

- Motivation
- Evolution of C++
- Status Quo
- Header files
- Blank slate?
- The proposal
- What is a module?
- The promise of modules
- The building blocks

Modularized C++

A modern approach to C++ project setup

Alexander Christensen

Copenhagen C/C++ Meetup, August 2022

Contents

- 1 Motivation
- 2 Evolution of C++
- 3 Status Quo
- 4 Header files
- 5 Blank slate?
- 6 The proposal
- 7 What is a module?
- 8 The promise of modules
- 9 The building blocks

Motivation

Evolution of C++

Status Quo

Header files

Blank slate?

The proposal

What is a module?

The promise of modules

The building blocks

Motivation

Motivation
Evolution of C++
Status Quo
Header files
Blank slate?
The proposal
What is a module?
The promise of modules
The building blocks

Evolution of C++

- Motivation
- Evolution of C++
- Status Quo**
- Header files
- Blank slate?
- The proposal
- What is a module?
- The promise of modules
- The building blocks

Status Quo

- Motivation
- Evolution of C++
- Status Quo
- Header files**
- Blank slate?
- The proposal
- What is a module?
- The promise of modules
- The building blocks

Header files are problematic

- Motivation
- Evolution of C++
- Status Quo
- Header files
- Blank slate?**
- The proposal
- What is a module?
- The promise of modules
- The building blocks

Blank slate is impossible

- Motivation
- Evolution of C++
- Status Quo
- Header files
- Blank slate?
- The proposal**
- What is a module?
- The promise of modules
- The building blocks

The modules proposal

- Motivation
- Evolution of C++
- Status Quo
- Header files
- Blank slate?
- The proposal
- What is a module?**
- The promise of modules
- The building blocks

What is a module?

The promise of modules

- That we may rid our projects of header files
- No more include directories
- No more inline functions and methods everywhere
- No more "header-only" libraries
- No more magic macro customizations for building our code
- The preprocessor has no understanding of types, but the compiler has - let's use it!
- Everything will be easier, simpler, and better
- My estimate: ~50% language complexity reduction
- Vastly improved build times
- No loss of expressiveness

The building blocks

We get 4 new "classifications" of files:

- Header unit (temporary solution of creating a BMI from a header file)
 - `g++ -std=c++20 -fmodules-ts -xc++-system-header iostream`
 - Creates BMI in
`./gcm.cache/usr/include/c++/11/iostream.gcm`
- Module interface unit / primary module interface unit
 - This is a translation unit which exports a module
- Module partition / submodule
 - Another translation unit which belongs to a module interface
- Module implementation unit
 - A translation unit which may provide implementations to declarations in module interface