10 Years of Meeting C++

JENS WELLER

Cppcon
The C++ Conference

2022
September 12th-16th

CppCon 2022

10 years of meeting C++

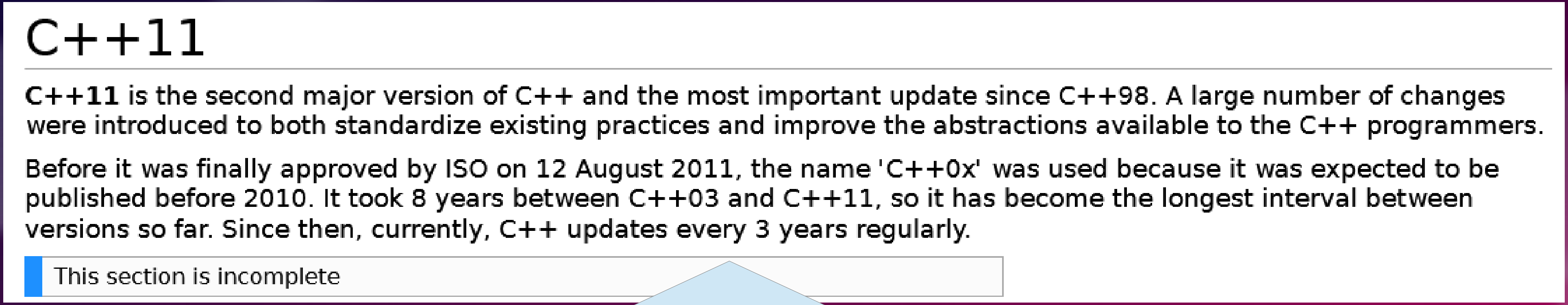
Jens Weller

Past
Present
Future

Questions at the end

10 years ago...

A new C++ standard. Fancy!



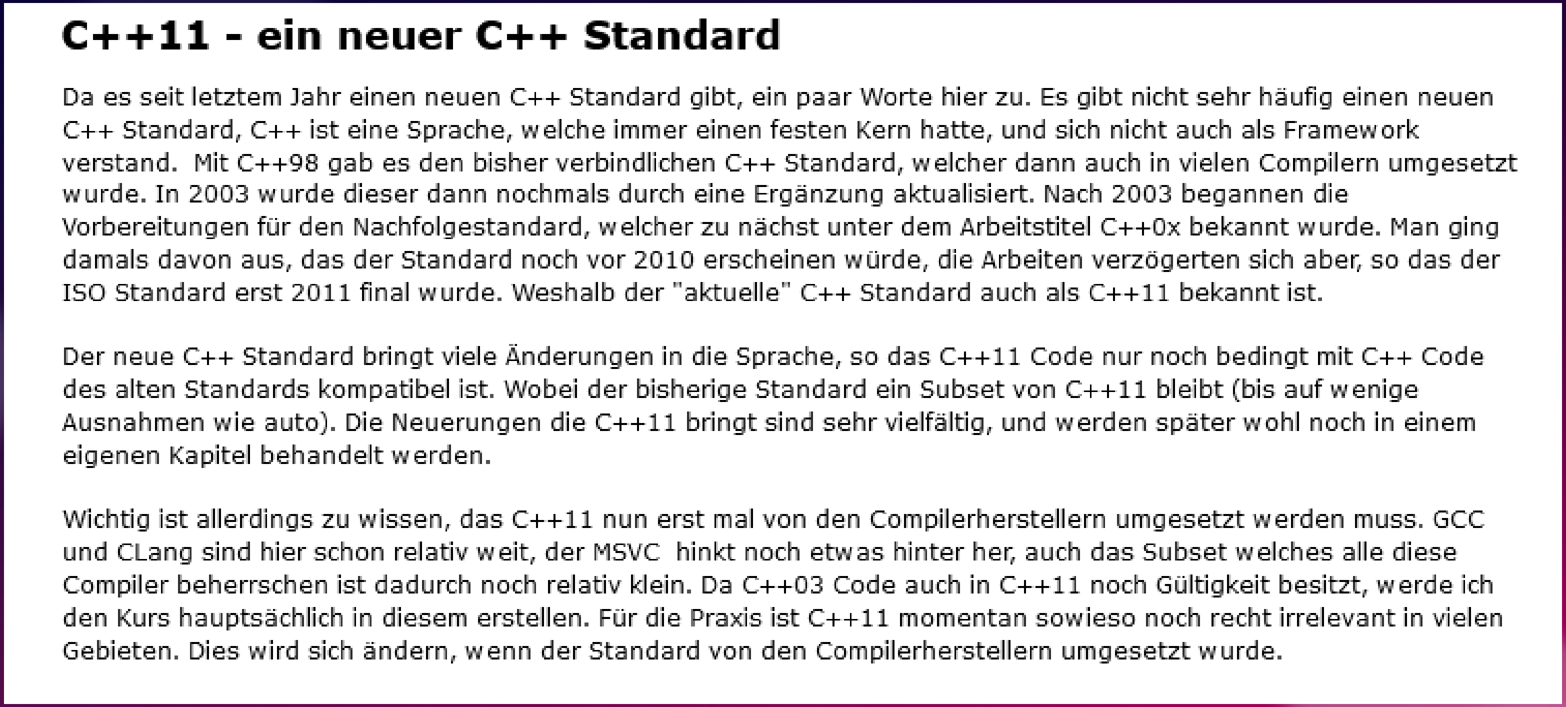
C++11

C++11 is the second major version of C++ and the most important update since C++98. A large number of changes were introduced to both standardize existing practices and improve the abstractions available to the C++ programmers.

Before it was finally approved by ISO on 12 August 2011, the name 'C++0x' was used because it was expected to be published before 2010. It took 8 years between C++03 and C++11, so it has become the longest interval between versions so far. Since then, currently, C++ updates every 3 years regularly.

This section is incomplete

And my opinion in 2012...



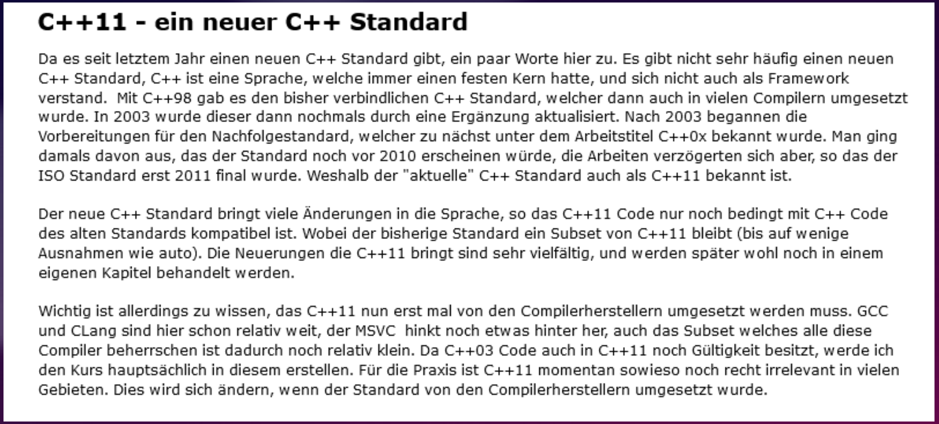
C++11 - ein neuer C++ Standard

Da es seit letztem Jahr einen neuen C++ Standard gibt, ein paar Worte hier zu. Es gibt nicht sehr häufig einen neuen C++ Standard, C++ ist eine Sprache, welche immer einen festen Kern hatte, und sich nicht auch als Framework verstand. Mit C++98 gab es den bisher verbindlichen C++ Standard, welcher dann auch in vielen Compilern umgesetzt wurde. In 2003 wurde dieser dann nochmals durch eine Ergänzung aktualisiert. Nach 2003 begannen die Vorbereitungen für den Nachfolgestandard, welcher zu nächst unter dem Arbeitstitel C++0x bekannt wurde. Man ging damals davon aus, das der Standard noch vor 2010 erscheinen würde, die Arbeiten verzögerten sich aber, so das der ISO Standard erst 2011 final wurde. Weshalb der "aktuelle" C++ Standard auch als C++11 bekannt ist.

Der neue C++ Standard bringt viele Änderungen in die Sprache, so das C++11 Code nur noch bedingt mit C++ Code des alten Standards kompatibel ist. Wobei der bisherige Standard ein Subset von C++11 bleibt (bis auf wenige Ausnahmen wie auto). Die Neuerungen die C++11 bringt sind sehr vielfältig, und werden später wohl noch in einem eigenen Kapitel behandelt werden.

Wichtig ist allerdings zu wissen, das C++11 nun erst mal von den Compilerherstellern umgesetzt werden muss. GCC und CLang sind hier schon relativ weit, der MSVC hinkt noch etwas hinter her, auch das Subset welches alle diese Compiler beherrschen ist dadurch noch relativ klein. Da C++03 Code auch in C++11 noch Gültigkeit besitzt, werde ich den Kurs hauptsächlich in diesem erstellen. Für die Praxis ist C++11 momentan sowieso noch recht irrelevant in vielen Gebieten. Dies wird sich ändern, wenn der Standard von den Compilerherstellern umgesetzt wurde.

And my opinion in 2012...



C++11 - ein neuer C++ Standard

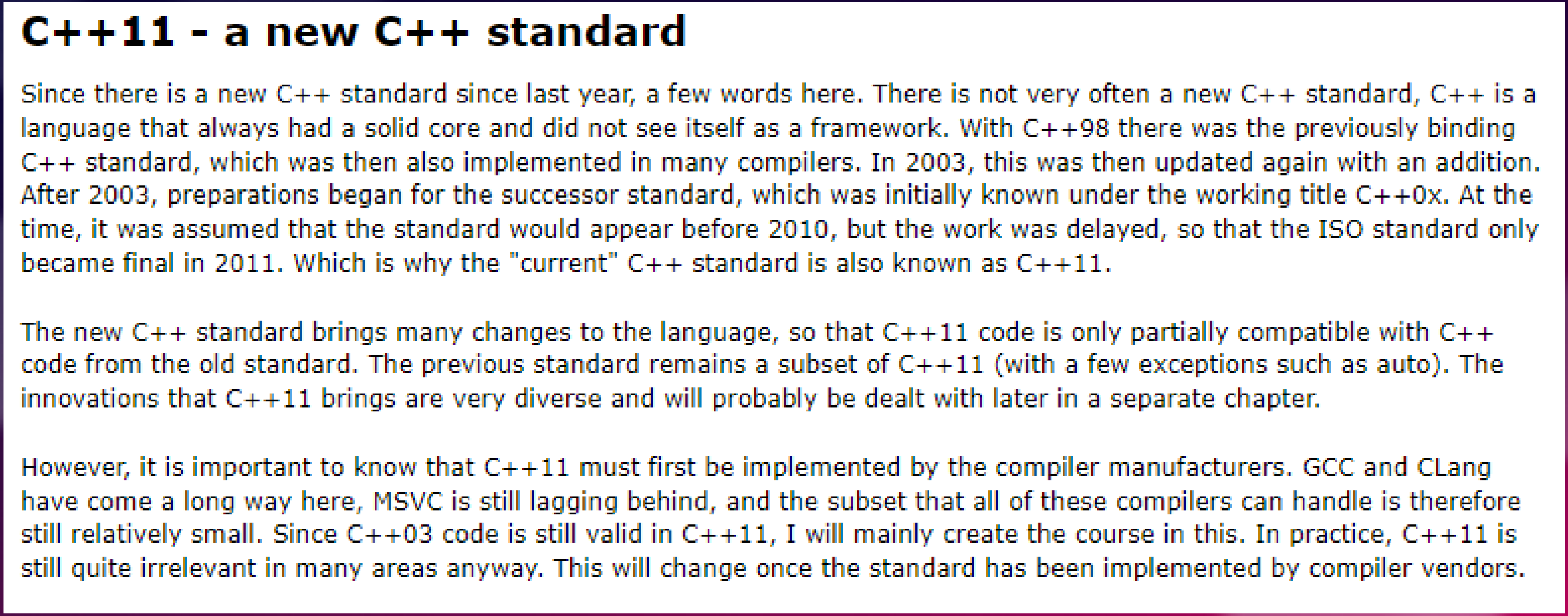
Da es seit letztem Jahr einen neuen C++ Standard gibt, ein paar Worte hier zu. Es gibt nicht sehr häufig einen neuen C++ Standard, C++ ist eine Sprache, welche immer einen festen Kern hatte, und sich nicht auch als Framework verstand. Mit C++98 gab es den bisher verbindlichen C++ Standard, welcher dann auch in vielen Compilern umgesetzt wurde. In 2003 wurde dieser dann nochmals durch eine Ergänzung aktualisiert. Nach 2003 begannen die Vorbereitungen für den Nachfolgestandard, welcher zu nächst unter dem Arbeitstitel C++0x bekannt wurde. Man ging damals davon aus, das der Standard noch vor 2010 erscheinen würde, die Arbeiten verzögerten sich aber, so das der ISO Standard erst 2011 final wurde. Weshalb der "aktuelle" C++ Standard auch als C++11 bekannt ist.

Der neue C++ Standard bringt viele Änderungen in die Sprache, so das C++11 Code nur noch bedingt mit C++ Code des alten Standards kompatibel ist. Wobei der bisherige Standard ein Subset von C++11 bleibt (bis auf wenige Ausnahmen wie auto). Die Neuerungen die C++11 bringt sind sehr vielfältig, und werden später wohl noch in einem eigenen Kapitel behandelt werden.

Wichtig ist allerdings zu wissen, das C++11 nun erst mal von den Compilerherstellern umgesetzt werden muss. GCC und CLang sind hier schon relativ weit, der MSVC hinkt noch etwas hinter her, auch das Subset welches alle diese Compiler beherrschen ist dadurch noch relativ klein. Da C++03 Code auch in C++11 noch Gültigkeit besitzt, werde ich den Kurs hauptsächlich in diesem erstellen. Für die Praxis ist C++11 momentan sowieso noch recht irrelevant in vielen Gebieten. Dies wird sich ändern, wenn der Standard von den Compilerherstellern umgesetzt wurde.

- First Paragraph
  - How did we get here?
- Second Paragraph
  - Its new, but compatible to C++03
  - May add a chapter later about it
- Third Paragraph
  - C++11 must first be implemented by the compiler vendors first
  - In practice, C++11 is still quite irrelevant in many areas anyway.

English translation



C++11 - a new C++ standard

Since there is a new C++ standard since last year, a few words here. There is not very often a new C++ standard, C++ is a language that always had a solid core and did not see itself as a framework. With C++98 there was the previously binding C++ standard, which was then also implemented in many compilers. In 2003, this was then updated again with an addition. After 2003, preparations began for the successor standard, which was initially known under the working title C++0x. At the time, it was assumed that the standard would appear before 2010, but the work was delayed, so that the ISO standard only became final in 2011. Which is why the "current" C++ standard is also known as C++11.

The new C++ standard brings many changes to the language, so that C++11 code is only partially compatible with C++ code from the old standard. The previous standard remains a subset of C++11 (with a few exceptions such as auto). The innovations that C++11 brings are very diverse and will probably be dealt with later in a separate chapter.

However, it is important to know that C++11 must first be implemented by the compiler manufacturers. GCC and CLang have come a long way here, MSVC is still lagging behind, and the subset that all of these compilers can handle is therefore still relatively small. Since C++03 code is still valid in C++11, I will mainly create the course in this. In practice, C++11 is still quite irrelevant in many areas anyway. This will change once the standard has been implemented by compiler vendors.

C++Now 2012

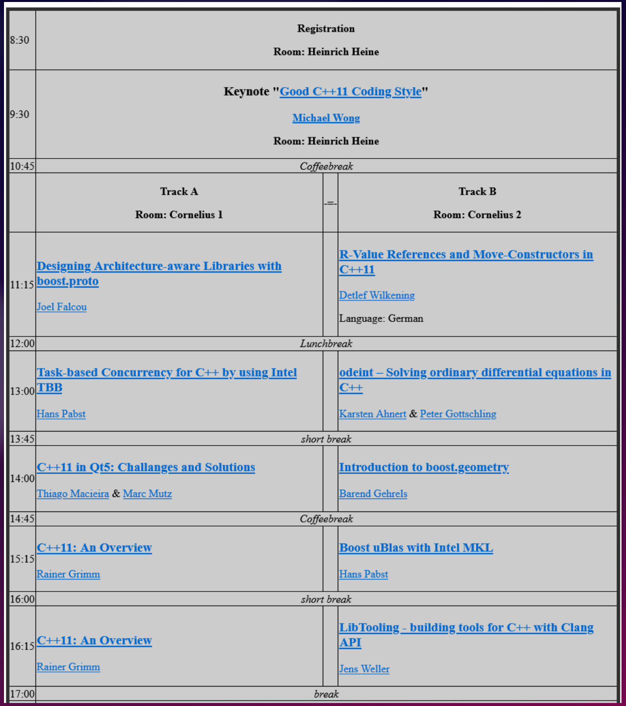
C++Now 2012

- Lots of C++ knowledge
- Less 150 folks there
  - How do we reach the rest?
- New standard
  - C++11 and beyond?
- The best weather

C++Now 2012 → Meeting C++ 2012

- Got lots of support from Europeans for the idea
- Had started the C++ UG Düsseldorf in December 2011
- Kinda talked my self into it

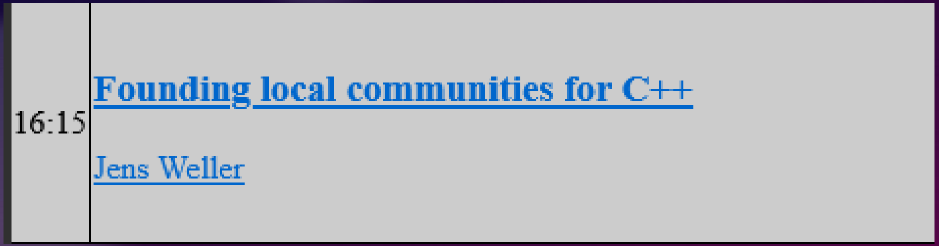
Meeting C++ 2012



8:30	**Registration** **Room: Heinrich Heine**	
9:30	**Keynote "Good C++11 Coding Style"** Michael Wong **Room: Heinrich Heine**	
10:45	*Coffeebreak*	
	Track A **Room: Cornelius 1**	**Track B** **Room: Cornelius 2**
11:15	**Designing Architecture-aware Libraries with boost.proto** Joel Falcou	**R-Value References and Move-Constructors in C++11** Detlef Wilkening Language: German
12:00	*Lunchbreak*	
13:00	**Task-based Concurrency for C++ by using Intel TBB** Hans Pabst	**odeint – Solving ordinary differential equations in C++** Karsten Ahnert & Peter Gottschling
13:45	*short break*	
14:00	**C++11 in Qt5: Challanges and Solutions** Thiago Macieira & Marc Mutz	**Introduction to boost.geometry** Barend Gehrels
14:45	*Coffeebreak*	
15:15	**C++11: An Overview** Rainer Grimm	**Boost uBlas with Intel MKL** Hans Pabst
16:00	*short break*	
16:15	**C++11: An Overview** Rainer Grimm	**LibTooling - building tools for C++ with Clang API** Jens Weller
17:00	*break*	

- Program
  - C++11 / boost heavy
  - Some Qt
- Results
  - Big success
  - german ISO C++ saved
- Proof of concept

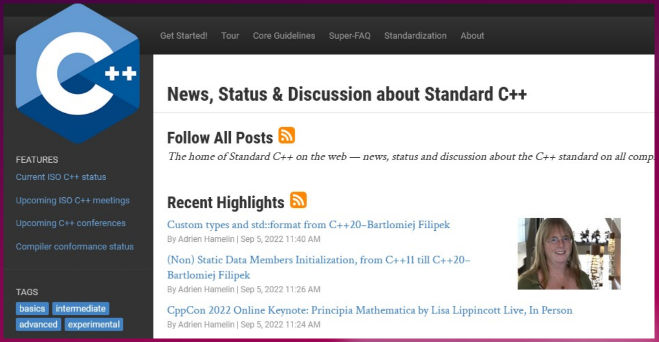
The conference as a motivational tool



16:15 [Founding local communities for C++](#)

[Jens Weller](#)

- Motivating folks to start their own User Groups
  - So that we reach more folks
- Connecting the audience
- After this talk Michael Wong gave an introduction to isocpp.org



Get Started! Tour Core Guidelines Super-FAQ Standardization About

FEATURES
Current ISO C++ status
Upcoming ISO C++ meetings
Upcoming C++ conferences
Compiler conformance status

TAGS
basics intermediate advanced experimental

News, Status & Discussion about Standard C++

Follow All Posts

The home of Standard C++ on the web — news, status and discussion about the C++ standard on all comp

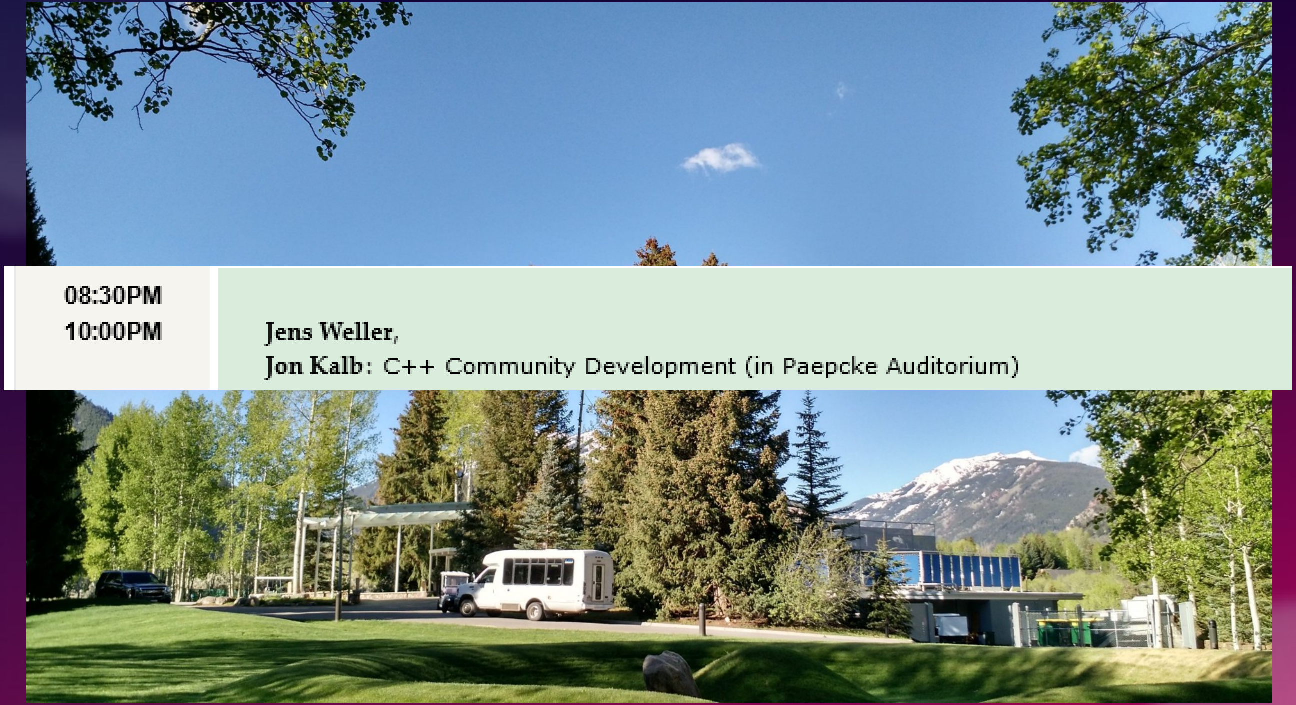
Recent Highlights

Custom types and std::format from C++20–Bartlomiej Filipek
By Adrien Hamelin | Sep 5, 2022 11:40 AM

(Non) Static Data Members Initialization, from C++11 till C++20–Bartlomiej Filipek
By Adrien Hamelin | Sep 5, 2022 11:26 AM

CppCon 2022 Online Keynote: Principia Mathematica by Lisa Lippincott Live, In Person
By Adrien Hamelin | Sep 5, 2022 11:24 AM

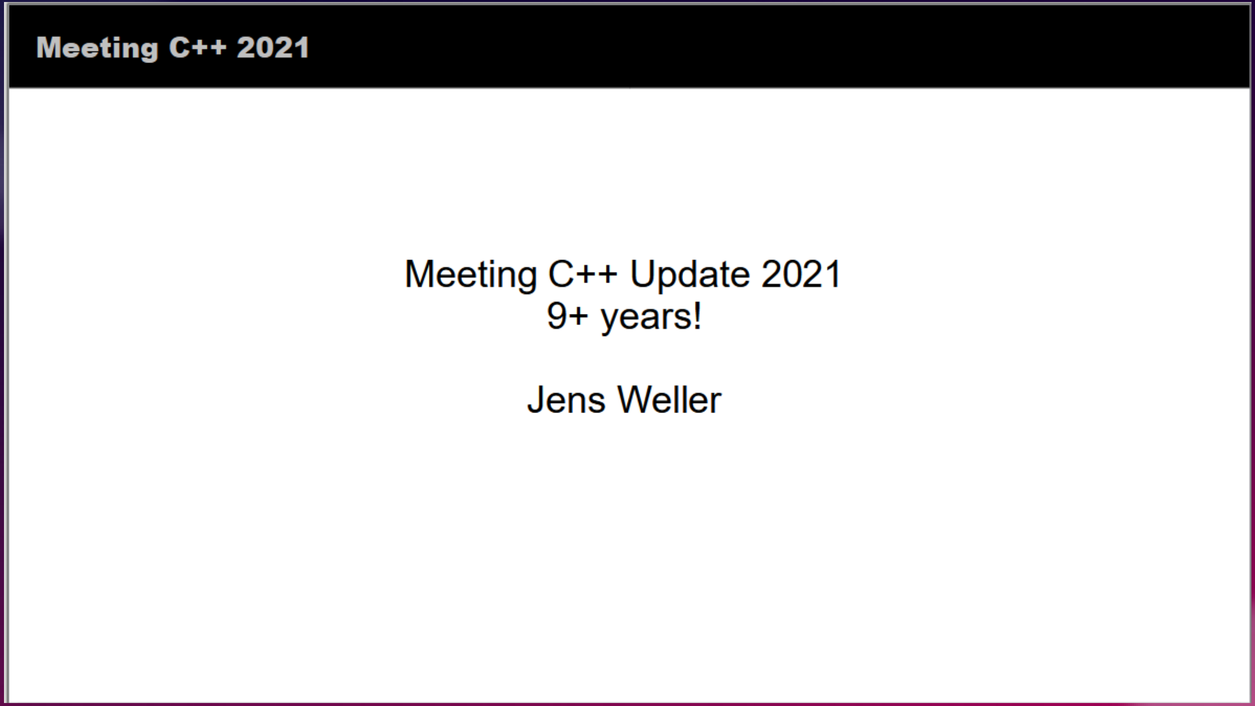
C++now 2013 / Meeting C++ 2013



08:30PM 10:00PM	**Jens Weller,** **Jon Kalb**: C++ Community Development (in Paepcke Auditorium)

Meeting C++ evolves into a Plattform
2013 - 2014

Meeting C++ update

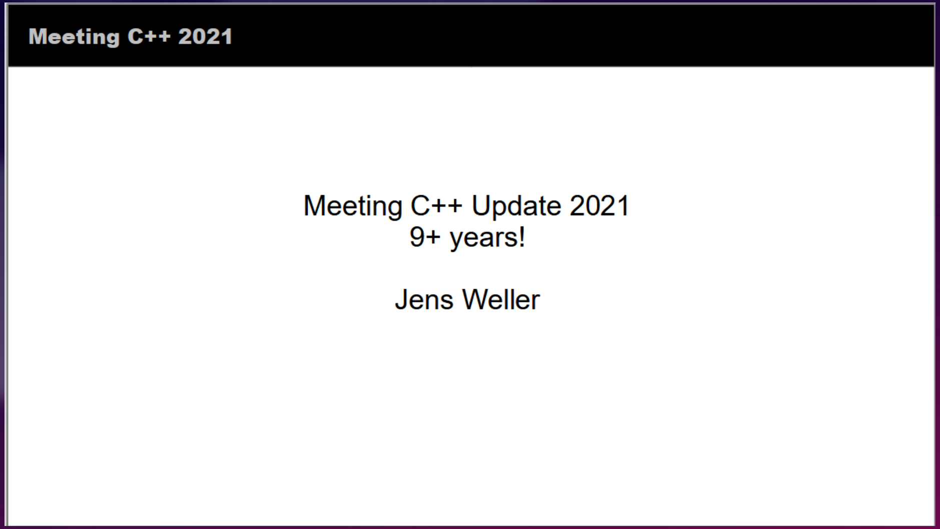


Meeting C++ 2021

Meeting C++ Update 2021
9+ years!

Jens Weller

Meeting C++ update

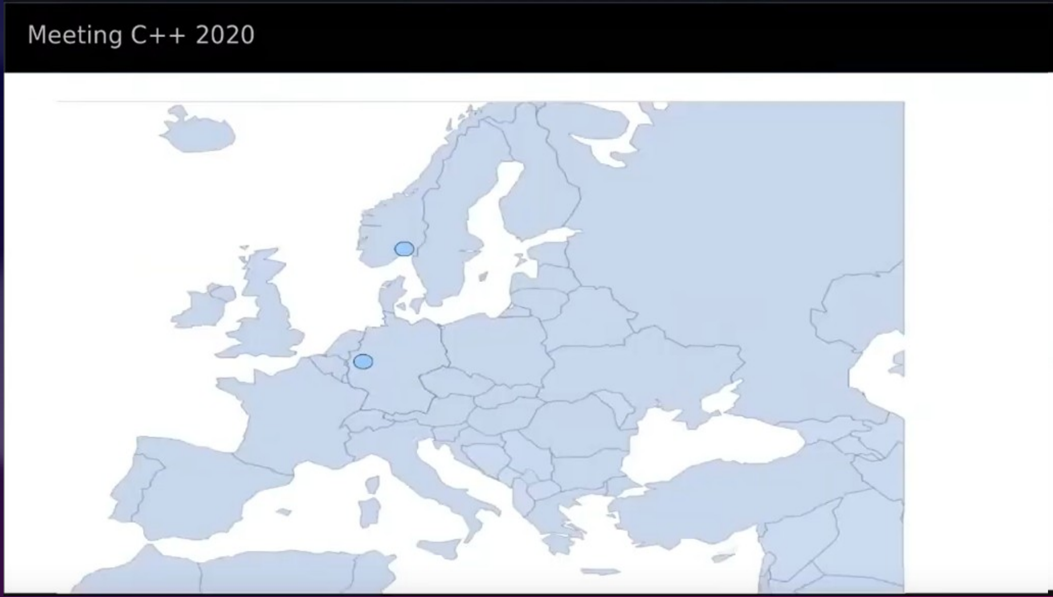

Meeting C++ 2021

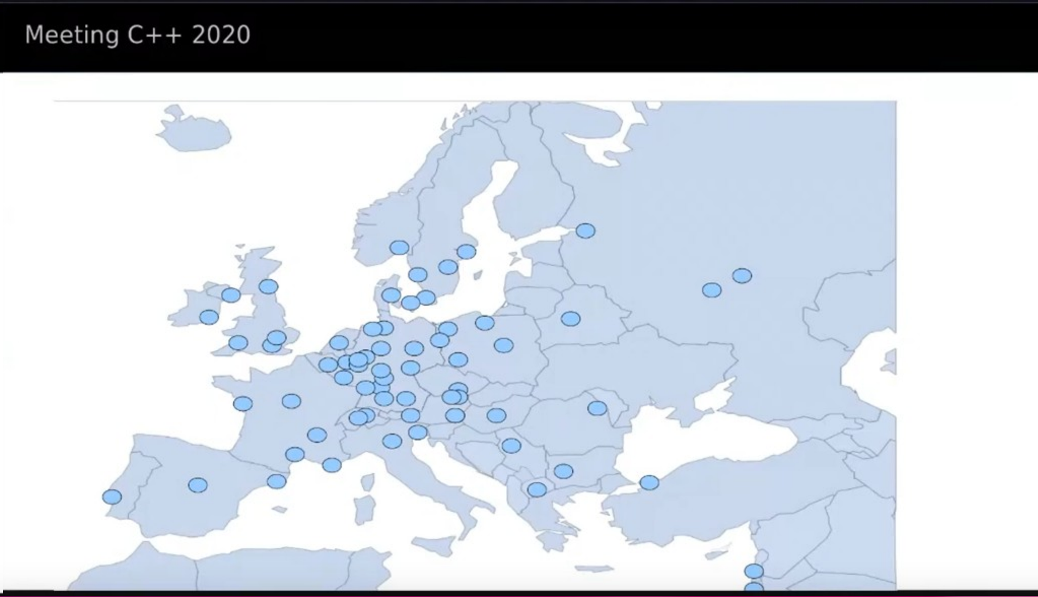
Meeting C++ Update 2021
9+ years!

Jens Weller

- Meeting C++ is funded by the conference
  - This *is* the room that pays me
- Community Manager
  - Giving feedback
- Motivating folks to join

C++ User Groups 2011 → 2019





- 2011

- 2019

And so the golden age begins...

Cppcon
The C++ Conference

The golden Age 2014 - 2019

Meeting C++

Cppcon
The C++ Conference

- Stable C++ Committee
  - Every 3 years a new standard
  - C++11 → C++14 → C++17 → C++20
- C++ Community grows
- Lots of C++ content
  - Blogs
  - CppCast
  - Videos

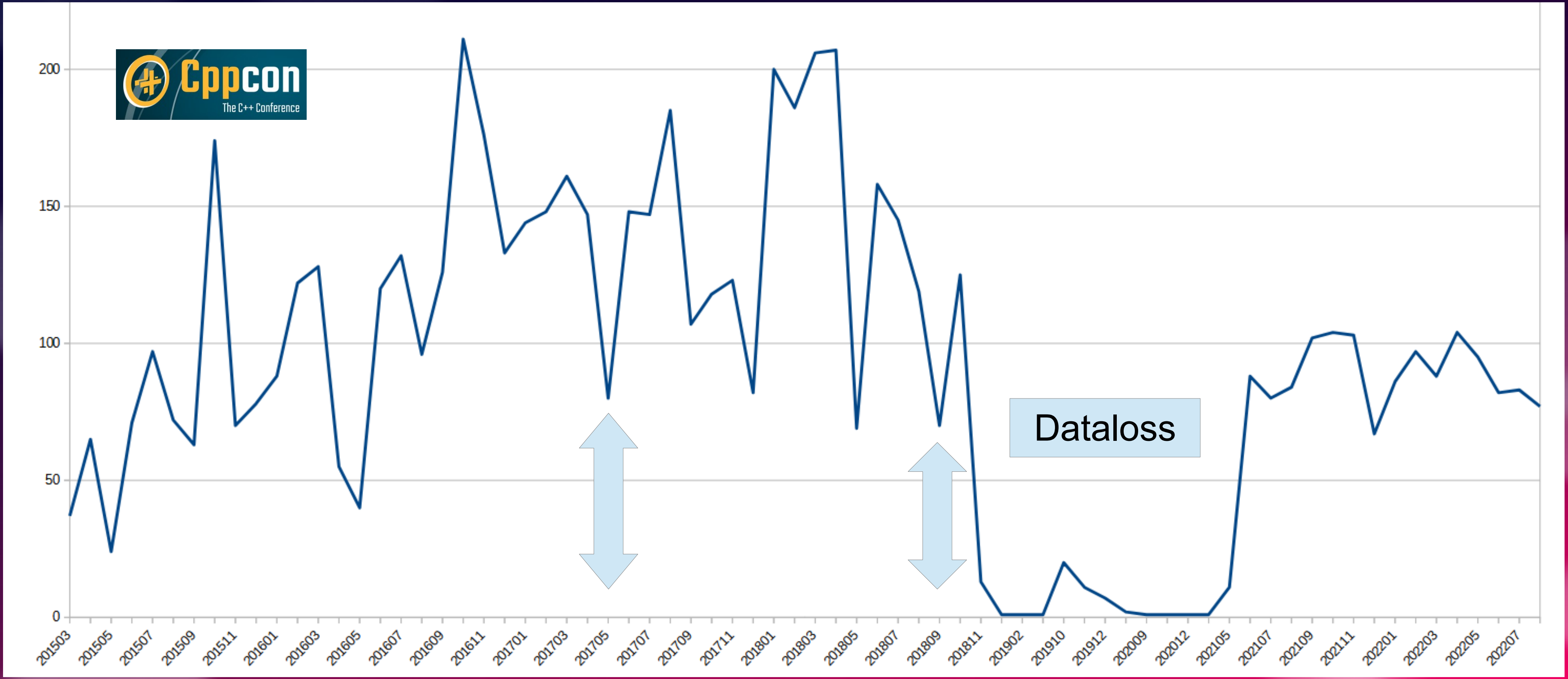
These 6 years lead to where we are at present

What do we know about the golden age?

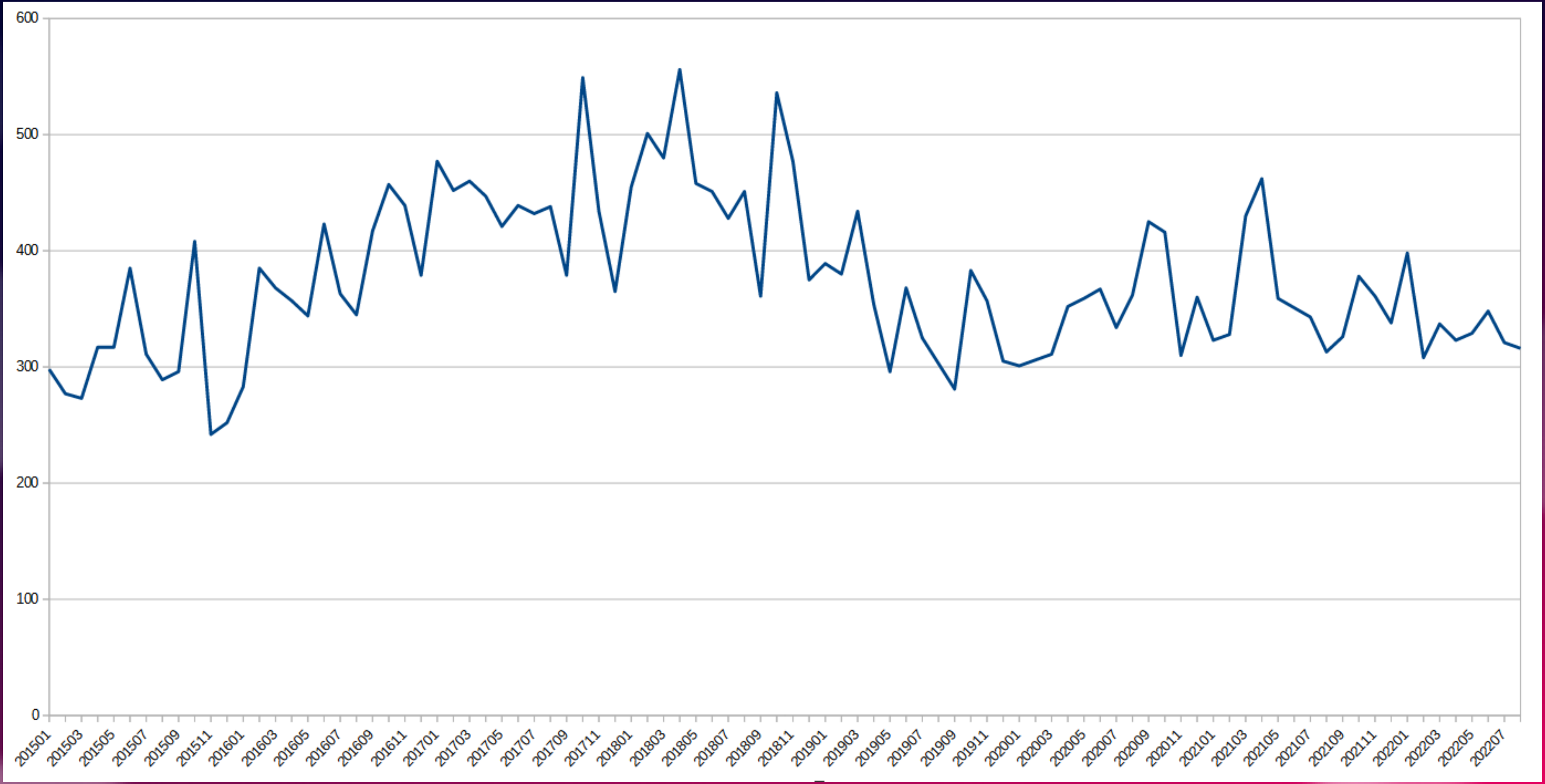
Some statistics on C++ content

- C++ content as
  - Blogs, videos
- Posted by Meeting C++
- Datasource
  - Database of my RssReader tool

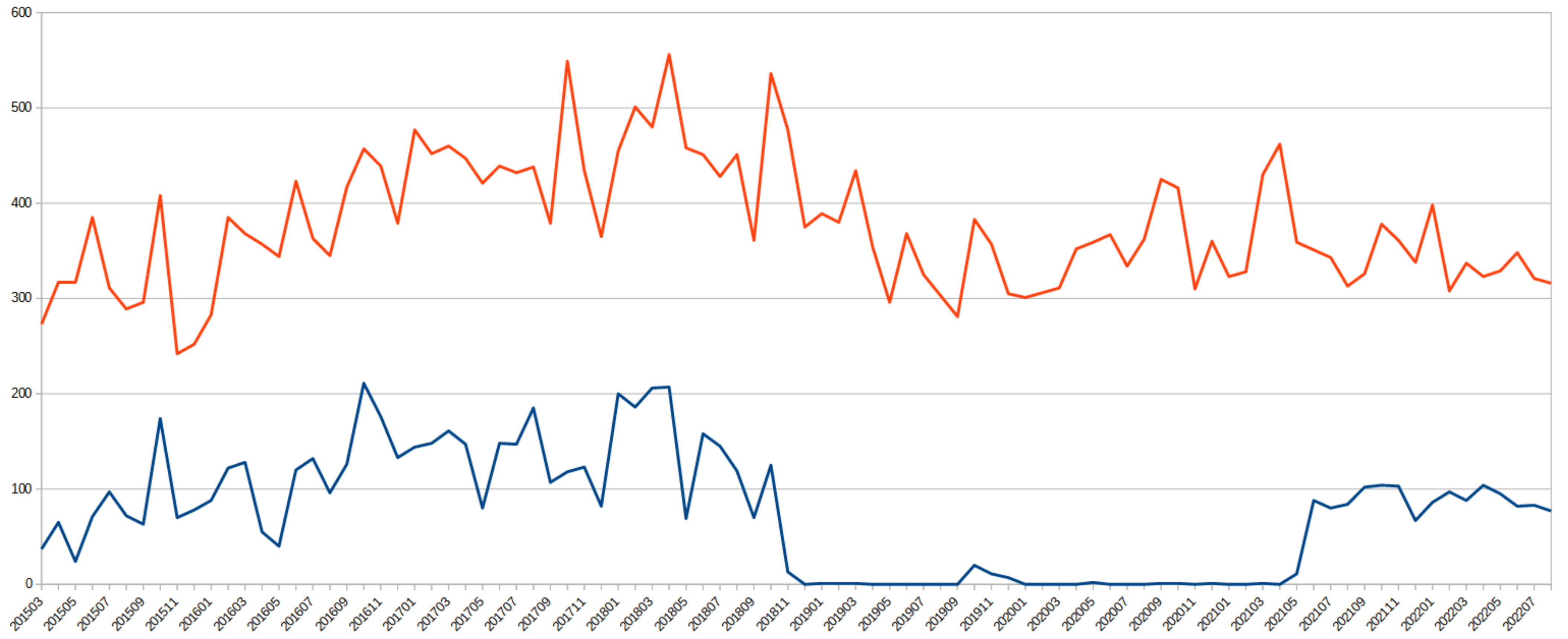
Monthly postings by Meeting C++



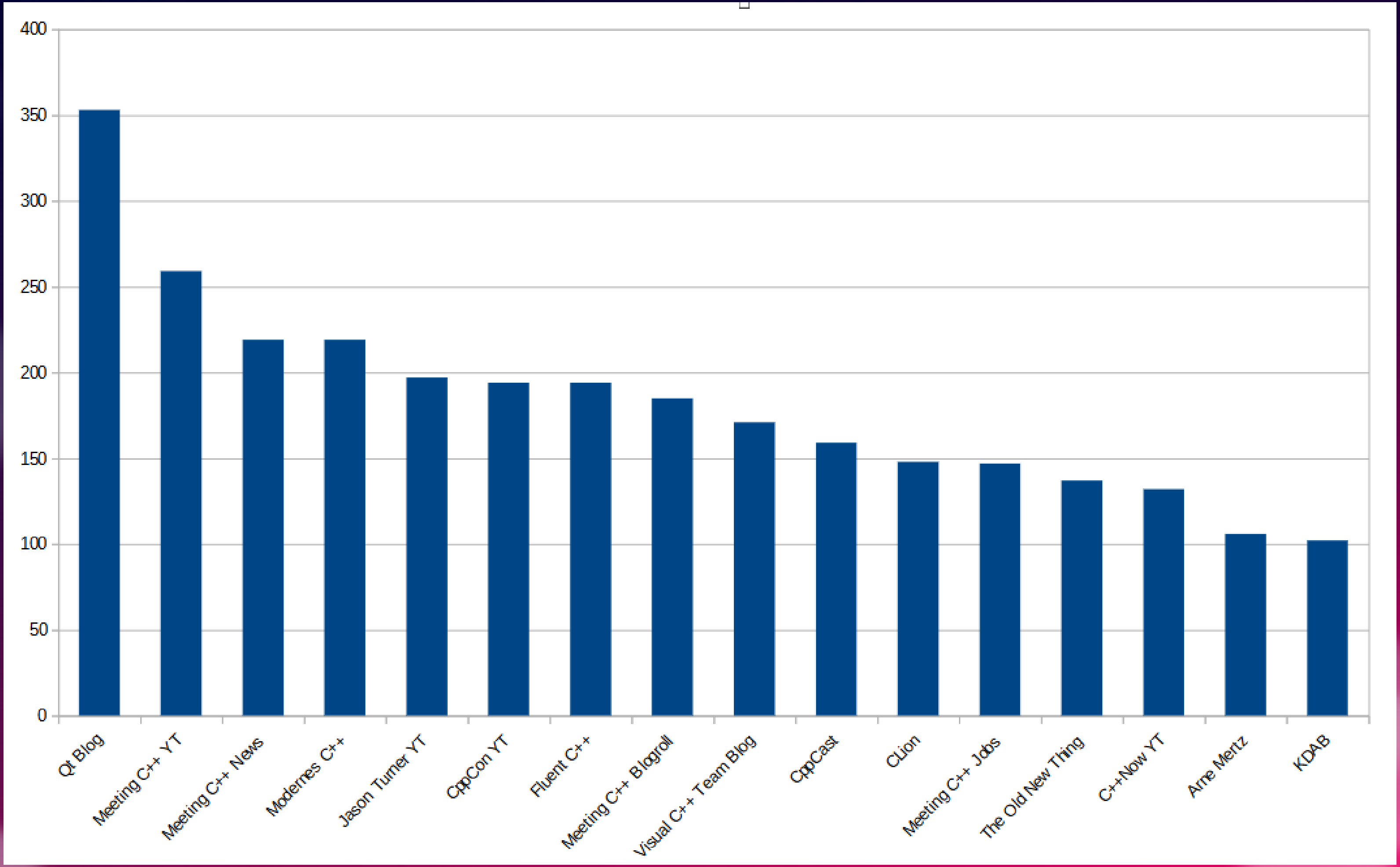
Known posts to my RSS Tooling



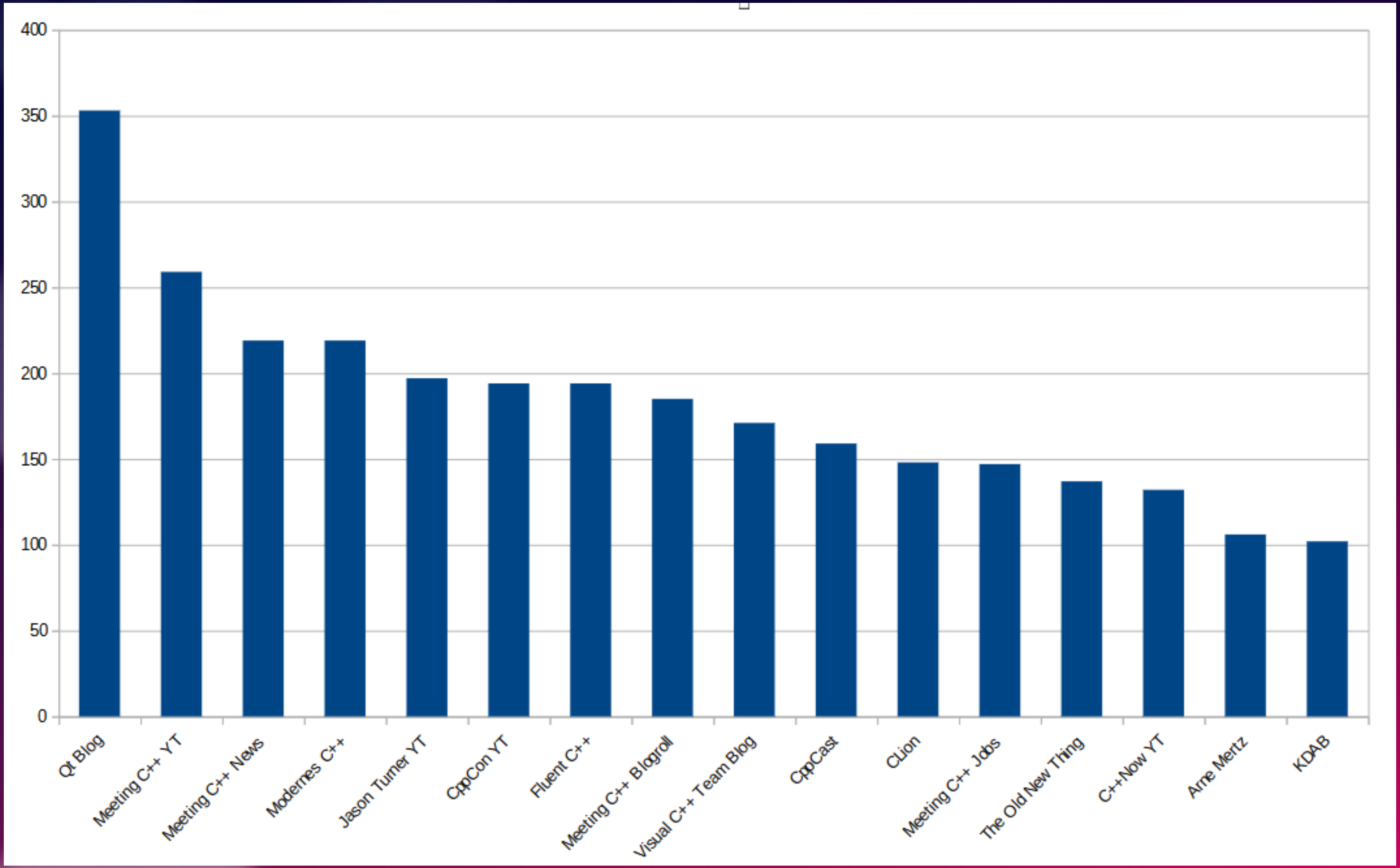
Combining both



Top 16 RSS Sources



Top 16 RSS Sources

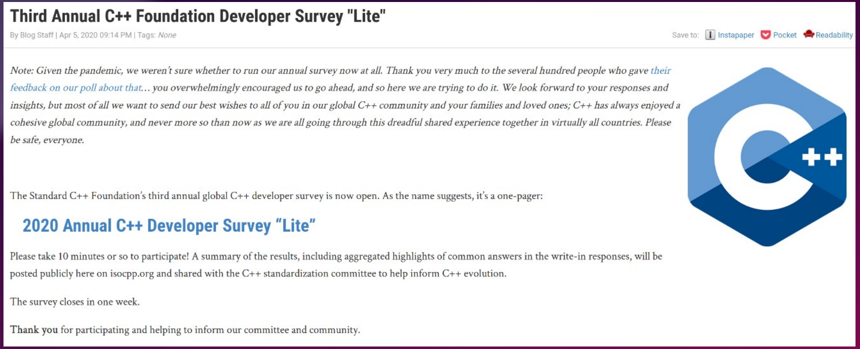


- Qt Blog 353
- Meeting C++ YT 259
- Meeting C++ News 219
- Modernes C++ 219
- Jason Turner YT 197
- CppCon YT 194
- Fluent C++ 194
- Meeting C++ Blogroll 185
- Visual C++ Team Blog 171
- CppCast 159
- CLion 148
- Meeting C++ Jobs 147
- The Old New Thing 137
- C++Now YT 132
- Arne Mertz 106
- KDAB 102

Annual Surveys

- ISOCPP
  - 2018?

- Jetbrains
  - 2017



Third Annual C++ Foundation Developer Survey "Lite"

By Blog Staff | Apr 5, 2020 09:14 PM | Tags: *None*

Save to: Instapaper Pocket Readability

Note: Given the pandemic, we weren't sure whether to run our annual survey now at all. Thank you very much to the several hundred people who gave their feedback on our poll about that… you overwhelmingly encouraged us to go ahead, and so here we are trying to do it. We look forward to your responses and insights, but most of all we want to send our best wishes to all of you in our global C++ community and your families and loved ones; C++ has always enjoyed a cohesive global community, and never more so than now as we are all going through this dreadful shared experience together in virtually all countries. Please be safe, everyone.

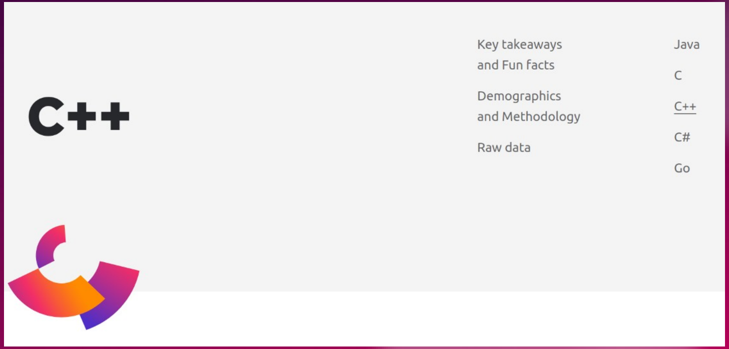
The Standard C++ Foundation's third annual global C++ developer survey is now open. As the name suggests, it's a one-pager:

2020 Annual C++ Developer Survey "Lite"

Please take 10 minutes or so to participate! A summary of the results, including aggregated highlights of common answers in the write-in responses, will be posted publicly here on isocpp.org and shared with the C++ standardization committee to help inform C++ evolution.

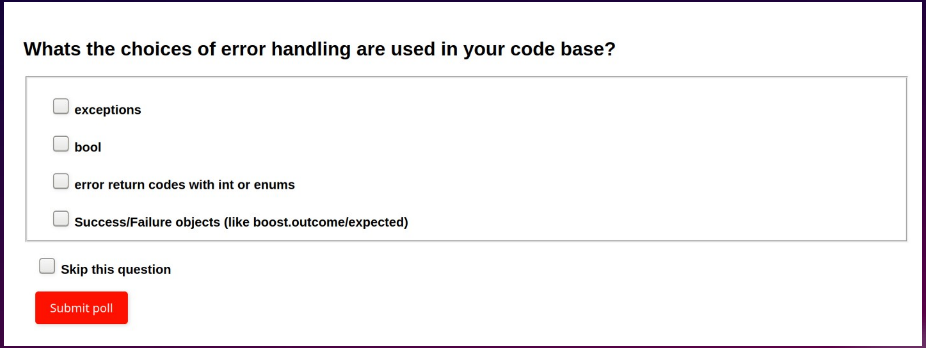
The survey closes in one week.

Thank you for participating and helping to inform our committee and community.



Meeting C++ Community Survey

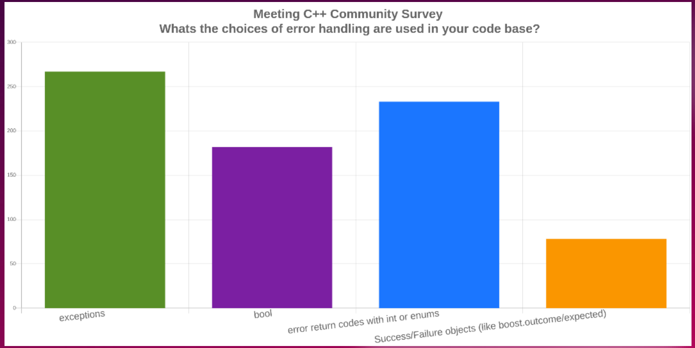
- 2020
- Continuous Survey
- Questions
  - Single & Multiple choice



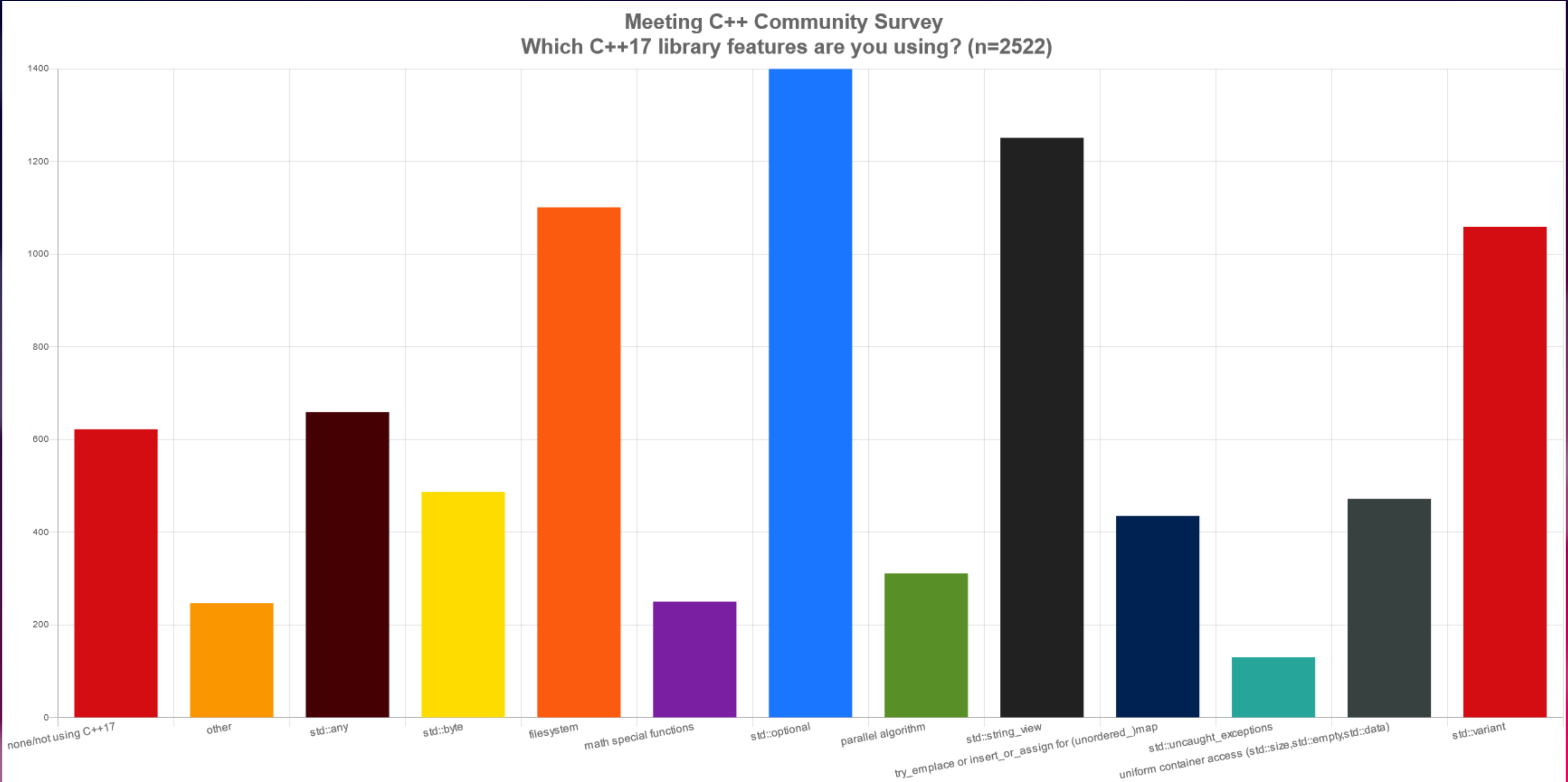
Whats the choices of error handling are used in your code base?

- ☐ exceptions
- ☐ bool
- ☐ error return codes with int or enums
- ☐ Success/Failure objects (like boost.outcome/expected)

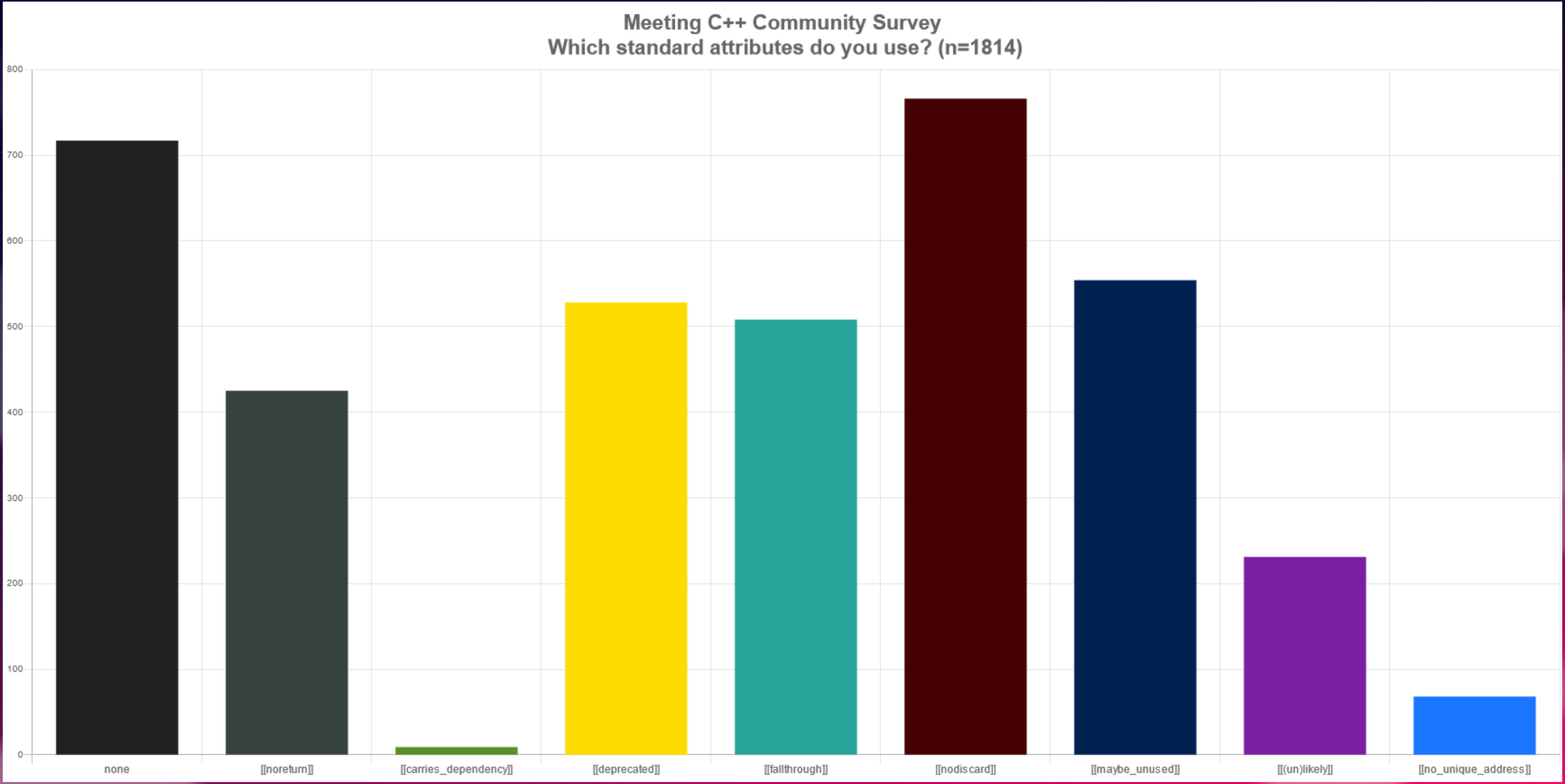
☐ Skip this question



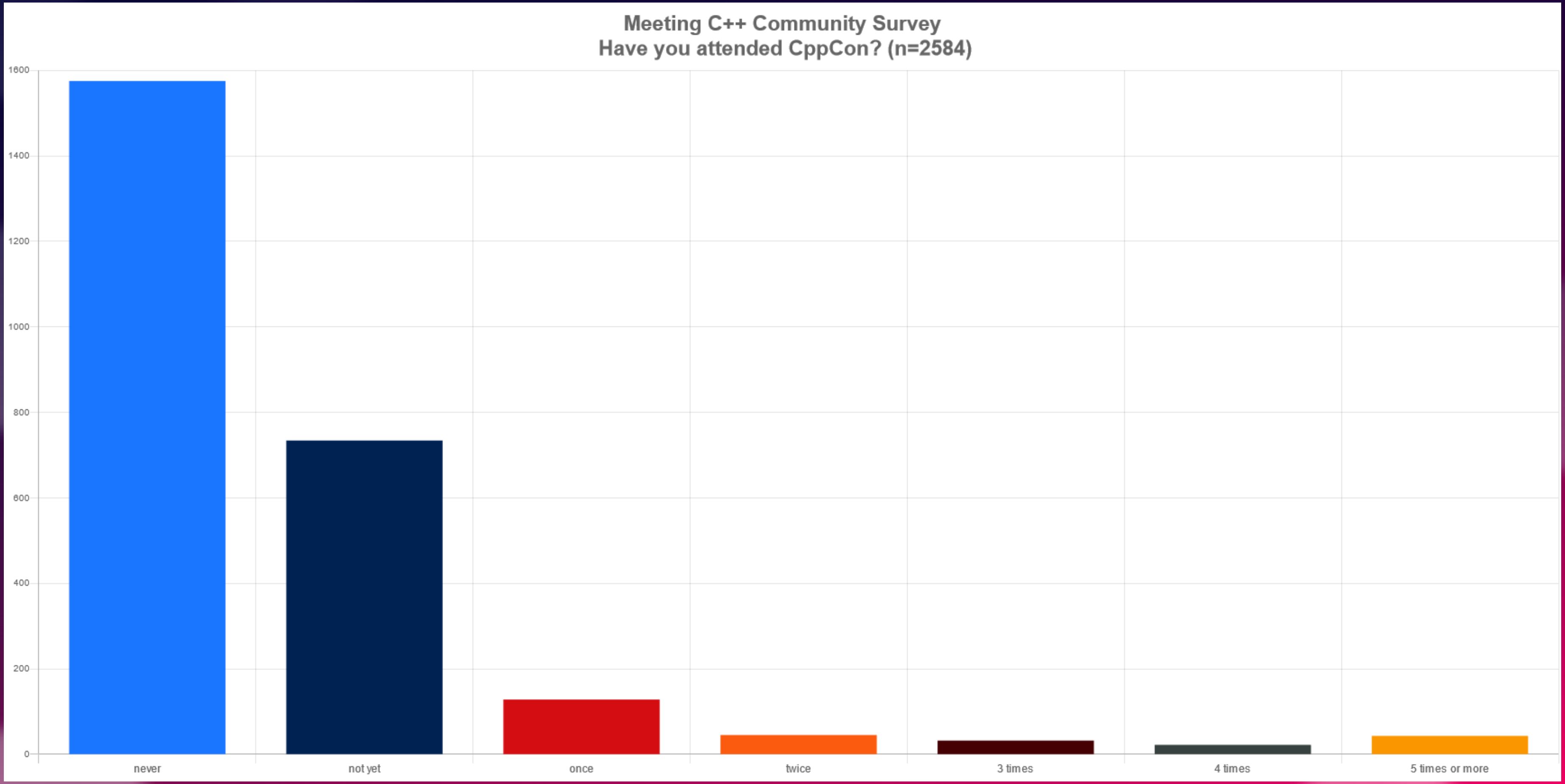
Which C++17 library features are you using?



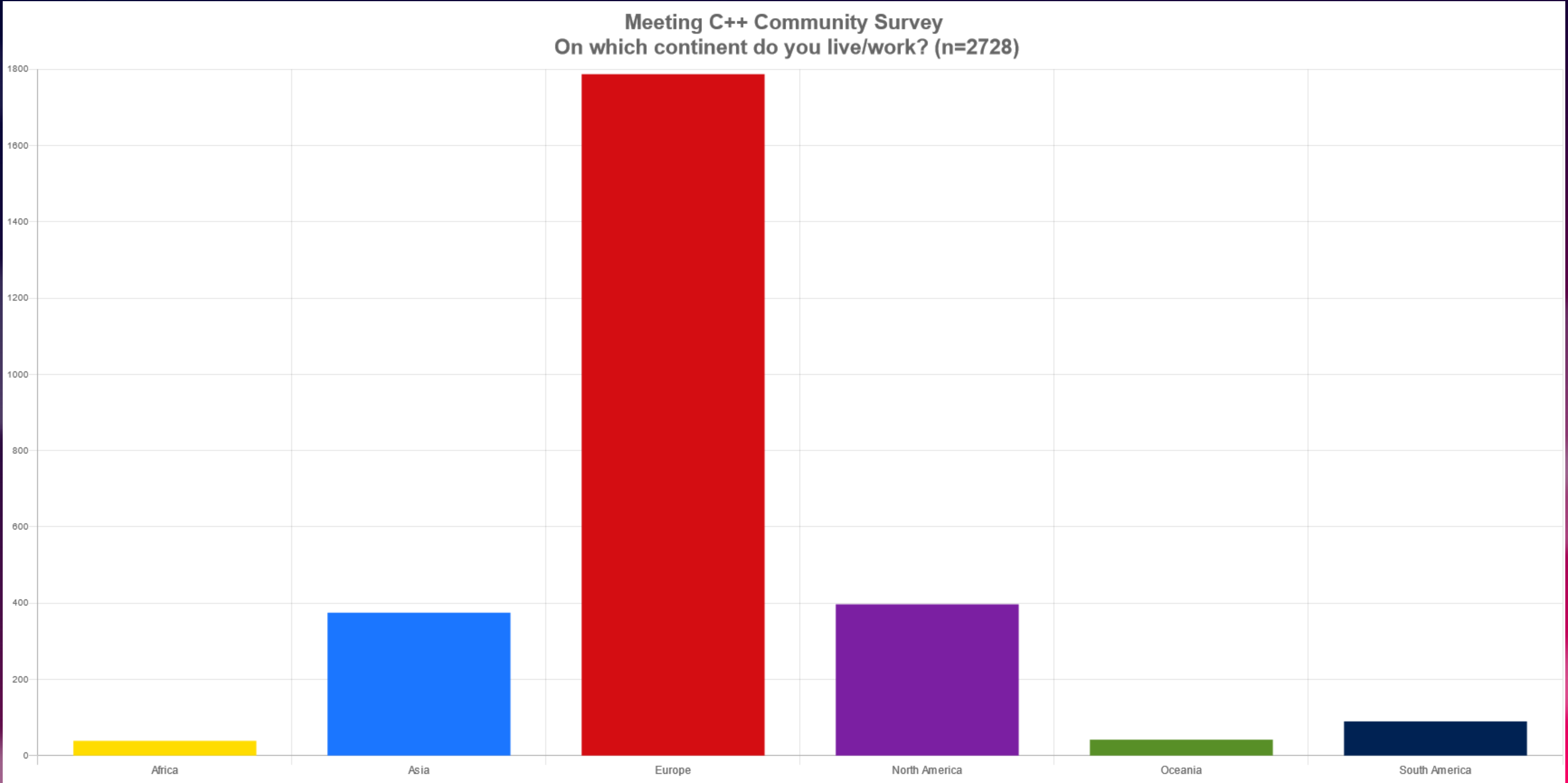
Which standard attributes do you use?



Have you attended CppCon?

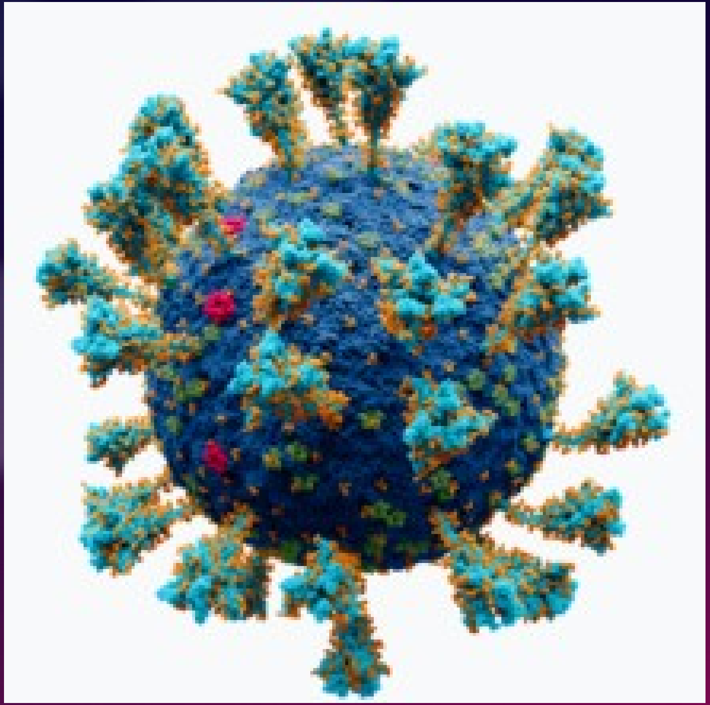


Its a little bit biased though



Which brings us to the present
2020 - today

Pandemic

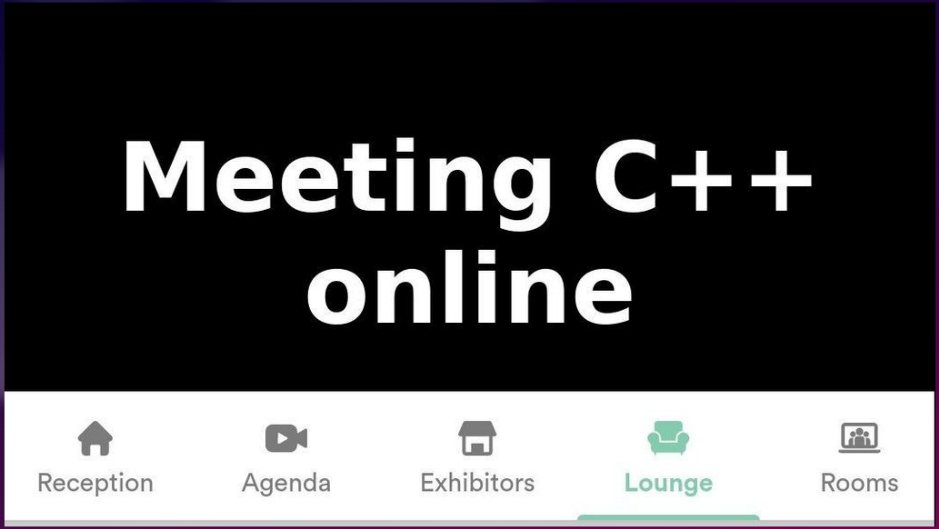


- With 2020 things changed
- The golden age ended
  - Conferences
  - Committee

2020 from Meeting C++ perspective

- Uncertain
- Having to plan for multiple cenarios from the start
- On-site limitations due to only one room fitting restrictions
- Lockdown
  - On-site cancelation late October

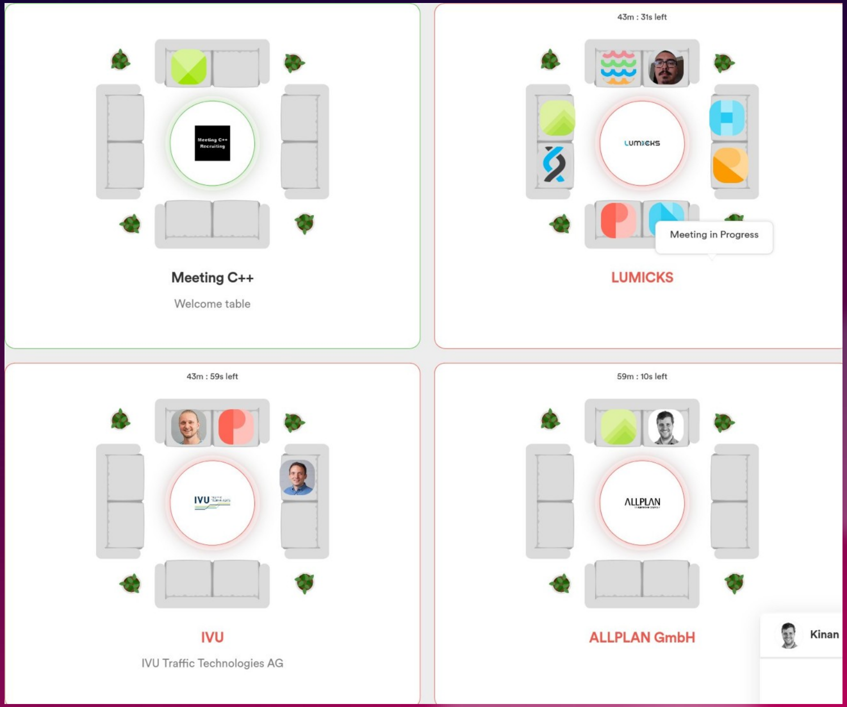
New things also came along



- Started as an experiment for online
- Over 50 meetings today
- Various formats
  - Fairs (jobs / books / tools)
  - AMAs
- https://meetup.meetingcpp.com

Open content sessions

- C++ Job fairs
- Book & Tool fair
- Lightning talks
- AMAs



Hybrid Events – the future?

- CppCon 2021/22
- Meeting C++ 2022
  - Prerecordings
  - Live stream main session
  - Some talks online live too
- Accu 2022

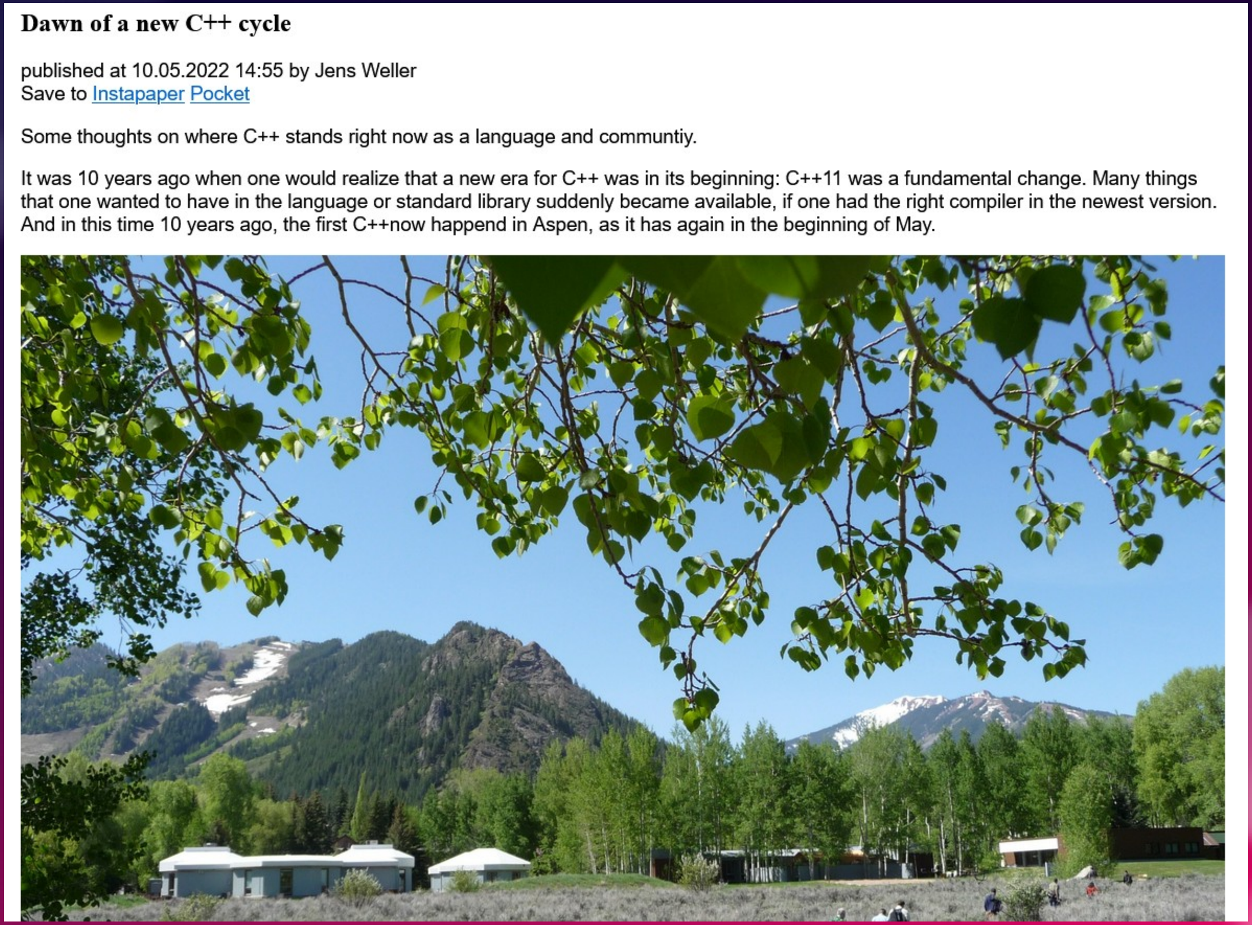
- Hybrid
  - Both onsite & online
  - Live Streams
- Lots of challenges
  - Cost overhead
  - Internet connection
  - Extra effort
  - 3 hours in meetings to speak online at CppCon ‘22.
  - Lots of communication

As conferences return

- What is still viable for you onsite?
  - Attendance is 1/3 to 2019 in 2022
  - “Minimal viable conference”
    - And you build from this again
    - Was the same in 2014 – 2019 cycle

And so a new cycle begins

Dawn of a new C++ cycle



Dawn of a new C++ cycle

published at 10.05.2022 14:55 by Jens Weller
Save to Instapaper Pocket

Some thoughts on where C++ stands right now as a language and communtiy.

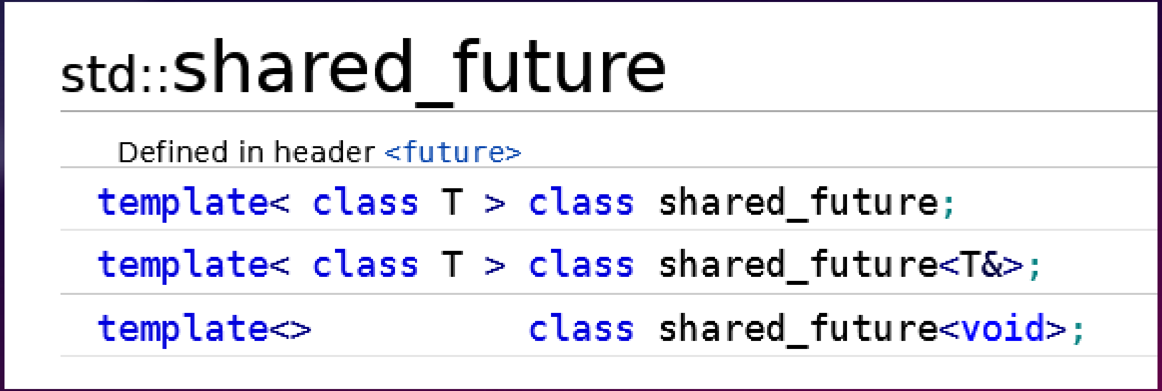
It was 10 years ago when one would realize that a new era for C++ was in its beginning: C++11 was a fundamental change. Many things that one wanted to have in the language or standard library suddenly became available, if one had the right compiler in the newest version. And in this time 10 years ago, the first C++now happend in Aspen, as it has again in the beginning of May.

All the same things

- C++20 is the new C++11
  - Implementations need to catch up at first
  - Lots of new things in the standards
  - Unknown best practice (Modules especially)
- C++23 is what C++14 was to C++11
  - Bugfix of the “big new standard”

C++26 will be the new C++17

Looking ahead



std::shared_future

Defined in header <future>

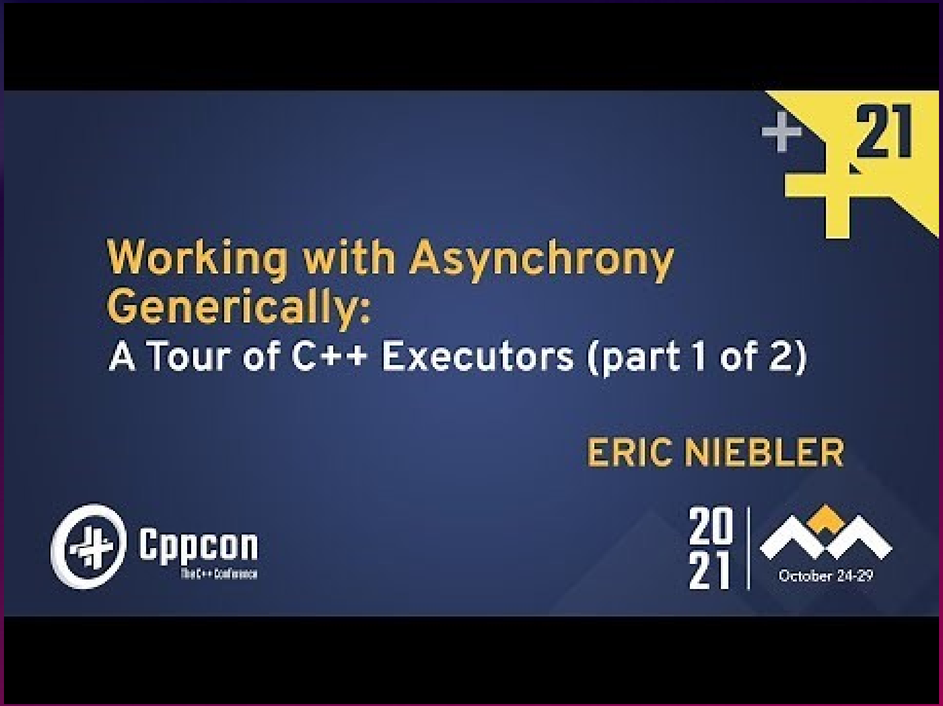
```
template< class T > class shared_future;
template< class T > class shared_future<T&>;
template<>          class shared_future<void>;
```

The two most fundamental issues

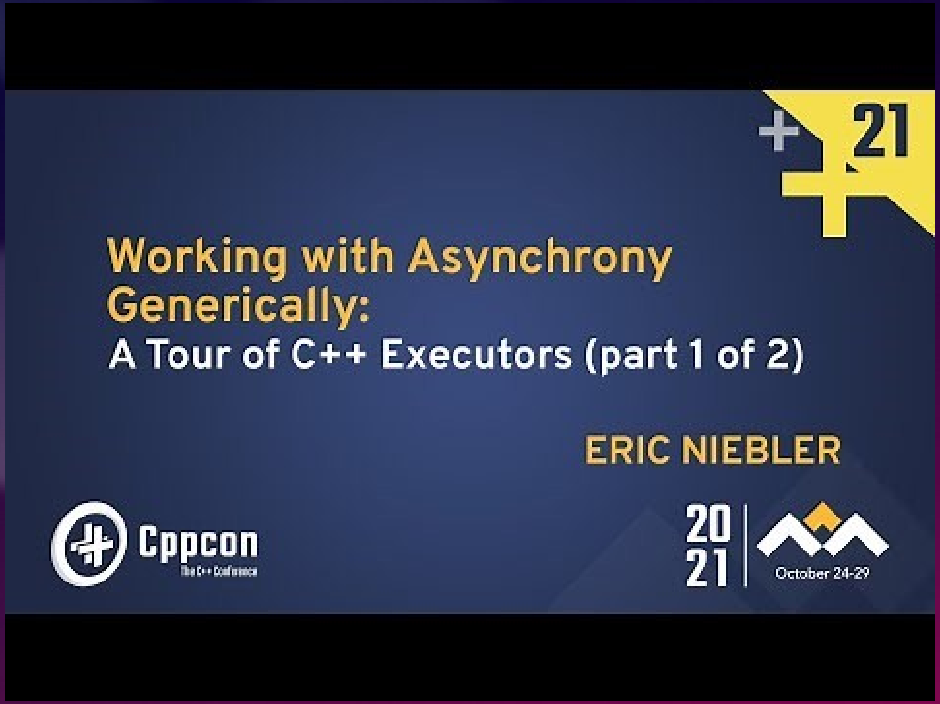
- Running C++ in context
  - “std::execution”
  - Scheduling
  - Parallelism
  - Concurrency
  - Coroutines

- ABI
  - P2123

Enabling async in C++

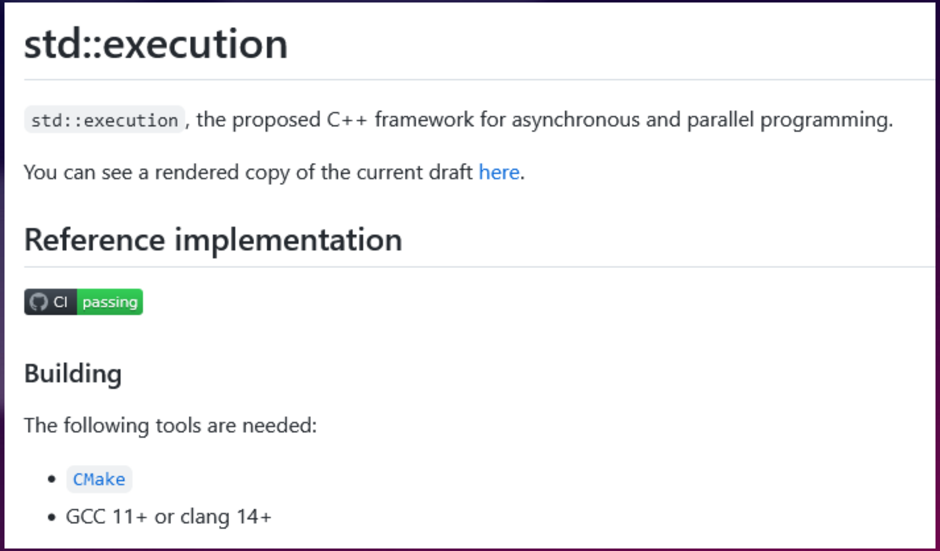

Working with Asynchrony Generically:
A Tour of C++ Executors (part 1 of 2)
ERIC NIEBLER

Enabling async in C++


Working with Asynchrony Generically:
A Tour of C++ Executors (part 1 of 2)
ERIC NIEBLER

- An important part
- The base of so much more
- Fundamental building block
  - Any error, oversight will carry heavy
  - Taking time is the right approach
- Unfortunately
  - Blocks progress for others

Reference implementation


std::execution

`std::execution`, the proposed C++ framework for asynchronous and parallel programming.

You can see a rendered copy of the current draft here.

Reference implementation

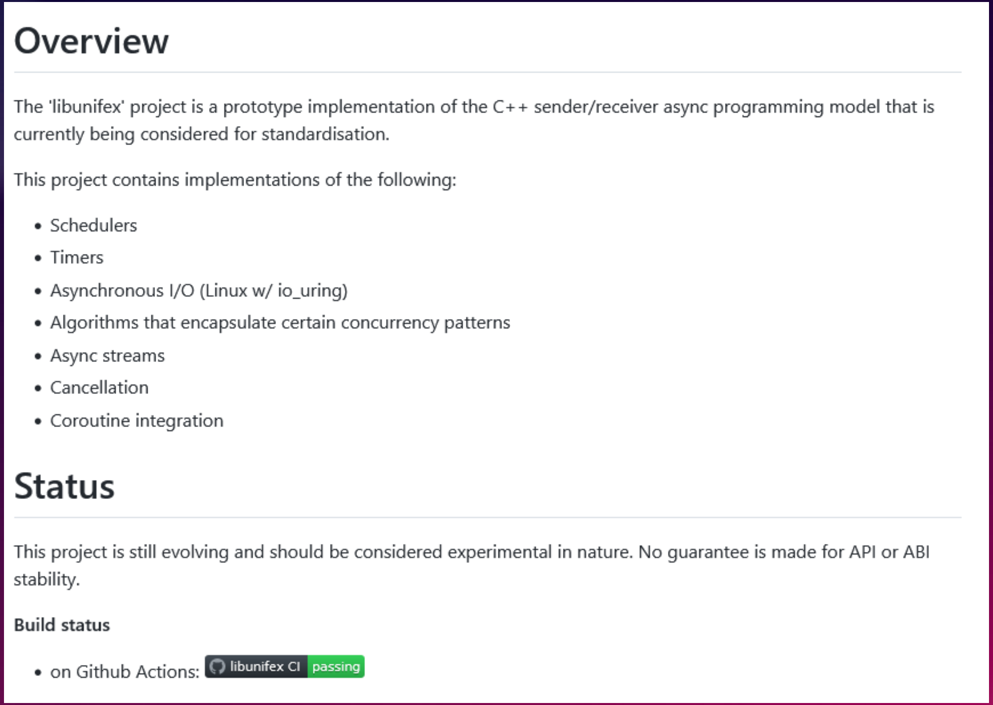
Building

The following tools are needed:

- `CMake`
- GCC 11+ or clang 14+

- Currently at
  - https://github.com/brycelelbach/wg21_p2300_std_execution
- Likely moving into a Repo at NVIDIA
- P2300

Executors today: libunifex



Overview

The 'libunifex' project is a prototype implementation of the C++ sender/receiver async programming model that is currently being considered for standardisation.

This project contains implementations of the following:

- Schedulers
- Timers
- Asynchronous I/O (Linux w/ io_uring)
- Algorithms that encapsulate certain concurrency patterns
- Async streams
- Cancellation
- Coroutine integration

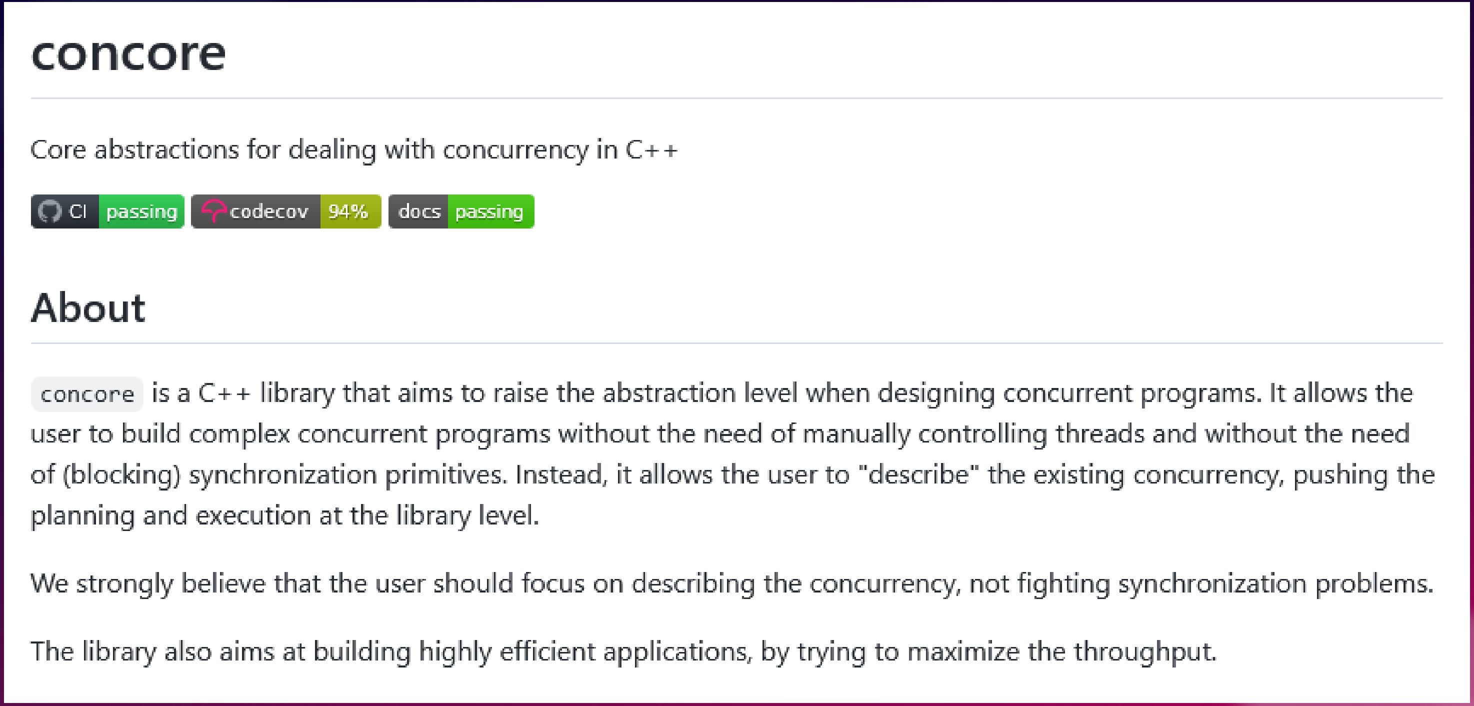
Status

This project is still evolving and should be considered experimental in nature. No guarantee is made for API or ABI stability.

Build status

- on Github Actions: libunifex CI passing

Executors today: concore



concore

Core abstractions for dealing with concurrency in C++

About

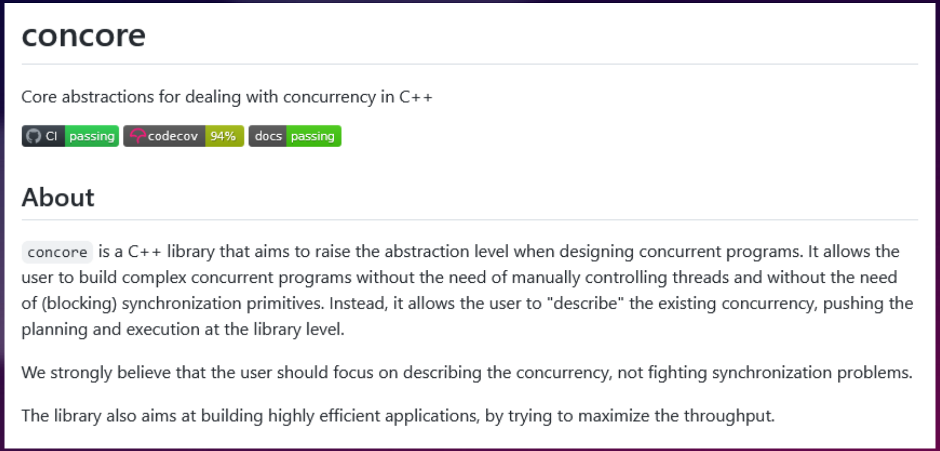
`concore` is a C++ library that aims to raise the abstraction level when designing concurrent programs. It allows the user to build complex concurrent programs without the need of manually controlling threads and without the need of (blocking) synchronization primitives. Instead, it allows the user to "describe" the existing concurrency, pushing the planning and execution at the library level.

We strongly believe that the user should focus on describing the concurrency, not fighting synchronization problems.

The library also aims at building highly efficient applications, by trying to maximize the throughput.

Executors today: concore

- No changes in one year+



concore

Core abstractions for dealing with concurrency in C++

About

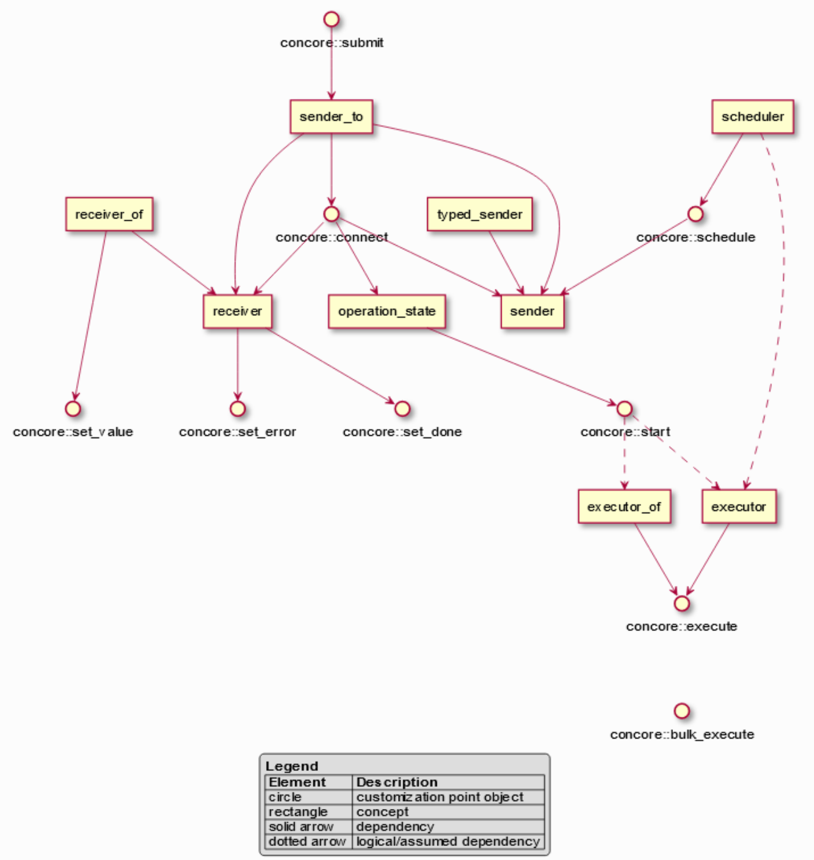
`concore` is a C++ library that aims to raise the abstraction level when designing concurrent programs. It allows the user to build complex concurrent programs without the need of manually controlling threads and without the need of (blocking) synchronization primitives. Instead, it allows the user to "describe" the existing concurrency, pushing the planning and execution at the library level.

We strongly believe that the user should focus on describing the concurrency, not fighting synchronization problems.

The library also aims at building highly efficient applications, by trying to maximize the throughput.

Though it has a really nice documentation

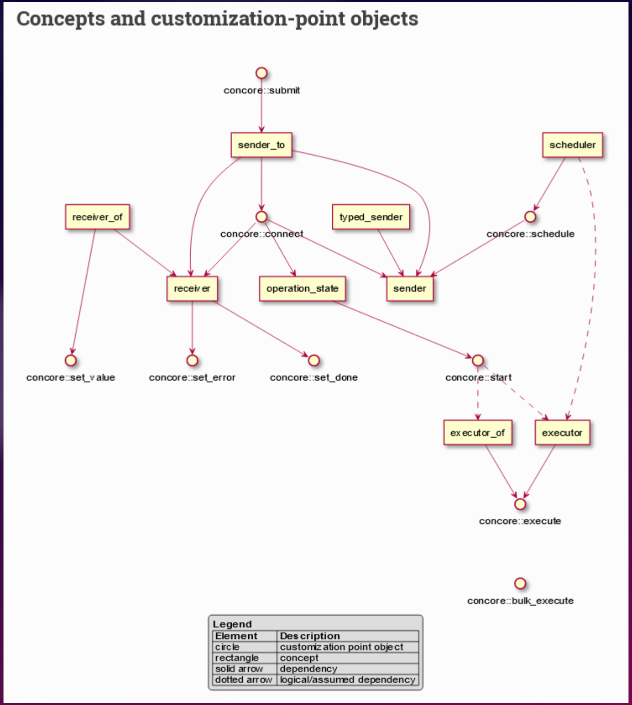
Concepts and customization-point objects



Legend

Element	Description
circle	customization point object
rectangle	concept
solid arrow	dependency
dotted arrow	logical/assumed dependency

Though it has a really nice documentation


Concepts and customization-point objects

Legend

Element	Description
circle	customization point object
rectangle	concept
solid arrow	dependency
dotted arrow	logical/assumed dependency

- C++23 executors search
- https://concore.readthedocs.io/

ABI
Application Binary Interface

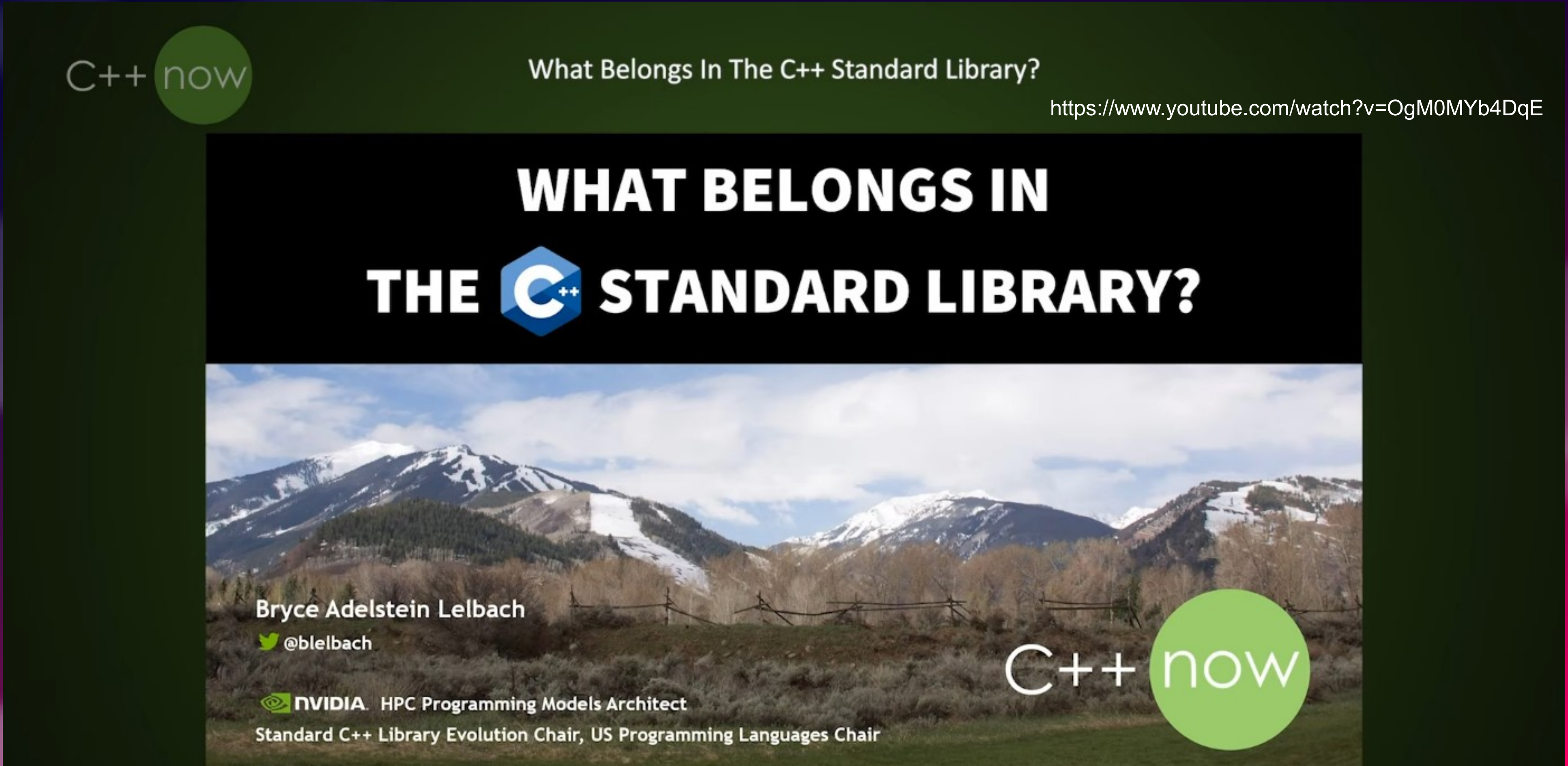
To break ABI or not is *not* the question

C++ needs to respect its users

C++ needs to respect its users

- Breaking ABI
  - Your arguments here
- Keeping ABI stable
  - Your arguments here
- Both are valid!
  - Gordian knot

Bryce Adelstein Lelbach – What belongs in the C++ standard library? - C++now 2021


C++ now

What Belongs In The C++ Standard Library?

WHAT BELONGS IN THE C++ STANDARD LIBRARY?

Bryce Adelstein Lelbach

@blelbach

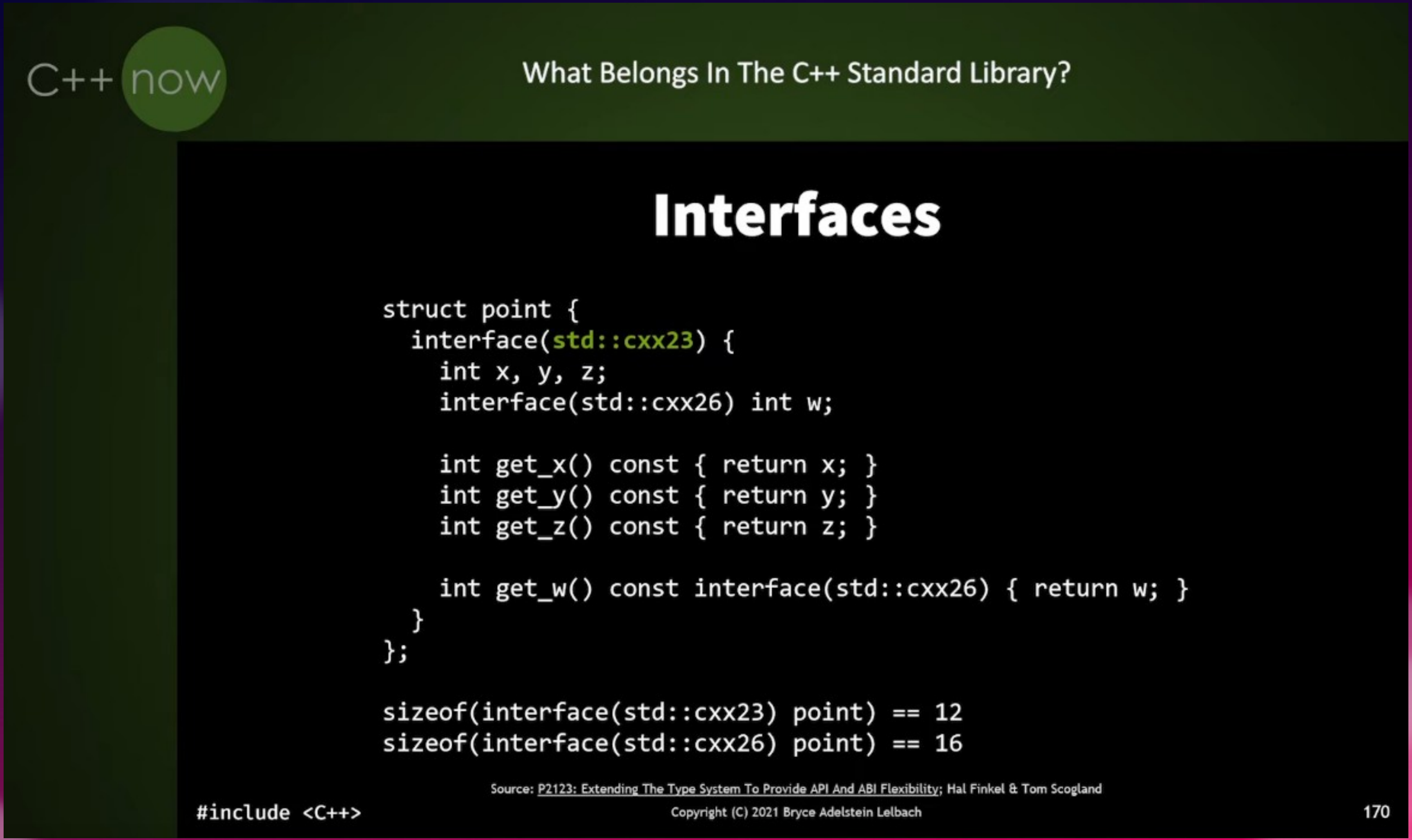
NVIDIA HPC Programming Models Architect

Standard C++ Library Evolution Chair, US Programming Languages Chair

C++ now

https://www.youtube.com/watch?v=OgM0MYb4DqE

Interface feature?



C++ now

What Belongs In The C++ Standard Library?

Interfaces

```
struct point {
  interface(std::cxx23) {
    int x, y, z;
    interface(std::cxx26) int w;

    int get_x() const { return x; }
    int get_y() const { return y; }
    int get_z() const { return z; }

    int get_w() const interface(std::cxx26) { return w; }
  }
};

sizeof(interface(std::cxx23) point) == 12
sizeof(interface(std::cxx26) point) == 16
```

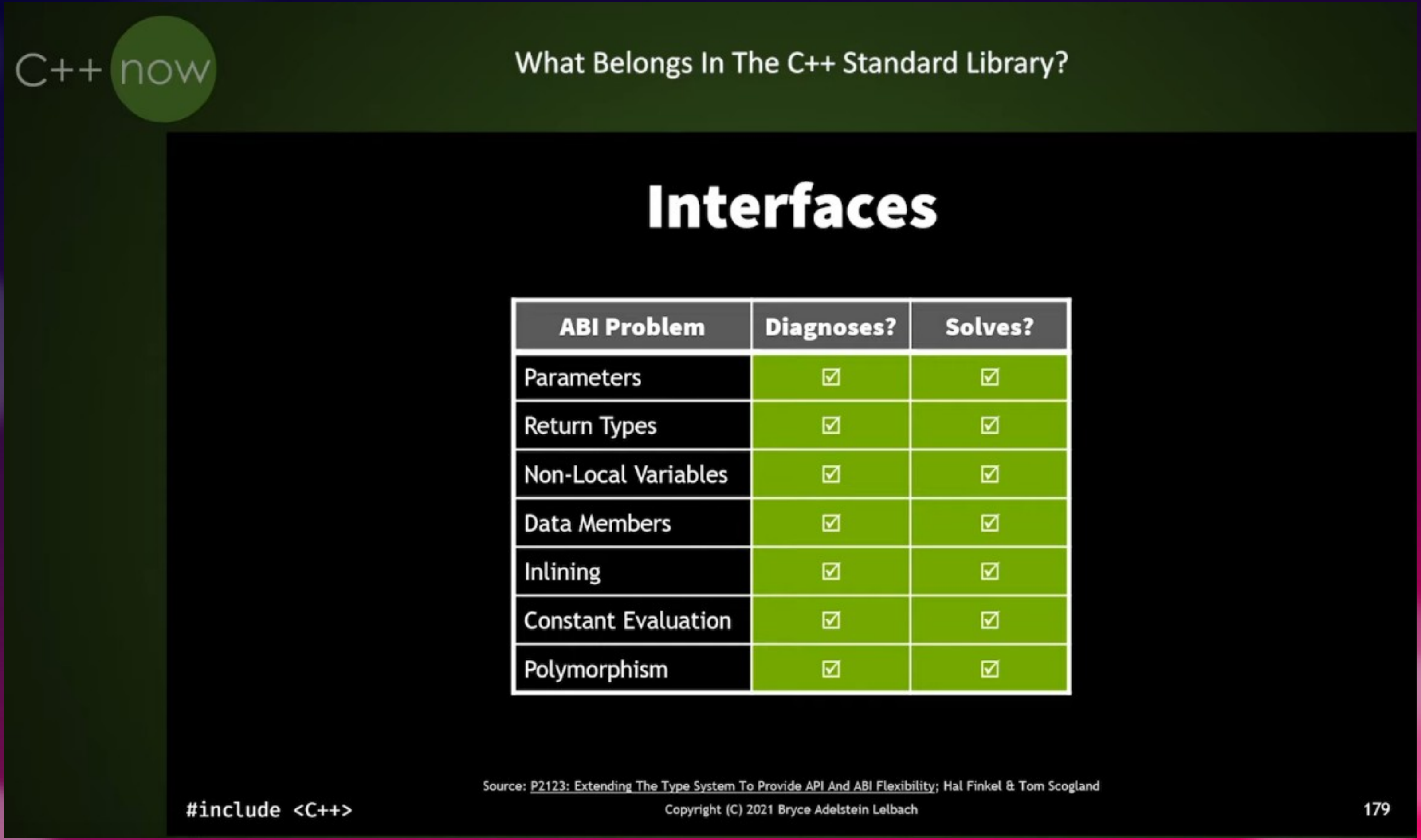
Source: P2123: Extending The Type System To Provide API And ABI Flexibility; Hal Finkel & Tom Scogland

#include <C++>

Copyright (C) 2021 Bryce Adelstein Lelbach

170

Could be a good approach?



C++ now

What Belongs In The C++ Standard Library?

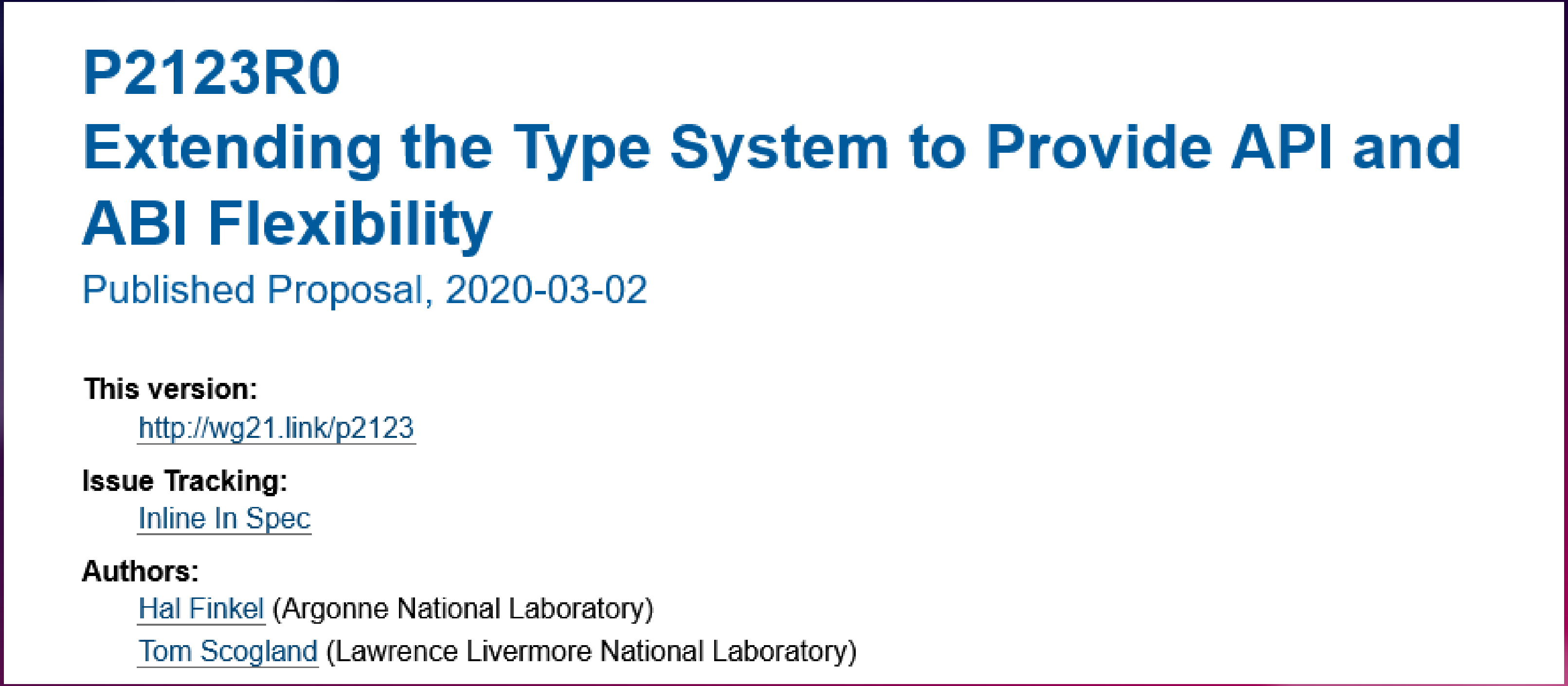
Interfaces

ABI Problem	Diagnoses?	Solves?
Parameters	☑	☑
Return Types	☑	☑
Non-Local Variables	☑	☑
Data Members	☑	☑
Inlining	☑	☑
Constant Evaluation	☑	☑
Polymorphism	☑	☑

Source: P2123: Extending The Type System To Provide API And ABI Flexibility; Hal Finkel & Tom Scogland

#include <C++>

P2123



P2123R0

Extending the Type System to Provide API and ABI Flexibility

Published Proposal, 2020-03-02

This version:

http://wg21.link/p2123

Issue Tracking:

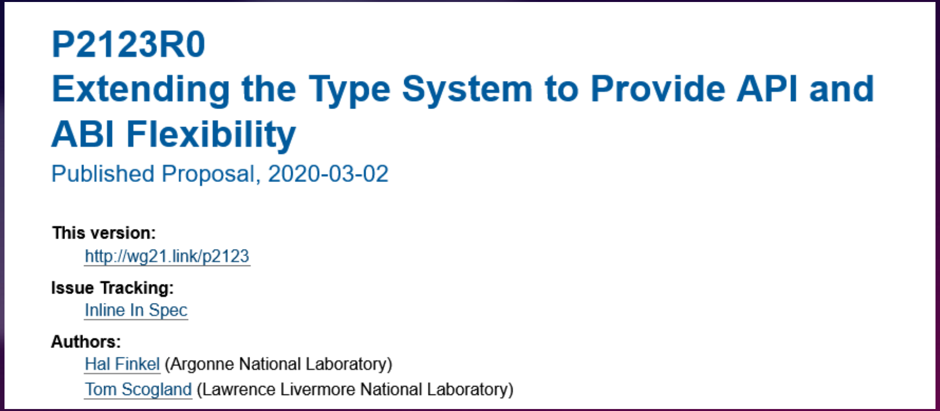
Inline In Spec

Authors:

Hal Finkel (Argonne National Laboratory)

Tom Scogland (Lawrence Livermore National Laboratory)

P2123



P2123R0
Extending the Type System to Provide API and ABI Flexibility
Published Proposal, 2020-03-02

This version:
http://wg21.link/p2123

Issue Tracking:
Inline In Spec

Authors:
Hal Finkel (Argonne National Laboratory)
Tom Scogland (Lawrence Livermore National Laboratory)

- A language level mechanism
  - To support various interfaces
    - “breaking ABI”
    - Conserving ABI

And the long wish list for this decade...

- Pattern matching
- Reflection
- Networking
- … and more

- All the things that need additions
  - Updating to new features

But, C++ is more then just the standard

Qt

{fmt}

Dlib C++ Library

MC++

POCO
C++ LIBRARIES

boost
C++ LIBRARIES

{C++ STORIES}
// STAY UP TO DATE WITH MODERN C++

Thank you!

info@meetingcpp.com

@meetingcpp