What's New for Visual Studio Code:

Performance, GitHub Copilot, and CMake Enhancements

ALEXANDRA KEMPER
& SINEM AKINCI

Cppcon
The C++ Conference

2024
September 15 - 20

Meet the Speakers

Alexandra Kemper

PM for the C++ Extension for VS Code

Sinem Akinci

PM for C++ Copilot and CMake in VS Code

Agenda: What's new in VS Code

Faster Performance

Smarter GitHub Copilot suggestions

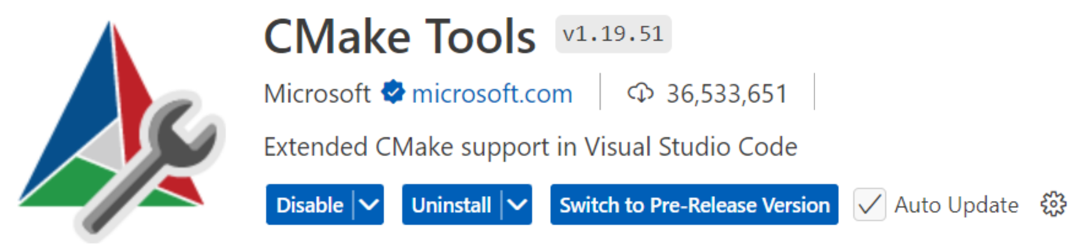
Easier to use Features for CMake & C/C++

Extensible and secure Copilot solutions

Wait I want to learn about Visual Studio!
Check out the talk tomorrow @ 13:30 "What's New in Visual Studio for C++ Developers" – Mryam Girmay & Michael Price

VS Code has a rich extension ecosystem

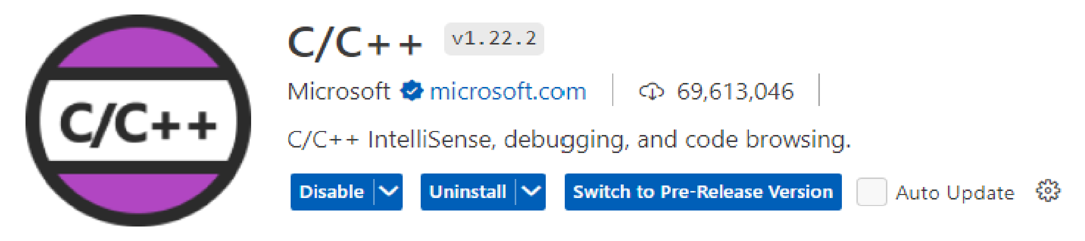
Extensions we'll cover today:



CMake Tools v1.19.51

Microsoft microsoft.com | 36,533,651

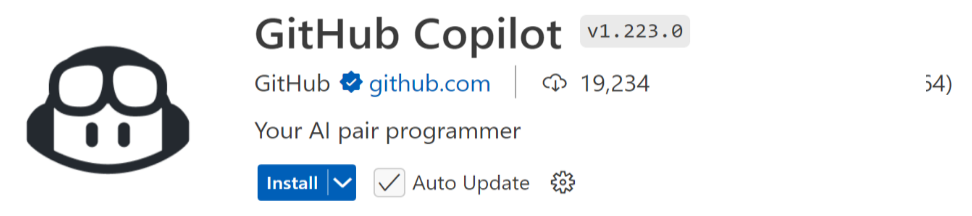
Extended CMake support in Visual Studio Code



C/C++ v1.22.2

Microsoft microsoft.com | 69,613,046

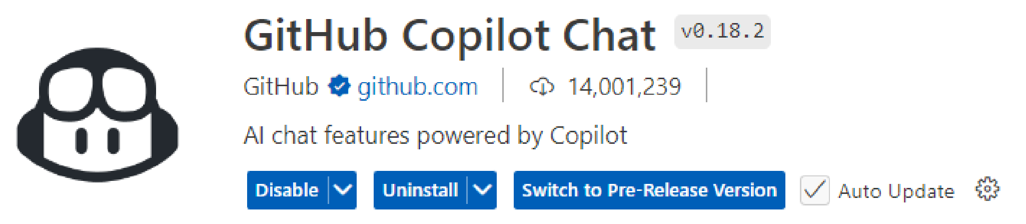
C/C++ IntelliSense, debugging, and code browsing.



GitHub Copilot v1.223.0

GitHub github.com | 19,234

Your AI pair programmer



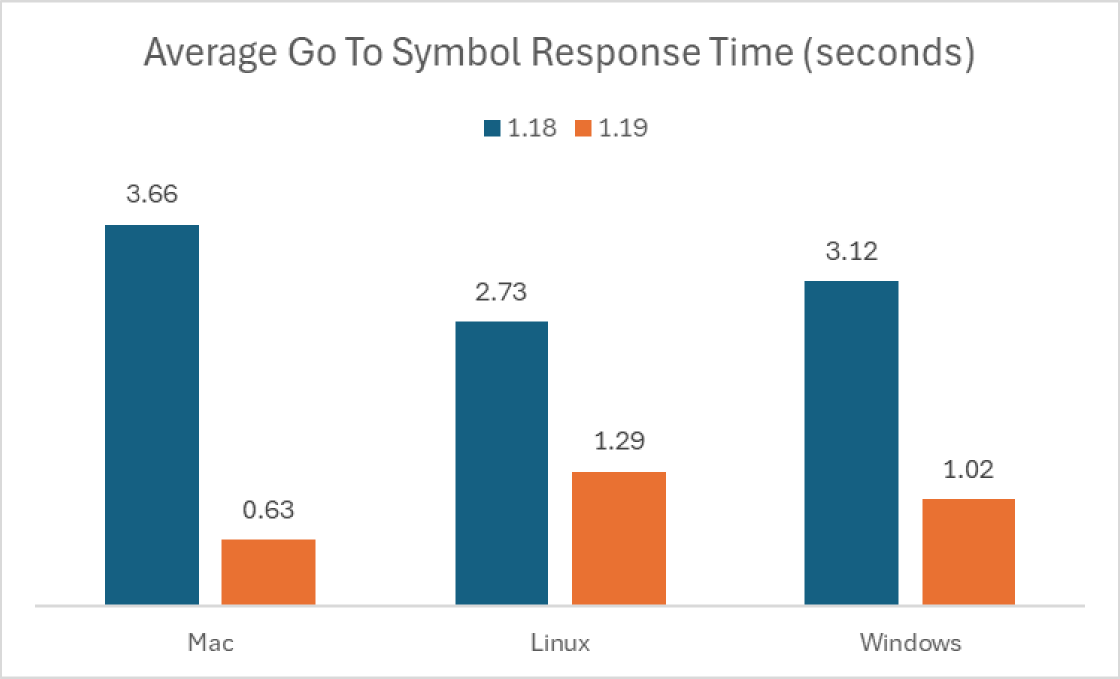
GitHub Copilot Chat v0.18.2

GitHub github.com | 14,001,239

AI chat features powered by Copilot

Code ***Faster*** with Performance Improvements

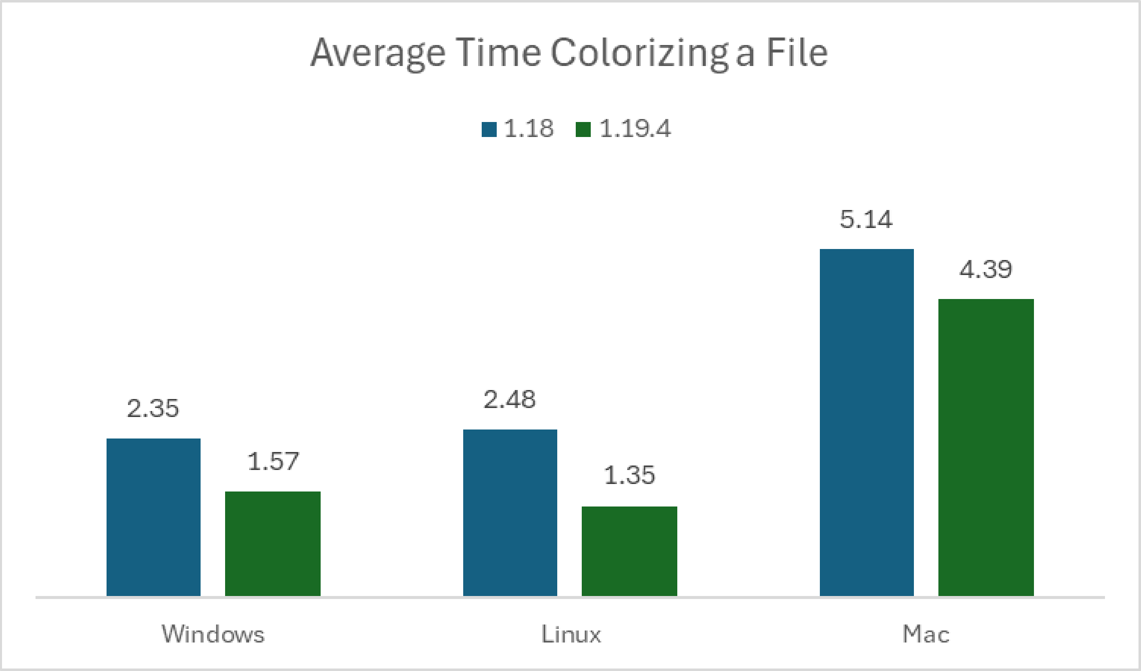
- 3.6x faster symbol search
  + new fuzzy search
- 1.5x faster time to IntelliSense
  + progressive IntelliSense
- 2-3x faster startup time [Insiders]



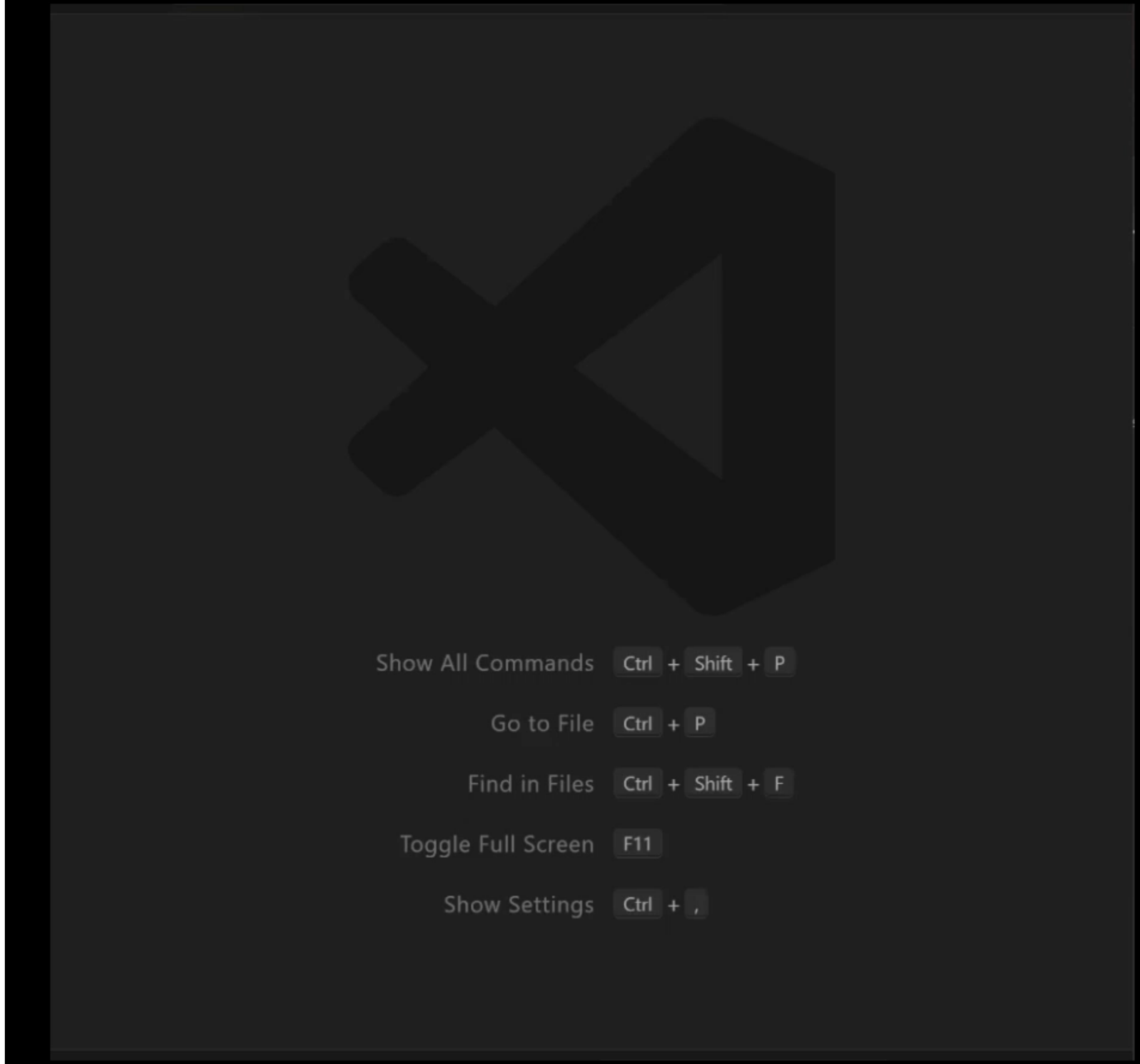
https://aka.ms/cpp/perf

Code ***Faster*** with Performance Improvements

- 3.6x faster symbol search
  + new fuzzy search
- 1.5x faster time to IntelliSense
  + progressive IntelliSense
- 2-3x faster startup time [Insiders]



https://aka.ms/cpp/perf



Show All Commands Ctrl + Shift + P

Go to File Ctrl + P

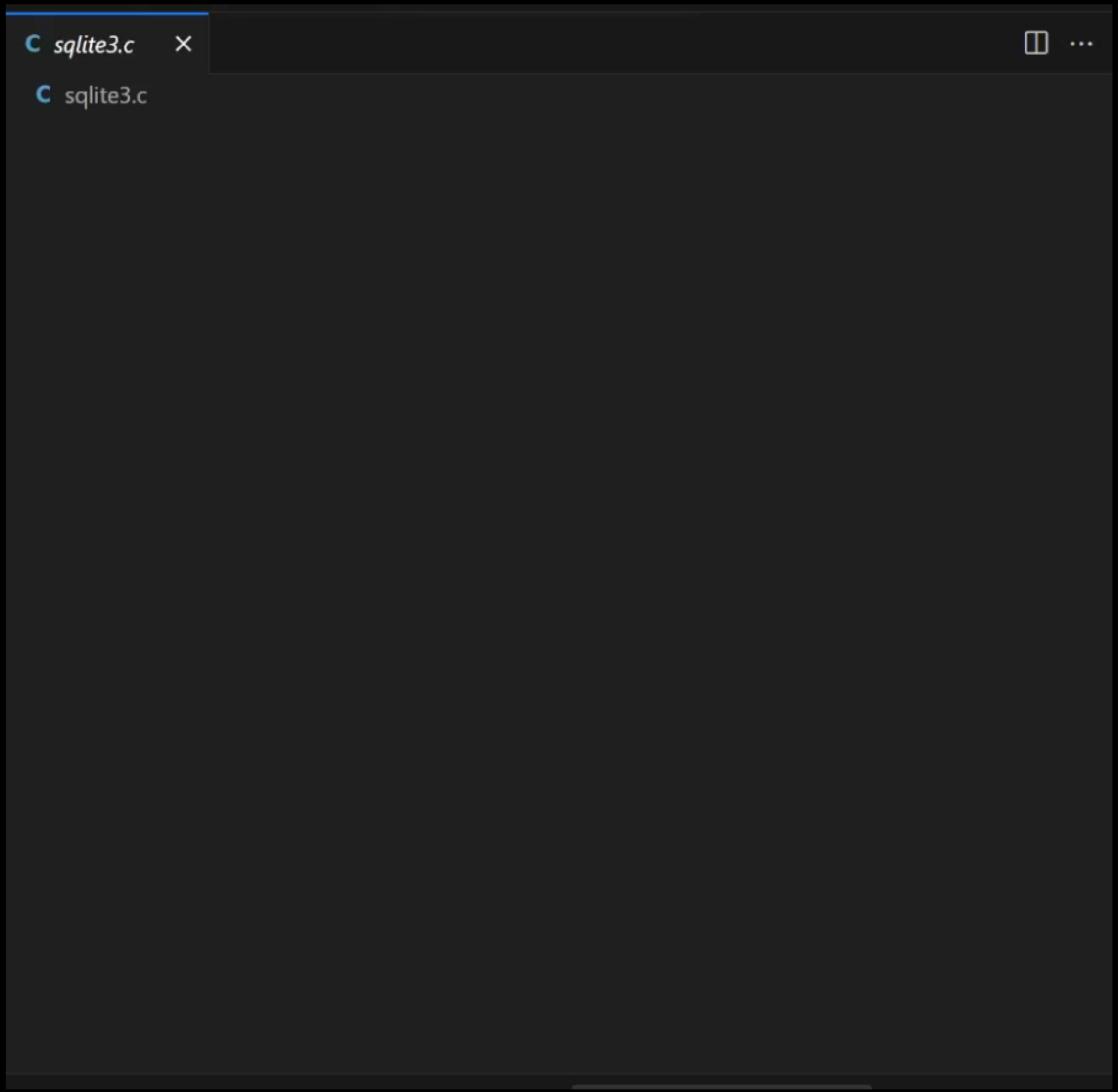
Find in Files Ctrl + Shift + F

Toggle Full Screen F11

Show Settings Ctrl + ,

00:00

Previously



sqlite3.c

sqlite3.c

00:00

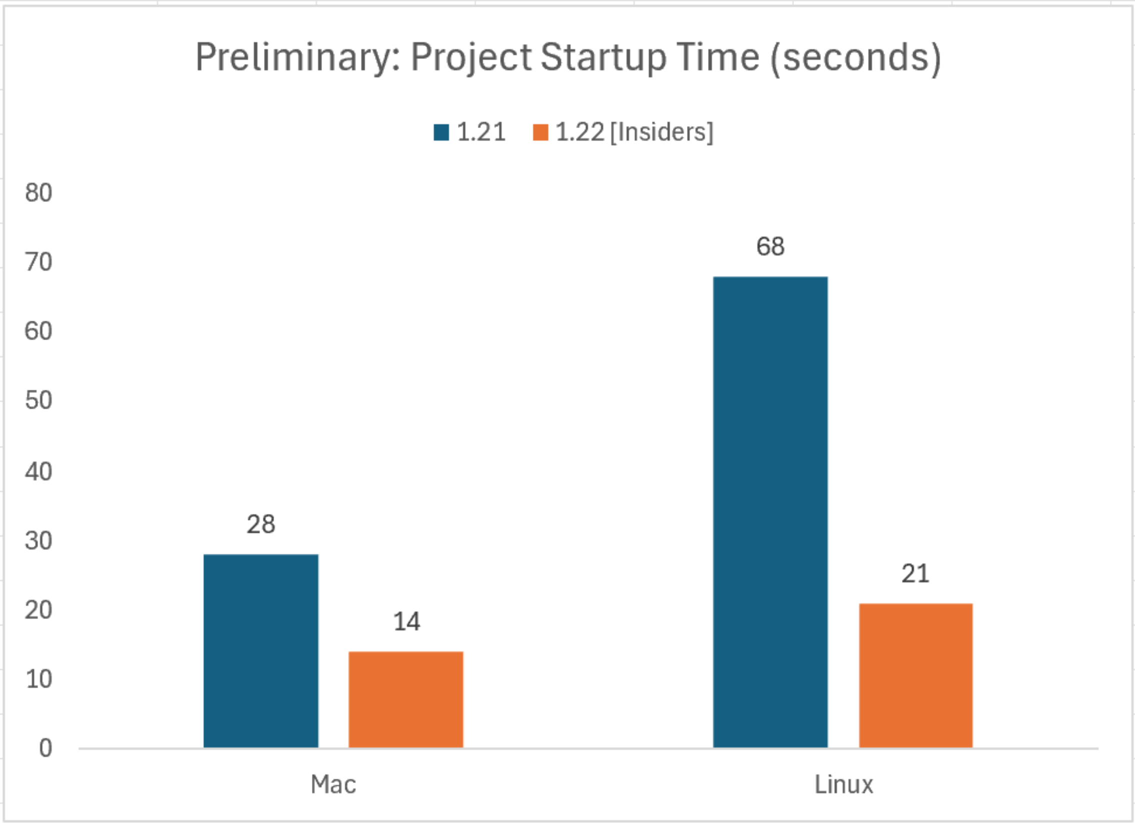
Version 1.19

Code *Faster* with Performance Improvements

- 3.6x faster symbol search
  + new fuzzy search

- 1.5x faster time to IntelliSense
  + progressive IntelliSense

- 2-3x faster startup time [Insiders]



Code ***Smarter*** with GitHub Copilot

The world's most widely adopted AI developer tool

Copilot can help you generate code, fix errors, and explain your code through *in-line suggestions* or answer questions through the *integrated chat* directly in VS Code

Our focus has been on making it ***smarter***, more ***reliable***, and more ***integrated***

- More relevant responses
- Deeper support for every part of your workflow

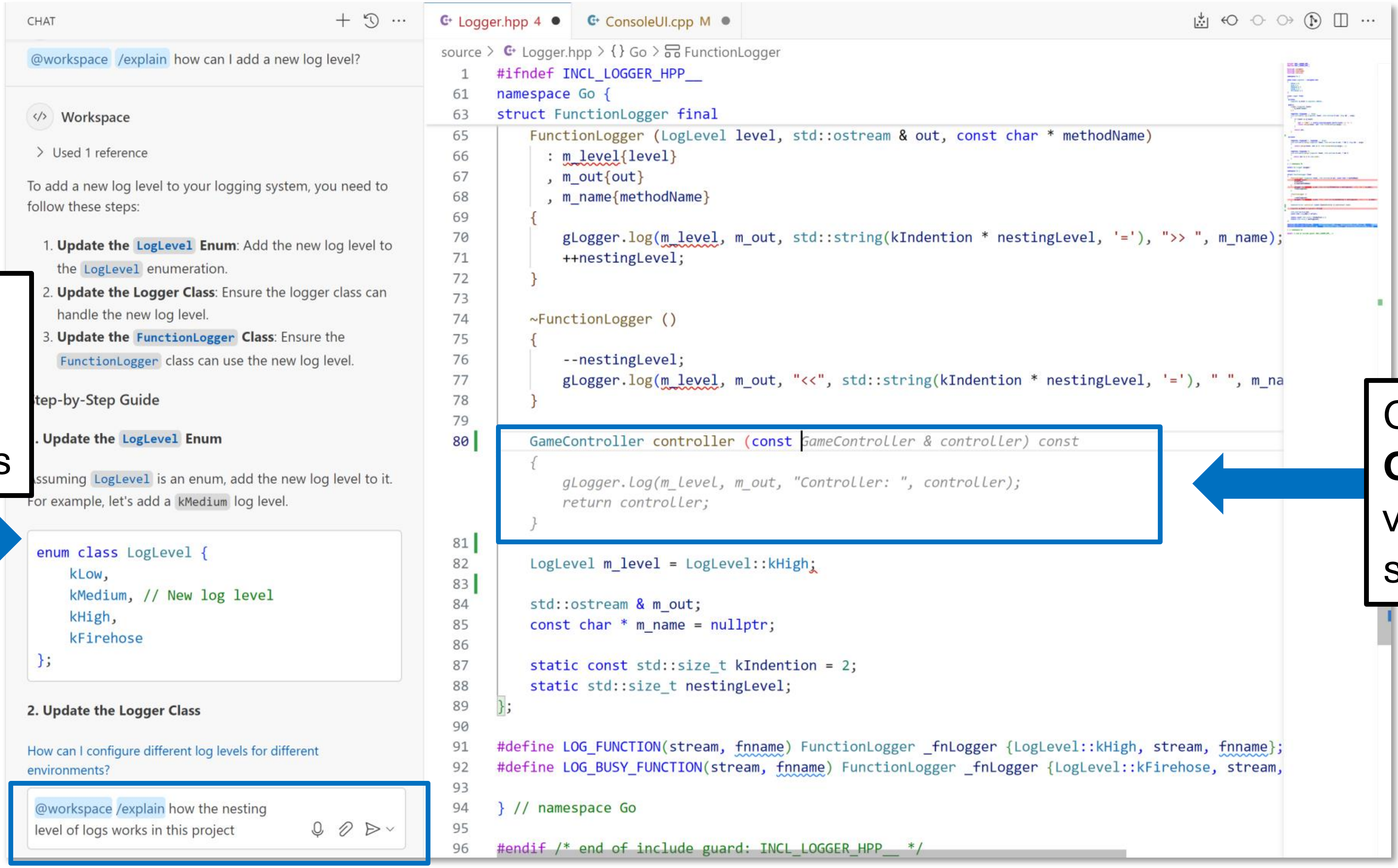
https://github.com/features/copilot

GitHub Copilot at your fingertips in VS Code

Dynamic context for custom suggestions

Ask about your code via **Chat**

Code **Completions** via Inline suggestions



@workspace /explain how can I add a new log level?

To add a new log level to your logging system, you need to follow these steps:

1. **Update the `LogLevel` Enum**: Add the new log level to the `LogLevel` enumeration.
2. **Update the Logger Class**: Ensure the logger class can handle the new log level.
3. **Update the `FunctionLogger` Class**: Ensure the `FunctionLogger` class can use the new log level.

tep-by-Step Guide

. Update the `LogLevel` Enum

ssuming `LogLevel` is an enum, add the new log level to it. For example, let's add a `kMedium` log level.

```
enum class LogLevel {
    kLow,
    kMedium, // New log level
    kHigh,
    kFirehose
};
```

2. Update the Logger Class

How can I configure different log levels for different environments?

@workspace /explain how the nesting level of logs works in this project

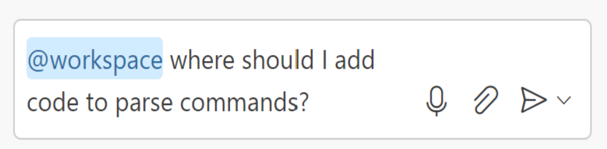
```
#ifndef INCL_LOGGER_HPP__
namespace Go {
struct FunctionLogger final
    FunctionLogger (LogLevel level, std::ostream & out, const char * methodName)
      : m_level{level}
      , m_out{out}
      , m_name{methodName}
    {
        gLogger.log(m_level, m_out, std::string(kIndention * nestingLevel, '='), ">> ", m_name);
        ++nestingLevel;
    }

    ~FunctionLogger ()
    {
        --nestingLevel;
        gLogger.log(m_level, m_out, "<<", std::string(kIndention * nestingLevel, '='), " ", m_na
    }

    GameController controller (const GameController & controller) const
    {
        gLogger.log(m_level, m_out, "Controller: ", controller);
        return controller;
    }

    LogLevel m_level = LogLevel::kHigh;

    std::ostream & m_out;
    const char * m_name = nullptr;

    static const std::size_t kIndention = 2;
    static std::size_t nestingLevel;
};

#define LOG_FUNCTION(stream, fnname) FunctionLogger _fnLogger {LogLevel::kHigh, stream, fnname};
#define LOG_BUSY_FUNCTION(stream, fnname) FunctionLogger _fnLogger {LogLevel::kFirehose, stream,

} // namespace Go

#endif /* end of include guard: INCL_LOGGER_HPP__ */
```

Add Chat Context with participants

More context provided leads to **more informed and relevant responses**

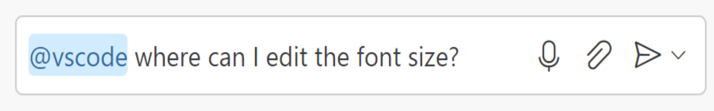
Add context to your chat commands by using Participants:

Questions about your Workspace?



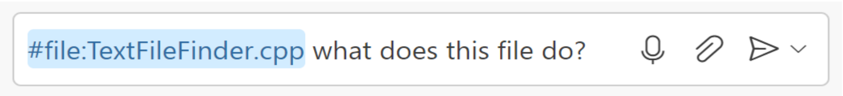
@workspace where should I add code to parse commands?

Questions about VS Code?



@vscode where can I edit the font size?

Questions about a specific file?



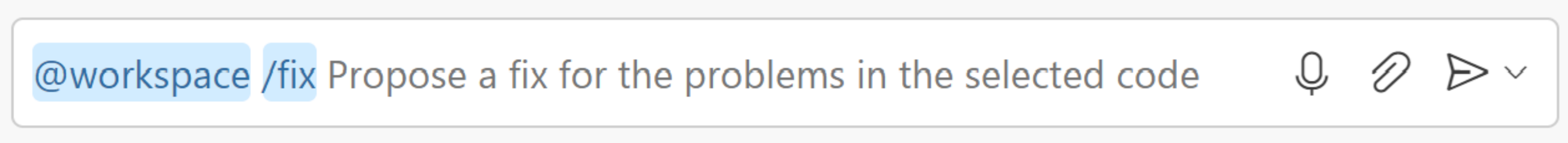
#file:TextFileFinder.cpp what does this file do?

Coming soon:

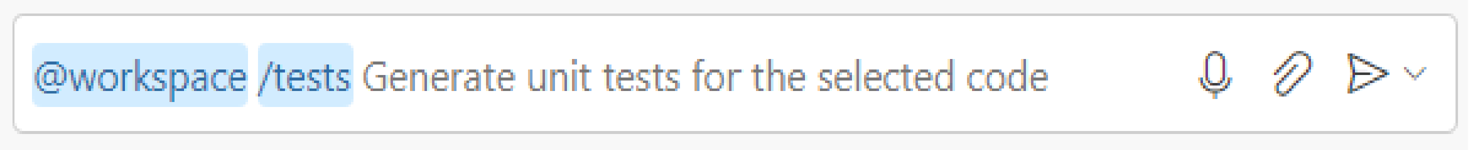
- #cpp for C++ project context (i.e. language standard, target OS, compiler name)
- Smart intent detection to detect participants + tools

Get specific with / commands

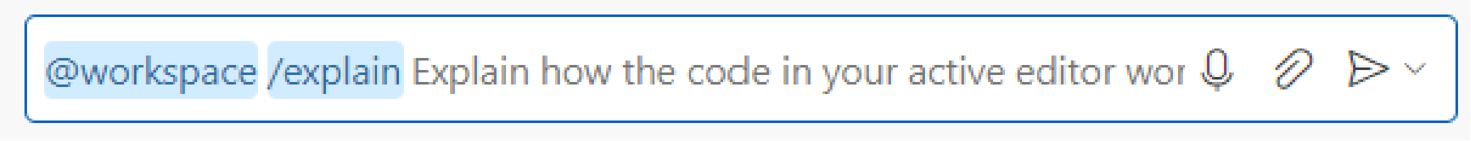
Use / commands to specify your intent:



@workspace /fix Propose a fix for the problems in the selected code



@workspace /tests Generate unit tests for the selected code

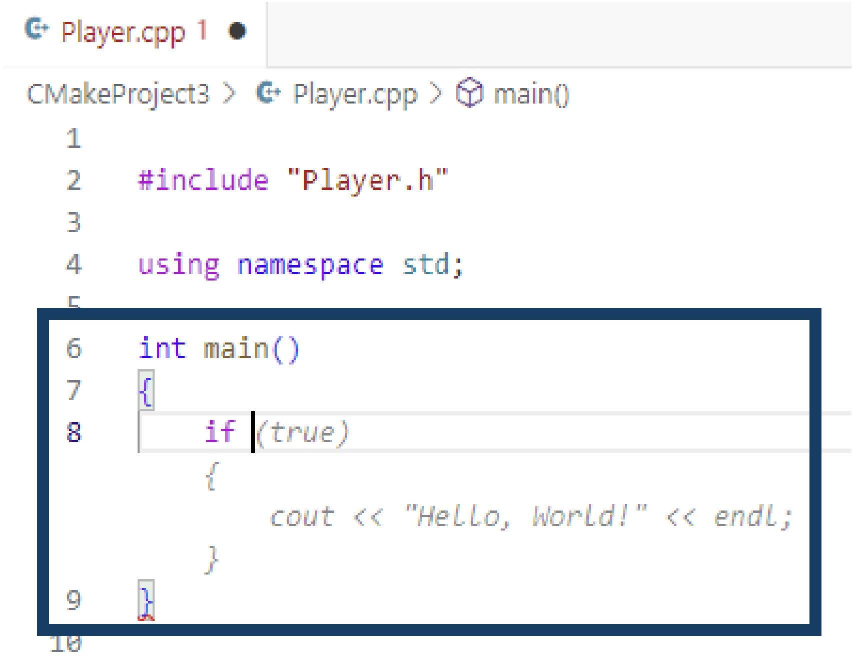


@workspace /explain Explain how the code in your active editor wor

In-line Completions tailored to C++

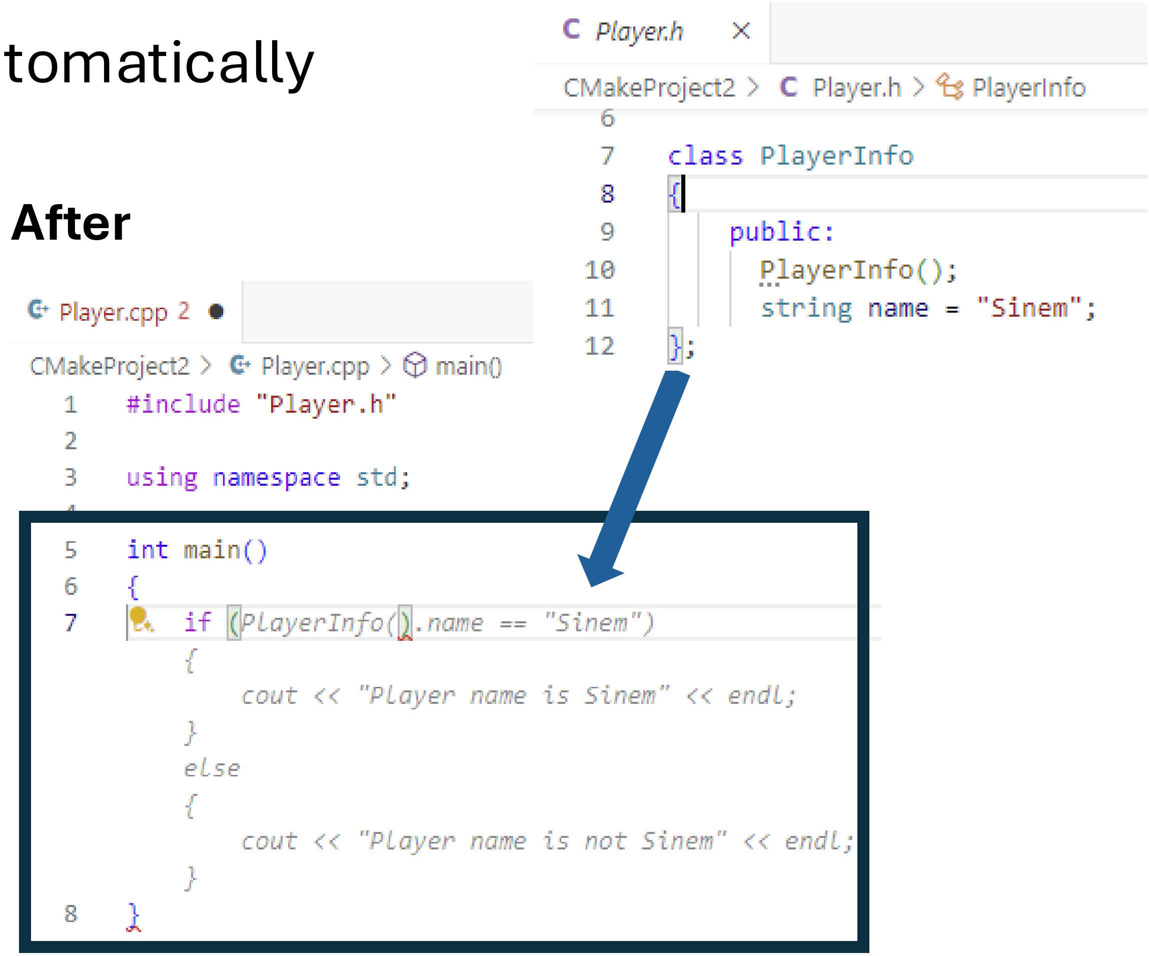
Directly-referenced C++ header files are automatically included as eligible context

Before



```
#include "Player.h"

using namespace std;

int main()
{
    if (true)
    {
        cout << "Hello, World!" << endl;
    }
}
```

After



Player.h ×
CMakeProject2 > Player.h > PlayerInfo
Player.cpp 2 ●
CMakeProject2 > Player.cpp > main()

```
class PlayerInfo
{
    public:
      PlayerInfo();
      string name = "Sinem";
};
```

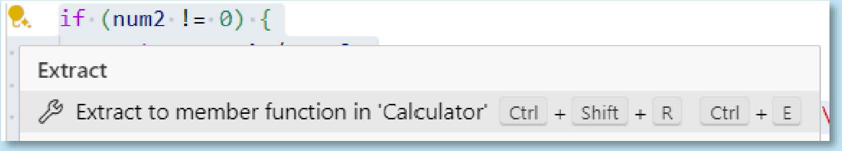
```
#include "Player.h"

using namespace std;

int main()
{
    if (PlayerInfo().name == "Sinem")
    {
        cout << "Player name is Sinem" << endl;
    }
    else
    {
        cout << "Player name is not Sinem" << endl;
    }
}
```

https://aka.ms/cpp/copilot/completions

Demo

Recap: New CMake & C++ Productivity features

Extract to Method

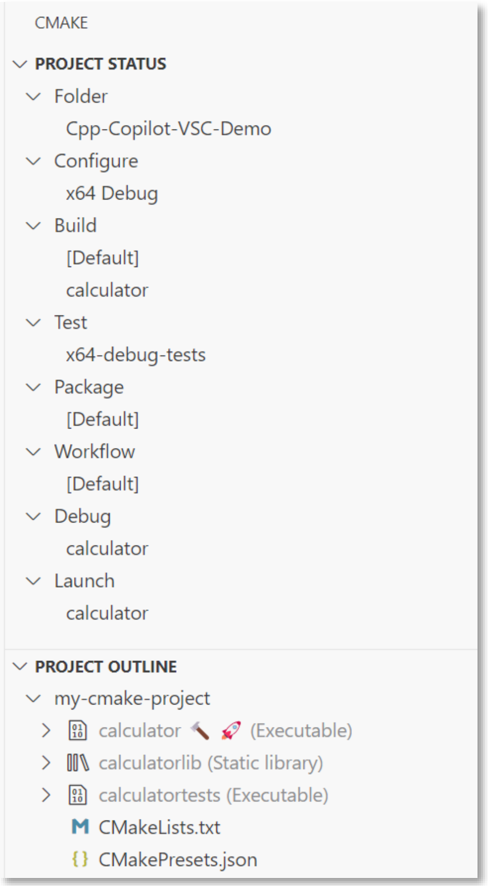


```
if (num2 != 0) {
```

Extract

Extract to member function in 'Calculator' Ctrl + Shift + R Ctrl + E

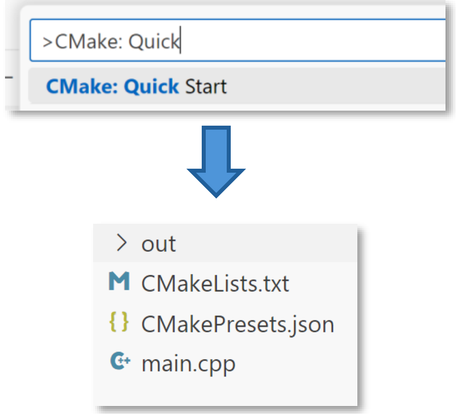
CMake Side Panel



CMAKE

- PROJECT STATUS
  - Folder
    - Cpp-Copilot-VSC-Demo
  - Configure
    - x64 Debug
  - Build
    - [Default]
    - calculator
  - Test
    - x64-debug-tests
  - Package
    - [Default]
  - Workflow
    - [Default]
  - Debug
    - calculator
  - Launch
    - calculator
- PROJECT OUTLINE
  - my-cmake-project
    - calculator (Executable)
    - calculatorlib (Static library)
    - calculatortests (Executable)
    - CMakeLists.txt
    - CMakePresets.json

Updated CMake Quick Start

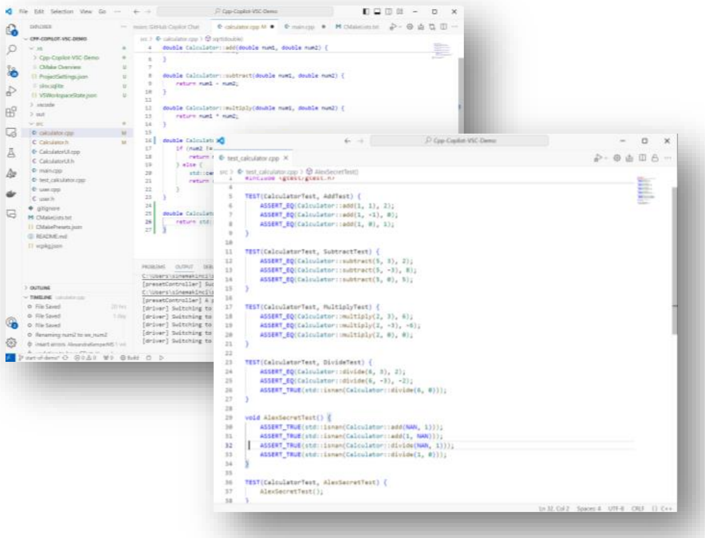


>CMake: Quick

CMake: Quick Start

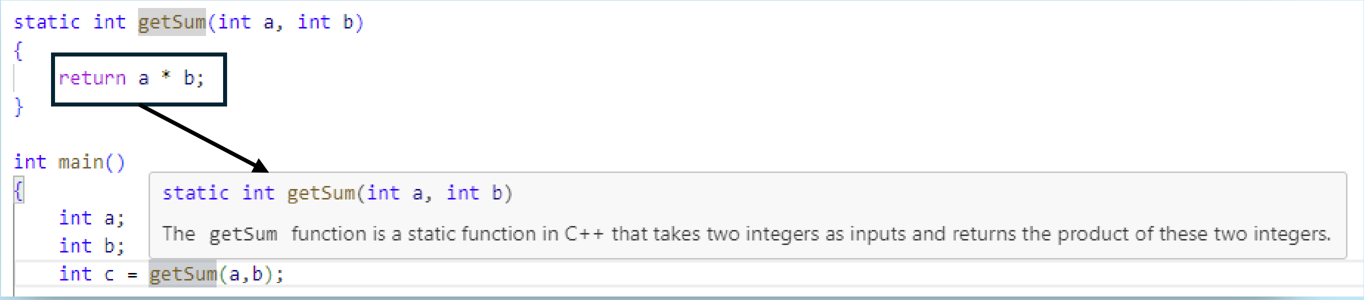
- out
- CMakeLists.txt
- CMakePresets.json
- main.cpp

Floating Editor Windows



Recap: New GitHub Copilot Features

Copilot in Quick Info (coming soon)

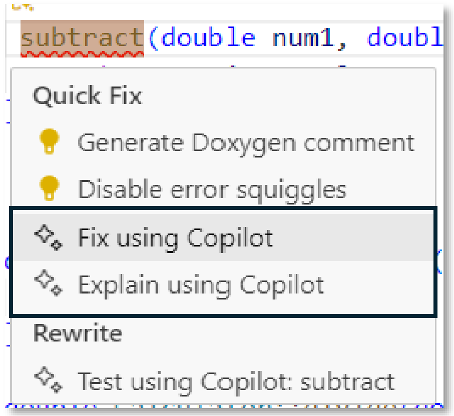


```
static int getSum(int a, int b)
{
    return a * b;
}

int main()
{
    int a;
    int b;
    int c = getSum(a,b);
```

```
static int getSum(int a, int b)
```

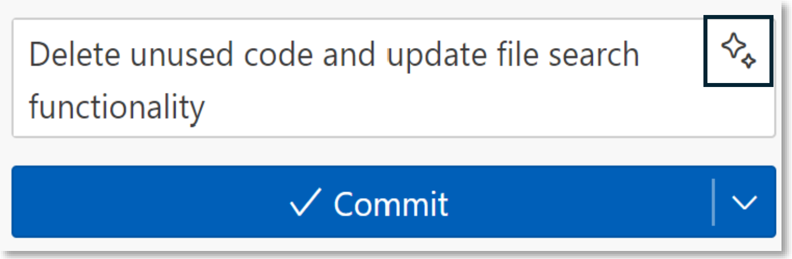
The getSum function is a static function in C++ that takes two integers as inputs and returns the product of these two integers.

Fix using Copilot



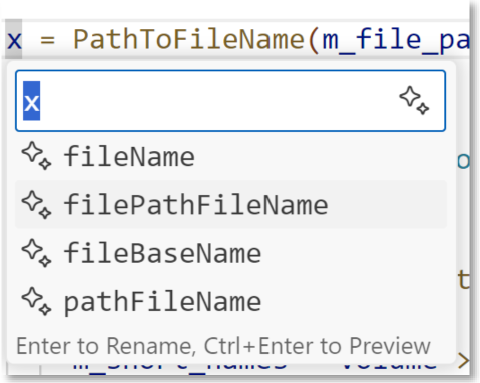
```
subtract(double num1, doubl
```

Commit Messages



Delete unused code and update file search functionality

Rename Suggestions



```
x = PathToFileName(m_file_pa
```

How can I tailor Copilot to fit my unique needs?

As a VS Code Extension Author

Build custom Copilot capabilities for your extension using new APIs

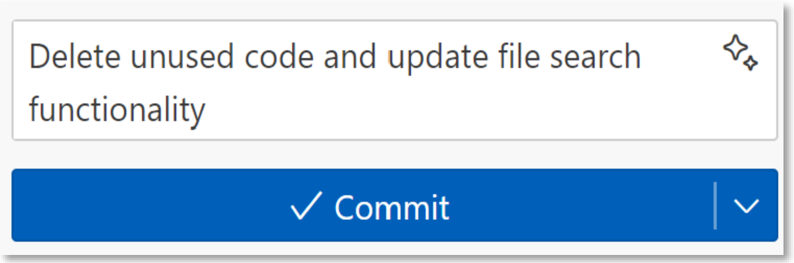
As a user or an organization with specific coding needs

Customize Copilot to your needs using custom models, instructions, and more…

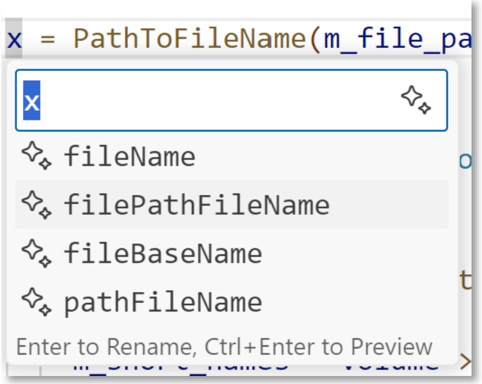
Build custom Copilot Chat features

Build your own custom features with the **Language Model API** to integrate AI-powered features and natural language processing into your VS Code extension

1. Build the language model prompt
2. Send the language model request
3. Interpret the response



Delete unused code and update file search functionality



```
x = PathToFileName(m_file_pa
```

x

Enter to Rename, Ctrl+Enter to Preview

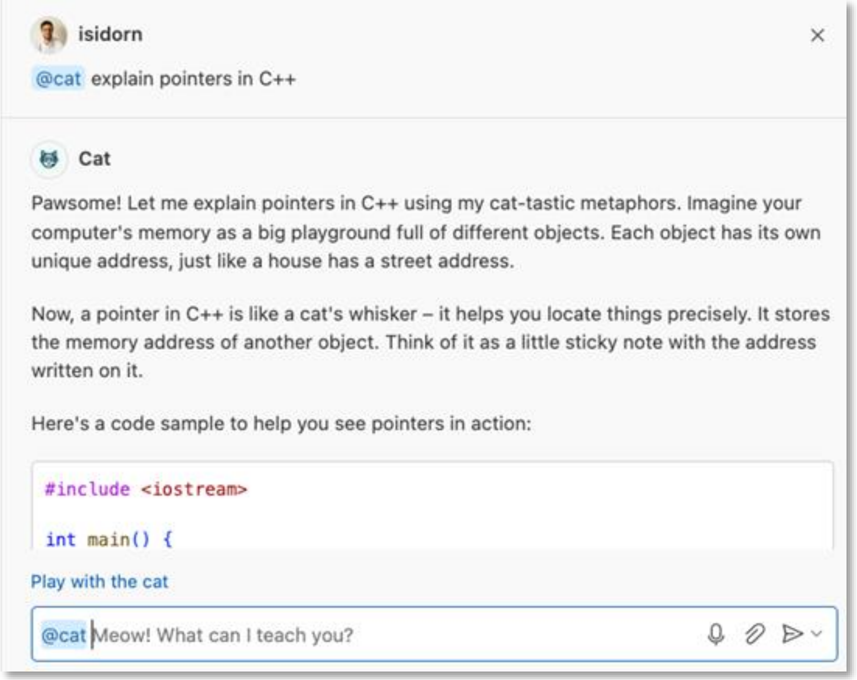
Extend Copilot Chat to match your needs

Build your own custom domain experts in the form of chat participants by using the **Chat Extension API**

- Get results for custom prompts
- Act as autonomous agents with invoke tools

Use this to provide responses in the form of:

- Buttons to invoke VS Code commands
- References to URLs
- Text, image, or code responses



isidorn

@cat explain pointers in C++

Cat

Pawsome! Let me explain pointers in C++ using my cat-tastic metaphors. Imagine your computer's memory as a big playground full of different objects. Each object has its own unique address, just like a house has a street address.

Now, a pointer in C++ is like a cat's whisker – it helps you locate things precisely. It stores the memory address of another object. Think of it as a little sticky note with the address written on it.

Here's a code sample to help you see pointers in action:

```
#include <iostream>

int main() {
```

Chat Participant Demo

How can I tailor Copilot to fit my unique needs?

As a VS Code Extension Author → Build custom Copilot capabilities for your extension using new APIs

As a user or an organization with specific coding needs → Customize Copilot to your needs using custom models, instructions, and more…

Custom Models for tailored Copilot suggestions

Receive more personalized and precise code suggestions by **fine-tuning a custom model to your unique needs**, such as

- training on a specific source code repository
- training on an organization's specific coding style, preferred libraries, and internal jargon

> *"Code completion is* ***much better than before****. It seems to generate functions quite well, given arguments, return type and a comment explaining what the function does. I personally used it to write a block of authentication code where a pop-up would be shown only if silent login fails". – Office AI, using fine-tuned model on their C++ repo*

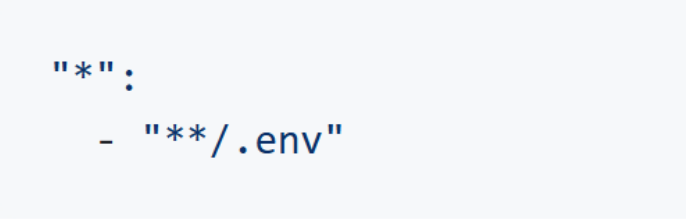
Free during beta, join the waitlist @ https://aka.ms/copilot/custommodels

Content Exclusion [Public Beta]

- Configure Copilot to ignore certain files (by path), these files
  - Will not have code completions available
  - Will not be used for context for code completion suggestions in other files
  - Will not be used to inform GitHub Copilot Chat responses

- Managed by organization owners
- Supports git & non-git based repos

Example: Ignore all .env files

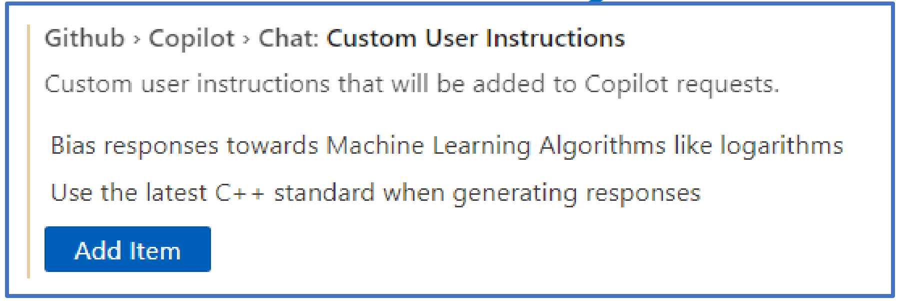


```
"*":
  - "**/.env"
```

https://aka.ms/copilot/exclude_content

Custom Instructions [Preview]

Provide knowledge and specify hints for Copilot Chat to take into context when answering requests ***specific to your workspace and needs***

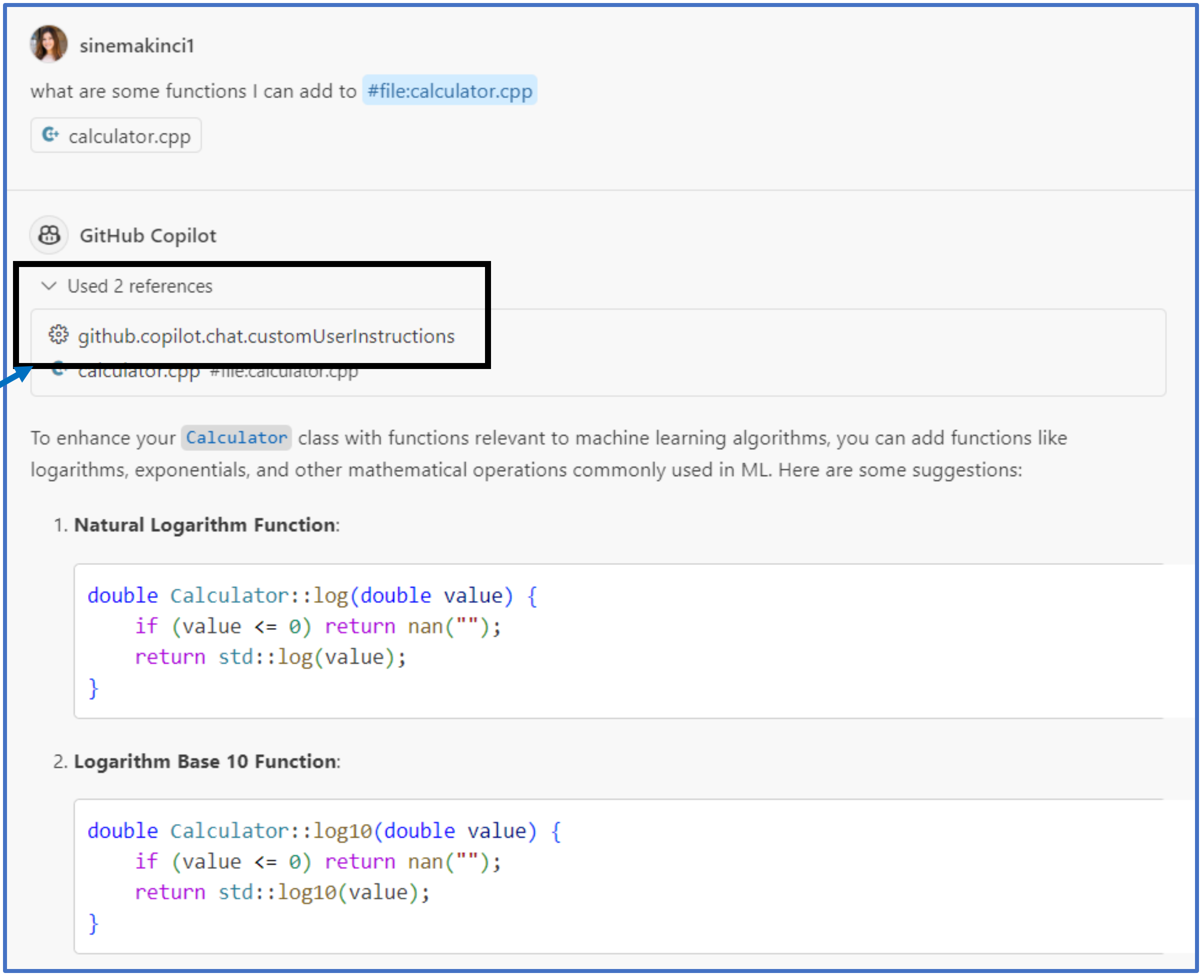


Github › Copilot › Chat: Custom User Instructions

Custom user instructions that will be added to Copilot requests.

Bias responses towards Machine Learning Algorithms like logarithms

Use the latest C++ standard when generating responses



sinemakinci1

what are some functions I can add to #file:calculator.cpp

GitHub Copilot

Used 2 references

github.copilot.chat.customUserInstructions

calculator.cpp #file:calculator.cpp

To enhance your `Calculator` class with functions relevant to machine learning algorithms, you can add functions like logarithms, exponentials, and other mathematical operations commonly used in ML. Here are some suggestions:

1. **Natural Logarithm Function**:

```
double Calculator::log(double value) {
    if (value <= 0) return nan("");
    return std::log(value);
}
```

2. **Logarithm Base 10 Function**:

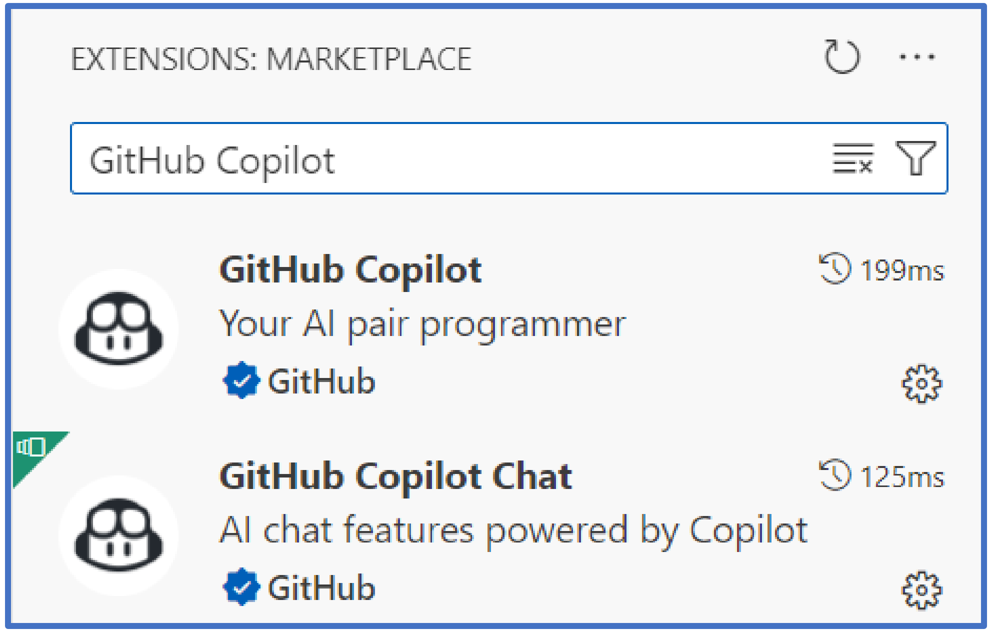
```
double Calculator::log10(double value) {
    if (value <= 0) return nan("");
    return std::log10(value);
}
```

Installing Copilot

You must have an active GitHub Copilot subscription to use Copilot

GitHub Copilot and Copilot Chat are **separate extensions** in VS Code downloadable through the extension marketplace

Learn More at aka.ms/copilotchatVSCode



EXTENSIONS: MARKETPLACE

GitHub Copilot

GitHub Copilot 199ms
Your AI pair programmer
GitHub

GitHub Copilot Chat 125ms
AI chat features powered by Copilot
GitHub

Questions?

Upcoming talks:

Friday @13:30: "What's New in Visual Studio for C++ Developers" – Mryam Girmay & Michael Price

@VisualC
@AlexandraKemperMS
@Sinem__Akinci (double underscore)

visualcpp@microsoft.com

Want to learn more? Watch our 2023 CppCon talk at aka.ms/CppCon2023/VSCode