

C++ Club Meeting Notes

Gleb Dolgich

2019-06-06

Apple SwiftUI and C++

Zhuowei Zhang (@zhuowei) 6 Replies, 10 Quotes

Apple: Swift is a system programming language that can replace C++
Also Apple: writes new **#SwiftUI** library in C++
#wwdc

04/06/2019, 21:03 (Yesterday)
Twitter Web Client

344 Likes | 97 Retweets | Thread >

Thread navigation icons: back, forward, reply, like, unlike, report.

Thread details:

- Thread 1 Queue: com.apple.main-thread (serial)
- 0 ContentView.body.getter
- 1 protocol witness for View.body.getter in conformance ContentView
- 2 ViewBody.apply(_)
- 3 protocol witness for static UntypedAttribute._update(_:graph:attribute:) in confo...
- 4 partial apply
- 5 AG::Graph::UpdateStack::update()
- 6 AG::Graph::update_attribute(unsigned int, bool)
- 7 **AG::Subgraph::update(unsigned int)** (highlighted)
- 8 ViewGraph.runTransaction(in:)
- 9 closure #1 in ViewGraph.updateOutputs(at:)
- 10 ViewGraph.updateOutputs(at:)
- 11 closure #1 in closure #1 in ViewRendererHost.render(interval:updateDisplayLi...
- 12 closure #1 in ViewRendererHost.render(interval:updateDisplayList:)
- 13 ViewRendererHost.render(interval:updateDisplayList:)

Myths about C++

<https://www.reddit.com/r/cpp/comments/bvf78q/myths/>

See also

- ▶ Bjarne Stroustrup - Five Popular Myths about C++

CppCon 2018: Jason Turner - Applied Best Practices

- ▶ Video: <https://youtu.be/DHOlsEd0eDE>
- ▶ Slides: https://github.com/CppCon/CppCon2018/blob/master/Presentations/applied_best_practices/applied_best_practices_jason_turner__cppcon_2018.pdf

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

The slide is titled "Accessors". It features a code snippet for a C++ struct named "Strongly Typed". The code uses a template parameter "Type" and a CTRP (Copy-and-Transpose Return Parameter) type. It includes a static assert for trivial types and a constexpr auto member function "value" that returns the private member variable "m_value". The code is numbered from 1 to 10.

```
1 template<typename Type, typename CTRP>
2 struct Strongly Typed {
3     // enforce this expectation of trivial types
4     static_assert(std::is_trivial_v<Type>);
5
6     [[nodiscard]] constexpr auto value() const noexcept { return m_value; }
7
8     protected:
9         Type m_value;
10    };

```

JASON TURNER

Applied
Best Practices

Copyright Jason Turner @lefticus 3.16

CppCon.org

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

JASON TURNER

Applied Best Practices

Trailing Return Types

Which do we prefer?

```
1 // A
2 [[nodiscard]] constexpr auto value() const noexcept {return m_value;}
```

```
1 // B
2 [[nodiscard]] constexpr auto value() const noexcept ->Type{return m_value;}
```

```
1 // C
2 [[nodiscard]] constexpr Type value() const noexcept {return m_value;}
```

Copyright Jason Turner @lefticus 4.2

CppCon.org

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

cppcon | 2018

THE C++ CONFERENCE • BELLEVUE, WASHINGTON



JASON TURNER

Applied Best Practices

CppCon.org

Trailing Return Types

What if the types are longer and full `auto` isn't practical?

```
1 // A
2 [[nodiscard]] constexpr auto value() const noexcept->Type;
3 [[nodiscard]] constexpr auto op() const noexcept->std::pair<bool,uint32_t>;
4 [[nodiscard]] constexpr auto type() const noexcept->Op_Type;
```



```
1 // B
2 [[nodiscard]] constexpr Type value() const noexcept;
3 [[nodiscard]] constexpr std::pair<bool, uint32_t> op() const noexcept;
4 [[nodiscard]] constexpr Op_Type type() const noexcept;
```

Copyright Jason Turner

@lefticus

4.3

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

cppcon | 2018

THE C++ CONFERENCE • BELLEVUE, WASHINGTON



JASON TURNER

Applied Best Practices

CppCon.org

How `constexpr` Helps

Cannot compile! By utilizing `constexpr` we can catch extra classes of undefined behavior that warnings cannot catch.

```
1  constexpr auto shift(int val, int distance)
2  {
3      return val << distance;
4  }
5
6  int main()
7  {
8      constexpr auto result = shift(1, 32);
9
10 }
```

Copyright Jason Turner

@lefticus

6.8

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

The slide is from a CppCon 2018 session. On the left, there's a video feed of Jason Turner speaking, with his name "JASON TURNER" below it. To the right of the video is a large text box containing a title and a bulleted list. At the bottom of the slide, there's footer information.

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

JASON TURNER

Applied Best Practices

constexpr Is Not The Point of This Talk

... but the subset of C++ that works in `constexpr` context is the language many people say they want.

- No (or minimal) undefined behavior
- No exception handling
- No dynamic allocation (that might be changing)
- Types tend to be `trivial` or at least `trivially destructible`
- Move semantics are mostly irrelevant when objects are `trivially copyable`

Copyright Jason Turner @lefticus 10.2

CppCon.org

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

The slide features a video feed of Jason Turner on the left, showing him from the waist up as he speaks. He is wearing a light blue striped short-sleeved shirt and dark trousers. To his right is a large text box containing the slide's content. At the bottom of the slide is a dark footer bar.

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

JASON TURNER

**Applied
Best Practices**

There Are Warnings

My prototyping command line looks like this:

```
g++ -std=c++17 -Wall -Wextra -Wshadow -Wpedantic test.cpp
```

Looks pretty good?

Copyright Jason Turner @lefticus 11.3

CppCon.org

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON



JASON TURNER

Applied
Best Practices

CppCon.org

And Then There Are Warnings

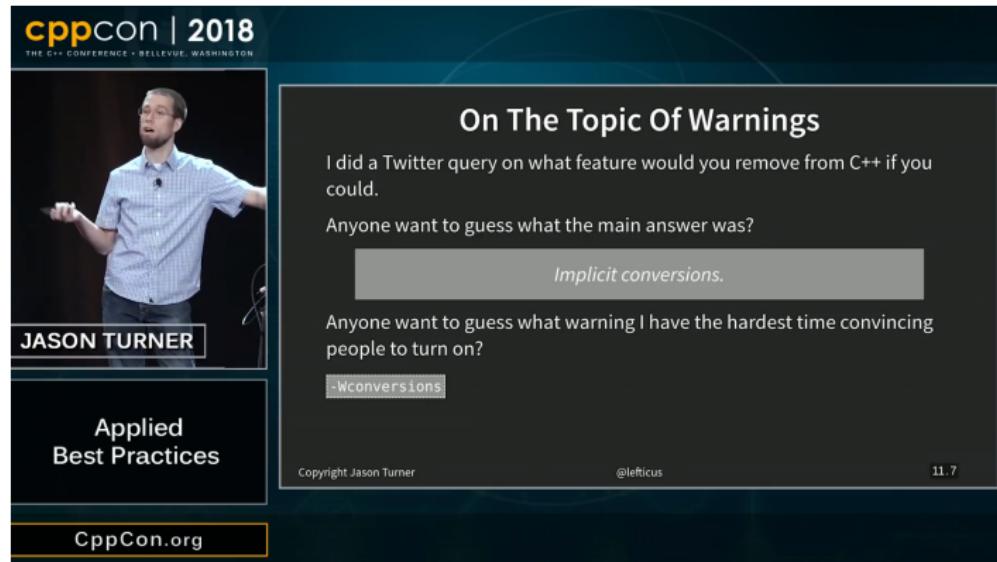
```
1  -Wall
2  -Wextra # reasonable and standard
3  -Wshadow # warn the user if a variable declaration shadows one from a
4   # parent context
5  -Wnon-virtual-dtor # warn the user if a class with virtual functions has a
6   # non-virtual destructor.
7  -Wold-style-cast # warn for c-style casts
8  -Wcast-align # warn for potential performance problem casts
9  -Wunused # warn on anything being unused
10 -Woverloaded-virtual # warn if you overload (not override) a virtual func
11 -Wpedantic # warn if non-standard C++ is used
12 -Wconversion # warn on type conversions that may lose data
13 -Wsign-conversion # warn on sign conversions
14 -Whull-dereference # warn if a null dereference is detected
15 -Wdouble-promotion # warn if float is implicitly promoted to double
16 -Wformat=2 # warn on security issues around functions that format output
17   # (ie printf)
18 -Wduplicated-cond # warn if if / else chain has duplicated conditions
19 -Wduplicated-branches # warn if if / else branches have duplicated code
20 -Wlogical-op # warn about logical operations being used where bitwise were
21   # probably wanted
22 -Wuseless-cast # warn if you perform a cast to the same type
23 -Wlifetime # ///
```

Copyright Jason Turner

@nektarios

11.5

CppCon 2018: Jason Turner - Applied Best Practices (cont.)



The image is a screenshot from a video of Jason Turner speaking at CppCon 2018. On the left, there's a video frame showing Jason Turner from the waist up, wearing a light blue checkered shirt and jeans, holding a microphone and gesturing with his hands. To his right is a dark slide with white text. The slide has a title, a question, and two highlighted answers. At the bottom, there are copyright information and social media handles.

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

JASON TURNER

Applied Best Practices

On The Topic Of Warnings

I did a Twitter query on what feature would you remove from C++ if you could.

Anyone want to guess what the main answer was?

Implicit conversions.

Anyone want to guess what warning I have the hardest time convincing people to turn on?

`-Wconversions`

Copyright Jason Turner @lefticus 11.7 CppCon.org

CppCon 2018: Jason Turner - Applied Best Practices (cont.)

cppcon | 2018
THE C++ CONFERENCE • BELLEVUE, WASHINGTON

JASON TURNER

Applied Best Practices

	SFML 2.5.0	Catch2 2.3.0	rang 3.1.0	{fmt} 5.1.0	spdlog 1.1.0
Buckaroo	2.4.2	-	2.0.0	3.0.1	0.13.0
cppget	-	-	-	-	-
Conan Center	-	2.3.0	-	5.1.0	1.1.0
Conan	2.5.0	2.3.0	3.1.0	5.1.0	1.1.0
CPPAN	2.5.0	2.2.3	2.0.0	5.1.0	1.0.0
Hunter	-	2.2.2	-	4.1.0	0.16.3
qpm	-	-	-	-	-
vcpkg	2.5.0 *	2.3.0	-	5.1.0	1.0.0

Copyright Jason Turner @lefticus 14.4

CppCon.org

Simplifying C++ with Herb Sutter

- ▶ CppCast <http://cppcast.com/2019/05/herb-sutter/>
 - ▶ Reddit https://www.reddit.com/r/cpp/comments/bv3bc9/cppcast_simplifying_c_with_herb_sutter/
- ▶ A theme: Simplifying C++ (& CppCast podcast)
<https://herbsutter.com/2019/06/01/a-theme-simplifying-c-cppcast-podcast/>
 - ▶ Reddit https://www.reddit.com/r/cpp/comments/bvukm6/a_theme_simplifying_c_cppcast_podcast/

Awesome Modern C++

<https://awesomecpp.com/>

C++ Operator Signatures

<https://gist.github.com/beached/38a4ae52fcadfab68cb6de05403fa393>

<https://github.com/glebd/cppclub/blob/next/3rd/C%2B%2B%2520normal%2520operators.md>

Hedley

Hedley: A C/C++ header to help move #ifdefs out of your code

- ▶ Home page: <https://nemequ.github.io/hedley/>
- ▶ Reddit: https://www.reddit.com/r/cpp/comments/bm2xyk/hedley_a_cc_header_to_help_move_ifdefs_out_of/

STXXL

STXXL: Standard Template Library for Extra Large Data Sets.

The core of STXXL is an implementation of the C++ standard template library for external memory (out-of-core) computations, i. e., STXXL implements containers and algorithms that can process huge volumes of data that only fit on disks. While the closeness to the STL supports ease of use and compatibility with existing applications, another design priority is high performance.

- ▶ Home: <http://stxxl.org/>
- ▶ Code: <https://github.com/stxxl/stxxl> (Boost Software License)
- ▶ Video:
<http://panthema.net/2014/0622-Talk-STXXL-1.4.0-and-Beyond/>

Function Poisoning in C++

<https://www.fluentcpp.com/2018/09/04/function-poisoning-in-cpp/>

```
1 #include <stdio.h>
2 #pragma GCC poison puts
3
4 int main() {
5     puts("a");
6 }
7
8 // error: attempt to use poisoned "puts"
```

Deprecating and Deleting Functions in C++

<https://www.fluentcpp.com/2018/11/20/deprecating-and-deleting-functions-in-cpp/>

```
1 [[deprecated("Replaced by fillmem, which has an improved interface")]]  
2 void* memset(void*, int, size_t);
```

Clear, Functional C++ Documentation with Sphinx + Breathe + Doxygen + CMake

<https://devblogs.microsoft.com/cppblog/clear-functional-c-documentation-with-sphinx-breathe-doxygen-cmake/>

See also

- ▶ CppCon 2017: Robert Ramey “How to Write Effective Documentation for C++ Libraries...”
- ▶ Eli Bendersky - reStructuredText vs. Markdown for technical documentation
- ▶ Viktor Zverovich - reStructuredText vs Markdown for documentation

Twitter



Timur Doumler
@timur_audio



How do you pronounce "tuple"?

/tupəl/ ("toople")

47%

/tʌpəl/ ("tuttle")

31%

/tjupəl/ ("tewple")

22%

411 votes · Final results

01:11 · 30/05/2019 · Twitter for iPhone

Quote

Cedric Guillemet @skaven_:

Any sufficiently advanced C++ codebase contains a Utils.cpp/h