C++ Club Meeting Notes

Gleb Dolgich

2018-10-18

Bjarne Stroustrup Interview at CppCon 2018

Video

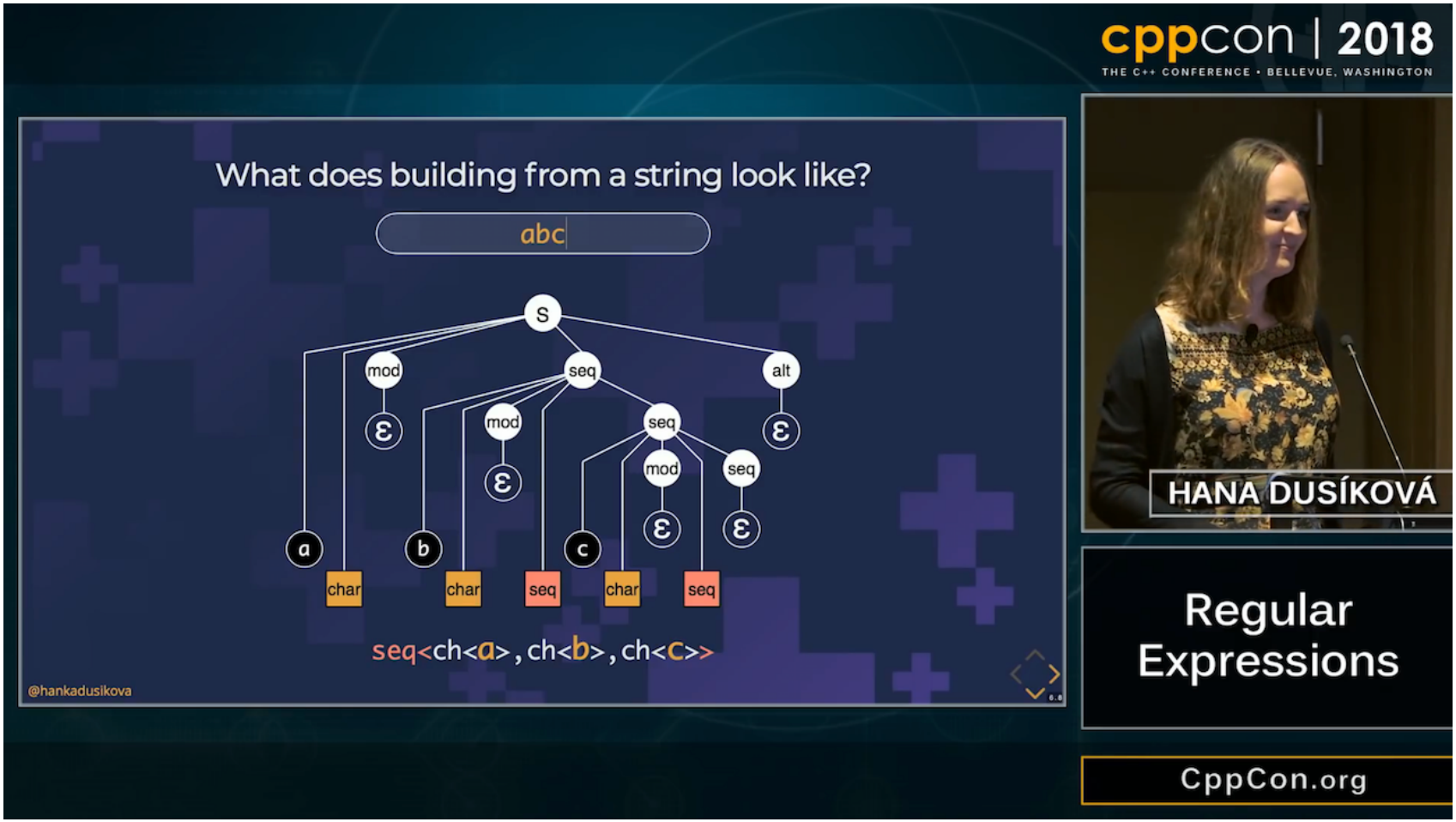
CppCon 2018 Trip Reports

- Mathieu Ropert
- Anny Gakhokidze
  - with sketches and talk notes!

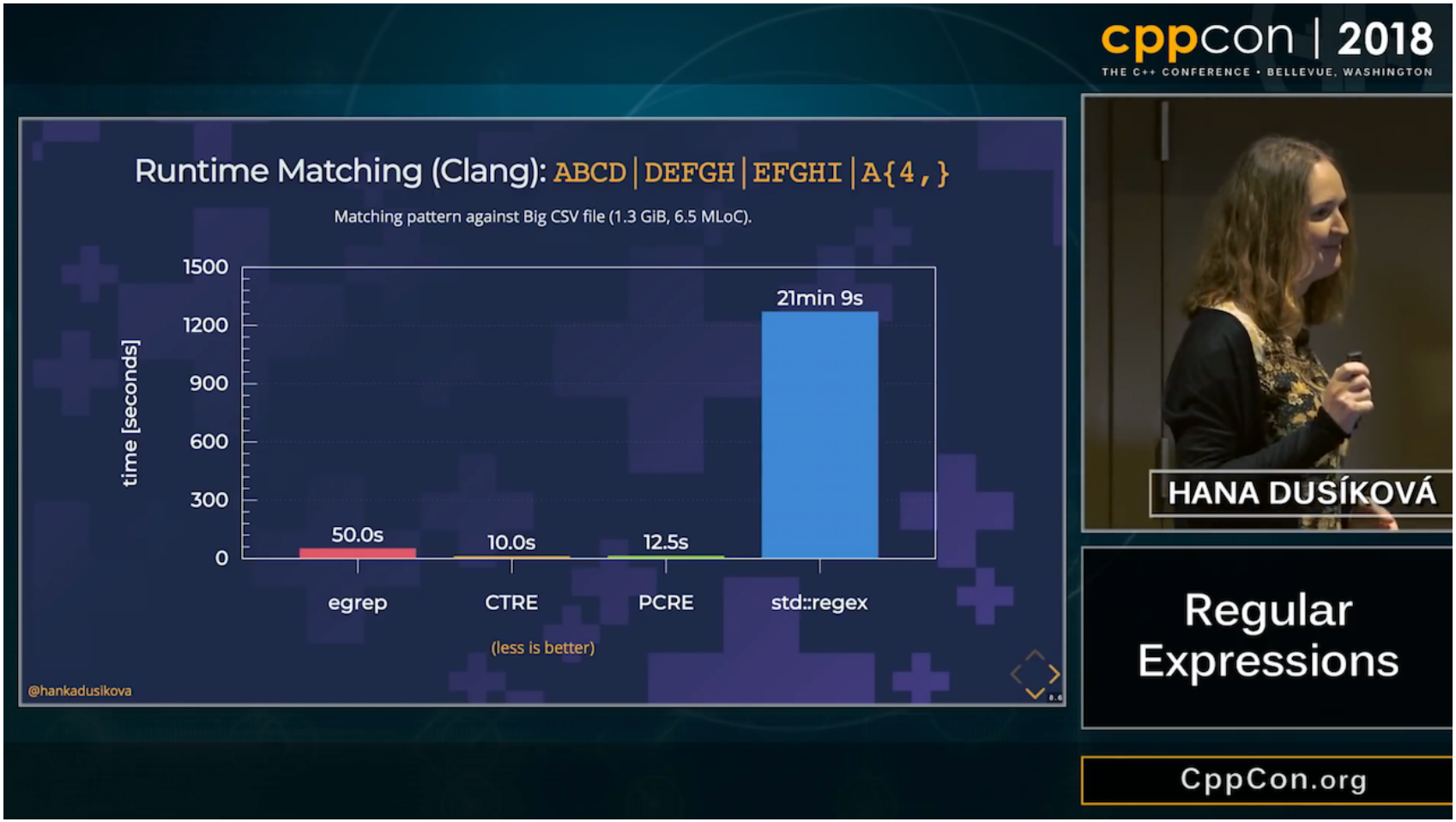
CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (1/6)

- Video
  - Reddit
- Slides
- GitHub (MIT)
- P0732R0 Class Types in Non-Type Template Parameters
- LL Parser

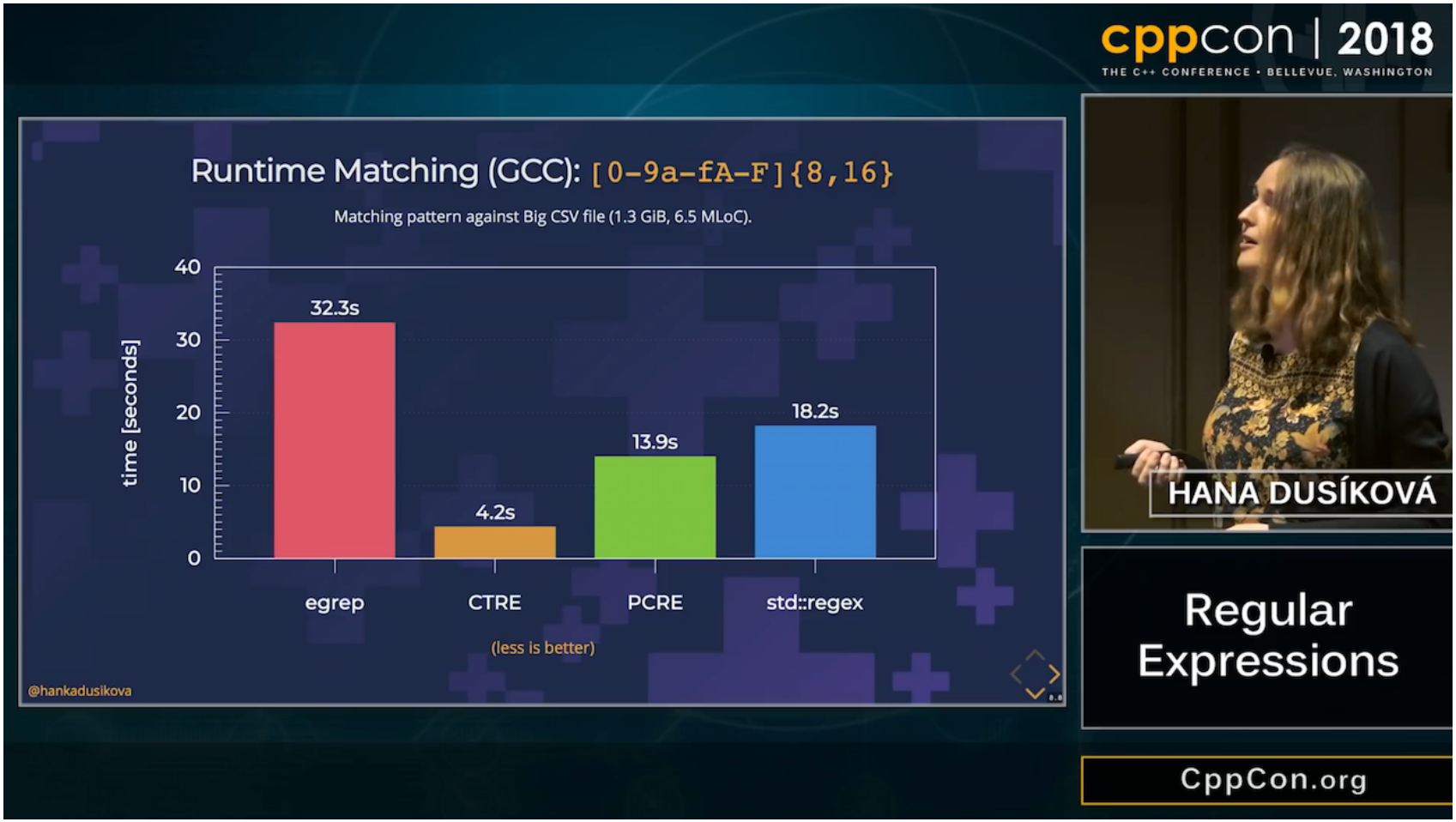
CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (2/6)



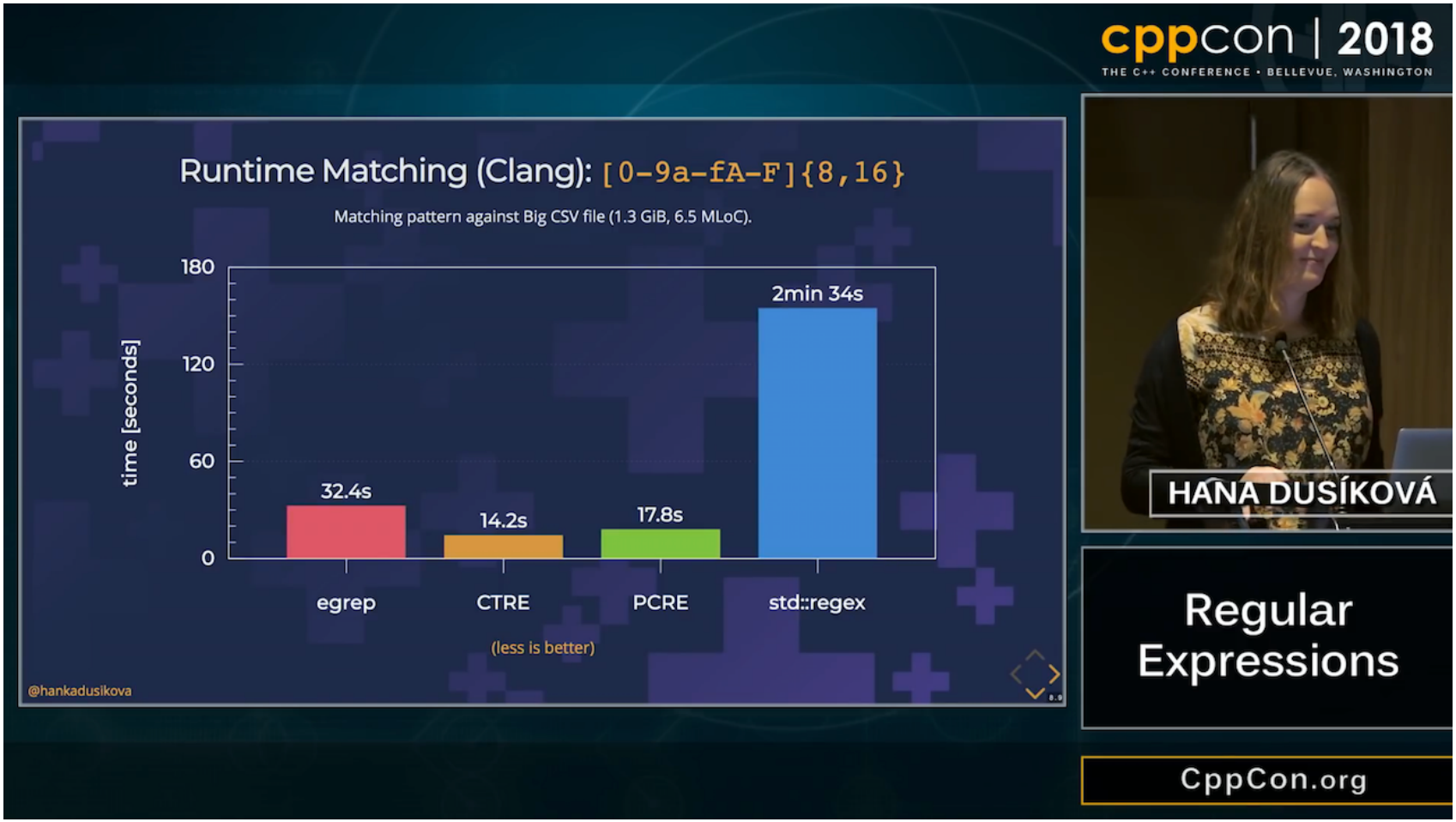
CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (3/6)



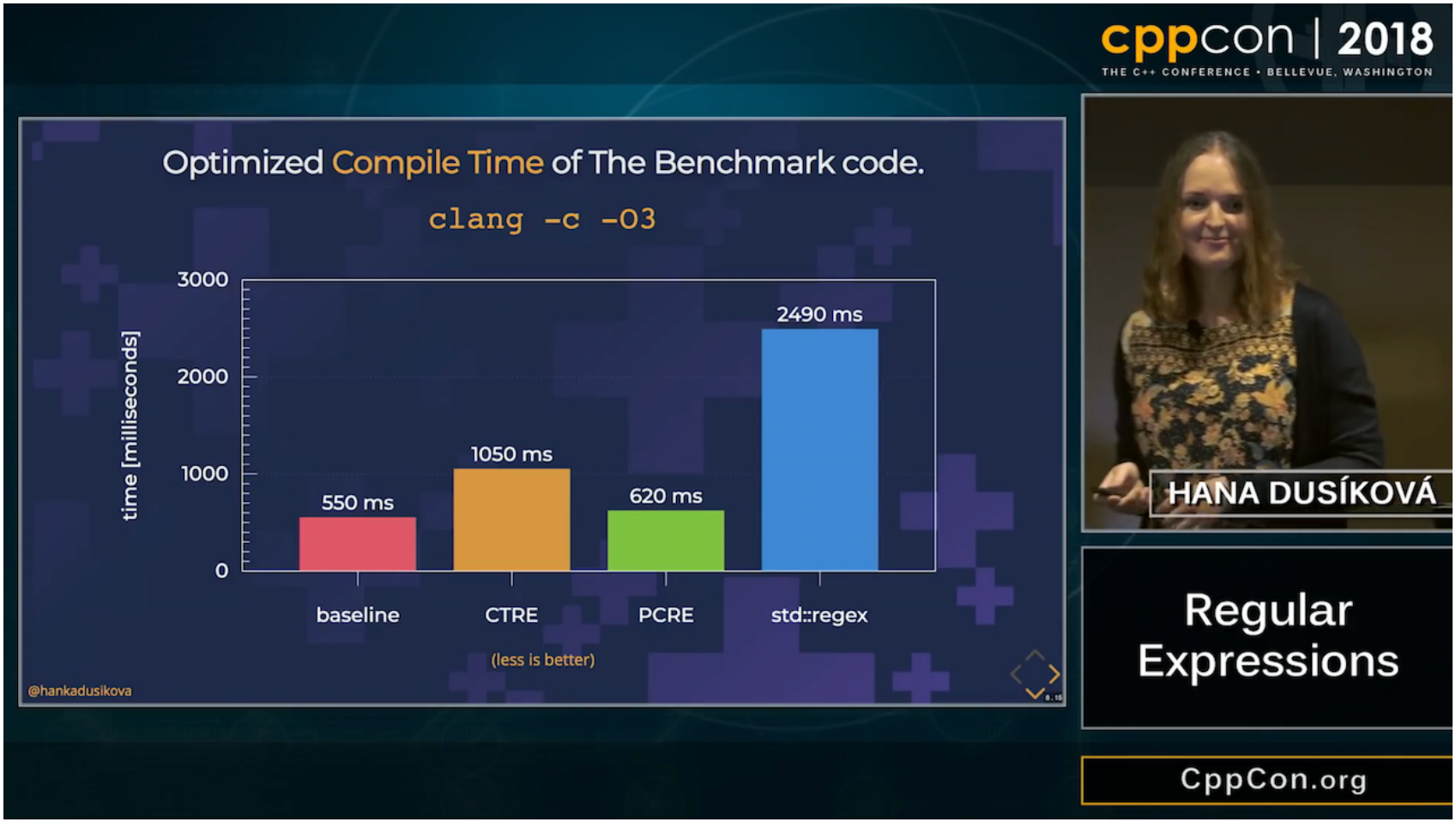
CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (4/6)



CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (5/6)



CppCon 2018: Hana Dusíková - Compile-Time Regular Expressions (6/6)



Pre-San Diego Mailing

- Papers
  - Reddit
- Reading guide (Google Groups)
  - Reddit

Pre-San Diego: Modules

- N4720: Working Draft, Extensions to C++ for Modules
  - Gabriel Dos Reis, gdr@microsoft.com
- P0581R1: Standard Library Modules
  - Marshall Clow, Beman Dawes, Gabriel Dos Reis, Stephan T. Lavavej, Billy O'Neal, Bjarne Stroustrup, Jonathan Wakely
- P0909R0: Module TS Supports Legacy Integration
  - Steve Downey, sdowney2@bloomberg.net
- P0877R0: Modular macros
  - Bruno Cardoso Lopes, blopes@apple.com

Pre-San Diego: Concept-constrained auto

- P0915R0
  - Vittorio Romeo vittorio.romeo@outlook.com, John Lakos jlakos@bloomberg.net

```
1 template <typename T>
2 void foo(std::vector<T>& v)
3 {
4     auto<RandomAccessIterator> it{std::begin(v)};
5     // ...
6 }
```

(Not in pre-San Diego) Type functions and beyond: An exploration of type functions and concept functions, by J. Monnon

P0844

> *This document proposes to extend functions to let them operate directly on types and concepts. The goal is to allow writing metaprogramming in the most intuitive and consistent way with the rest of the language.*

```
1 ForwardIterator IteratorType(typename T) {
2     // In a type function, an `if` behaves as a `if constexpr`.
3     if (Container(T))  // `Container` is a concept
4         return T::iterator;
5     else if (Array(T)) // `Array` is a concept
6         return Decay(T);
7 }
8 // On call site:
9 typename I = IteratorType(C);
```

(Not in pre-San Diego) Type functions and beyond: An exploration of type functions and concept functions, by J. Monnon

P0844

> *Concept functions are introduced to manipulate and transform concepts. One of the simplest examples of concept function is to create a new concept by adding constraints to an existing one:*

```
1 // Adds the constraints of the `Serialize` concept to any concept.
2 concept Serializable(concept C) {
3     return C && Serialize;
4 };
5
6 // On call site:
7 template<Serializable(Container) C>
```

Pre-San Diego: A sane variant converting constructor

- P0608R3
  - Zhihao Yuan zy@miator.net
  - This paper proposes to constrain the variant converting constructor and the converting assignment operator to prevent narrowing conversions and conversions to bool.

```
1 using T = variant<float, long>;
2 T v;
3 v = 0;    // what is stored in v: 1) float 2) long 3) ill-formed
```

Pre-San Diego: Monadic operations for std::optional

- P0798R2
  - Simon Brand, simon.brand@microsoft.com

```
std::optional<image> get_cute_cat (const image& img) {
    return crop_to_cat(img)
            .and_then(add_bow_tie)
            .and_then(make_eyes_sparkle)
            .map(make_smaller)
            .map(add_rainbow);
}
```

Pre-San Diego: C++ Monadic interface

- P0650R2
  - Vicente J. Botet Escribá vicente.botet@nokia.com
- p0323r5 std::expected

Pre-San Diego: Adding a workflow operator to C++

- P1282R0
  - Isabella Muerte isabella.muerte@target.com

```
int total = accumulate { 0 } <| view::iota(1)
                             |> view::transform([](int x){return x*x;})
                             |> view::take(10);
```

Pre-San Diego: Language Variants (1/2)

- P0095R2
  - David Sankel (dsankel@bloomberg.net)
  - Dan Sarginson (dsarginson@bloomberg.net)
  - Sergei Murzin (smurzin@bloomberg.net)

```
1 lvariant command {
2   std::size_t set_score; // Set the score to the specified value
3   std::monostate fire_missile; // Fire a missile
4   unsigned fire_laser; // Fire a laser with the specified intensity
5   double rotate; // Rotate the ship by the specified degrees.
6 };
```

Pre-San Diego: Language Variants (2/2)

```
std::ostream& operator<<( std::ostream& stream, const command cmd ) {
  return inspect( cmd ) {
    set_score value =>
      stream << "Set the score to " << value << ".\n"
    fire_missile m =>
      stream << "Fire a missile.\n"
    fire_laser intensity:
      stream << "Fire a laser with " << intensity << " intensity.\n"
    rotate degrees =>
      stream << "Rotate by " << degrees << " degrees.\n"
  };
}
```

Pre-San Diego: Pattern matching

- P1308R0
  - David Sankel (dsankel@bloomberg.net)
  - Dan Sarginson (dsarginson@bloomberg.net)
  - Sergei Murzin (smurzin@bloomberg.net)

```
1 enum color { red, yellow, green, blue };
2 const Vec3 opengl_color =
3   inspect(c) {
4     red    => Vec3(1.0, 0.0, 0.0)
5     yellow => Vec3(1.0, 1.0, 0.0)
6     green  => Vec3(0.0, 1.0, 0.0)
7     blue   => Vec3(0.0, 0.0, 1.0)
8   };
```

- P1260R0 (?)

Pre-San Diego: Reflection

- N4766 C++ Extensions for reflection
- P0953R1 constexpr reflexpr
- P1240R0 Scalable Reflection in C++
- P1306R0 Expansion statements

Pre-San Diego: A polymorphic value type for C++

- P0201R4
  - Jonathan Coe (jonathanbcoe@gmail.com)
  - Sean Parent (sparent@adobe.com)

Add a class template, `polymorphic_value<T>`, to the standard library to support polymorphicobjects with value like semantics.

The class template, `polymorphic_value`, confers value like semantics on a free store allocated object. A `polymorphic_value<T>` may hold an object of a class publicly derived from `T`, and copying the `polymorphic_value<T>`will copy the object of the derived type.

Pre-San Diego: A Unified Executors Proposal for C++

- P0443R9
  - Jared Hoberock, jhoberock@nvidia.com
  - Michael Garland, mgarland@nvidia.com
  - Chris Kohlhoff, chris@kohlhoff.com
  - Chris Mysen, mysen@google.com
  - Carter Edwards, hcedwar@sandia.gov
  - Gordon Brown, gordon@codeplay.com
- P1244R0 Dependent Execution for a Unified Executors Proposal for C++
- P0797R1 Handling Concurrent Exceptions with Executors
- P1194 The Compromise Executors Proposal: A lazy simplification of P0443
  - This paper seeks to add support for lazy task creation and deferred execution to P0443, while also simplifying the fundamental concepts involved in asynchronous execution.

Pre-San Diego: Text Formatting

- P0645R3
  - Viktor Zverovich, victor.zverovich@gmail.com

This paper proposes a new text formatting library that can be used as a safe and extensible alternative to the printf family of functions. It is intended to complement the existing C++ I/O streams library and reuse some of its infrastructure such as overloaded insertion operators for user-defined types.

```
1 string message = format("The answer is {}.", 42);
```

A full implementation of this proposal is available at https://github.com/fmtlib/fmt/tree/std.

Pre-San Diego: Freestanding Proposal

- P0829R3
  - Ben Craig, ben.craig@gmail.com
  - “The current set of freestanding libraries provides too little to kernel and embedded programmers. I propose we provide the (nearly) maximal subset of the library that does not require an OS or space overhead.”
- P1105R1: Leaving no room for a lower-level language: A C++ Subset
  - Making exceptions, dynamic RTTI, TLS, heap, floating point, program teardown, and blocking operations optional in freestanding mode.
- P1212R0: Modules and Freestanding
  - Standard library modules needs to know a direction for freestanding.

Pre-San Diego: Down with *typename*!

- P0634R2
  - Nina Ranns, Daveed Vandevoorde

Pre-San Diego: Parametric Functions (1/2)

- P0671R1
  - Axel Naumann (axel@cern.ch)

```
1  double Gauss(double x, double mean = 0., double width = 1., double height =
       1.);
2  Gauss(0.1, mean := 0., width := 2.);
3
4  // same as:
5
6  Gauss(0.1, 0., 2.);
7
8  // potential error:
9
10 Gauss(0.1, width := 2., age := 65.);
```

Pre-San Diego: Parametric Functions (2/2)

- P1229R0 Labelled Parameters
  - Jorg Brown jorg.brown@gmail.com

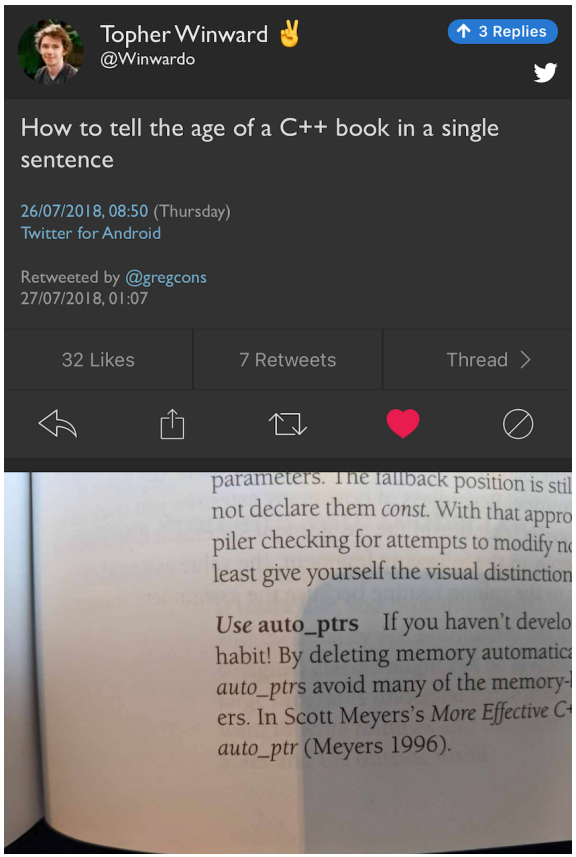
```
1 // declaration
2 void memcpy(to: char *, from: const char *, n: size_t);
3 // call site
4 memcpy(to: buf, from: in, n: bytes);
```

Pre-San Diego: Signed size() functions

- P1227R0
  - Jorg Brown jorg.brown@gmail.com

```
1 template <class C>
2 constexpr ptrdiff_t ssize(const C& c);
3
4 template <class T, ptrdiff_t N>
5 constexpr ptrdiff_t ssize(const T (&array)[N]) noexcept;
```

Twitter


Topher Winward ✌
@Winwardo
3 Replies
How to tell the age of a C++ book in a single sentence
26/07/2018, 08:50 (Thursday)
Twitter for Android
Retweeted by @gregcons
27/07/2018, 01:07
32 Likes
7 Retweets
Thread
parameters. The fallback position is stil
not declare them const. With that appro
piler checking for attempts to modify n
least give yourself the visual distinction
Use auto_ptrs If you haven't develo
habit! By deleting memory automatic
auto_ptrs avoid many of the memory-
ers. In Scott Meyers's More Effective C+
auto_ptr (Meyers 1996).

Twitter

Lars Doucet
@larsiusprime
3 Replies, 64 Quotes
Due to my elite programming skills, I figured out how to shave off THIRTY SECONDS from my app's startup time. Here's some optimization tips:
1. Remove the sleep(30) call you added a month ago and forgot about
29/06/2018, 22:51 (Friday)
TweetDeck
10002 Likes
2066 Retweets
Thread