

C++ Club UK

Gleb Dolgich

2019-09-05

<http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2019/#mailing2019-08>

► [Reddit](#)

Arthur O'Dwyer

<https://quuxplusone.github.io/blog/2019/08/18/post-cologne-mailing/>

► [Reddit](#)

<https://cppcast.com/vittorio-romeo-epochs/>

https://www.reddit.com/r/cpp/comments/cxbkae/cppcast_c_epochs/

Article: https://vittorioromeo.info/index/blog/fixing_cpp_with_epochs.html

Reddit: Is the C++ committee or any key influencers in the C++ community working on anything to tackle the advantages that Rust has over C++ (eg. Rust's borrow checking, compiler-checked concurrency safety and cargo package management)

<https://devblogs.microsoft.com/cppblog/vcpkg-2019-07-update/>

https://www.reddit.com/r/cpp/comments/cqta79/vcpkg_201907_update/

Hopefully you will not require specific versions of packages, as the workflow of vcpkg doesn't really support the (very common) scenario well.

{fmt} V6.0.0 released

<https://github.com/fmtlib/fmt/releases/tag/6.0.0>

https://www.reddit.com/r/cpp/comments/cvogofmt_60_released_with_format_string_compilation/

Cpp.Chat: <https://youtu.be/PlELO-H9ZZE>

CppCast: <https://cppcast.com/victor-zverovich/>

Boost 1.71 released

https://www.boost.org/users/history/version_1_71_0.html

- ▶ NEW: **Variant2**: A never-valueless, strong guarantee implementation of `std::variant`, from Peter Dimov.
- ▶ [Reddit](#)

OSes built using C++

Reddit

- ▶ TempleOS
- ▶ Haiku
- ▶ Google Fuchsia
- ▶ IncludeOS
- ▶ DistortOS (RTOS)
- ▶ Symbian OS (Dead)
- ▶ SerenityOS

Lambdas vs. Closures

Scott Meyers

► [Reddit](#)

Web Framework Benchmarks

<https://www.techempower.com/benchmarks/#section=test&runid=26a79c95-5eec-4572-8c94-dd710df659d7&hw=ph&test=update>

► Reddit

<https://github.com/an-tao/drogon>

Drogon: A C++14/17 based HTTP web application framework running on Linux/macOS/Unix

Introducing the Rule of DesDeMovA (1/4)

Blog post by Peter Sommerlad

<https://blog.safecpp.com/2019/07/01/initial.html>

https://accu.org/content/conf2014/Howard_Hinnant_Accu_2014.pdf

Rule of Zero:

Code that you do not write cannot be wrong.

Introducing the Rule of DesDeMovA (2/4)

C++ now

2019
MAY 6-10
cppnow.org



Peter Sommerlad

Rule of DesDeMovA

Video Sponsorship
Provided By:



03:54

Voting Closed

Do you Remember: What Special Member Functions Do You Get?

3

What you write	What you get					
	default constructor	destructor	copy constructor	copy assignment	move constructor	move assignment
nothing	defaulted	defaulted	defaulted	defaulted	defaulted	defaulted
any constructor	not declared	defaulted	defaulted	defaulted	defaulted	defaulted
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	defaulted
destructor	defaulted	user declared	defaulted (!)	defaulted (!)	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted (!)	not declared	not declared
copy assignment	defaulted	defaulted	defaulted (!)	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	defaulted	deleted	deleted	not declared	user declared

Howard Hinnant's Table: https://ericniebler.com/2014/05/14/Howard_Hinnant_Accu_2014.pdf

Note: Declaring the defaulted special members denoted with a (!) is a bug in the standard.

© Peter Sommerlad

Introducing the Rule of DesDeMovA (3/4)

C++ now

2019
MAY 6-10
cppnow.org



Peter Sommerlad

Rule of DesDeMovA

Video Sponsorship
Provided By:



02:38

Voting Closed

Rule of DesDeMovA: T&& operator=(T&&) noexcept=delete;

6


What you write	DesDeMovA Rule of if Destructor defined Deleted Move Assignment					
	default constructor	copy constructor	move constructor	move assignment	move assignment	move assignment
nothing	defaulted	defaulted	defaulted	defaulted	defaulted	defaulted
any constructor	not declared	defaulted	defaulted	defaulted	defaulted	defaulted
default constructor	user declared	defaulted	defaulted	defaulted	defaulted	defaulted
destructor	defaulted	user declared	defaulted (!)	defaulted (!)	not declared	not declared
copy constructor	not declared	defaulted	user declared	defaulted (!)	not declared	not declared
copy assignment	defaulted	defaulted	defaulted (!)	user declared	not declared	not declared
move constructor	not declared	defaulted	deleted	deleted	user declared	not declared
move assignment	defaulted	user declared	deleted	deleted	not declared	=delete

Howard Hinnant's Table: https://ericniebler.com/2014/hinnant_howard_2014.pdf
Note: Defining the defaulted special members denoted with a (!) is a bug in the standard.

Introducing the Rule of DesDeMovA (3/4)


C++ now

2019
MAY 6-10
cppnow.org




Peter Sommerlad

Rule of DesDeMovA


Video Sponsorship
Provided By: 

02:19

Voting Closed


Summary  7

1. Rule of Zero
2. Rule of DesDeMovA (no copy, no move for SBRM/RAII and OO-Base classes)
3. Rule of Unique Resource Managers (move-only, no copy)
4. Rule of Five for Resource Managers with Value Semantics, or other really special cases



Cevelop
Your C++ deserves it

Download IDE at:
www.cevelop.com



Sponsors welcome!

Commercial licensing possible!



Simon Brand @TartanLlama

Threw every C++ proposal title in history into a recurrent neural network and some sound surprisingly legit

```
Introducing the Proposal for an Resumable memory related initialization from string_user enuming_values
Integer C++ Function (Revision 9)
Fixing The Konates of Capture
feature_lock_view-expressions
GENDA, Tiny Proposed Regrogish to Standard Library
Transparentic atomic operations (rev. 3)
Editor's Report for Deprecate Vector Technical Specification and Allocation
Containers for Parflock Points
std::function without a requirement and optimization for the C++2 Defects Record
Propose-a the Generic Memory Modification
Equalized Interaction for the Model Proposal for P0107R2
Introduce types in computing and string_that in C++
Contract Design Aliation for be Business
Gegropage initialization finesy policies in point memory scheduler be_rodng_robot and enumerations
Reflection of noexcept in C++20
std:::haluter_tokence_related (revision 1)
Meeting Not C++
Parallellative Intermate() and unlumbing_cip
What madened: Propose-tight member of Painfreacted Default
Lookup Asiomation TS
```

2d • 16/07/2019 • 20:07 •

