Copyie Elesion from the C++11 mass

9 Sep 2016

Pete Williamson

C++ 11 is a whole new language

Links to learning more:

http://en.cppreference.com/w/cpp/language/copy_elision

https://engdoc.corp.google.com/eng/doc/devguide/cpp/cpp11.shtml?cl=head

https://chromium-cpp.appspot.com/

http://www.codeproject.com/Articles/570638/Ten-Cplusplus11-Features-Every-Cplusplus-Developer

http://blog.smartbear.com/c-plus-plus/the-biggest-changes-in-c11-and-why-you-should-care/

Perfect Forwarding

You can construct return values in temporary variables that are actually the place being returned to, so no temporary copy is made at all, and no copying to the return value happens. C++ does this all for you under the hood under the name of “Copy Elision”.

lvalue vs rvalue

An lvalue has a name and allocated storage.

A rvalue is a temporary, has storage (which may be short-lived), and *no name*.

&& is a rvalue reference.

```
const std::vector<std::unique_ptr<MyClass>>&&
```

Return Value Optimization

When a nameless temporary, not bound to any references, would be moved or copied into an object of the same type (ignoring top-level cv-qualification), the copy/move is omitted. When that temporary is constructed, it is constructed directly in the storage where it would otherwise be moved or copied to. When the nameless temporary is the argument of a return statement, this variant of copy elision is known as RVO, "return value optimization".

New move constructor

```
// Move constructor.
MemoryBlock(MemoryBlock&& other)
   : _data(nullptr)
   , _length(0)
{
   std::cout << "In MemoryBlock(MemoryBlock&&). length = "
             << other._length << ". Moving resource." << std::endl;

   // Copy the data pointer and its length from the
   // source object.
   _data = other._data;
   _length = other._length;

   // Release the data pointer from the source object so that
   // the destructor does not free the memory multiple times.
   other._data = nullptr;
   other._length = 0;
}
```

Move assignment operator

```
// Move assignment operator.
MemoryBlock& operator=(MemoryBlock&& other)
{
   std::cout << "In operator=(MemoryBlock&&). length = "
             << other._length << "." << std::endl;

   if (this != &other)
   {
      // Free the existing resource.
      delete[] _data;

      // Copy the data pointer and its length from the
      // source object.
      _data = other._data;
      _length = other._length;

      // Release the data pointer from the source object so that
      // the destructor does not free the memory multiple times.
      other._data = nullptr;
      other._length = 0;
   }
   return *this;
}
```

Using std::move()

std::move is really a cast to rvalue reference, which guides the compiler to do a move instead of a copy, it all happens behind the scenes. So, std::move() doesn’t really do anything but cast, the compiler does the magic.

Can’t use const with moving

Normally, we like to return const references to things that we pass across. However, moving a value actually destroys the place it came from, so it isn’t being treated as const.

base::Bind and callbacks

Can’t use const - move will modify old item

Must use const - base::Bind demands it, it makes fresh copies of everything to marshall to other threads.

What to do?

base::Passed()

It does the cast for us, so we don’t also need std::move().

Tells base::Bind to move this parameter instead of copying it.

Pattern for passing something up

Passing something through to another callback

```
void RequestCoordinator::GetQueuedRequestsCallback(
    const GetRequestsCallback& callback,
    RequestQueue::GetRequestsResult result,
    std::vector<std::unique_ptr<SavePageRequest>> requests) {
  callback.Run(std::move(requests));
}
```

Passing something to the first callback

```
   runner->PostTask(FROM_HERE, base::Bind(
       callback, statement.Succeeded(), base::Passed(&requests)));
```

Reference changelist

https://codereview.chromium.org/2262423002/

I return a vector of smartpointers all the way up the chain. You could also return a vector of items the same way

Gotchas

Leaving a “&” somewhere

Marking something const

Template errors very hard to read - inspect your code to make sure all the types are what you think they should be. Then experiment to see what types the compiler likes.

Sample template errors

How to read these: Actual error at the top. Normally, a type somewhere doesn't match declarations. Could also be that base::Bind requires something that you didn't do.

Look for the first line of your code, double check all the types, std::move() or base::Passed()

Look for the word "to"

Sample compile error

In file included from ../../build/linux/debian_wheezy_amd64-sysroot/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../include/c++/4.6/memory:66:

../../build/linux/debian_wheezy_amd64-sysroot/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../include/c++/4.6/bits/stl_construct.h:76:38: error: call to deleted constructor of 'std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >' **(Type mistakenly declared const)**

{ ::new(static_cast<void*>(__p)) _T1(std::forward<_Args>(__args)...); }

^ ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

(and a lot more paragraphs here that look like the following one …)

../../build/linux/debian_wheezy_amd64-sysroot/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../include/c++/4.6/debug/vector:105:9: note: in instantiation of member function 'std::__cxx1998::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > >::vector' requested here

: _Base(__x), _Safe_base(), _M_guaranteed_capacity(__x.size()) { }

^

../../components/offline_pages/background/request_coordinator.cc:136:16: note: in instantiation of member function 'std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > >::vector' requested here

callback.Run(std::move(requests));

^

../../build/linux/debian_wheezy_amd64-sysroot/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../include/c++/4.6/bits/unique_ptr.h:256:7: note: 'unique_ptr' has been explicitly marked deleted here

unique_ptr(const unique_ptr&) = delete;

^

1 error generated.

[16/21] CXX obj/components/offline_pag...adapter/download_notifying_observer.o

Another sample compile error

../../components/offline_pages/background/request_coordinator.cc:136:16: error: no viable conversion from 'vector<offline_pages::SavePageRequest, allocator<offline_pages::SavePageRequest>>' to 'vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >>>'

```
  callback.Run(std::move(requests));
```

(type mistmatch)

x86_64-linux-gnu/4.6/../../../../include/c++/4.6/memory:66:

../../build/linux/debian_wheezy_amd64-sysroot/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../include/c++/4.6/bits/stl_construct.h:76:38: error: call to deleted constructor of 'std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >'

```
    { ::new(static_cast<void*>(__p)) _T1(std::forward<_Args>(__args)...); }
                                       ^   ~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

...

../../components/offline_pages/background/request_coordinator.cc:136:16: note: in instantiation of member function 'std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > >::vector' requested here

```
  callback.Run(requests)
```

(Forgot std::move())

One more compile error

In file included from ../../base/bind.h:8:

../../base/bind_internal.h:265:9: **error**: no viable conversion from 'std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > > *const' to 'std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > >'

```
        std::forward<RunArgs>(args)...);
```

(**about two pages of error messages here ...)**

../../components/offline_pages/background/request_queue_store_sql.cc:308:37: note: in instantiation of function template specialization 'base::Bind<const base::Callback<void (bool, std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > >), base::internal::CopyMode::Copyable, base::internal::RepeatMode::Repeating> &, bool, std::__debug::vector<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> >, std::allocator<std::unique_ptr<offline_pages::SavePageRequest, std::default_delete<offline_pages::SavePageRequest> > > > *>' requested here

```
  runner->PostTask(FROM_HERE, base::Bind(callback, statement.Succeeded(),
                              ^
```

(another page of messages here …)

(forgot base::Passed)

Thanks!

I'll get the slides to Lloyd to put up on the website, and a video of this talk should be available soon.

Send any late breaking questions to petewil@google.com