C++ 17: Beyond the Basics

MODERN C++

Kate Gregory

@gregcons www.gregcons.com/kateblog

What Makes C++ Modern?

Expressive

Choose keywords and constructs that suit specific needs

Fully C++

Don’t reject lambdas, templates, const, or other additions to the language

Readable

Don’t write obscure or opaque code

Stack Semantics

Pointers not your first choice; avoid manual memory management

Fundamentals Review

C++ and Libraries

C++ Standard Library is not the biggest

It is growing

Smaller isn't non existent

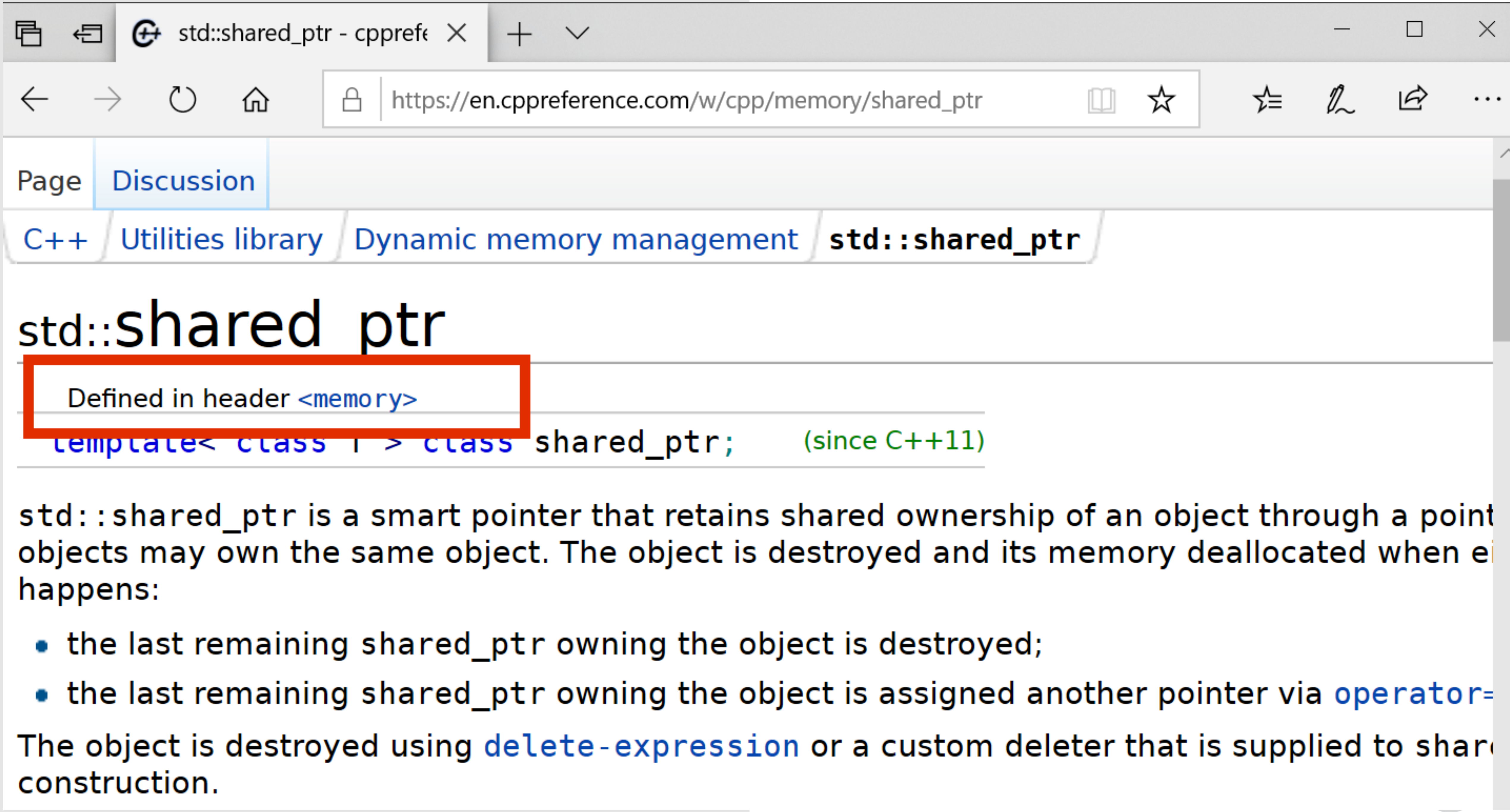
Don't ignore what is there

The Standard Library

Each compiler vendor includes an implementation

Function signatures and performance characteristics set by Standards Committee

Shipped as (only) header files; include what you use


std::shared_ptr - cppref

https://en.cppreference.com/w/cpp/memory/shared_ptr

Page Discussion

C++ Utilities library Dynamic memory management std::shared_ptr

std::shared_ptr

Defined in header <memory>

```
template< class T > class shared_ptr;        (since C++11)
```

std::shared_ptr is a smart pointer that retains shared ownership of an object through a point
objects may own the same object. The object is destroyed and its memory deallocated when ei
happens:

- the last remaining shared_ptr owning the object is destroyed;
- the last remaining shared_ptr owning the object is assigned another pointer via operator=

The object is destroyed using delete-expression or a custom deleter that is supplied to shar
construction.

Standard Library Smart Pointers

shared_ptr

- Reference counted

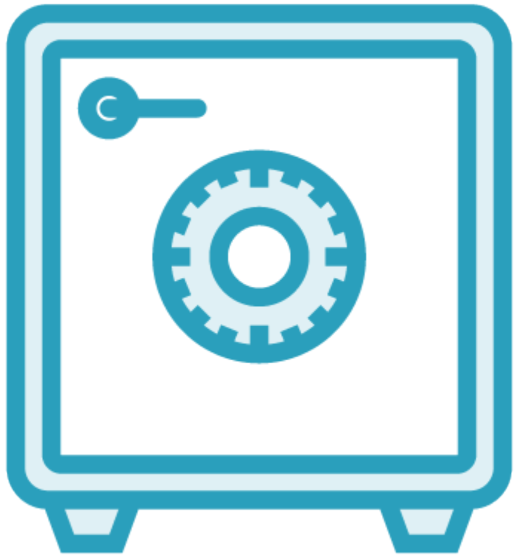
weak_ptr

- Lets you “peek” at a shared_ptr without bumping the reference count

unique_ptr

- Noncopyable (use std::move)

const



A way to commit to the compiler you won't change something

- When declaring a local variable

```
int const zero = 0;
```

- As a function parameter

```
int taxes(int const total)
int something(Person const& p)
```

- Modifier on a member function

```
int GetName() const;
```

Const correctness can be difficult

const: Before or After?

```
int const ci = 3;

const int ci = 3;
```

Compiler doesn’t care

Humans do

The Standards Process

ISO Committee

Study groups and technical specifications

Updates every three years

Standard Releases

C++ 11

- Move semantics and rvalues
- auto
- Range-based for
- Lambdas
- Scoped enums (enum classes)
- Variadic templates
- Defaulted and deleted functions
- Tuple
- Smart pointers

Standard Releases

C++ 14

- Generic lambdas
- Capture expressions in lambdas
- Standard user defined literals

C++ 17

- Structured bindings
- if initializers
- Class template argument deduction
- string_view
- optional
- Parallel algorithms

Summary

Modern C++ is readable and simple

Use the full power of the language and library

Emphasize expressing your intent and minimizing your effort

Very different from “C with Classes” style C++