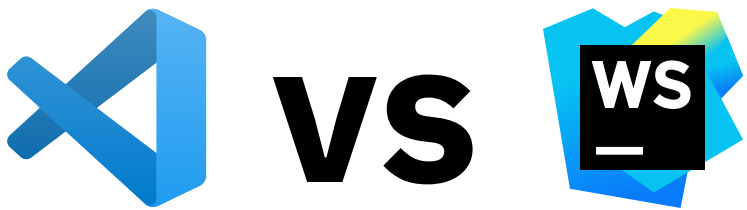
Как работает LSP?

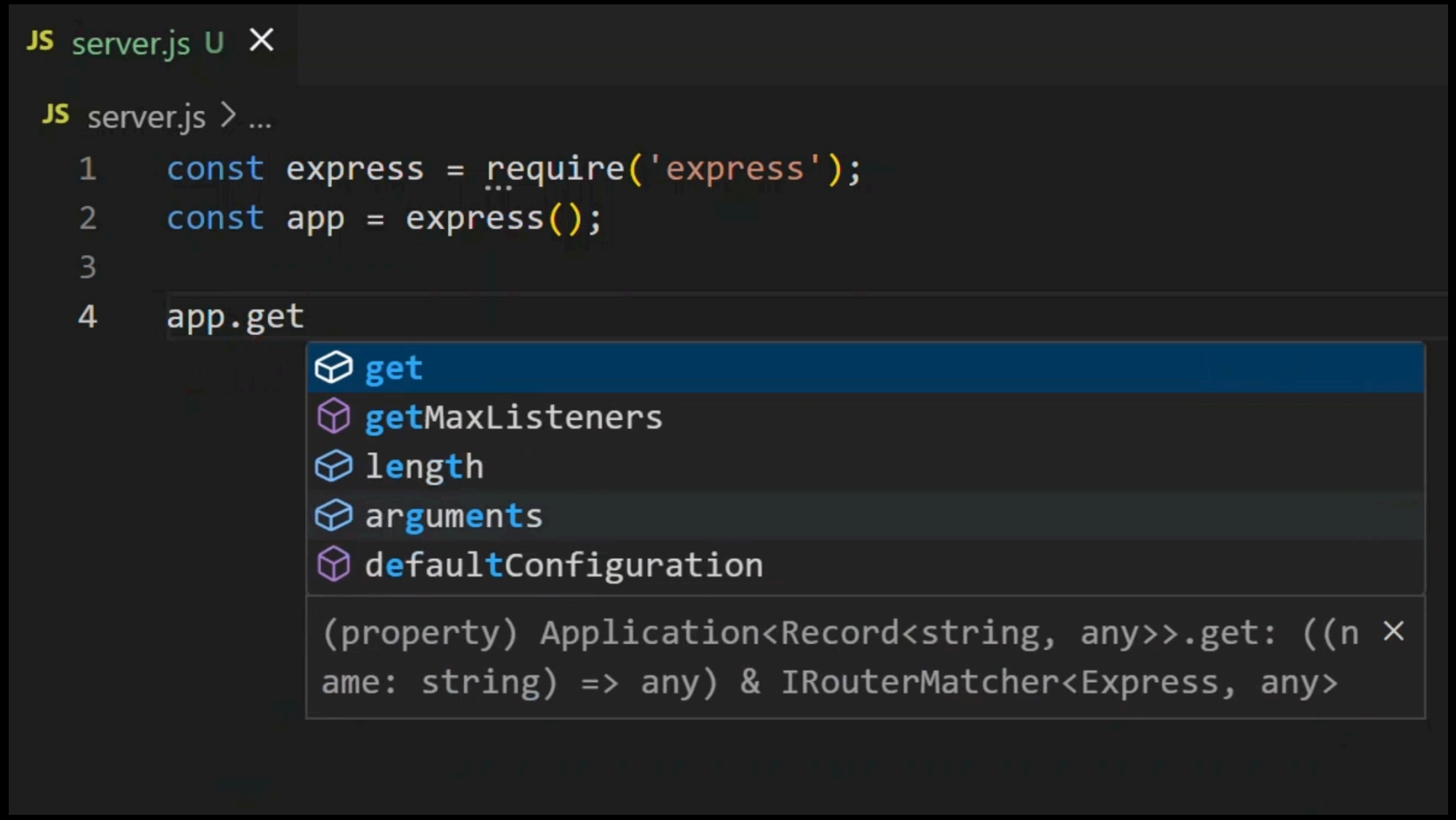
Сэм Булатов

- В разработке 8+ лет
- Работаю в Т-Банке, CoreTech Frontend
- Организатор конференций KRD.DEV сообщества
- Спикер


VS
WS

Редактор кода vs IDE

- **Раньше:** редактор = текст + подсветка
- **IDE:** понимание кода + refactoring
- **Сейчас:** LSP стирает границы


JS server.js U
JS server.js > ...
1 const express = require('express');
2 const app = express();
3
4 app.get
get
getMaxListeners
length
arguments
defaultConfiguration
(property) Application<Record<string, any>>.get: ((n
ame: string) => any) & IRouterMatcher<Express, any>

Что такое LSP?

Проблема

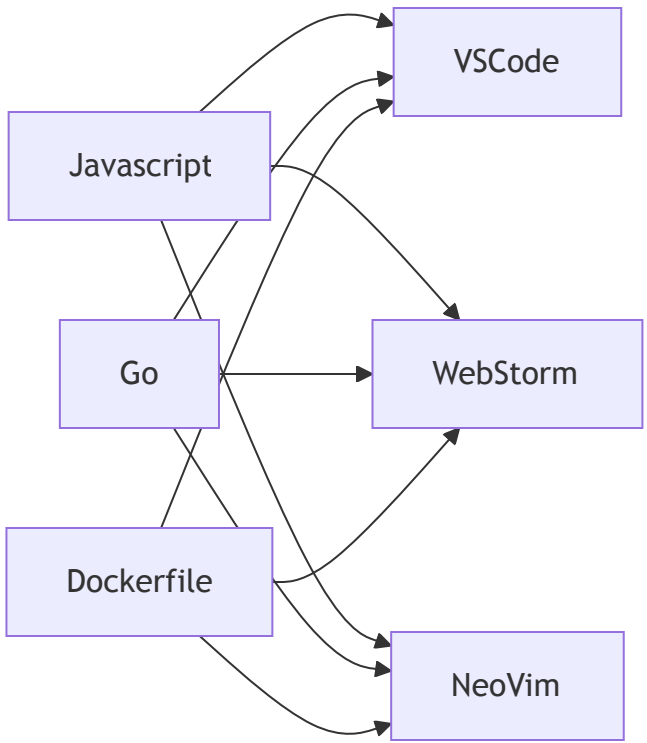
N фреймворков × M редакторов = **N×M** интеграций


astro

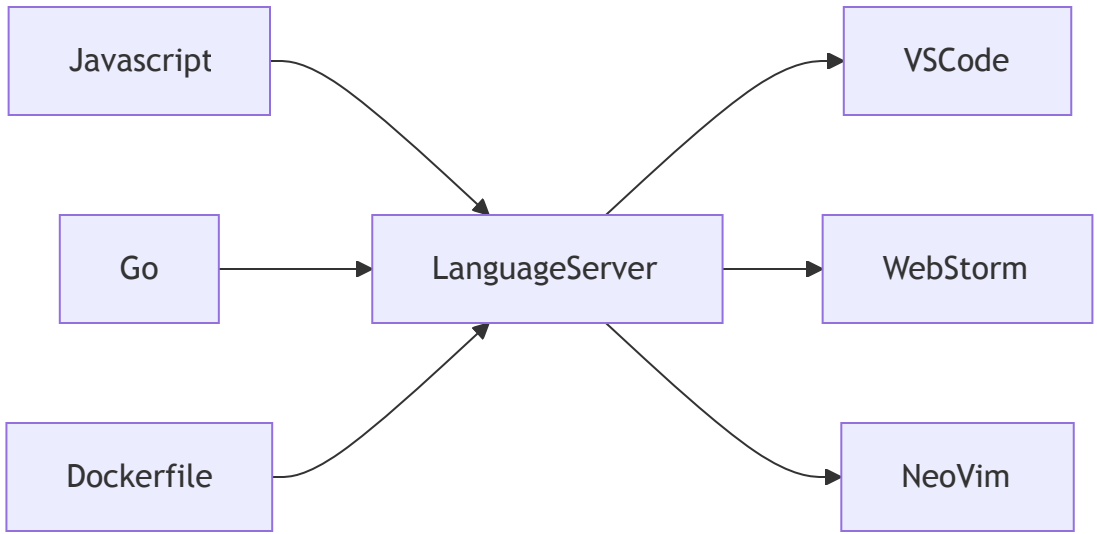
```
---
import SomeAstroComponent from '../components/SomeAstroComponent.astro';
import SomeReactComponent from '../components/SomeReactComponent.jsx';
import someData from '../data/pokemon.json';

// Access passed-in component props, like ``
const { title } = Astro.props;

// Fetch external data, even from a private API or database
const data = await fetch('SOME_SECRET_API_URL/users').then(r => r.json());
---
<h1>{title}</h1>
```


VSCode
Javascript
Go
WebStorm
Dockerfile
NeoVim

LSP — это решение

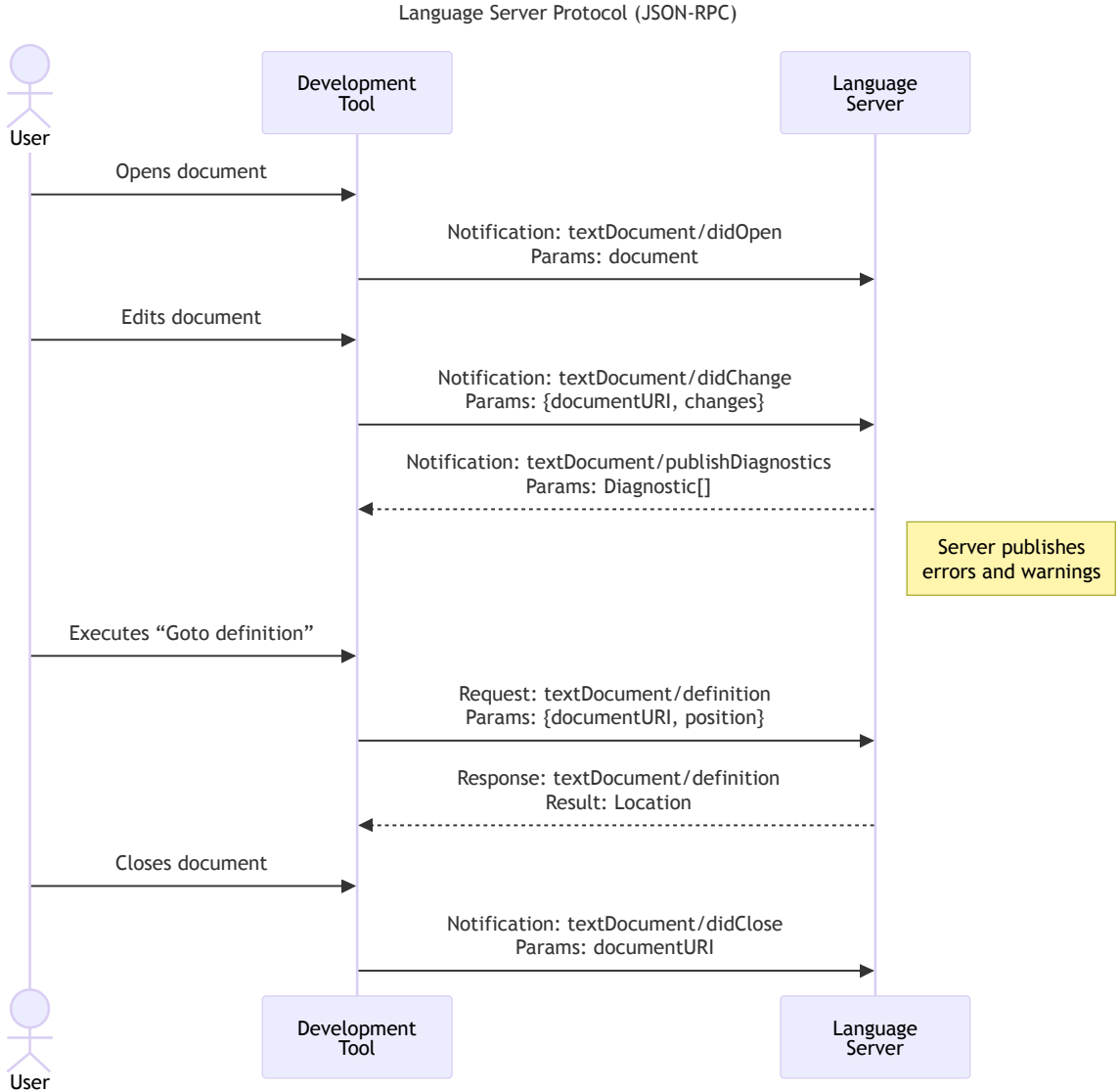


Плюсы LSP

- Пишешь один раз — для всех редакторов
- Можно написать свой сервер
- Поддержка сообществом
- Любой язык реализации

Устройство LSP

```
Content-Length: ...\r\n
\r\n
{
    "jsonrpc": "2.0",
    "id": 1,
    "method": "textDocument/completion",
    "params": {
        ...
    }
}
```


Language Server Protocol (JSON-RPC)
User
Development Tool
Language Server
Opens document
Notification: textDocument/didOpen
Params: document
Edits document
Notification: textDocument/didChange
Params: {documentURI, changes}
Notification: textDocument/publishDiagnostics
Params: Diagnostic[]
Server publishes errors and warnings
Executes “Goto definition”
Request: textDocument/definition
Params: {documentURI, position}
Response: textDocument/definition
Result: Location
Closes document
Notification: textDocument/didClose
Params: documentURI
User
Development Tool
Language Server

Установка соединения

```
import {fork} from "node:child_process"

const server = fork('./server.js', {
  stdio: ['pipe', 'pipe', 'pipe', 'ipc'],
});

server.on("data", () => {...})
```

Не HTTP! Через **stdio** (stdin/stdout)

capabilities

Language Features

- Go to definition
- Hover
- Find references
- ...

Workspace Features

- Workspace symbols
- Apply edit
- Get configuration
- ...

Window Features

- Show message notification
- Log Message
- Show Document
- ...

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "method": "textDocument/definition",
  "params": {
    "textDocument": {
      "uri": "file:///someFolder/main.go"
    },
    "position": {
      "line": 1,
      "character": 13
    }
  }
}
```

```
{
  "jsonrpc": "2.0",
  "id": 1,
  "result": {
    "uri": "file:///someFolder/exampleStructure.go",
    "range": {
      "start": {
        "line": 0,
        "character": 6
      },
      "end": {
        "line": 0,
        "character": 12
      }
```

textDocument/completion

```
// Пользователь пишет:
console.
```

completion: Запрос

```
{
  "method": "textDocument/completion",
  "params": {
    "position": {"line": 0, "character": 8},
    "context": {"triggerKind": 2, "triggerCharacter": "."}
  }
}
```

completion: Ответ


```
{
  "result": {
    "items": [{
      "label": "log",
      "kind": 2,
      "detail": "function log(...data: any[]): void",
      "documentation": {"kind": "markdown", "value": "Prints..."},
      "insertText": "log($1)"
    }]
  }
}
```

textDocument/codeAction

```
// Запрос (ошибка: Cannot find name 'useState')
{
  "method": "textDocument/codeAction",
  "params": {
    "context": {
      "diagnostics": [{
        "severity": 1,
        "code": "2304",
        "message": "Cannot find name 'useState'."
      }],
      "only": ["quickfix"]
    }
  }
}
```

codeAction: Ответ

```
// Ответ:
{
  "result": [{
    "title": "Add import from 'react'",
    "kind": "quickfix",
    "edit": {
      "changes": {
        "file:///project/src/App.tsx": [{
          "range": {"start": {"line": 0, "character": 0}, ...},
          "newText": "import { useState } from 'react';\n\n"
        }]
      }
    }
  }]
}
```

Gitlab CI Language Server

Проблема

- Много пайплайнов в разных файлах
- Нет **Go to Definition** для `include.local`
- Редактор не понимает структуру Gitlab CI

Пример .gitlab-ci.yml

```
include:
  - local: simple-documentation.yml
  - project: ci-cd-pipelines/js
    ref: 1.x
    file:
      - /app.yml


stages:
  - test
  - deploy
```

Хочу перейти к `simple-documentation.yml` по клику!

Инициализация сервера

```
const connection = createConnection(ProposedFeatures.all);
const documents = new TextDocuments(TextDocument);

connection.onInitialize((params: InitializeParams) => {
  return {
    capabilities: {
      textDocumentSync: TextDocumentSyncKind.Incremental,
      definitionProvider: true,
      workspaceSymbolProvider: true,
    },
    serverInfo: {
      version: "0.0.1",
      name: "Gitlab CI LS by Sam Bulatov"
    }
  }
```

Go to Definition: часть 1

```
connection.onDefinition((event) => {
  const textDocument = documents.get(event.textDocument.uri);
  const fileName = URI.parse(event.textDocument.uri).path;

  if (!isGitLabCiFile(fileName) && !hasGitLabCiContent(textDocument)) {
    return null;
  }

  const offset = textDocument.offsetAt(event.position);
  const yamlDocument = parseDocument(textDocument.getText());
  let best: FoundYamlNode | null = null;

  visit(yamlDocument, {
    Node(key, node, path) {
      if (!containsOffset(node.range, offset)) return visit.SKIP;
```

Go to Definition: часть 2

```
  const localPath = findIncludeLocalPath(best);
  if (!localPath) return null;

  const targetUri = resolveLocalPath(
    event.textDocument.uri,
    localPath
  );

  return Location.create(
    targetUri,
    Range.create(0, 0, 0, 0)
  );
})
```

Алгоритм

1. Проверяем что это Gitlab CI файл
2. Ищем AST узел под курсором
3. Проверяем что это `include.local`
4. Получаем путь до файла
5. Возвращаем Location клиенту

Спасибо!