lsp-mode

Table of Contents

Introduction

lsp-mode is a client/library package that allows Emacs to communicate with language servers [1] using Microsoft's Language Server Protocol. `lsp-mode` aims to plug-in to the vast collection of extensive Emacs tools and packages to provide a familiar, yet powerful interface for programmers. `lsp-mode` also provides an API for Emacs package developers to provide LSP support to users (see: company-lsp, lsp-treemacs, helm-lsp). It supports a wide array of standard LSP methods, and can be extended with optimally work with non-standard, language-specific functionality as well (see: lsp-java, emacs-ccls).

Out of the box, it supports 25+ programming languages, with various other external packages that provide more specific language features. Adding support for a new language server can be accomplished in a few lines of code, with more fine tuning and configuration options available.

Basic Features

- Focuses on providing an asynchronous/non-blocking interface for as much functionality as possible.
- Real-time diagnostics/linting support (either via `flymake` for Emacs versions 26 and above, or flycheck, using lsp-ui)
- Code completion - support for the (relatively basic) `completion-at-point` interface in Emacs, documented and asynchronous completion support provided by company-lsp.
- Documentation/highlight on hover: Basic, out of the box support provided using `eldoc`. Richer and more extensive functionality available in lsp-ui.
- Code Actions: Text based commands that provide refactoring/linting/code fixing. Supported natively with `lsp-execute-code-action`, and by lsp-ui. Otherwise known as "quick fixes" in Visual Studio Code.
- Code outline/explorer: In-built support for Imenu/ helm-imenu.
- Cross referencing, goto-definition, goto-implementation with Xref.
- Code/Symbol Highlighting
- Formatting, On-type code formatting
- Debugging support with the Debug Adapter Protocol with dap-mode.
- Helm support using helm-lsp

Installation

`lsp-mode` is available as a package on both MELPA and MELPA Stable. If you haven't added either to Emacs' package sources, you can do that with

```
(require 'packages)
(add-to-list 'package-archives
         '("melpa" . "https://melpa.org/packages/"))
```

For MELPA, or

```
(require 'package)
(add-to-list 'package-archives
         '("melpa-stable" . "https://stable.melpa.org/packages/"))
```

for MELPA Stable.

After updating the package list with `M-x package-refresh-contents RET`, you can install `lsp-mode` using `M-x package-install RET lsp-mode RET`.

Installing language servers

Because language servers exist outside of Emacs, they need to be installed seperately. A list of language servers is available at https://langserver.org/ and https://microsoft.github.io/language-server-protocol/implementors/servers/. Additionally, this documentation also provides a list of supported languages.

Configuration

Once `lsp-mode` is installed, a minimal configuration would consist of installing the language server(s) for your preferred languages, and enabling `lsp-mode` for the respective major-mode with:

```
(require 'lsp-mode)
(add-hook '<major-mode-hook> #'lsp)
;; Major mode hooks tend to be named as xxx-mode-hook (python-mode-hook,
;; c++-mode-hook, etc)
```

If `lsp-auto-configure` and `lsp-auto-require-clients` are non-nil, then `lsp-mode` loads all the clients that have been bundled with the package. Otherwise, or if you're using an external package for LSP support, you might need to `require` it as well. For instance, if you want to use `lsp-java`:

```
(require 'lsp-mode)
(require 'lsp-java)
(add-hook 'java-mode-hook 'lsp)
```

Alternatively, `lsp-mode` can be hooked to `prog-mode-hook`, which will cause it to try initialising LSP support for all supported programming languages.

use-package

A minimal `use-package` configuration for `lsp-mode` might look like this:

```
(use-package lsp-mode
  :commands lsp
  :init
  (setq ...))
```

Spacemacs

`lsp-mode` is included in Spacemacs' `develop` branch. Documentation for the `lsp` layer is available here.

Doom Emacs

`lsp-mode` is included in the `lsp` module. Documentation for it is available here.

`lsp-mode` settings

- `lsp-print-io`

> Default value: `nil`
>
> If non-nil, print all messages to and from the language server to **lsp-log**.

- `lsp-print-performance`

> Default value: `nil`
>
> If non-nil, print performance info in the logs.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-use-native-json`

> Default value: `t`
>
> If non-nil, use native json parsing if available.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-json-use-lists`

> Default value: `nil`
>
> If non-nil, use lists instead of vectors when doing json deserialization.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-log-max`

> Default value: `1000`
>
> Maximum number of lines to keep in the log buffer. If nil, disable message logging. If t, log messages but don't truncate the buffer when it becomes large.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-io-messages-max`

> Default value: `t`
>
> Maximum number of messages that can be locked in a 'lsp-io' buffer.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-report-if-no-buffer`

> Default value: `t`
>
> If non nil the errors will be reported even when the file is not open.

- `lsp-keep-workspace-alive`

> Default value: `t`
>
> If non nil keep workspace alive when the last workspace buffer is closed.

- `lsp-enable-snippet`

> Default value: `nil`
>
> Enable/disable snippet completion support.

- `lsp-enable-folding`

> Default value: `t`
>
> Enable/disable code folding support.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-folding-range-limit`

> Default value: `nil`
>
> The maximum number of folding ranges to receive from the language server.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-folding-line-folding-only`

> Default value: `nil`
>
> If non-nil, only fold complete lines.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-auto-require-clients`

> Default value: `t`
>
> Auto require lsp-clients.

- `lsp-auto-guess-root`

> Default value: `nil`
>
> Automatically guess the project root using projectile/project.

- `lsp-restart`

> Default value: `interactive`
>
> Defines how server exited event must be handled.

- `lsp-session-file`

> Default value: `"~/.emacs.d/.lsp-session-v1"`
>
> Automatically guess the project root using projectile/project.

- `lsp-auto-configure`

> Default value: `t`
>
> Auto configure 'lsp-mode'. When set to t 'lsp-mode' will auto-configure 'lsp-ui' and 'company-lsp'.

- `lsp-disabled-clients`

> Default value: `nil` A list of disabled/blacklisted clients. Each entry in the list can be either: a symbol, the server-id for the LSP client, or a cons pair (MAJOR-MODE . CLIENTS), where MAJOR-MODE is the major-mode, and CLIENTS is either a client or a list of clients.
>
> This option can also be used as a file or directory local variable to disable a language server for individual files or directories/projects respectively.

- `lsp-before-initialize-hook`

> Default value: `nil`
>
> List of functions to be called before a Language Server has been initialized for a new workspace.

- `lsp-after-initialize-hook`

> Default value: `nil`
>
> List of functions to be called after a Language Server has been initialized for a new workspace.

- `lsp-before-open-hook`

> Default value: `nil`
>
> List of functions to be called before a new file with LSP support is opened.

- `lsp-after-open-hook`

> Default value: `nil`
>
> List of functions to be called after a new file with LSP support is opened.

- `lsp-enable-file-watchers`

> Default value: `t`
>
> If non-nil lsp-mode will watch the files in the workspace if the server has requested that.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-file-watch-ignored`

> Default value: `("[/\\]\\.git$" "[/\\]\\.hg$" "[/\\]\\.bzr$" "[/\\]_darcs$" "[/\\]\\.svn$" "[/\\]_FOSSIL_$" "[/\\]\\.idea$" "[/\\]\\.ensime_cache$" "[/\\]\\.eunit$" "[/\\]node_modules$" "[/\\]\\.fslckout$" "[/\\]\\.tox$" "[/\\]\\.stack-work$" "[/\\]\\.bloop$" "[/\\]\\.metals$" "[/\\]target$" "[/\\]\\.deps$" "[/\\]build-aux$" "[/\\]autom4te.cache$" "[/\\]\\.reference$")`
>
> List of regexps matching directory paths which won't be monitored when creating file watches.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-after-uninitialized-hook`

> Default value: `(doom-modeline-update-lsp)`
>
> List of functions to be called after a Language Server has been uninitialized.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-debounce-full-sync-notifications`

> Default value: `t`
>
> If non-nil debounce full sync events. This flag affects only server which do not support incremental update.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-debounce-full-sync-notifications-interval`

> Default value: `1.0`
>
> Time to wait before sending full sync synchronization after buffer modication.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-document-sync-method`

> Default value: `nil`
>
> How to sync the document with the language server.

- `lsp-auto-execute-action`

> Default value: `t`
>
> Auto-execute single action.

- `lsp-enable-links`

> Default value: `t`
>
> If non-nil, all references to links in a file will be made clickable, if supported by the language server.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-links-check-internal`

> Default value: `0.1`
>
> The interval for updating document links.

- `lsp-eldoc-enable-hover`

> Default value: `t`
>
> If non-nil, eldoc will display hover info when it is present.

- `lsp-eldoc-enable-signature-help`

> Default value: `t`
>
> If non-nil, eldoc will display signature help when it is present.

- `lsp-eldoc-prefer-signature-help`

> Default value: `t`
>
> If non-nil, eldoc will display signature help when both hover and signature help are present.

- `lsp-eldoc-render-all`

> Default value: `nil`
>
> Define whether all of the returned by document/onHover will be displayed. If 'lsp-eldoc-render-all' is set to nil 'eldoc' will show only the symbol information.

- `lsp-signature-render-all`

> Default value: `t`
>
> Define whether all of the returned by textDocument/signatureHelp will be displayed. If 'lsp-signature-render-all' is set to nil 'eldoc' will show only the active signature.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-enable-completion-at-point`

> Default value: `t`
>
> Enable 'completion-at-point' integration.

- `lsp-enable-symbol-highlighting`

> Default value: `t`
>
> Highlight references of the symbol at point.

- `lsp-enable-xref`

> Default value: `t`
>
> Enable xref integration.

- `lsp-enable-indentation`

> Default value: `t`
>
> Indent regions using the file formatting functionality provided by the language server.

- `lsp-enable-on-type-formatting`

> Default value: `t`
>
> Enable ‘textDocument/onTypeFormatting’ integration.

- `lsp-before-save-edits`

> Default value: `t`
>
> If non-nil, ‘lsp-mode’ will apply edits suggested by the language server before saving a document.

- `lsp-after-diagnostics-hook`

> Default value: `nil`
>
> Hooks to run after diagnostics are received.

- `lsp-workspace-folders-changed-hook`

> Default value: `nil`
>
> Hooks to run after the folders has changed. The hook will receive two parameters list of added and removed folders.

- `lsp-on-hover-hook`

> Default value: `nil`
>
> The hooks that run after on hover and signature information has been loaded. The hook is called with two params: the signature information and hover data.

- `lsp-eldoc-hook`

> Default value: `(lsp-hover)`
>
> Hooks to run for eldoc.

- `lsp-response-timeout`

> Default value: `10`
>
> Number of seconds to wait for a response from the language server before timing out.

- `lsp-prefer-flymake`

> Default value: `t`
>
> Auto-configure to prefer 'flymake' over 'lsp-ui' if both are present. If set to ':none' neither of two will be enabled.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-lens-check-interval`

> Default value: `0.1`
>
> The interval for checking for changes in the buffer state.

- `lsp-lens-debounce-interval`

> Default value: `0.7`
>
> Debounce interval for loading lenses.

- `lsp-symbol-highlighting-skip-current`

> Default value: `nil`
>
> If non-nil skip current symbol when setting symbol highlights.

- `lsp-document-highlight-delay`

> Default value: `0.2`
>
> Seconds of idle time to wait before showing symbol highlight.

Using `lsp-mode`

Workspace/Session Management

For a language server to work with a file, it needs to know the **project root**, a (usually higher-level) directory, containing the project to which the file belongs to.

On opening a file in a new project, `lsp-mode` will try to guess the project root, and present you with several options:

- Import the root directory `lsp-mode` guessed.
- Select a project root directory interactively (ask the user), and import it.
- Blacklist the guessed root directory, and ignore it in the future.
- Select a project root directory interactively, and blacklist it.

- Do nothing, but try importing the project again when any other file in the project is opened.

The root directory is then known to `lsp-mode` as a **workspace**, and every file in the project is associated with it.

The list of imported/whitelisted and blacklisted directories is stored in a **Session File**, located at `~/.emacs.d/.lsp-session-v1` by default. Its location can be changed by modifying `lsp-session-file`.

`lsp-mode` will try using Projectile (if installed), or the in-built project library `project.el` to guess the root.

Logging

By default, `lsp-mode` will log everything to the the **`lsp-log`** buffer. This can be controlled using [lsp-log-max], which is set to message-log-max by default. Additionally, `lsp-mode` can log all LSP requests, responses and notifications sent and received from a language server.

Workspace Management Commands

`lsp-workspace-folders-remove PROJECT-ROOT`

If called interactively (through `M-x lsp-workspace-folders-remove RET`), it will ask the user to select a directory from a list of project roots that `lsp-mode` has yet imported. The selected directory will be removed from the session file, and currently opened workspaces under it will be shut down (they will no longer be associated with a language server, and wouldn't support `lsp-mode` features). This only removes the directory from the session file, and opening any file from the removed project will cause `lsp-mode` to ask the user about importing/blacklisting the directory again.

Function: `lsp-workspace-folders-add PROJECT-ROOT`

If called interactively, it will ask the user to select a directory, and import it as a project root. All files located under the directory will be treated as belonging to the project rooted at this directory.

Function: `lsp-workspace-folders-open PROJECT-ROOT`

If called interactively, it asks the user to choose from a list of imported project roots, and opens the selected directory in `dired`.

Function: `lsp-workspace-shutdown WORKSPACE`

If called interactively, it asks the user to select an opened project, and shuts down the workspace and the language server associated with it.

Function: `lsp-workspace-restart WORKSPACE`

If called interactively, it asks the user to select an opened project, and restarts the workspace and the language server associated with it.

Function: `lsp-workspace-blacklist-remove WORKSPACE`

If called interactively, it asks the user to select a blacklisted project, and remove the project folder from the blacklist.

Common Issues/Fixes

File watches

Some language servers request file notifications from Emacs using `workspace/didChangeWatchedFiles`, in which case `lsp-mode` will monitor every file in the path recursively for changes. If your project contains a large number of files, this might slow `lsp-mode` down. In this case, you can either disable file notifications entirely via `lsp-enable-file-watchers`, or disable it for a specific project using Directory Variables.

Frequent freezes/stuttering

`lsp-mode` in Emacs versions < 27 uses the json.el library bundled with Emacs to parse and encode JSON objects in order to communicate with a language server. When working with large objects, this can create a significant garbage collection overhead which slows down Emacs significantly. This can be either fixed by setting `gc-cons-threshold` to a large enough value (like `30000000`, causing it to start a garbage collection cycle on every 30 MB of allocated memory), or using Emacs 27 compiled with native JSON support (`--with-json`).

If you're using company-lsp, setting `company-lsp-cache-candidates` to either `'auto` or `t` can help reduce freezes when performing code completion.

Adding support for new languages

Adding support for new languages/language servers is fairly straightforward:

```
(lsp-register-client
 (make-lsp-client :new-connection <connection-plist>
                  :major-modes '(<supported-major-modes>)
                  :server-id <server-id>))
```

Creating a connection

A `connection-plist` is a property list that tells `lsp-mode` how to launch and connect to a language server process. For most language servers, you do not need to create an actual list yourself, as helper functions `lsp-stdio-connection` or `lsp-tcp-connection` can be used to create one.

Helper functions

Function: `lsp-stdio-connection COMMAND`

Returns a connection property list using `COMMAND`, for launching a language server instance that sends and receives messages over standard I/O. `COMMAND` can be:

- A string, denoting the command to launch the language server.
- A list of strings, denoting an executable with its command line arguments.
- A function, that either returns a string or a list of strings.

Function: `lsp-tcp-connection COMMAND-FN`

Returns a connection property list similar to `lsp-stdio-connection`, but `COMMAND-FN` can only be a function that takes a single argument, a port number. It should return a command for launches a language server process listening for TCP connections on the provided port.

Other arguments to `make-lsp-client`

- `:server-id`: A unique symbol that represents the client object created by `make-lsp-client`. Examples include `'clangd`, `'pyls`, `'css-ls`, etc.
- `:major-modes`: A list of major-modes supported by the language server.
- `:activation-fn`: Function that returns `t` if the client can manage given buffer. It should take a single argument (the buffer object). Can be used instead of `:major-modes`.
- `:language-id`: Function that should take a single buffer as an argument, and return the language identifier (`languageId`) for that buffer.
- `:add-on`: If non-nil, the client will be started even when there is another server handling the same mode/buffer.
- `:ignore-regexps`: A list of regular expressions. If any incoming data from the language server matches any of these regexps, it will be ignored. This is useful for filtering out unwanted messages, such as servers that send extraneous non-LSP messages.
- `:notification-handlers`: Hash table mapping notification method strings to functions handling them. See: Method handlers.
- `:request-handlers`: Hash table mapping request method strings to functions handling them. See: Method handlers.
- `:prefix-function`: A prefix function takes no arguments, and return the bounds for the entity under point as a cons pair (START . END), where both `START` and `END` are point values. The prefix function is used to compute the bounds for the entity being completed during completion, specifically, the `start` and `end` values in the list returned by Completion functions.
- `:uri-handlers`: Hash table mapping non-standard file URI schemes to a function that opens the file pointed to by the URI. The function should accept a single string (the file URI).
- `:action-handlers`: Hash table mapping code action strings to a function that executes them. While executing a code action, it is used to determine whether a particular code action should be executed by a client, or sent to the server. The function should take a single string (the code action) as an argument.

- `:multi-root`: Non-nil if the client supports multi-root workspaces.
- `:initialization-options`: A JSON-encodable object, or a function that returns one. The value will be used for additional initialization options for the `initialize` request.
- `before-file-open-fn`: Function called before a LSP-managed file (by this client) is opened. Should accept a single argument, the currently active workspace.
- `initialized-fn`: Function called right after a workspace has been initialized. Should accept a single argument, the newly initialized workspace.
- `completion-in-comments?`: Non-nil if the client supports completion within comments.

Method handlers

A method handler is a function that takes two arguments, a `lsp—workspace` object and the method parameters as a deserialized object. A defined client can use method handlers to support language specific features that aren't specified in the official LSP specification. For instance, method handlers are used by the `lsp-rust` rls client to support progress indicators sent from the server to the editor:

```
(defun lsp-clients--rust-window-progress (_workspace params)
  ;; Minimal implementation - we could show the progress as well.
  (lsp-log (gethash "title" params)))
(defvar lsp-rust-notification-handlers (make-hash-table :test 'equal))
(puthash "window/progress" #'lsp-clients--rust-window-progress lsp-rust-notification-
handlers)

(lsp-register-client
 (make-lsp-client :notification-handlers lsp-rust-notification-handlers
          <other-arguments>))
```

Languages/Language servers supported by `lsp-mode`

Language	Language Server	Support	Installation command	Debugger
Bash	bash-language-server	Built-in	`npm i -g bash-language-server`	
C++	ccls	emacs-ccls	ccls	Yes (gdb or lldb)
C++	clangd	Built-in	clangd	Yes (gdb or lldb)
C++	cquery	emacs-cquery	cquery	Yes (gdb or lldb)
Clojure	clojure-lsp	Built-in	clojure-lsp	
CSS/LessCSS/SASS/SCSS	css	Built-in	`npm install -g vscode-css-languageserver-bin`	

Language	Language Server	Support	Installation command	Debugger
Dart	dart_language_server	Built-in	`pub global activate dart_language_server`	
Elixir	elixir-ls	Built-in	elixir-ls	Yes
Elm	elmLS	Built-in	Clone the repository, run `npm install` in the root, then run tsc in the server directory. Set [lsp-elm-server-install-dir] to wherever you cloned the repo.	
Erlang	erlang_ls	Built-in	erlang_ls	
F#	FsAutoComplete	Built-in	Clone the repository, run ./build.sh LocalRelease. Set lsp-fsharp-server-path, and lsp-fsharp-server-runtime.	
Fortran	fortran-language-server	Built-in	`pip install fortran-language-server`	Yes
Go	gopls	Built-in	`go get -u golang.org/x/tools/cmd/gopls`	Yes
Go	bingo	Built-in	bingo	Yes
Groovy	groovy-language-server	Built-in	groovy-language-server	
Hack	hhvm	Built-in	hhvm	
HTML	html	Built-in	`npm install -g vscode-html-languageserver-bin`	
Haskell	IDE engine	lsp-haskell	IDE engine	
Java	Eclipse JDT LS	lsp-java	lsp-java	Yes
JavaScript/TypeScript	typescript-language-server (recommended)	Built-in	`npm i -g typescript-language-server; npm i -g typescript`	Yes (Firefox/Chrome)

Language	Language Server	Support	Installation command	Debugger
JavaScript/TypeScript	javascript-typescript-stdio	Built-in	`npm i -g javascript-typescript-langserver`	Yes (Firefox/Chrome)
JavaScript Flow	flow (add-on if working on a Flow file)	Built-in	flow	Yes (Firefox/Chrome)
Julia	lsp-julia	lsp-julia	LanguageServer.jl	
Kotlin	kotlin-language-server	Built-in	kotlin-language-server	
Ocaml	ocaml-language-server	Built-in	ocaml-language-server	
PHP(recommended)	intelephense	Built-in	`npm i intelephense -g`	Yes
PHP	php-language-server	Built-in	php-language-server	Yes
Powershell	PowerShellEditorServices	lsp-powershell	Automatic by lsp-powershell	
Python	pyls	Built-in	`pip install 'python-language-server[all]'`	Yes
Python(Microsoft)	Microsoft Python Language Server	lsp-python-ms	lsp-python-ms	Yes
Ruby	solargraph	Built-in	`gem install solargraph`	Yes
Rust	rls	Built-in	rls	Yes
Scala	Metals	Built-in	Metals	
Swift	sourcekit-LSP	lsp-sourcekit	sourcekit-LSP	Yes (via llvm debug adapter)
Vue	vue-language-server	Built-in	`npm install -g vue-language-server`	Yes (Firefox/Chrome)
XML	lsp4xml	Built-in	Download from lsp4xml releases	

Clangd support: `lsp-clangd`

`lsp-clangd` provides support for C-like languages (C, C, Objective-C/) using clangd.

`lsp-clangd` settings

- `lsp-clients-clangd-executable`

> Default value: `"clangd"`
>
> The clangd executable to use. Leave as just the executable name to use the default behavior of finding the executable with 'exec-path'.

- `lsp-clients-clangd-args`

> Default value: `nil`
>
> Extra arguments for the clangd executable.

Clojure support: `lsp-clojure`

`lsp-clangd` provides support for Clojure/ClojureScript clojure-lsp.

`lsp-clojure` settings

- `lsp-clojure-server-command`

> Default value: `("bash" "-c" "clojure-lsp")`
>
> The clojure-lisp server command.

CSS support: `lsp-css`

`lsp-css` provides support for CSS using vscode-css-languageserver.

`lsp-css` settings

- `lsp-css-experimental-custom-data`

> Default value: `nil`
>
> A list of JSON file paths that define custom CSS data that loads custom properties, at directives, pseudo classes / elements.

- `lsp-css-completion-trigger-property-value-completion`

> Default value: `t`
>
> By default, VS Code triggers property value completion after selecting a CSS property. Use this setting to disable this behavior.

- `lsp-css-validate`

> Default value: `t`
>
> Enables or disables all validations.

- `lsp-css-lint-compatible-vendor-prefixes`

> Default value: `"ignore"`
>
> When using a vendor-specific prefix make sure to also include all other vendor-specific properties.

- `lsp-css-lint-vendor-prefix`

> Default value: `"warning"`
>
> When using a vendor-specific prefix, also include the standard property.

- `lsp-css-lint-duplicate-properties`

> Default value: `"ignore"`
>
> Do not use duplicate style definitions.

- `lsp-css-lint-empty-rules`

> Default value: `"warning"`
>
> Do not use empty rulesets.

- `lsp-css-lint-import-statement`

> Default value: `"ignore"`
>
> Import statements do not load in parallel.

- `lsp-css-lint-box-model`

> Default value: `"ignore"`
>
> nil

- `lsp-css-lint-universal-selector`

> Default value: `"ignore"`
>
> nil

- `lsp-css-lint-zero-units`

> Default value: `"ignore"`
>
> No unit for zero needed.

- `lsp-css-lint-font-face-properties`

> Default value: `"warning"`
>
> nil

- `lsp-css-lint-hex-color-length`

> Default value: `"error"`
>
> Hex colors must consist of three or six hex numbers.

- `lsp-css-lint-arguments-in-color-function`

> Default value: `"error"`
>
> Invalid number of parameters.

- `lsp-css-lint-unknown-properties`

> Default value: `"warning"`
>
> Unknown property.

- `lsp-css-lint-valid-properties`

> Default value: `nil`
>
> A list of properties that are not validated against the ‘unknownProperties‘ rule.

- `lsp-css-lint-ie-hack`

> Default value: `"ignore"`
>
> IE hacks are only necessary when supporting IE7 and older.

- `lsp-css-lint-unknown-vendor-specific-properties`

> Default value: `"ignore"`
>
> Unknown vendor specific property.

- `lsp-css-lint-property-ignored-due-to-display`

> Default value: `"warning"`
>
> nil

- `lsp-css-lint-important`

> Default value: `"ignore"`
>
> nil

- `lsp-css-lint-float`

> Default value: `"ignore"`
>
> nil

- `lsp-css-lint-id-selector`

> Default value: `"ignore"`
>
> Selectors should not contain IDs because these rules are too tightly coupled with the HTML.

- `lsp-css-lint-unknown-at-rules`

> Default value: `"warning"`
>
> Unknown at-rule.

- `lsp-css-trace-server`

> Default value: `"off"`
>
> Traces the communication between VS Code and the CSS language server.

Dart support: `lsp-dart`

`lsp-dart` provides support for Dart using the Dart Language Server.

`lsp-dart` settings

- `lsp-clients-dart-server-command`

> Default value: `"~/.pub-cache/bin/dart_language_server"`
>
> The dart_language_server executable to use.

Elixir support: `lsp-css`

`lsp-elixir` provides support for Elixir using ElixirLS.

`lsp-elixir` settings

- `lsp-clients-elixir-server-executable`

> Default value: `"language_server.sh"`
>
> The elixir-language-server executable to use. Leave as just the executable name to use the default behavior of finding the executable with 'exec-path'.

Erlang support: `lsp-erlang`

`lsp-erlang` provides support for Erlang via erlang_ls.

`lsp-erlang` settings

- `lsp-erlang-server-install-dir`

> Default value: `"."`
>
> Path to the Erlang Language Server installation dir.

Flow support: `lsp-flow`

`lsp-flow` provides support for the Flow Javascript type checker.

`lsp-flow` settings

- `lsp-clients-flow-server`

> Default value: `"flow"`
>
> The Flow executable to use. Leave as just the executable name to use the default behavior of finding the executable with variable 'exec-path'.

- `lsp-clients-flow-server-args`

> Default value: `("lsp")`
>
> Extra arguments for starting the Flow language server.

Fortran support: `lsp-fortran`

`lsp-fortran` provides support for Fortran using the Fortran Language Server.

`lsp-fortran` settings

- `lsp-clients-fortls-executable`

> Default value: `"fortls"`
>
> The fortls executable to use. Leave as just the executable name to use the default behavior of finding the executable with 'exec-path'.

- `lsp-clients-fortls-args`

> Default value: `nil`
>
> Extra arguments for the fortls executable

F# support: `lsp-fsharp`

`lsp-fsharp` provides support for F# using FsAutoComplete.

`lsp-fsharp` settings

- `lsp-fsharp-server-runtime`

> Default value: `net-core`
>
> The .NET runtime to use.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-fsharp-server-path`

> Default value: `nil`
>
> The path to fsautocomplete.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-fsharp-server-args`

> Default value: `nil`
>
> Extra arguments for the F# language server.
>
>  Introduced in `lsp-mode` 6.1

Groovy support: `lsp-groovy`

`lsp-groovy` provides support for Groovy using Palantir's groovy-language-server.

`lsp-groovy` settings

- `lsp-groovy-server-install-dir`

> Default value: `"~/.emacs.d/groovy-language-server/"`
>
> Install directory for groovy-language-server. A slash is expected at the end. This directory shoud contain a file matching groovy-language-server-*.jar

Hack support: `lsp-css`

`lsp-hack` provides support for Hack using HHVM.

`lsp-hack` settings

- `lsp-clients-hack-command`

> Default value: `("hh_client" "lsp" "--from" "emacs")`
>
> Command to start hh_client.

HTML support: `lsp-html`

`lsp-html` provides support for HTML using vscode-html-languageserver.

`lsp-html` settings

- `lsp-html-experimental-custom-data`

> Default value: `nil`
>
> A list of JSON file paths that define custom tags, properties and other HTML syntax constructs. Only workspace folder setting will be read.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-html-format-enable`

> Default value: `t`
>
> Enable/disable default HTML formatter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-wrap-line-length`

> Default value: `120`
>
> Maximum amount of characters per line (0 = disable).
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-unformatted`

> Default value: `"wbr"`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-content-unformatted`

> Default value: `"pre,code,textarea"`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-indent-inner-html`

> Default value: `nil`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-preserve-new-lines`

> Default value: `t`
>
> Controls whether existing line breaks before elements should be preserved. Only works before elements, not inside tags or for text.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-max-preserve-new-lines`

> Default value: `nil`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-indent-handlebars`

> Default value: `nil`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-end-with-newline`

> Default value: `nil`
>
> End with a newline.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-extra-liners`

> Default value: `"head, body, /html"`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-format-wrap-attributes`

> Default value: `"auto"`
>
> Wrap attributes.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-suggest-html5`

> Default value: `t`
>
> Controls whether the built-in HTML language support suggests HTML5 tags, properties and values.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-validate-scripts`

> Default value: `t`
>
> Controls whether the built-in HTML language support validates embedded scripts.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-validate-styles`

> Default value: `t`
>
> Controls whether the built-in HTML language support validates embedded styles.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-auto-closing-tags`

> Default value: `t`
>
> Enable/disable autoclosing of HTML tags.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-trace-server`

> Default value: `"off"`
>
> Traces the communication between VS Code and the HTML language server.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-html-server-command`

> Default value: `("html-languageserver" "--stdio")`
>
> Command to start html-languageserver.
>
>  Introduced in `lsp-mode` 6.1

PHP support: `lsp-intelephense`

`lsp-php` provides support for PHP using Intelephense.

`lsp-intelephense` settings

- `lsp-intelephense-files-max-size`

> Default value: `1000000`
>
> Maximum file size in bytes.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-intelephense-files-associations`

> Default value: `["*.php" "*.phtml" ]`
>
> Configure glob patterns to make files available for language server features.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-intelephense-files-exclude`

Default value: `["**/.git/**" "**/.svn/**" "**/.hg/**" "**/CVS/**" "**/.DS_Store/**" "**/node_modules/**" "**/bower_components/**" "**/vendor/**/{Test,test,Tests,tests}/**" ]`

Configure glob patterns to exclude certain files and folders from all language server features.

Introduced in `lsp-mode` 6.1

- `lsp-intelephense-stubs`

Default value: `["apache" "bcmath" "bz2" "calendar" "com_dotnet" "Core" "csprng" "ctype" "curl" "date" "dba" "dom" "enchant" "exif" "fileinfo" "filter" "fpm" "ftp" "gd" "hash" "iconv" "imap" "interbase" "intl" "json" "ldap" "libxml" "mbstring" "mcrypt" "mssql" "mysql" "mysqli" "oci8" "odcb" "openssl" "password" "pcntl" "pcre" "PDO" "pdo_ibm" "pdo_mysql" "pdo_pgsql" "pdo_sqlite" "pgsql" "Phar" "posix" "pspell" "readline" "recode" "Reflection" "regex" "session" "shmop" "SimpleXML" "snmp" "soap" "sockets" "sodium" "SPL" "sqlite3" "standard" "superglobals" "sybase" "sysvmsg" "sysvsem" "sysvshm" "tidy" "tokenizer" "wddx" "xml" "xmlreader" "xmlrpc" "xmlwriter" "Zend OPcache" "zip" "zlib"]`

Configure stub files for built in symbols and common extensions. The default setting includes PHP core and all bundled extensions.

Introduced in `lsp-mode` 6.1

- `lsp-intelephense-completion-insert-use-declaration`

Default value: `t`

Use declarations will be automatically inserted for namespaced classes, traits, interfaces, functions, and constants.

Introduced in `lsp-mode` 6.1

- `lsp-intelephense-completion-fully-qualify-global-constants-and-functions`

Default value: `nil`

Global namespace constants and functions will be fully qualified (prefixed with a backslash).

Introduced in `lsp-mode` 6.1

- `lsp-intelephense-format-enable`

> Default value: `t`
>
> Enables formatting
>
>  Introduced in `lsp-mode` 6.1

- `lsp-intelephense-trace-server`

> Default value: `"off"`
>
> Traces the communication between VSCode and the intelephense language server.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-intelephense-storage-path`

> Default value: `"~/.emacs.d/lsp-cache"`
>
> Optional absolute path to storage dir.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-intelephense-server-command`

> Default value: `("intelephense" "--stdio")`
>
> Command to start Intelephense.
>
>  Introduced in `lsp-mode` 6.1

Kotlin support: `lsp-kotlin`

`lsp-kotlin` provides support for Kotlin using KotlinLanguageServer.

`lsp-kotlin` settings

Ocaml support: `lsp-ocaml`

`lsp-ocaml` provides support for Ocaml using ocaml-language-server.

`lsp-ocaml` settings

- `lsp-ocaml-lang-server-command`

> Default value: `("ocaml-language-server" "--stdio")`
>
> Command to start ocaml-language-server.

PHP support: `lsp-php`

`lsp-css` provides support for CSS using PHP Language Server.

`lsp-php` settings

- `lsp-clients-php-server-command`

> Default value: `("php" "~/.composer/vendor/felixfbecker/language-server/bin/php-language-server.php")`
>
> Install directory for php-language-server.

Python support: `lsp-pyls`

`lsp-pyls` provides support for Python using pyls

`lsp-pyls` settings

- `lsp-clients-python-library-directories`

> Default value: `("/usr/")`
>
> List of directories which will be considered to be libraries.
>
> 
> Introduced in `lsp-mode` 6.1

- `lsp-pyls-server-command`

> Default value: `("pyls")`
>
> Command to start pyls.
>
> 
> Introduced in `lsp-mode` 6.1

- `lsp-pyls-configuration-sources`

> Default value: `["pycodestyle" ]`
>
> List of configuration sources to use.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-completion-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-completion-include-params`

> Default value: `t`
>
> Auto-completes methods and classes with tabstops for each parameter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-definition-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-definition-follow-imports`

> Default value: `t`
>
> The goto call will follow imports.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-definition-follow-builtin-imports`

> Default value: `t`
>
> If follow_imports is True will decide if it follow builtin imports.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-hover-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-references-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-signature-help-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-symbols-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-jedi-symbols-all-scopes`

> Default value: `t`
>
> If True lists the names of all scopes instead of only the module namespace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-mccabe-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-mccabe-threshold`

> Default value: `15`
>
> The minimum threshold that triggers warnings about cyclomatic complexity.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-preload-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-preload-modules`

> Default value: `nil`
>
> List of modules to import on startup
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pylint-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-exclude`

> Default value: `nil`
>
> Exclude files or directories which match these patterns.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-filename`

> Default value: `nil`
>
> When parsing directories, only check filenames matching these patterns.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-select`

> Default value: `nil`
>
> Select errors and warnings
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-ignore`

> Default value: `nil`
>
> Ignore errors and warnings
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-hang-closing`

> Default value: `nil`
>
> Hang closing bracket instead of matching indentation of opening bracket's line.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pycodestyle-max-line-length`

> Default value: `nil`
>
> Set maximum allowed line length.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-enabled`

> Default value: `nil`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-convention`

> Default value: `nil`
>
> Choose the basic list of checked errors by specifying an existing convention.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-add-ignore`

> Default value: `nil`
>
> Ignore errors and warnings in addition to the specified convention.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-add-select`

Default value: `nil`

Select errors and warnings in addition to the specified convention.

Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-ignore`

Default value: `nil`

Ignore errors and warnings

Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-select`

Default value: `nil`

Select errors and warnings

Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-match`

Default value: `"(?!test_).*\\.py"`

Check only files that exactly match the given regular expression; default is to match files that don't start with 'test_' but end with '.py'.

Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pydocstyle-match-dir`

Default value: `"[^\\..*"]`

Search only dirs that exactly match the given regular expression; default is to match dirs which do not begin with a dot.

Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-pyflakes-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-rope-completion-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-plugins-yapf-enabled`

> Default value: `t`
>
> Enable or disable the plugin.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-rope-extension-modules`

> Default value: `nil`
>
> Builtin and c-extension modules that are allowed to be imported and inspected by rope.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-pyls-rope-rope-folder`

> Default value: `nil`
>
> The name of the folder in which rope stores project configurations and data. Pass ‘null‘ for not using such a folder at all.
>
>  Introduced in `lsp-mode` 6.1

Rust support: `lsp-rust`

`lsp-rust` provides support for Rust using the Rust Language Server.

`lsp-rust` settings

- `lsp-rust-library-directories`

> Default value: `("~/.cargo/registry/src" "~/.rustup/toolchains")`
>
> List of directories which will be considered to be libraries.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-sysroot`

> Default value: `nil`
>
> If non-nil, use the given path as the sysroot for all rustc invocations instead of trying to detect the sysroot automatically.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-target`

> Default value: `nil`
>
> If non-nil, use the given target triple for all rustc invocations.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-rustflags`

> Default value: `nil`
>
> Flags added to RUSTFLAGS.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-clear-env-rust-log`

> Default value: `t`
>
> Clear the RUST_LOG environment variable before running rustc or cargo.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-build-lib`

Default value: `nil`

If non-nil, checks the project as if you passed the '--lib' argument to cargo. Mutually exclusive with, and preferred over, 'lsp-rust-build-bin'. (Unstable)

Introduced in `lsp-mode` 6.1

- `lsp-rust-build-bin`

Default value: `nil`

If non-nil, checks the project as if you passed `-- bin <build_bin>` argument to cargo. Mutually exclusive with 'lsp-rust-build-lib'. (Unstable)

Introduced in `lsp-mode` 6.1

- `lsp-rust-cfg-test`

Default value: `nil`

If non-nil, checks the project as if you were running 'cargo test' rather than cargo build. I.e., compiles (but does not run) test code.

Introduced in `lsp-mode` 6.1

- `lsp-rust-unstable-features`

Default value: `nil`

Enable unstable features.

Introduced in `lsp-mode` 6.1

- `lsp-rust-wait-to-build`

Default value: `nil`

Time in milliseconds between receiving a change notification and starting build. If not specified, automatically inferred by the latest build duration.

Introduced in `lsp-mode` 6.1

- `lsp-rust-show-warnings`

> Default value: `t`
>
> Show warnings.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-use-crate-blacklist`

> Default value: `t`
>
> Don’t index crates on the crate blacklist.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-build-on-save`

> Default value: `nil`
>
> Only index the project when a file is saved and not on change.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-features`

> Default value: `[ ]`
>
> A list of Cargo features to enable.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-all-features`

> Default value: `nil`
>
> Enable all Cargo features.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-no-default-features`

> Default value: `nil`
>
> Do not enable default Cargo features.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-racer-completion`

> Default value: `t`
>
> Enables code completion using racer.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-clippy-preference`

> Default value: `"opt-in"`
>
> Controls eagerness of clippy diagnostics when available. Valid values are (case-insensitive): - "off": Disable clippy lints. - "opt-in": Clippy lints are shown when crates specify ‘#![warn(clippy)]‘. - "on": Clippy lints enabled for all crates in workspace. You need to install clippy via rustup if you haven’t already.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-jobs`

> Default value: `nil`
>
> Number of Cargo jobs to be run in parallel.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-all-targets`

> Default value: `t`
>
> Checks the project as if you were running cargo check --all-targets (I.e., check all targets and integration tests too).
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-target-dir`

> Default value: `nil`
>
> When specified, it places the generated analysis files at the specified target directory. By default it is placed target/rls directory.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-rustfmt-path`

> Default value: `nil`
>
> When specified, RLS will use the Rustfmt pointed at the path instead of the bundled one
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-build-command`

> Default value: `nil`
>
> EXPERIMENTAL (requires ‘unstable_features‘) If set, executes a given program responsible for rebuilding save-analysis to be loaded by the RLS. The program given should output a list of resulting .json files on stdout. Implies ‘rust.build_on_save‘: true.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-full-docs`

> Default value: `nil`
>
> Instructs cargo to enable full documentation extraction during save-analysis while building the crate.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-show-hover-context`

> Default value: `t`
>
> Show additional context in hover tooltips when available. This is often the type local variable declaration.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-rust-rls-server-command`

> Default value: `("rls")`
>
> Command to start RLS.
>
>  Introduced in `lsp-mode` 6.1

Metals support: lsp-metals

lsp-metals provides support for Scala using Metals.

lsp-metals settings

- lsp-metals-server-command

> Default value: "metals-emacs"
>
> The command to launch the Scala language server.
>
>  Introduced in lsp-mode 6.1

- lsp-metals-server-args

> Default value: nil
>
> Extra arguments for the Scala language server.
>
>  Introduced in lsp-mode 6.1

- lsp-metals-java-home

> Default value: ""
>
> The Java Home directory used for indexing JDK sources and locating the ‘java‘ binary.
>
>  Introduced in lsp-mode 6.1

- lsp-metals-scalafmt-config-path

> Default value: ""
>
> Optional custom path to the .scalafmt.conf file. Should be relative to the workspace root directory and use forward slashes / for file separators (even on Windows).
>
>  Introduced in lsp-mode 6.1

- lsp-metals-sbt-script

> Default value: ""
>
> Optional absolute path to an ‘sbt‘ executable to use for running ‘sbt bloopInstall‘. By default, Metals uses ‘java -jar sbt-launch.jar‘ with an embedded launcher while respecting ‘.jvmopts‘ and ‘.sbtopts‘. Update this setting if your ‘sbt‘ script requires more customizations like using environment variables.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-metals-gradle-script`

> Default value: ""
>
> Optional absolute path to a ‘gradle‘ executable to use for running ‘gradle bloopInstall‘. By default, Metals uses gradlew with 5.3.1 gradle version. Update this setting if your ‘gradle‘ script requires more customizations like using environment variables.
>
>  Introduced in `lsp-mode` 6.1

Ruby support: `lsp-solargraph`

`lsp-solargraph` provides support for Ruby using Solargraph.

`lsp-solargraph` settings

- `lsp-solargraph-completion`

> Default value: `t`
>
> Enable completion
>
>  Introduced in `lsp-mode` 6.1

- `lsp-solargraph-hover`

> Default value: `t`
>
> Enable hover
>
>  Introduced in `lsp-mode` 6.1

- `lsp-solargraph-diagnostics`

Default value: `t`

Enable diagnostics

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-autoformat`

Default value: `nil`

Enable automatic formatting while typing (WARNING: experimental)

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-formatting`

Default value: `t`

Enable document formatting

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-symbols`

Default value: `t`

Enable symbols

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-definitions`

Default value: `t`

Enable definitions (go to, etc.)

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-rename`

Default value: `t`

Enable symbol renaming

Introduced in `lsp-mode` 6.1

- `lsp-solargraph-references`

> Default value: `t`
>
> Enable finding references
>
>  Introduced in `lsp-mode` 6.1

- `lsp-solargraph-folding`

> Default value: `t`
>
> Enable folding ranges
>
>  Introduced in `lsp-mode` 6.1

- `lsp-solargraph-log-level`

> Default value: `"warn"`
>
> Level of debug info to log. ‘warn‘ is least and ‘debug‘ is most.
>
>  Introduced in `lsp-mode` 6.1

Typescript support: `lsp-typescript`

`lsp-typescript` provides support for TypeScript, using Theia/Typefox’s TypeScript Language Server

`lsp-typescript` settings

- `lsp-clients-typescript-server`

> Default value: `"typescript-language-server"`
>
> The typescript-language-server executable to use. Leave as just the executable name to use the default behavior of finding the executable with variable ‘exec-path’.

- `lsp-clients-typescript-server-args`

> Default value: `("--stdio")`
>
> Extra arguments for the typescript-language-server language server.

Typescript support: `lsp-typescript-javascript`

`lsp-javascript-typescript` provides support for Javascript and TypeScript, using Sourcegraph's JavaScript/TypeScript language server

`lsp-typescript-javascript` settings

- `lsp-clients-javascript-typescript-server`

> Default value: `"javascript-typescript-stdio"`
>
> The javascript-typescript-stdio executable to use. Leave as just the executable name to use the default behavior of finding the executable with variable 'exec-path'.

- `lsp-clients-typescript-javascript-server-args`

> Default value: `nil`
>
> Extra arguments for the typescript-language-server language server.

Vue support: `lsp-vetur`

`lsp-vetur` provides support for Vue using the Vue Language Server.

`lsp-vetur` settings

- `lsp-vetur-use-workspace-dependencies`

> Default value: `nil`
>
> Use dependencies from workspace. Currently only for TypeScript.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-vetur-completion-auto-import`

> Default value: `t`
>
> Include completion for module export and auto import them
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-vetur-completion-use-scaffold-snippets`

> Default value: `t`
>
> Enable/disable Vetur's built-in scaffolding snippets
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-completion-tag-casing`

> Default value: `"kebab"`
>
> Casing conversion for tag completion
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-grammar-custom-blocks`

> Default value: `((docs . "md") (i18n . "json"))`
>
> Mapping from custom block tag name to language name. Used for generating grammar to support syntax highlighting for custom blocks.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-validation-template`

> Default value: `t`
>
> Validate vue-html in <template> using eslint-plugin-vue
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-validation-style`

> Default value: `t`
>
> Validate css/scss/less/postcss in <style>
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-validation-script`

> Default value: `t`
>
> Validate js/ts in <script>
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-enable`

> Default value: `t`
>
> Enable/disable the Vetur document formatter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-options-tab-size`

> Default value: `2`
>
> Number of spaces per indentation level. Inherited by all formatters.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-options-use-tabs`

> Default value: `nil`
>
> Use tabs for indentation. Inherited by all formatters.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-html`

> Default value: `"prettyhtml"`
>
> Default formatter for <template> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-css`

> Default value: `"prettier"`
>
> Default formatter for <style> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-postcss`

> Default value: `"prettier"`
>
> Default formatter for <style lang='postcss'> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-scss`

> Default value: `"prettier"`
>
> Default formatter for <style lang='scss'> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-less`

> Default value: `"prettier"`
>
> Default formatter for <style lang='less'> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-stylus`

> Default value: `"stylus-supremacy"`
>
> Default formatter for <style lang='stylus'> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-js`

> Default value: `"prettier"`
>
> Default formatter for <script> region
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-ts`

Default value: `"prettier"`

Default formatter for <script> region

Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-default-formatter-options`

Default value: `((js-beautify-html (wrap_attributes . "force-expand-multiline")) (prettyhtml (printWidth . 100) (singleQuote . :json-false) (wrapAttributes . :json-false) (sortAttributes . :json-false)))`

Options for all default formatters

Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-style-initial-indent`

Default value: `nil`

Whether to have initial indent for <style> region

Introduced in `lsp-mode` 6.1

- `lsp-vetur-format-script-initial-indent`

Default value: `nil`

Whether to have initial indent for <script> region

Introduced in `lsp-mode` 6.1

- `lsp-vetur-trace-server`

Default value: `"off"`

Traces the communication between VS Code and Vue Language Server.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-tsdk`

> Default value: `nil`
>
> Specifies the folder path containing the tsserver and lib*.d.ts files to use.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-disable-automatic-type-acquisition`

> Default value: `nil`
>
> Disables automatic type acquisition. Automatic type acquisition fetches ‘@types‘ packages from npm to improve IntelliSense for external libraries.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-npm`

> Default value: `nil`
>
> Specifies the path to the NPM executable used for Automatic Type Acquisition. Requires using TypeScript 2.3.4 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-check-npm-is-installed`

> Default value: `t`
>
> Check if NPM is installed for Automatic Type Acquisition.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-references-code-lens-enabled`

> Default value: `nil`
>
> Enable/disable references CodeLens in JavaScript files.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-references-code-lens-enabled`

> Default value: `nil`
>
> Enable/disable references CodeLens in TypeScript files.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-implementations-code-lens-enabled`

> Default value: `nil`
>
> Enable/disable implementations CodeLens. This CodeLens shows the implementers of an interface.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-tsserver-log`

> Default value: `"off"`
>
> 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.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-tsserver-plugin-paths`

> Default value: `nil`
>
> Additional paths to discover Typescript Language Service plugins. Requires using TypeScript 2.3.0 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-tsserver-trace`

> Default value: `"off"`
>
> 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 information from your project.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-complete-function-calls`

> Default value: `nil`
>
> Complete functions with their parameter signature.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggest-complete-function-calls`

> Default value: `nil`
>
> Complete functions with their parameter signature.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-report-style-checks-as-warnings`

> Default value: `t`
>
> Report style checks as warnings.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-validate-enable`

> Default value: `t`
>
> Enable/disable TypeScript validation.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-enable`

> Default value: `t`
>
> Enable/disable default TypeScript formatter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-comma-delimiter`

> Default value: `t`
>
> Defines space handling after a comma delimiter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-constructor`

> Default value: `nil`
>
> Defines space handling after the constructor keyword. Requires using TypeScript 2.3.0 or newer in the workspace.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-semicolon-in-for-statements`

> Default value: `t`
>
> Defines space handling after a semicolon in a for statement.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-before-and-after-binary-operators`

> Default value: `t`
>
> Defines space handling after a binary operator.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-keywords-in-control-flow-statements`

> Default value: `t`
>
> Defines space handling after keywords in a control flow statement.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-function-keyword-for-anonymous-functions`

> Default value: `t`
>
> Defines space handling after function keyword for anonymous functions.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-before-function-parenthesis`

Default value: `nil`

Defines space handling before function argument parentheses.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-opening-and-before-closing-nonempty-parenthesis`

Default value: `nil`

Defines space handling after opening and before closing non-empty parenthesis.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-opening-and-before-closing-nonempty-brackets`

Default value: `nil`

Defines space handling after opening and before closing non-empty brackets.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-opening-and-before-closing-nonempty-braces`

Default value: `t`

Defines space handling after opening and before closing non-empty braces. Requires using TypeScript 2.3.0 or newer in the workspace.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-opening-and-before-closing-template-string-braces`

Default value: `nil`

Defines space handling after opening and before closing template string braces.

Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-opening-and-before-closing-jsx-expression-braces`

> Default value: `nil`
>
> Defines space handling after opening and before closing JSX expression braces.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-insert-space-after-type-assertion`

> Default value: `nil`
>
> Defines space handling after type assertions in TypeScript. Requires using TypeScript 2.4 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-place-open-brace-on-new-line-for-functions`

> Default value: `nil`
>
> Defines whether an open brace is put onto a new line for functions or not.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-format-place-open-brace-on-new-line-for-control-blocks`

> Default value: `nil`
>
> Defines whether an open brace is put onto a new line for control blocks or not.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-validate-enable`

> Default value: `t`
>
> Enable/disable JavaScript validation.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-enable`

> Default value: `t`
>
> Enable/disable default JavaScript formatter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-comma-delimiter`

> Default value: `t`
>
> Defines space handling after a comma delimiter.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-constructor`

> Default value: `nil`
>
> Defines space handling after the constructor keyword. Requires using TypeScript 2.3.0 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-semicolon-in-for-statements`

> Default value: `t`
>
> Defines space handling after a semicolon in a for statement.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-before-and-after-binary-operators`

> Default value: `t`
>
> Defines space handling after a binary operator.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-keywords-in-control-flow-statements`

Default value: `t`

Defines space handling after keywords in a control flow statement.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-function-keyword-for-anonymous-functions`

Default value: `t`

Defines space handling after function keyword for anonymous functions.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-before-function-parenthesis`

Default value: `nil`

Defines space handling before function argument parentheses.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-opening-and-before-closing-nonempty-parenthesis`

Default value: `nil`

Defines space handling after opening and before closing non-empty parenthesis.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-opening-and-before-closing-nonempty-brackets`

Default value: `nil`

Defines space handling after opening and before closing non-empty brackets.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-opening-and-before-closing-nonempty-braces`

Default value: `t`

Defines space handling after opening and before closing non-empty braces. Requires using TypeScript 2.3.0 or newer in the workspace.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-opening-and-before-closing-template-string-braces`

Default value: `nil`

Defines space handling after opening and before closing template string braces.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-insert-space-after-opening-and-before-closing-jsx-expression-braces`

Default value: `nil`

Defines space handling after opening and before closing JSX expression braces.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-place-open-brace-on-new-line-for-functions`

Default value: `nil`

Defines whether an open brace is put onto a new line for functions or not.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-format-place-open-brace-on-new-line-for-control-blocks`

Default value: `nil`

Defines whether an open brace is put onto a new line for control blocks or not.

Introduced in `lsp-mode` 6.1

- `lsp-javascript-implicit-project-config-check-js`

> Default value: `nil`
>
> 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.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-implicit-project-config-experimental-decorators`

> Default value: `nil`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-names`

> Default value: `t`
>
> Enable/disable including unique names from the file in JavaScript suggestions.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-tsc-auto-detect`

> Default value: `"on"`
>
> Controls auto detection of tsc tasks.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-paths`

> Default value: `t`
>
> Enable/disable suggestions for paths in import statements and require calls.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggest-paths`

> Default value: `t`
>
> Enable/disable suggestions for paths in import statements and require calls.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-auto-imports`

> Default value: `t`
>
> Enable/disable auto import suggestions. Requires using TypeScript 2.6.1 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggest-auto-imports`

> Default value: `t`
>
> Enable/disable auto import suggestions. Requires using TypeScript 2.6.1 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-complete-js-docs`

> Default value: `t`
>
> Enable/disable suggestion to complete JSDoc comments.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggest-complete-js-docs`

> Default value: `t`
>
> Enable/disable suggestion to complete JSDoc comments.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-locale`

> Default value: `nil`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggestion-actions-enabled`

> Default value: `t`
>
> Enable/disable suggestion diagnostics for JavaScript files in the editor. Requires using TypeScript 2.8 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggestion-actions-enabled`

> Default value: `t`
>
> Enable/disable suggestion diagnostics for TypeScript files in the editor. Requires using TypeScript 2.8 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-preferences-quote-style`

> Default value: `"auto"`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-preferences-quote-style`

> Default value: `"auto"`
>
> nil
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-preferences-import-module-specifier`

> Default value: `"auto"`
>
> Preferred path style for auto imports.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-preferences-import-module-specifier`

> Default value: `"auto"`
>
> Infer the shortest path type.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-preferences-rename-shorthand-properties`

> Default value: `t`
>
> Enable/disable introducing aliases for object shorthand properties during renames. Requires using TypeScript 3.4 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-preferences-rename-shorthand-properties`

> Default value: `t`
>
> Enable/disable introducing aliases for object shorthand properties during renames. Requires using TypeScript 3.4 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-update-imports-on-file-move-enabled`

> Default value: `"prompt"`
>
> 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.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-update-imports-on-file-move-enabled`

> Default value: `"prompt"`
>
> Prompt on each rename.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-auto-closing-tags`

> Default value: `t`
>
> Enable/disable automatic closing of JSX tags. Requires using TypeScript 3.0 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-auto-closing-tags`

> Default value: `t`
>
> Enable/disable automatic closing of JSX tags. Requires using TypeScript 3.0 or newer in the workspace.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-javascript-suggest-enabled`

> Default value: `t`
>
> Enabled/disable autocomplete suggestions.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-suggest-enabled`

> Default value: `t`
>
> Enabled/disable autocomplete suggestions.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-typescript-surveys-enabled`

> Default value: `t`
>
> Enabled/disable occasional surveys that help us improve VS Code's JavaScript and TypeScript support.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-vetur-server-command`

> Default value: `("vls")`
>
> Command to start vetur.
>
>  Introduced in `lsp-mode` 6.1

XML support: `lsp-xml`

`lsp-xml` provides support for XML using lsp4xml

`lsp-xml` settings

- `lsp-xml-trace-server`

> Default value: `"off"`
>
> Traces the communication between VS Code and the XML language server.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-catalogs`

> Default value: `nil`
>
> Array of XML Catalogs
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-logs-client`

> Default value: `t`
>
> Should the server log to client output
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-split-attributes`

> Default value: `nil`
>
> Split multiple attributes each onto a new line
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-join-cdata-lines`

> Default value: `nil`
>
> Join lines in a CDATA tag's content
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-join-comment-lines`

> Default value: `nil`
>
> Join comment content on format
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-space-before-empty-close-tag`

> Default value: `t`
>
> Insert space before end of self closing tag. Example: <tag/> → <tag />
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-join-content-lines`

> Default value: `nil`
>
> Normalize the whitespace of content inside an element. Newlines and excess whitespace are removed.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-preserve-empty-content`

> Default value: `nil`
>
> Preserve empty content/whitespace in a tag.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-enabled`

> Default value: `t`
>
> Enable/disable ability to format document
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-format-quotations`

> Default value: `"doubleQuotes"`
>
> Which type of quotes to use for attribute values when formatting.
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-file-associations`

> Default value: `nil`
>
> Allows XML schemas to be associated to file name patterns. Example: [{ "systemId":"path/to/file.xsd","pattern": "file1.xml" },{ "systemId": "http://www.w3.org/2001/XMLSchema.xsd","pattern": "*/.xsd" }]
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-completion-auto-close-tags`

> Default value: `t`
>
> Enable/disable autoclosing of XML tags. IMPORTANT: Turn off editor.autoClosingTags for this to work
>
>  Introduced in `lsp-mode` 6.1

- `lsp-xml-server-vmargs`

Default value: `"-noverify -Xmx64M\n -XX:+UseG1GC -XX:+UseStringDeduplication"`

Specifies extra VM arguments used to launch the XML Language Server. Eg. use ‘-noverify -Xmx1G -XX:+UseG1GC -XX:+UseStringDeduplication‘ to bypass class verification, increase the heap size to 1GB and enable String deduplication with the G1 Garbage collector

Introduced in `lsp-mode` 6.1

- `lsp-xml-server-work-dir`

Default value: `"~/.lsp4xml"`

Set a custom folder path for cached XML Schemas. An absolute path is expected, although the ~ prefix (for the user home directory) is supported.

Introduced in `lsp-mode` 6.1

- `lsp-xml-validation-no-grammar`

Default value: `"hint"`

The message severity when a document has no associated grammar.

Introduced in `lsp-mode` 6.1

- `lsp-xml-validation-enabled`

Default value: `t`

Enable/disable all validation.

Introduced in `lsp-mode` 6.1

- `lsp-xml-validation-schema`

Default value: `t`

Enable/disable schema based validation. Ignored if "xml.validation.enabled": false.

Introduced in `lsp-mode` 6.1

- `lsp-xml-jar-file`

> Default value: `"~/.emacs.d/org.eclipse.lsp4xml-0.3.0-uber.jar"`
>
> Xml server jar command.
>
> 
>
> Introduced in `lsp-mode` 6.1

- `lsp-xml-server-command`

> Default value: `("java" "-jar" "~/.emacs.d/org.eclipse.lsp4xml-0.3.0-uber.jar")`
>
> Xml server command.
>
> 
>
> Introduced in `lsp-mode` 6.1

[1] A stand-alone, editor-agnostic program that provides IDE-like functionality (completion, references, error checking, etc) over a defined protocol. irony-mode and rls are both language servers that use different protocols to communicate.