: /bin/bash). [Read more about configuring the shell](https://code.visualstudio.com/docs/editor/integrated-terminal#_configuratio
n).
  "terminal.integrated.shell.linux": null,

  // The path of the shell that the terminal uses on macOS (default: /bin/bash). [Read more about configuring the shell](https://code.visualstudio.com/docs/editor/integrated-terminal#_configuratio
n).
  "terminal.integrated.shell.osx": null,

  // The path of the shell that the terminal uses on Windows (default: C:\WINDOWS\System32\WindowsPowerShell\v1.0\powershell.exe). [Read more about configuring the shell](https://code.visualstudio.c
om/docs/editor/integrated-terminal#_configuration).
  "terminal.integrated.shell.windows": null,

  // The command line arguments to use when on the Linux terminal. [Read more about configuring the shell](https://code.visualstudio.com/docs/editor/integrated-terminal#_configuration).
  "terminal.integrated.shellArgs.linux": [],

  // The command line arguments to use when on the macOS terminal. [Read more about configuring the shell](https://code.visualstudio.com/docs/editor/integrated-terminal#_configuration).
  "terminal.integrated.shellArgs.osx": [
    "-l"
  ],

  // The command line arguments to use when on the Windows terminal. [Read more about configuring the shell](https://code.visualstudio.com/docs/editor/integrated-terminal#_configuration).
  "terminal.integrated.shellArgs.windows": [],

  // Controls whether to show the alert "The terminal process terminated with exit code" when exit code is non-zero.
  "terminal.integrated.showExitAlert": true,

  // Controls the working directory a split terminal starts with.
  //  - workspaceRoot: A new split terminal will use the workspace root as the working directory. In a multi-root workspace a choice for which root folder to use is offered.
  //  - initial: A new split terminal will use the working directory that the parent terminal started with.
  //  - inherited: On macOS and Linux, a new split terminal will use the working directory of the parent terminal. On Windows, this behaves the same as initial.
  "terminal.integrated.splitCwd": "inherited",

  // Whether to use ConPTY for Windows terminal process communication (requires Windows 10 build number 18309+). Winpty will be used if this is false.
  "terminal.integrated.windowsEnableConpty": true,

// Problems

  // Controls whether Problems view should automatically reveal files when opening them.
  "problems.autoReveal": true,

  // Show Errors & Warnings on files and folder.
  "problems.decorations.enabled": true,

// Breadcrumb Navigation

  // Enable/disable navigation breadcrumbs.
  "breadcrumbs.enabled": true,
```

```
  // Controls whether and how file paths are shown in the breadcrumbs view.
  //  - on: Show the file path in the breadcrumbs view.
  //  - off: Do not show the file path in the breadcrumbs view.
  //  - last: Only show the last element of the file path in the breadcrumbs view.
  "breadcrumbs.filePath": "on",

  // Controls whether and how symbols are shown in the breadcrumbs view.
  //  - on: Show all symbols in the breadcrumbs view.
  //  - off: Do not show symbols in the breadcrumbs view.
  //  - last: Only show the current symbol in the breadcrumbs view.
  "breadcrumbs.symbolPath": "on",

  // Controls how symbols are sorted in the breadcrumbs outline view.
  //  - position: Show symbol outline in file position order.
  //  - name: Show symbol outline in alphabetical order.
  //  - type: Show symbol outline in symbol type order.
  "breadcrumbs.symbolSortOrder": "position",

// Telemetry

  // Enable crash reports to be sent to a Microsoft online service.
  // This option requires restart to take effect.
  "telemetry.enableCrashReporter": true,

  // Enable usage data and errors to be sent to a Microsoft online service.
  "telemetry.enableTelemetry": true,

// Outline

  // Render Outline Elements with Icons.
  "outline.icons": true,

  // Use badges for Errors & Warnings.
  "outline.problems.badges": true,

  // Use colors for Errors & Warnings.
  "outline.problems.colors": true,

  // Show Errors & Warnings on Outline Elements.
  "outline.problems.enabled": true,

// Git

  // Controls whether force push (with or without lease) is enabled.
  "git.allowForcePush": false,

  // Always show the Staged Changes resource group.
  "git.alwaysShowStagedChangesResourceGroup": false,

  // Controls the signoff flag for all commits.
  "git.alwaysSignOff": false,

  // When enabled, commits will automatically be fetched from the default remote of the current Git repository.
  "git.autofetch": false,

  // Duration in seconds between each automatic git fetch, when `git.autofetch` is enabled.
  "git.autofetchPeriod": 180,

  // Whether auto refreshing is enabled.
  "git.autorefresh": true,
```

```
// Configures when repositories should be automatically detected.
//  - true: Scan for both subfolders of the current opened folder and parent folders of open files.
//  - false: Disable automatic repository scanning.
//  - subFolders: Scan for subfolders of the currently opened folder.
//  - openEditors: Scan for parent folders of open files.
"git.autoRepositoryDetection": true,

// Stash any changes before pulling and restore them after successful pull.
"git.autoStash": false,

// A regular expression to validate new branch names.
"git.branchValidationRegex": "",

// The character to replace whitespace in new branch names.
"git.branchWhitespaceChar": "-",

// Controls what type of branches are listed when running `Checkout to...`.
//  - all: Show all references.
//  - local: Show only local branches.
//  - tags: Show only tags.
//  - remote: Show only remote branches.
"git.checkoutType": "all",

// Always confirm the creation of empty commits.
"git.confirmEmptyCommits": true,

// Controls whether to ask for confirmation before force-pushing.
"git.confirmForcePush": true,

// Confirm before synchronizing git repositories.
"git.confirmSync": true,

// Controls the git badge counter.
//  - all: Count all changes.
//  - tracked: Count only tracked changes.
//  - off: Turn off counter.
"git.countBadge": "all",

// Controls whether Git contributes colors and badges to the explorer and the open editors view.
"git.decorations.enabled": true,

// The default location to clone a git repository.
"git.defaultCloneDirectory": null,

// Controls whether to automatically detect git submodules.
"git.detectSubmodules": true,

// Controls the limit of git submodules detected.
"git.detectSubmodulesLimit": 10,

// Enables commit signing with GPG.
"git.enableCommitSigning": false,

// Whether git is enabled.
"git.enabled": true,

// Commit all changes when there are no staged changes.
"git.enableSmartCommit": false,

// Fetch all branches when pulling or just the current one.
"git.fetchOnPull": false,
```

```
  // List of git repositories to ignore.
  "git.ignoredRepositories": [],

  // Ignores the legacy Git warning.
  "git.ignoreLegacyWarning": false,

  // Ignores the warning when there are too many changes in a repository.
  "git.ignoreLimitWarning": false,

  // Ignores the warning when Git is missing.
  "git.ignoreMissingGitWarning": false,

  // Controls when to show commit message input validation.
  "git.inputValidation": "warn",

  // Controls the commit message length threshold for showing a warning.
  "git.inputValidationLength": 72,

  // Controls the commit message subject length threshold for showing a warning. Unset it to inherit the value of `config.inputValidationLength`.
  "git.inputValidationSubjectLength": 50,

  // Controls whether the diff editor should be opened when clicking a change. Otherwise the regular editor will be opened.
  "git.openDiffOnClick": true,

  // Path and filename of the git executable, e.g. `C:\Program Files\Git\bin\git.exe` (Windows).
  "git.path": null,

  // Runs a git command after a successful commit.
  //  - none: Don't run any command after a commit.
  //  - push: Run 'Git Push' after a successful commit.
  //  - sync: Run 'Git Sync' after a successful commit.
  "git.postCommitCommand": "none",

  // Controls whether Git should check for unsaved files before committing.
  "git.promptToSaveFilesBeforeCommit": true,

  // Fetch all tags when pulling.
  "git.pullTags": true,

  // Force git to use rebase when running the sync command.
  "git.rebaseWhenSync": false,

  // List of paths to search for git repositories in.
  "git.scanRepositories": [],

  // Controls whether to show an inline Open File action in the Git changes view.
  "git.showInlineOpenFileAction": true,

  // Controls whether git actions should show progress.
  "git.showProgress": true,

  // Controls whether to show a notification when a push is successful.
  "git.showPushSuccessNotification": false,

  // Controls whether force pushing uses the safer force-with-lease variant.
  "git.useForcePushWithLease": true,

// Default Configuration Overrides

  // Configure editor settings to be overridden for [git-commit] language.
```

```
  "[git-commit]":  {
    "editor.rulers": [
        72
    ]
  },

  // Configure editor settings to be overridden for [go] language.
  "[go]":  {
    "editor.insertSpaces": false
  },

  // Configure editor settings to be overridden for [json] language.
  "[json]":  {
    "editor.quickSuggestions": {
        "strings": true
    }
  },

  // Configure editor settings to be overridden for [makefile] language.
  "[makefile]":  {
    "editor.insertSpaces": false
  },

  // Configure editor settings to be overridden for [markdown] language.
  "[markdown]":  {
    "editor.wordWrap": "on",
    "editor.quickSuggestions": false
  },

  // Configure editor settings to be overridden for [yaml] language.
  "[yaml]":  {
    "editor.insertSpaces": true,
    "editor.tabSize": 2,
    "editor.autoIndent": false
  },

// Remote

  // Override the kind of an extension. `ui` extensions are installed and run on the local machine while `workspace` extensions are run on the remote. By overriding an extension's default kind using
this setting, you specify if that extension should be installed and enabled locally or remotely.
  "remote.extensionKind": {
    "pub.name": "ui"
  },

// Npm

  // Controls whether npm scripts should be automatically detected.
  "npm.autoDetect": "on",

  // Enable an explorer view for npm scripts.
  "npm.enableScriptExplorer": false,

  // Configure glob patterns for folders that should be excluded from automatic script detection.
  "npm.exclude": "",

  // Fetch data from https://registry.npmjs/org and https://registry.bower.io to provide auto-completion and information on hover features on npm dependencies.
  "npm.fetchOnlinePackageInfo": true,

  // The package manager used to run scripts.
  "npm.packageManager": "npm",
```

```
  // Run npm commands with the `--silent` option.
  "npm.runSilent": false,

  // The default click action used in the scripts explorer: `open` or `run`, the default is `open`.
  "npm.scriptExplorerAction": "open",

// Reference Search View

  // Controls whether 'Peek References' or 'Find References' is invoked when selecting code lens references
  //  - peek: Show references in peek editor.
  //  - view: Show references in separate view.
  "references.preferredLocation": "peek",

// Merge Conflict

  // Whether to automatically navigate to the next merge conflict after resolving a merge conflict.
  "merge-conflict.autoNavigateNextConflict.enabled": false,

  // Create a Code Lens for merge conflict blocks within editor.
  "merge-conflict.codeLens.enabled": true,

  // Create decorators for merge conflict blocks within editor.
  "merge-conflict.decorators.enabled": true,

  // Controls where the diff view should be opened when comparing changes in merge conflicts.
  //  - Current: Open the diff view in the current editor group.
  //  - Beside: Open the diff view next to the current editor group.
  //  - Below: Open the diff view below the current editor group.
  "merge-conflict.diffViewPosition": "Current",

// Node debug

  // Automatically attach node debugger when node.js was launched in debug mode from integrated terminal.
  //  - disabled: Auto attach is disabled and not shown in status bar.
  //  - on: Auto attach is active.
  //  - off: Auto attach is inactive.
  "debug.node.autoAttach": "disabled",

// Emmet

  // An array of languages where Emmet abbreviations should not be expanded.
  "emmet.excludeLanguages": [
    "markdown"
  ],

  // Path to a folder containing Emmet profiles and snippets.
  "emmet.extensionsPath": null,

  // Enable Emmet abbreviations in languages that are not supported by default. Add a mapping here between the language and emmet supported language.
  //  E.g.: `{"vue-html": "html", "javascript": "javascriptreact"}`
  "emmet.includeLanguages": {},

  // When set to `false`, the whole file is parsed to determine if current position is valid for expanding Emmet abbreviations. When set to `true`, only the content around the current position in cs
s/scss/less files is parsed.
  "emmet.optimizeStylesheetParsing": true,

  // Preferences used to modify behavior of some actions and resolvers of Emmet.
  "emmet.preferences": {},

  // Shows possible Emmet abbreviations as suggestions. Not applicable in stylesheets or when emmet.showExpandedAbbreviation is set to `"never"`.
  "emmet.showAbbreviationSuggestions": true,
```

```
  // Shows expanded Emmet abbreviations as suggestions.
  // The option `"inMarkupAndStylesheetFilesOnly"` applies to html, haml, jade, slim, xml, xsl, css, scss, sass, less and stylus.
  // The option `"always"` applies to all parts of the file regardless of markup/css.
  "emmet.showExpandedAbbreviation": "always",

  // If `true`, then Emmet suggestions will show up as snippets allowing you to order them as per `editor.snippetSuggestions` setting.
  "emmet.showSuggestionsAsSnippets": false,

  // Define profile for specified syntax or use your own profile with specific rules.
  "emmet.syntaxProfiles": {},

  // When enabled, Emmet abbreviations are expanded when pressing TAB.
  "emmet.triggerExpansionOnTab": false,

  // Variables to be used in Emmet snippets
  "emmet.variables": {},

// Jake

  // Controls whether auto detection of Jake tasks is on or off. Default is on.
  "jake.autoDetect": "on",

// Grunt

  // Controls whether auto detection of Grunt tasks is on or off. Default is on.
  "grunt.autoDetect": "on",

// Gulp

  // Controls whether auto detection of Gulp tasks is on or off. Default is on.
  "gulp.autoDetect": "on",
}
```

Common questions

VS Code says "Unable to write settings."

If you try to change a setting (for example turning on Auto Save or selecting a new Color Theme) and you see "Unable to write settings. Please open User Settings to correct errors/warnings in the file and try again.", it means your `settings.json` file is ill-formed or has errors. The errors can be as simple as a missing comma or setting value. Open the Settings editor **File** > **Preferences** > **Settings** (**Code** > **Preferences** > **Settings** on macOS) (`Ctrl+,`) and you should see the error highlighted with red squiggles.

How can I reset my user settings?

The easiest way to reset VS Code back to the default settings is to clear your user `settings.json` file contents in the Settings editor. Delete everything between the two curly braces, save the file, and VS Code will go back to using the default values.

When does it make sense to use workspace settings?

If you're using a workspace that needs custom settings but you don't want to apply them to your other VS Code projects. A good example is language-specific linting rules.

Was this documentation helpful?

Yes **No**

7/3/2019

IN THIS ARTICLE

Tweet this link (https://twitter.com/intent/tweet?original_referer=https://code.visualstudio.com/docs/getstarted/settings&ref_src=twsrc%5Etfw&text=Visual%20Studio%20Code%20User%20and%20Workspace%20Settings&tw_p=tweetbutton&url=https://code.visualstudio.com

Subscribe(/feed.xml)

Ask questions(https://stackoverflow.com/questions/tagged/vscode)

Follow @code(https://go.microsoft.com/fwlink/?LinkID=533687)

Request features(https://go.microsoft.com/fwlink/?LinkID=533482)

Report issues(https://www.github.com/Microsoft/vscode/issues)

Watch videos(https://www.youtube.com/channel/UCs5Y5_7XK8HLDX0SLNwkd3w)

Hello from Seattle. Follow @code (https://go.microsoft.com/fwlink/?LinkID=533687) Star 79,511