Atitit ide功能表

# Basic fun

## 语法着色，自动完成

### **完成清单**

使用完成列表找出可用的标准库类型和函数签名。

# Run debug

函数文档功能，类似idea在源码显示，或者悬停文档等功能

# Terninal

# 几大重要功能菜单

Edit nav run terninal 重构 other

View （stutts

# Other

## 重构支持

Code formatter

自动完成、转到定义或悬停文档

## 什么是语言服务器协议？

为编程语言添加自动完成、转到定义或悬停文档等功能需要付出巨大的努力。传统上，必须为每个开发工具重复这项工作，因为每个工具都提供不同的 API 来实现相同的功能。

## 什么是语言服务器协议？

为编程语言添加自动完成、转到定义或悬停文档等功能需要付出巨大的努力。传统上，必须为每个开发工具重复这项工作，因为每个工具都提供不同的 API 来实现相同的功能。

一个*语言服务器*是为了提供特定语言智慧和通过协议，使进程间通信开发工具沟通。

*语言服务器协议 (LSP)*背后的想法是标准化此类服务器和开发工具如何通信的协议。这样，单个*语言服务器*可以在多个开发工具中重复使用，从而可以以最少的努力支持多种语言。

LSP 是语言供应商和工具供应商的双赢！

该协议定义了在开发工具和语言服务器之间使用 JSON-RPC 发送的消息的格式。LSIF 定义了一种图形格式来存储有关编程工件的信息。

### [规格](https://microsoft.github.io/language-server-protocol/specifications/specification-current)

LSP 规范的最新版本是 3.15 版。现在还有针对即将推出的语言服务器索引格式 (LSIF) 的规范。

### [实现](https://microsoft.github.io/language-server-protocol/implementors/servers/)

LSP 已针对多种语言实现，并且许多开发工具正在集成这些语言服务器。

当用户使用不同的语言时，开发工具通常会为每种编程语言启动一个语言服务器。下面的示例显示了用户处理 Java 和 SASS 文件的会话。

## 能力

并非每个语言服务器都可以支持协议定义的所有功能。因此，LSP 提供了“能力”。能力对一组语言特性进行分组。开发工具和语言服务器使用功能宣布它们支持的特性。例如，服务器宣布它可以处理“textDocument/definition”请求，但它可能无法处理“workspace/symbol”请求。类似地，开发工具宣布其能够在保存文档之前提供“即将保存”通知，以便服务器可以在保存之前计算文本编辑以格式化编辑的文档。

****请注意****，语言服务器与特定工具的实际集成不是由语言服务器协议定义的，而是由工具实现者决定的。

## LSP 提供者和消费者的库 (SDK)

为了简化语言服务器和客户端的实现，有一些库或 SDK：

开发工具 SDKs每个开发工具通常都提供一个用于集成语言服务器的库。例如，对于 JavaScript/TypeScript，有[语言客户端 npm 模块](https://www.npmjs.com/package/vscode-languageclient)

用于不同实现语言的语言服务器 SDK有一个 SDK 来实现特定语言的语言服务器。例如，要使用 Node.js 实现语言服务器，有[语言服务器 npm 模块](https://www.npmjs.com/package/vscode-languageserver)。

此扩展为 VS Code 添加了对 Swift 语言的丰富语言支持。这些功能由 Swift 框架本身通过[SourceKit](https://github.com/apple/swift/tree/master/tools/SourceKit)和[Swift 语言服务器](https://github.com/RLovelett/langserver-swift)实现提供。