Sciware - Introduction to static analysis tools in Python and C++

03/28/24

Thomas Hahn

FLATIRON INSTITUTE

Outline

- Motivation for static analysis tools
- Overview of language servers and the LSP
- Static analysis tools for C++ in VS Code using clangd

Motivation for static analysis tools

Python example in VS Code

- Minimal setup:
  - Make sure to install/enable **Microsoft's Python extension**
    - This will also install/enable the language server **Pylance**
    - Choose the correct Python interpreter so that the linter knows where to look for modules

Overview of static analysis tools

- What is static code analysis?
    - Analysis of computer programs without executing it (vs. dynamic code analysis)
- What can static analysis tools do for us?
    - Highlight semantic and stylistic problems in our code (undefined variable, missing parentheses, etc.)
    - Help us with refactoring, renaming symbols, formatting, etc.
    - Perform code autocompletion and suggest code snippets
    - Encourage us to stick to good coding practices
    - And many more…

Overview of static analysis tools

- What is static code analysis?
  - Analysis of computer programs without executing it (vs. dynamic code analysis)
- What can static analysis tools do for us?
  - Highlight semantic and stylistic problems in our code (undefined variable, missing parentheses, etc.)
  - Help us with refactoring, renaming symbols, formatting, etc.
  - Perform code autocompletion and suggest code snippets — Github copilot
  - Encourage us to stick to good coding practices — Ruff + pre-commit
  - And many more…

Overview of language servers and the LSP

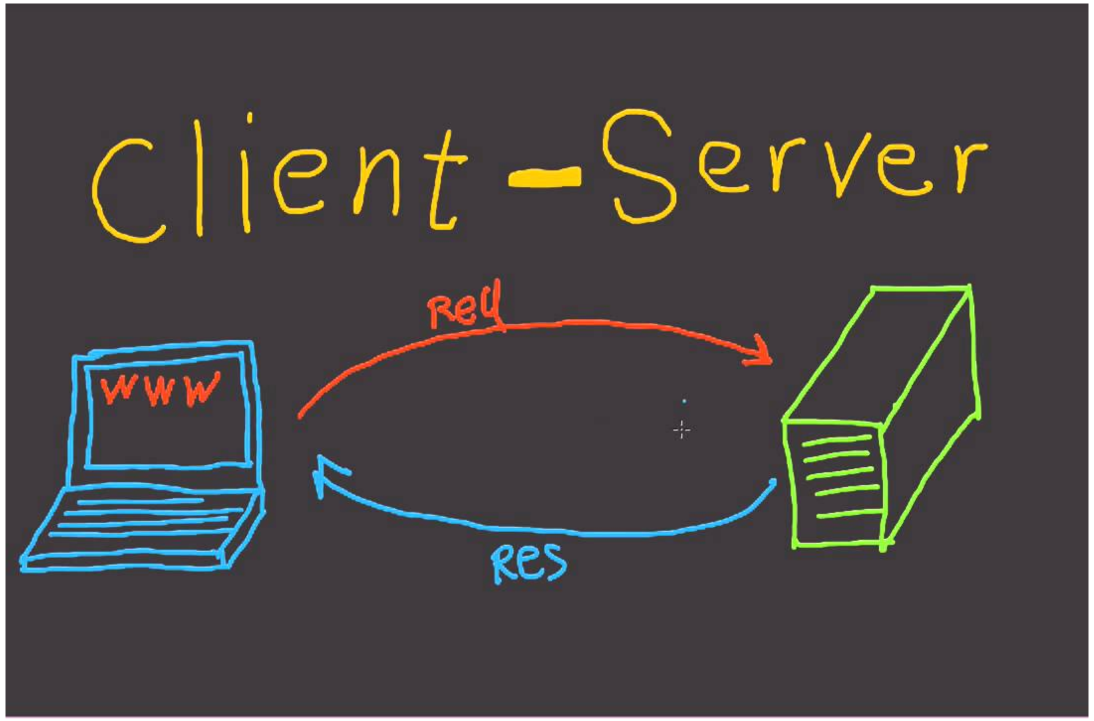
Language servers and the LSP

- **Language servers** provide language-specific smarts and communicate them to clients (usually IDEs or editors)

- The **Language server protocol** (LSP) is a standardized way for communication between a language server and a client

    - Single server for multiple development tools/clients

    - Easy for clients to support multiple languages via plugins



https://www.youtube.com/watch?app=desktop&v=SwLdKeC8scE

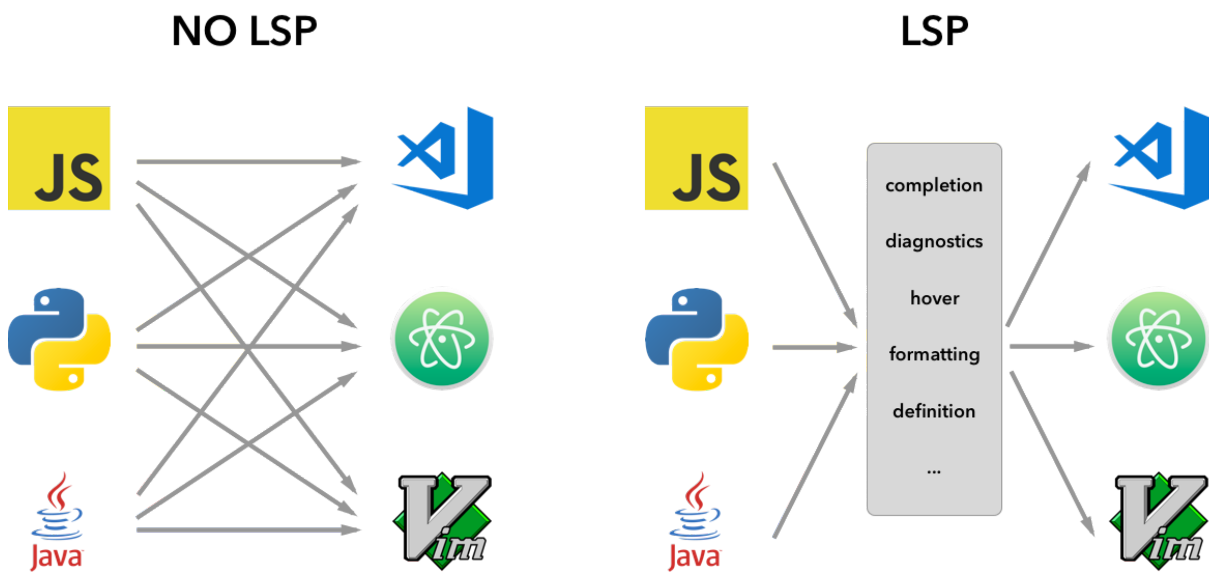
Language servers and the LSP

- **Language servers** provide language-specific smarts and communicate them to clients (usually IDEs or editors)

- The **Language server protocol** (LSP) is a standardized way for communication between a language server and a client

  - Single server for multiple development tools/clients

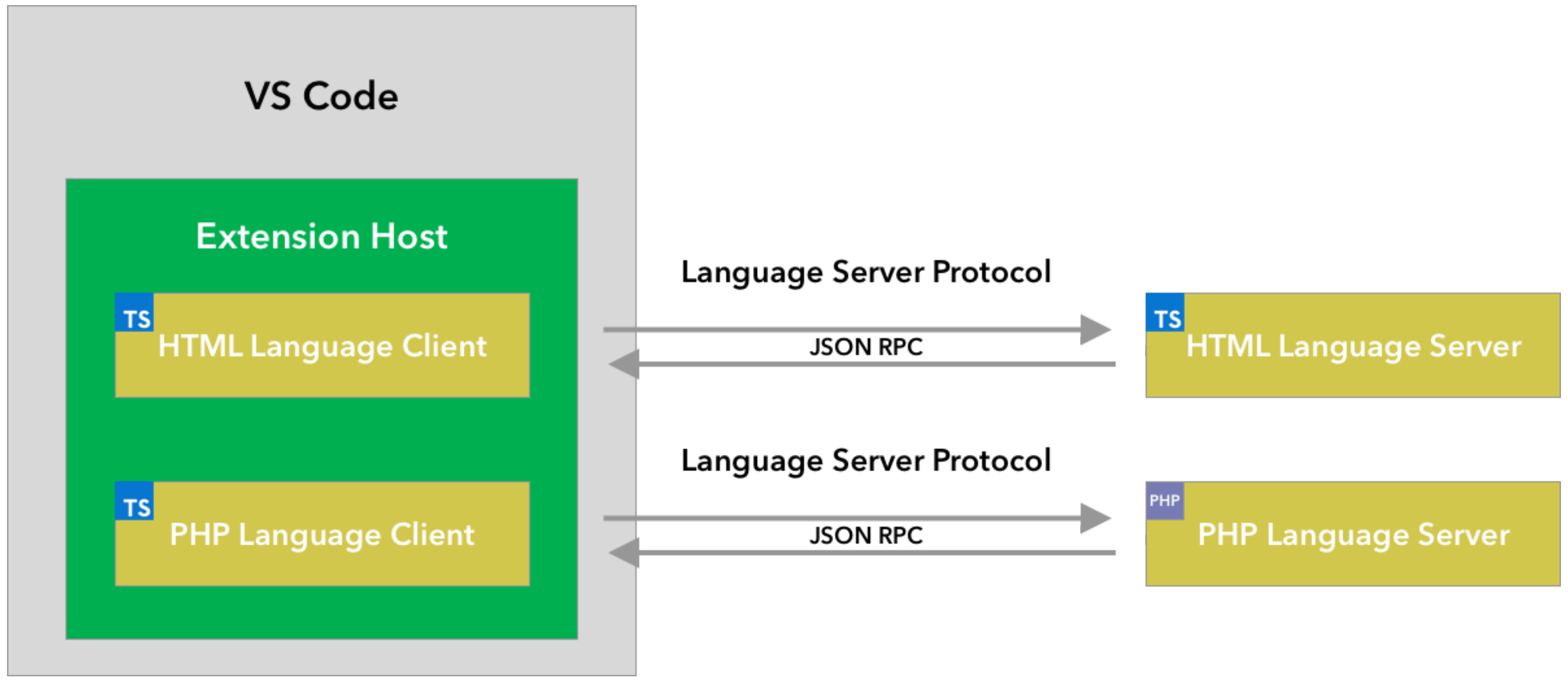
  - Easy for clients to support multiple languages via plugins



https://code.visualstudio.com/api/language-extensions/language-server-extension-guide

FLATIRON INSTITUTE

Language servers and the LSP



https://code.visualstudio.com/api/language-extensions/language-server-extension-guide

FLATIRON INSTITUTE

List of language servers

- See https://microsoft.github.io/language-server-protocol/implementors/servers/

Language	Language server	Maintainer
	pylance	Microsoft
	jedi	Samuel Roeca
	clangd	LLVM
	VS Code C++ extension	Microsoft

Static analysis tools for C++ in VS Code using clangd

What is clangd?

- Language server for C++ and part of the **LLVM** project
- Plugins for various editors and IDEs:
  - vim, Emacs, **VS Code**, Sublime Text, etc.
- Features:
  - Errors and warnings + possible fixes
  - clang-tidy checks, formatting with clang-format
  - Code completion + suggestions
  - Cross-references, refactoring, code navigation
  - and more …

C++ example in VS Code

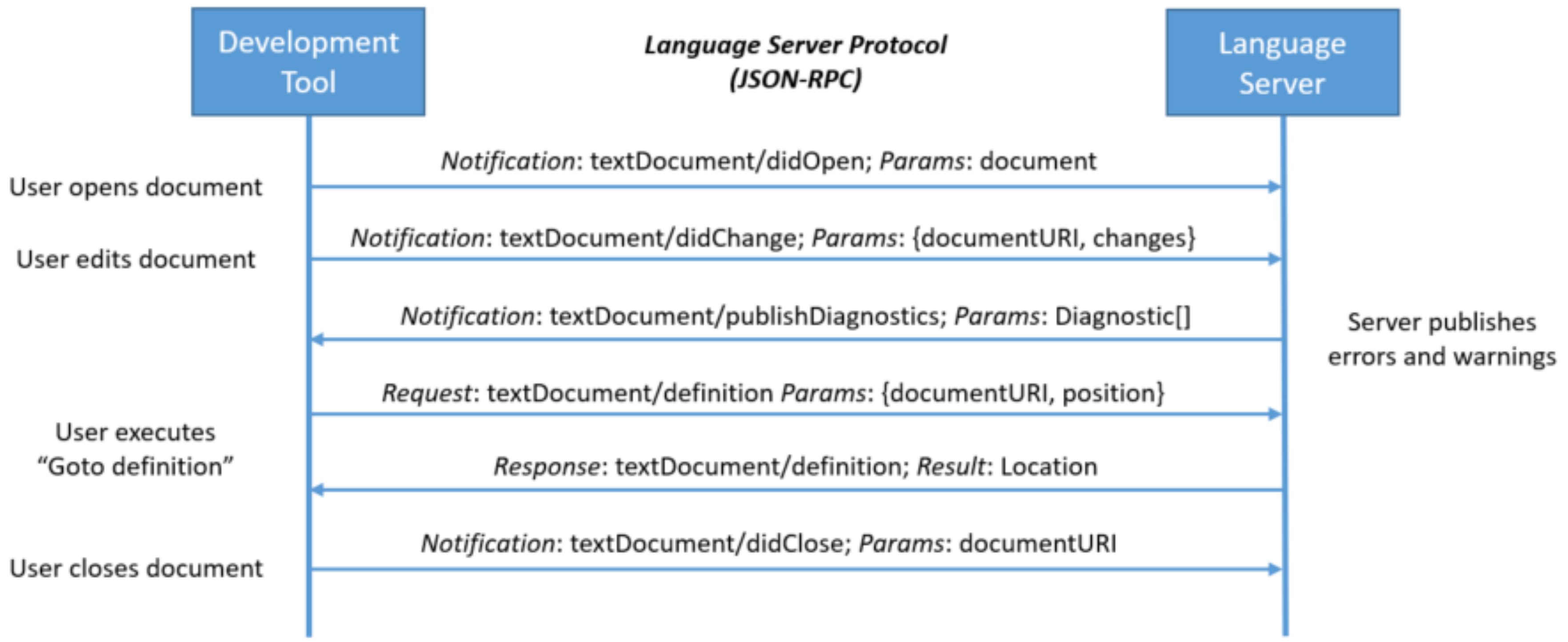
- Minimal setup:
  - Make sure you have **clangd** installed on your system
  - Install the **clangd extension** for VS Code and tell it where to find the clangd executable
  - clangd needs to know how you compiled your code to function properly
    - provide a **compile_commands.json** file
    - if you use **Microsoft's CMake Tools** extension the file generated automatically

Thank you.

FLATIRON INSTITUTE

FLATIRON
INSTITUTE

How does the LSP communication work?


Development Tool
Language Server Protocol (JSON-RPC)
Language Server
Notification: textDocument/didOpen; *Params*: document
User opens document
Notification: textDocument/didChange; *Params*: {documentURI, changes}
User edits document
Notification: textDocument/publishDiagnostics; *Params*: Diagnostic[]
Server publishes errors and warnings
Request: textDocument/definition *Params*: {documentURI, position}
User executes "Goto definition"
Response: textDocument/definition; *Result*: Location
Notification: textDocument/didClose; *Params*: documentURI
User closes document

https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/

How does the LSP communication work?

- Request from the IDE/client to the language server

```
{
    "jsonrpc": "2.0",
    "id" : 1,
    "method": "textDocument/definition",
    "params": {
        "textDocument": {
            "uri": "file:///p%3A/mseng/VSCode/Playgrounds/cpp/use.cpp"
        },
        "position": {
            "line": 3,
            "character": 12
        }
    }
}
```

https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/

How does the LSP communication work?

- Response from the language server for the IDE/client

```
{
    "jsonrpc": "2.0",
    "id": 1,
    "result": {
        "uri": "file:///p%3A/mseng/VSCode/Playgrounds/cpp/provide.cpp",
        "range": {
            "start": {
                "line": 0,
                "character": 4
            },
            "end": {
                "line": 0,
                "character": 11
            }
        }
    }
}
```

https://microsoft.github.io/language-server-protocol/overviews/lsp/overview/