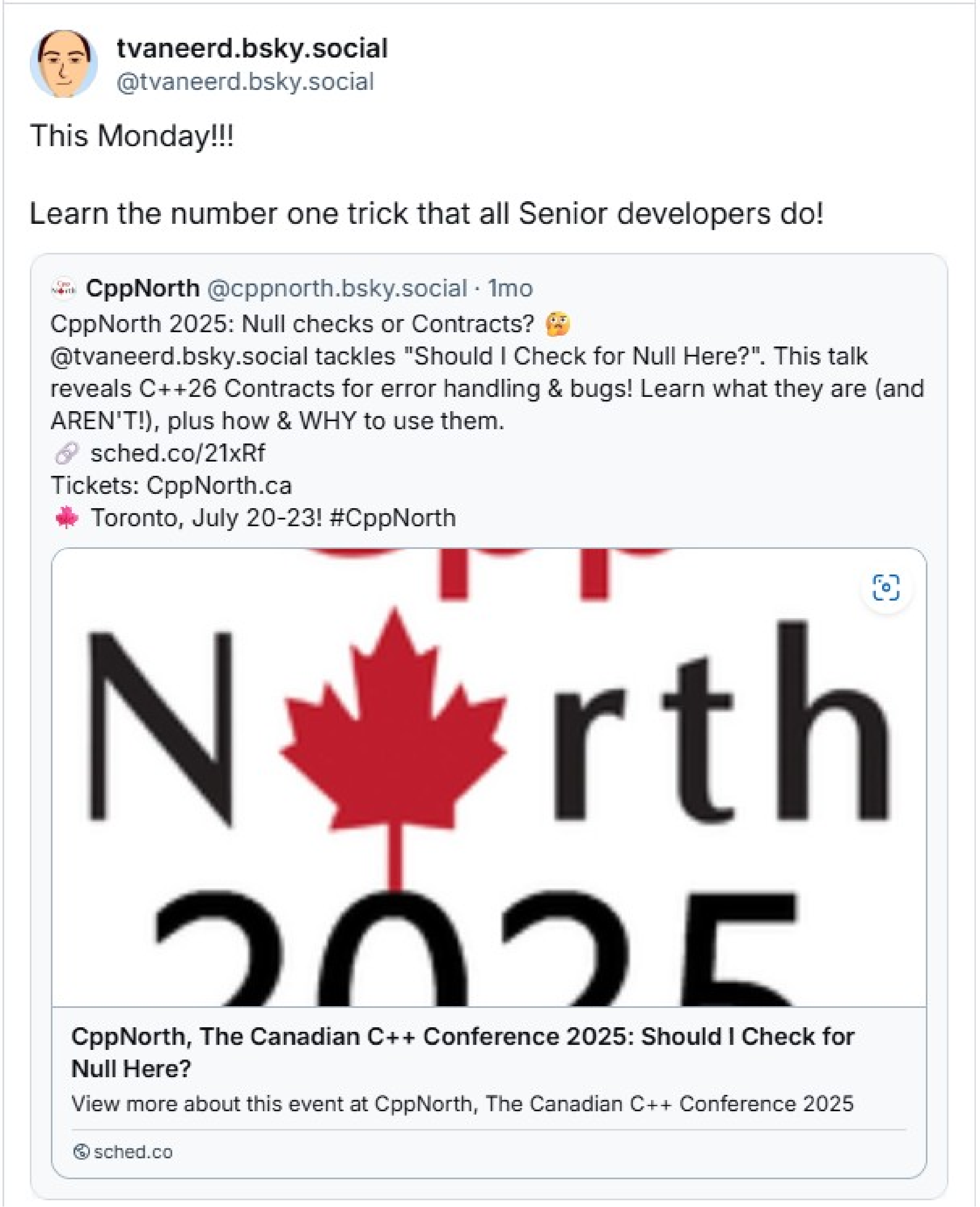
Cpp North 2025

Should I Check for Null Here?

Tony Van Eerd



tvaneerd.bsky.social
@tvaneerd.bsky.social

This Monday!!!

Learn the number one trick that all Senior developers do!

CppNorth @cppnorth.bsky.social · 1mo
CppNorth 2025: Null checks or Contracts? 🤔
@tvaneerd.bsky.social tackles "Should I Check for Null Here?". This talk reveals C++26 Contracts for error handling & bugs! Learn what they are (and AREN'T!), plus how & WHY to use them.
🔗 sched.co/21xRf
Tickets: CppNorth.ca
🍁 Toronto, July 20-23! #CppNorth

CppNorth, The Canadian C++ Conference 2025: Should I Check for Null Here?
View more about this event at CppNorth, The Canadian C++ Conference 2025
sched.co

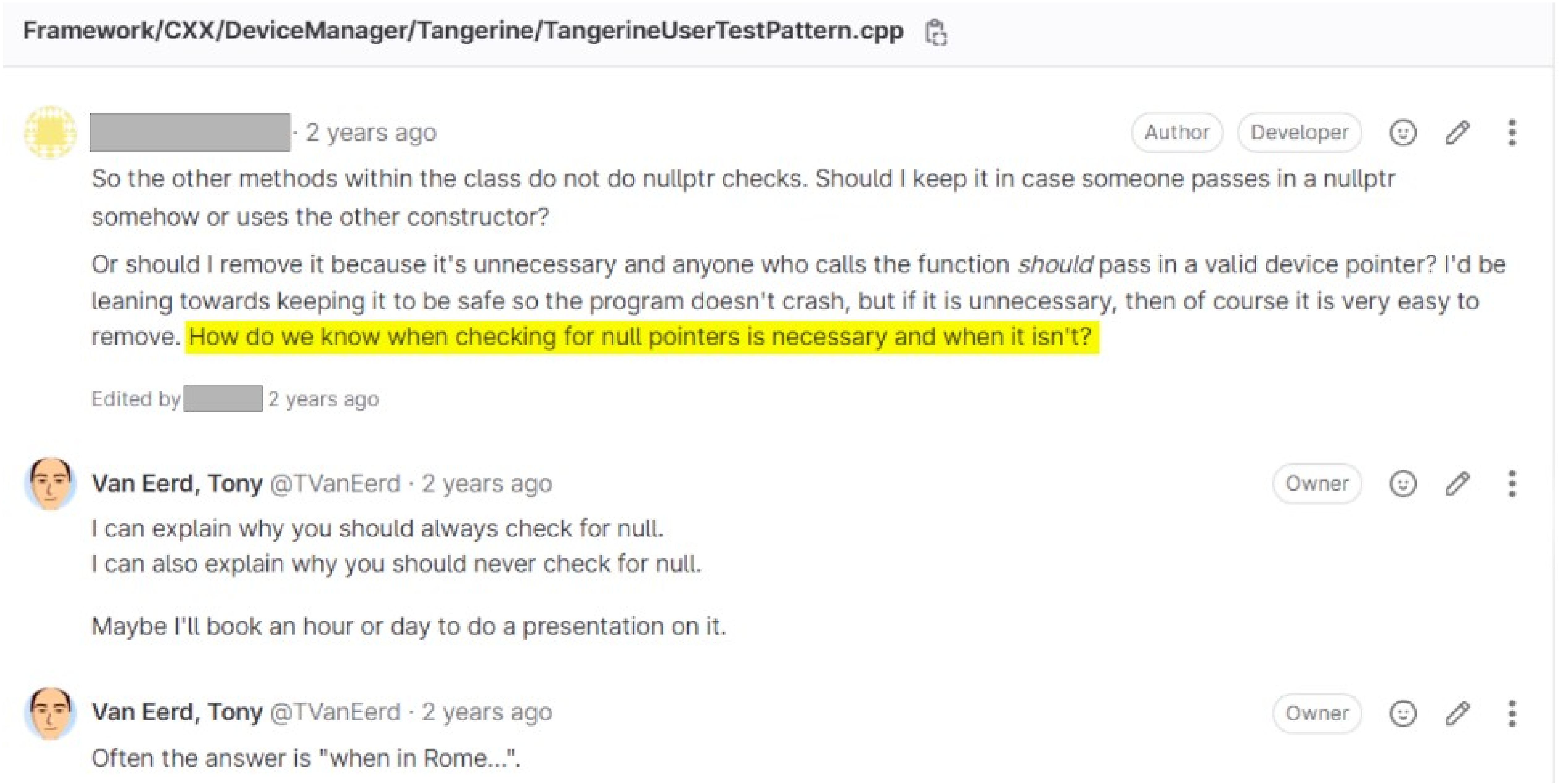
“It Depends”

Should I Check for Null Here?

“It Depends”

The End

A long time ago in a galaxy far,
far away. . . .



Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

[redacted] · 2 years ago — Author · Developer

So the other methods within the class do not do nullptr checks. Should I keep it in case someone passes in a nullptr somehow or uses the other constructor?

Or should I remove it because it's unnecessary and anyone who calls the function *should* pass in a valid device pointer? I'd be leaning towards keeping it to be safe so the program doesn't crash, but if it is unnecessary, then of course it is very easy to remove. How do we know when checking for null pointers is necessary and when it isn't?

Edited by [redacted] 2 years ago

Van Eerd, Tony @TVanEerd · 2 years ago — Owner

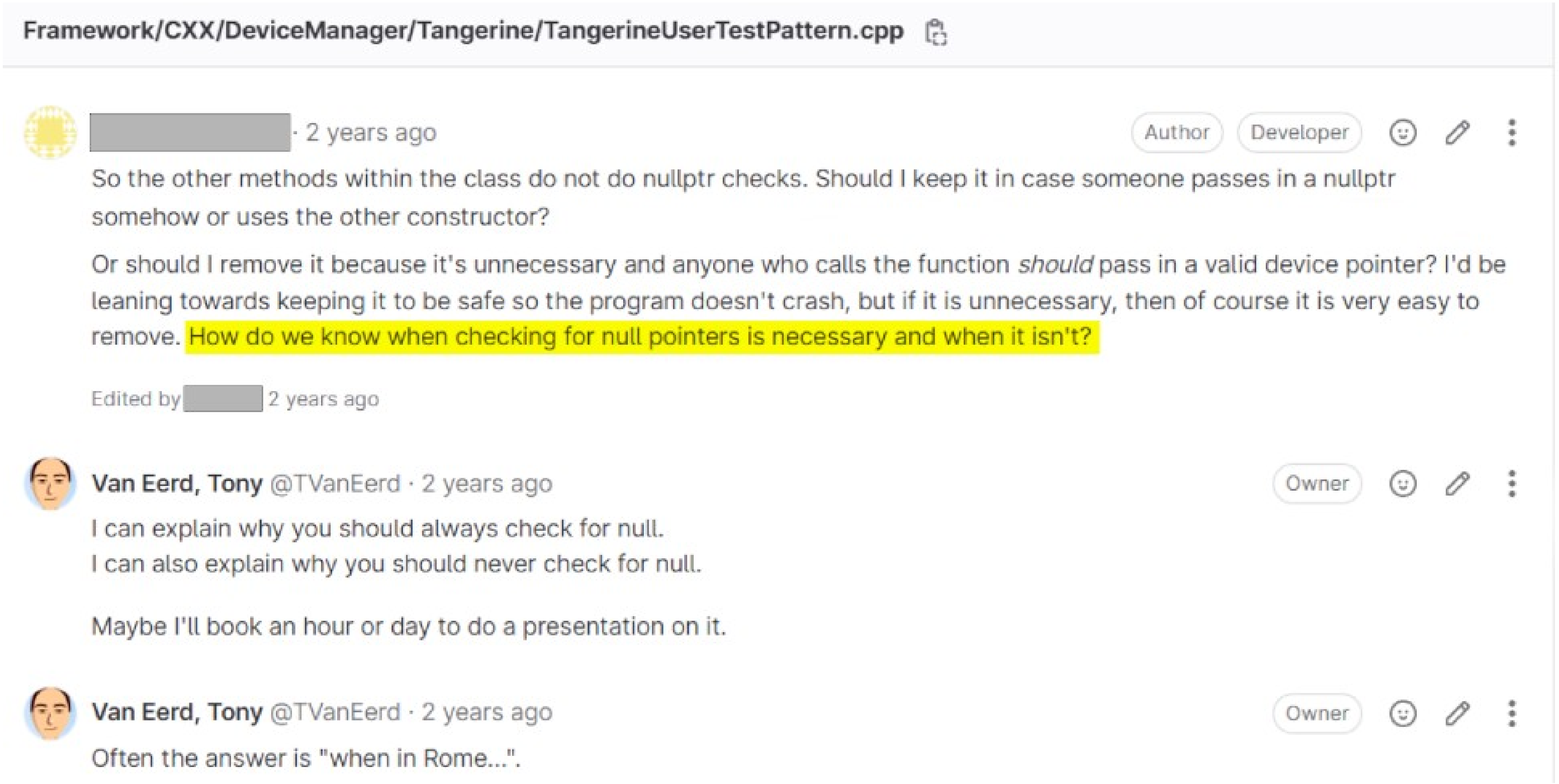
I can explain why you should always check for null.
I can also explain why you should never check for null.

Maybe I'll book an hour or day to do a presentation on it.

Van Eerd, Tony @TVanEerd · 2 years ago — Owner

Often the answer is "when in Rome...".

“It Depends”


Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

[redacted] · 2 years ago

Author Developer

So the other methods within the class do not do nullptr checks. Should I keep it in case someone passes in a nullptr somehow or uses the other constructor?

Or should I remove it because it's unnecessary and anyone who calls the function *should* pass in a valid device pointer? I'd be leaning towards keeping it to be safe so the program doesn't crash, but if it is unnecessary, then of course it is very easy to remove. How do we know when checking for null pointers is necessary and when it isn't?

Edited by [redacted] 2 years ago

Van Eerd, Tony @TVanEerd · 2 years ago

Owner

I can explain why you should always check for null.
I can also explain why you should never check for null.

Maybe I'll book an hour or day to do a presentation on it.

Van Eerd, Tony @TVanEerd · 2 years ago

Owner

Often the answer is "when in Rome...".

Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

Van Eerd, Tony @TVanEerd · 2 years ago Owner

[illegible]

[illegible]

[illegible]

[illegible]

1. [illegible]
2. [illegible]
3. [illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
[illegible]
```

[illegible]

[illegible]

[illegible]

[illegible]

Edited by Van Eerd, Tony 2 years ago

Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

Van Eerd, Tony @TVanEerd · 2 years ago Owner

[illegible]

[illegible]

[illegible]

[illegible] There are two possibilities here:

1. [illegible]
2. [illegible]
3. [illegible]

(haha, off-by-one-error joke)

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

```
[illegible]
```

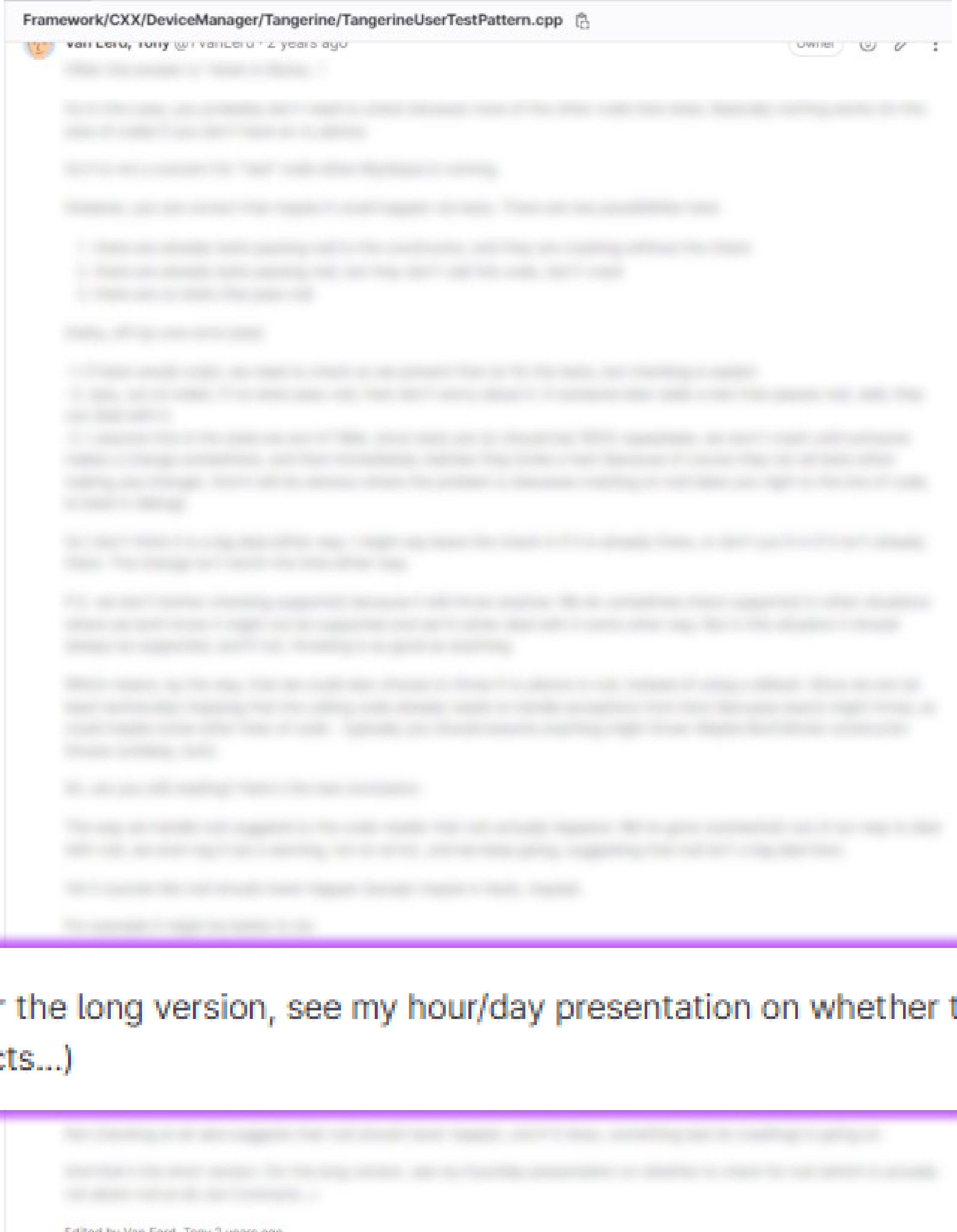
[illegible]

[illegible]

[illegible]

[illegible]

[illegible]



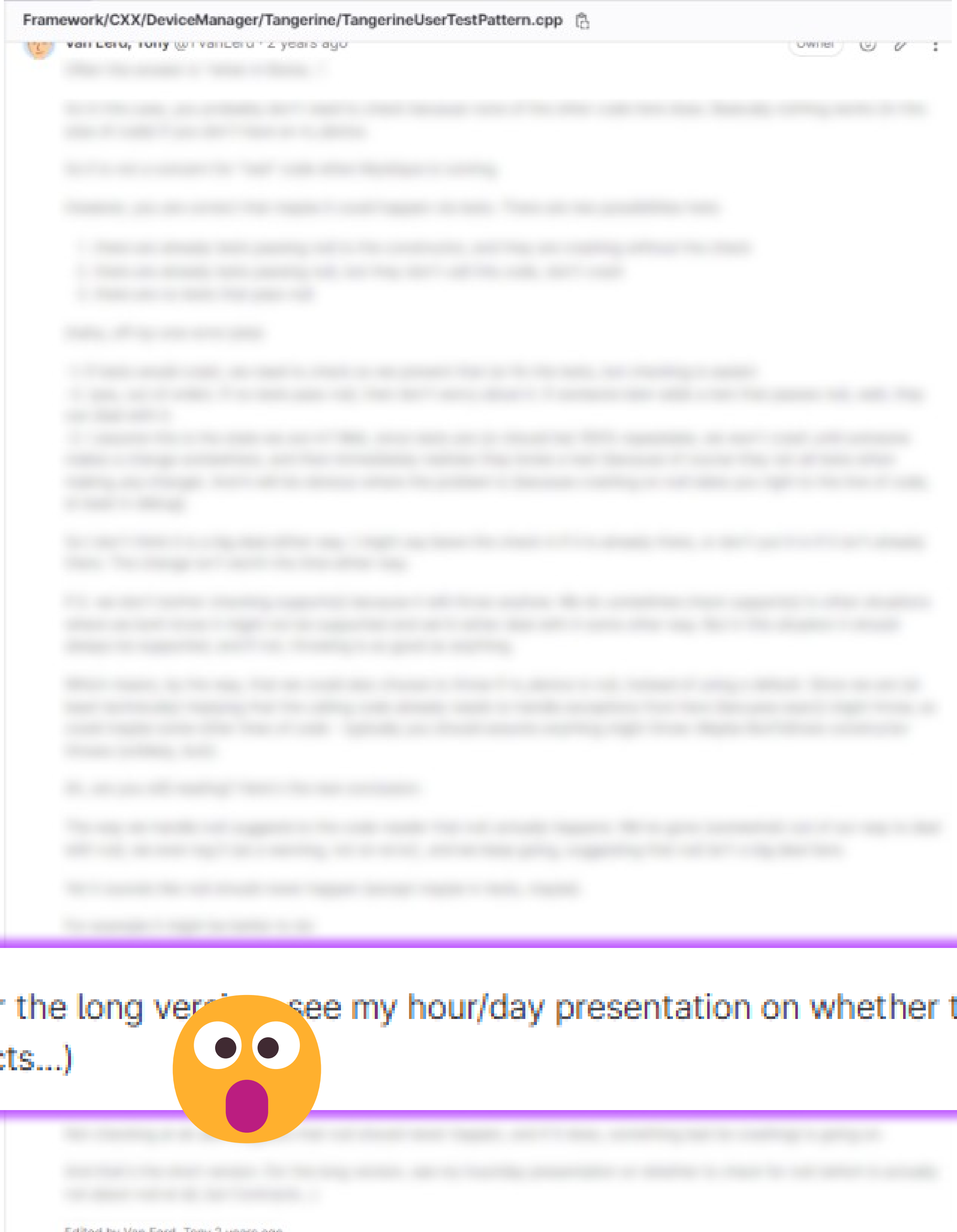
Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

Van Lerd, Tony @TVanLerd · 2 years ago

Owner

[illegible]

And that's the short version. For the long version, see my hour/day presentation on whether to check for null (which is actually not about null at all, but Contracts...)



Framework/CXX/DeviceManager/Tangerine/TangerineUserTestPattern.cpp

Van Eerd, Tony @TVanEerd · 2 years ago

Owner

And that's the short version. For the long ver see my hour/day presentation on whether to check for null (which is actually not about null at all, but Contracts...)

Why should we check for null?

Why Shouldn’t we check for null?

Don’t Crash?

Avoid Undefined Behaviour

Robust (debugging/spelunking)

Viral

Noise

Is it a bug??

Cognitive Load

Program State? Trust?

Partial State (moved from?)

Don’t Crash

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Avoid Undefined Behaviour

```
int someFunction(int index)
{
  if (index >= 0 && index < length)
        buffer[index]...


};
```

“It Depends”

Avoid Undefined Behaviour

```
int someFunction(int index)
{
  if (index >= 0 && index < length)
        buffer[index]...



};
```

Robust (debugging/spelunking)

Robust (debugging/spelunking)

inscriber TECHNOLOGY

CHRISTIE

Robust (debugging/spelunking)

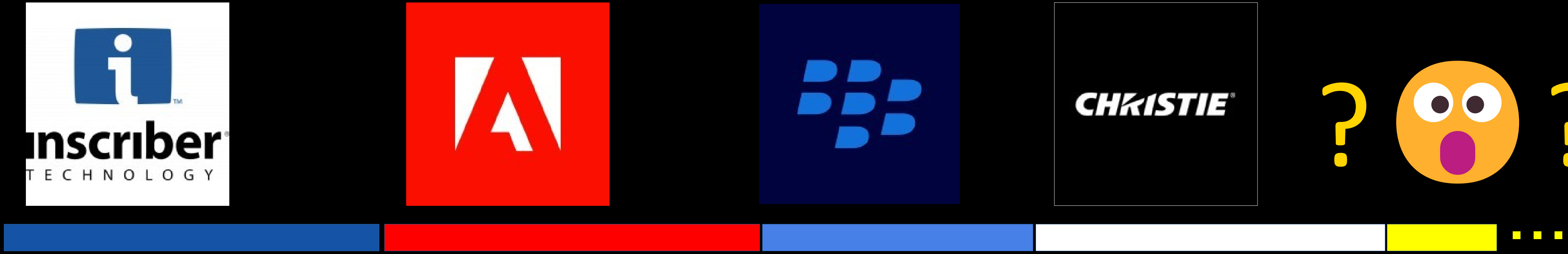
inscriber TECHNOLOGY

CHRISTIE

? ?

...

Robust (debugging/spelunking)



```
int someFunction()
{
  if (foo != nullptr)
        foo->...


};
```

Viral

Viral

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int oneMoreFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral
Noise

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int oneMoreFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral / Noise

Is it a BUG???

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int oneMoreFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral / Noise

Is it a BUG???

```
int someFunction()
{
    m_device->...


};
```

```
int anotherFunction()
{
    m_device->...


};
```

```
int oneMoreFunction()
{
    m_device->...


};
```

Viral / Noise

Is it a BUG???

```
int someFunction()
{
    m_device->...


};
```

```
int anotherFunction()
{
    m_device->...


};
```

```
int yetAnotherFunction()
{
    int x;

    int y = x + 2;


};
```

```
int oneMor
{
    m_device->...


};
```

Viral / Noise

Is it a BUG???

```
int someFunction()
{
    m_device->...


};
```

```
int anotherFunction()
{
    m_device->...


};
```

```
int oneMor
{
    m_device->...


};
```

```
int yetAnotherFunction()
{
    int x;

    int y = x + 2;


};
```

Erroneous Behaviour

Viral / Noise

Is it a BUG???

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int oneMoreFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral / Noise Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int anotherFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

```
int oneMoreFunction()
{
  if (m_device != nullptr)
        m_device->...


};
```

Viral / Noise

Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->…


  return ????

};
```

Viral / Noise

Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->…


  return ????

};
```

Do you TRUST Program State?

Viral / Noise

Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->…


  return ????

};
```

```
Class Projector
{
    AbstractDevice * m_device;
    ...
    ...


};
```

Do you TRUST Program State?

Viral / Noise

Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->…


  return ????

};
```

```
Class Projector
{
    AbstractDevice * m_device;
    ...
    ...


};
```

Do you TRUST Program State?

Partial State?

Viral / Noise

Is it a BUG???

Cognitive Load

```
int someFunction()
{
  if (m_device != nullptr)
        m_device->…


  return ????

};
```

```
Class Projector
{
    AbstractDevice * m_device;
    ...
    ...


};
```

Do you TRUST Program State?

Partial State? (moved-from?)

Don’t Crash?

Avoid Undefined Behaviour

Robust (debugging/spelunking)

Viral

Noise

Is it a bug??

Cognitive Load

Program State? Trust?

Partial State (moved from?)

“It Depends”

Error Handling

- what
- who

Error Handling
WHAT

Error Handling

WHAT

Function didn’t “succeed”

Program didn’t do what you wanted

Error Handling

WHAT

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

User Input

File not found

…

Error Handling

WHAT

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

User Input

File not found

…

Expected

Unexpected

Error Handling

WHAT

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

User Input

File not found

…

Expected

Unexpected (bugs)

Error Handling

WHAT

WHO??

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

Expected

User Input

Unexpected (bugs)

File not found

…

Error Handling
WHO

Error Handling
WHO

```
int someFunction()
{
  if (something_is_bad())
        ...


};
```

```
int someFunction()
{
  if (something_is_bad())
         ...

};
```

Error Handling
WHO

C –
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        ...
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        return -1;
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
};
```

Error Handling

WHO

```
std::expected<string, Excuse> someFunction();
```

C – Calling Code
D –
F –
U –

Error Handling

WHO??

WHAT

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

Expected

User Input

Unexpected (bugs)

File not found

…

Error Handling

WHAT

WHO??

Function didn’t “succeed”

Program didn’t do what you wanted

Bug

User Input

File not found

…

Expected
PLANNED?
NORMAL?

Unexpected
UNPLANNED?
ABNORMAL?

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
        - throw some_exception();
};
```

Error Handling

WHO

C – Calling Code

D –

F –

U –

```
auto someFunction()
{
  if (something_NORMAL_bad())
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
        - throw some_exception();
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

“It Depends”

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
        …???
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        …???
};
```

Error Handling
WHO

C – Calling Code
D –
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        …???
};
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

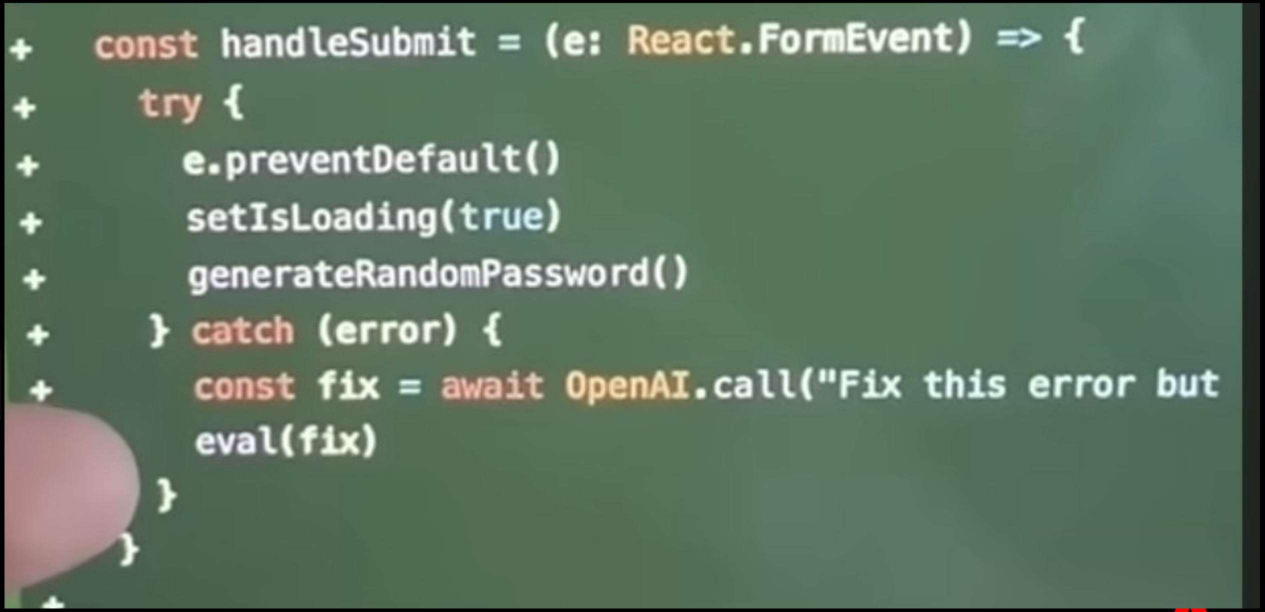
Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
      …???
};
```

Error Handling



```
+   const handleSubmit = (e: React.FormEvent) => {
+     try {
+       e.preventDefault()
+       setIsLoading(true)
+       generateRandomPassword()
+     } catch (error) {
+       const fix = await OpenAI.call("Fix this error but
        eval(fix)
      }
    }
```

C – Call

D – Call

F –

U –

expected

planned

normal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        email_calling_dev(x,y);
};
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        - email_calling_dev(x,y);
        - log();
};
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        - email_calling_dev(x,y);
        - log();
        - terminate();
};
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        - email_calling_dev(x,y);
        - log();
        - terminate();
        - throw std::logic_error();
};
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        - email_calling_dev(x,y);
        - log();
        - terminate();
        - throw std::logic_error();
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
        - throw some_exception();
```

Error Handling

WHO

de

D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction(Foo x, Bar y)
{
  if (something_ABNORMAL_bad(x, y))
        - email_calling_dev(x,y);
        - log();
        - terminate();
        - throw std::logic_error();
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
        - throw some_exception();
        - return in_range_value;
```

Error Handling

WHO

de

D – Calling DEVELOPER

F –

U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction()
{
  if (something_ABNORMAL_bad(this))
          ...
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F –
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction()
{
  if (something_ABNORMAL_bad(this))
        ...
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction()
{
  if (something_ABNORMAL_bad(this))
        email_function_author();
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction()
{
  if (something_ABNORMAL_bad(this))
        - email_function_author();
        - log();
        - terminate();
        - throw std::logic_error();
        - return -1;
        - return E_PRINTER_ON_FIRE;
        - return std::nullopt;
        - return std::unexpected(…);
        - throw some_exception();
        - return in_range_value;
```

Error Handling

WHO

de

D – Calling DEVELOPER

F – Function Author

U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

“It Depends”

```
auto someFunction()
{
  if (something_ABNORMAL_bad(this))
        - email_calling_developer();
        - email_function_author();
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U –

Unexpected
Unplanned
Abnormal

A BUG

code change needed

```
auto someFunction()
{
  if (something_bad())
        ...
```

Error Handling
WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U –

```
auto someFunction()
{
  if (something_bad())
        ...
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Expected
PLANNED?
NORMAL?

Unexpected
Unplanned
Abnormal
BUG

```
auto someFunction()
{
  if (something_bad())
        ...
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Expected
PLANNED?
NORMAL?

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
        ...
```

Error Handling

WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Expected
PLANNED?
NORMAL?

Unexpected
Unplanned
Abnormal
BUG

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
        ...
```

Error Handling

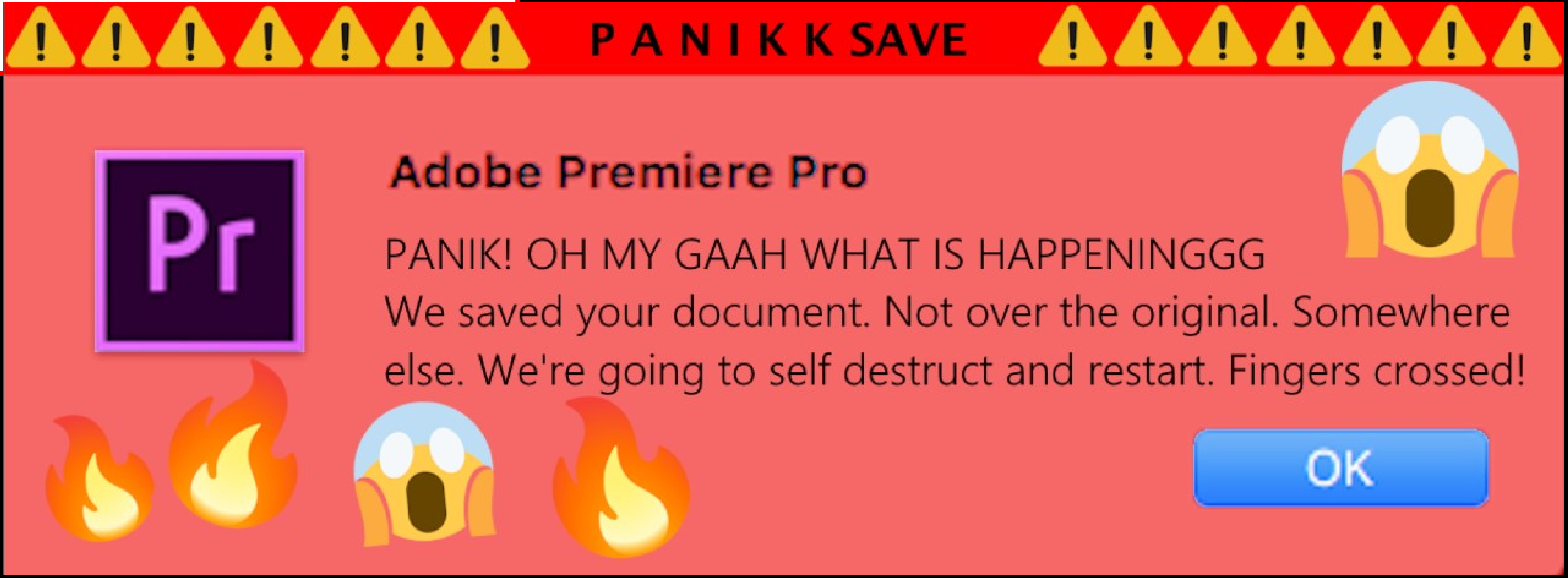
WHO

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Expected
PLANNED?
NORMAL?

Unexpected
Unplanned
Abnormal
BUG

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
```



P A N I K K SAVE

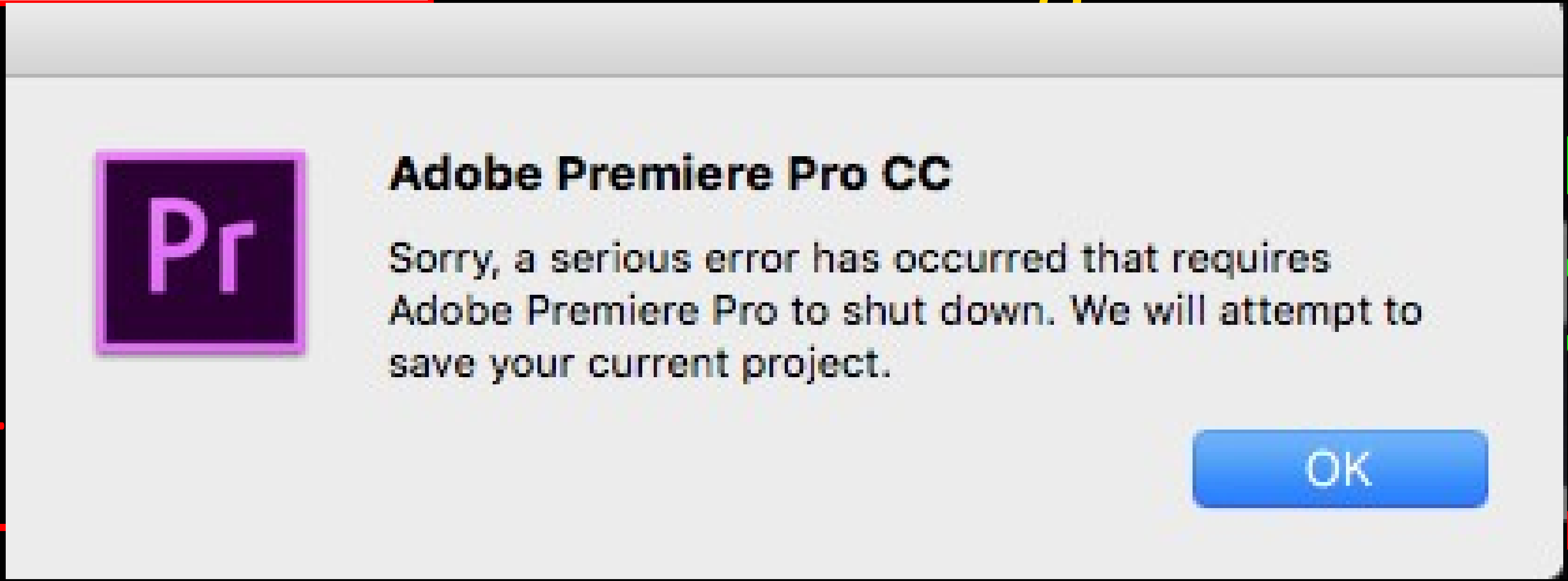
Adobe Premiere Pro

PANIK! OH MY GAAH WHAT IS HAPPENINGGG
We saved your document. Not over the original. Somewhere else. We're going to self destruct and restart. Fingers crossed!

Abnormal
BUG

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
        ...
```

Error Handling


Adobe Premiere Pro CC

Sorry, a serious error has occurred that requires Adobe Premiere Pro to shut down. We will attempt to save your current project.

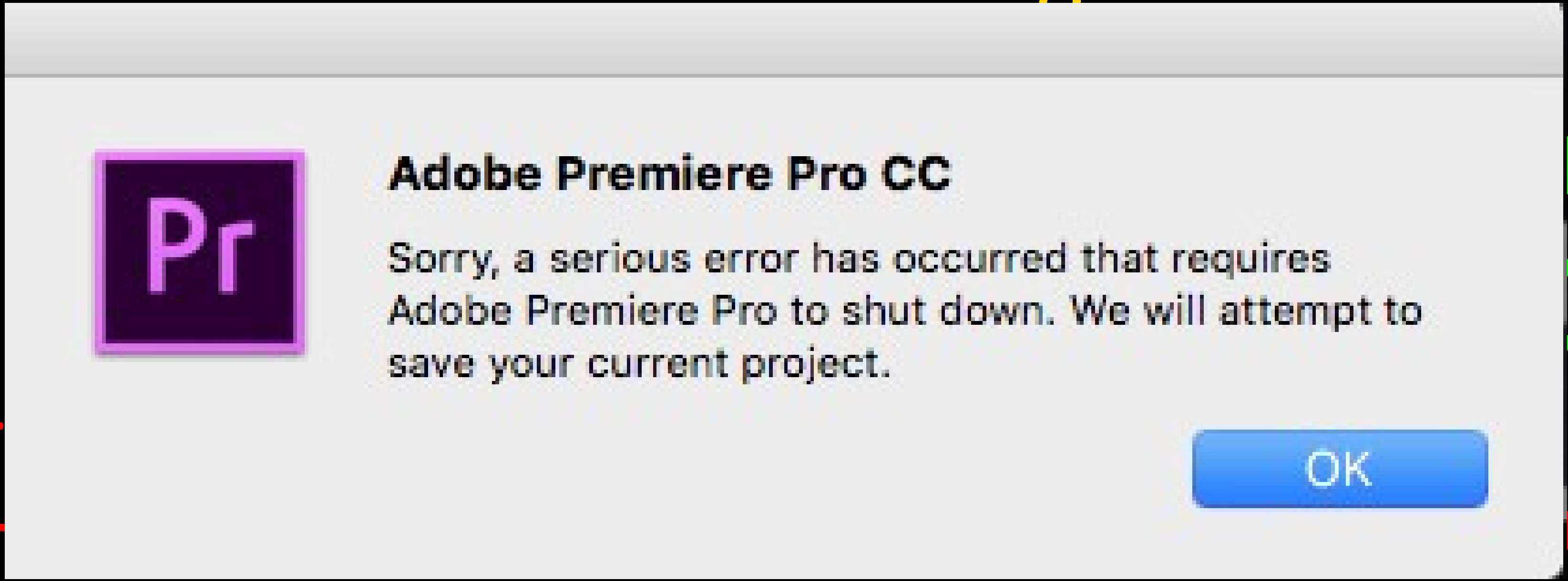
C –
D –
F –
U – End User

ted
NED?
IAL?

ected
Unplanned
Abnormal
BUG

```
auto someFunction()
{
  REQUIRES(x > 0 && p != nullptr);
```

Error Handling


Adobe Premiere Pro CC

Sorry, a serious error has occurred that requires Adobe Premiere Pro to shut down. We will attempt to save your current project.

C –

D –

F –

U – End User

ted
NED?
IAL?

ected
Unplanned
Abnormal
BUG

```
auto someFunction()
{
  if (something_ABNORMAL_bad())
        ...
```

Is it a BUG??

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Unexpected
Unplanned
Abnormal
BUG

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

check the docs! 😊

Is it a BUG??

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Unexpected
Unplanned
Abnormal
BUG

```
auto someFunction(int x, Foo * ptr)
{
if (something_ABNORMAL_bad(x,ptr))
        ...
```

Is it a BUG??

check the code 😬

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Unexpected
Unplanned
Abnormal
BUG

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

check the docs! 😊

Is it a BUG??

C – Calling Code
D – Calling DEVELOPER
F – Function Author
U – End User

Unexpected
Unplanned
Abnormal
BUG

Contracts

Contracts

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

this is a contract

Contracts

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

this is TWO contracts

Contracts

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

this is TWO contracts

Another contract

Contracts

Yet another contract

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
int someFunction(int x, Foo * ptr);
```

this is TWO contracts

Another contract

Contracts

Yet another contract

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
// returns highest prime < x
int someFunction(int x, Foo * ptr);
```

this is TWO contracts

Another contract

contract

Contracts

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
// returns highest prime < x
int someFunction(int x, Foo * ptr);
```

😢

- not parsed
-

Contracts

```
//
// does some thing
// requires x > 0, ptr != null
// requires thing to be valid
// returns highest prime < x
int someFunction(int x, Foo * ptr);
```

- not parsed
    - falls out of sync with code
    - limits possibilities...

Contracts - C++26

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

- C++ Code!!
- stays in sync better
- unlocks possibilities...

Contracts - C++26

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

😍

- C++ Code!!
- stays in sync better
- unlocks possibilities…

But what does it actually DO?

“It Depends”

Contracts - C++

😍

- C++ Co
- stays in
- unlocks possibilities…

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
✅ - email_function_author();
✅ - log();
✅ - terminate();
✅ - throw std::logic_error();
✅ - throw some_exception();
🚫 - return -1;
🚫 - return E_PRINTER_ON_FIRE;
🚫 - return std::nullopt;
🚫 - return std::unexpected(…);
🚫 - return in_range_value;
```

But what does it actually DO?

- whatever handler() wants

Contracts - C++

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
✓ - email_function_author();
✓ - log();
✓ - terminate();
✓ - throw std::logic_error();
✓ - throw some_exception();
🚫 - return -1;
🚫 - return E_PRINTER_ON_FIRE;
🚫 - return std::nullopt;
🚫 - return std::unexpected(…);
🚫 - return in_range_value;
```

😍

- C++ Co
- stays in
- unlocks possibilities…

But what does it actually DO?
- whatever handler() wants

Why can’t return??

Contracts - C++

```
✓ - email_function_author();
✓ - log();
ate();
std::logic_error();
some_exception();
 -1;
 E_PRINTER_ON_FIRE;
 std::nullopt;
 std::unexpected(…);
 in_range_value;
```

```
//
// does some thing
int someFunction(int 
   pre(x > 0)
   pre(ptr != nullptr
   pre(is_valid(thing
   post(r : is_highest
```

ties…

ctually DO?

r() wants

```
void handle_contract_violation( std::contracts::contract_violation );
```

Why can’t return??

Contracts - C++

- C++ Co
- stays in
- unlocks

```
- email_function_author();
- log();
- terminate();
- throw std::logic_error();
- throw some_exception();
- return -1;
- return E_PRINTER_ON_FIRE;
- return std::nullopt;
- return std::unexpected(…);
- return in_range_value;

- Absolutely nothing
```

✓	- **email_function_author();**
✓	- log();
✓	- terminate();
✓	- throw std::logic_error();
✓	- throw some_exception();
🚫	- return -1;
🚫	- return E_PRINTER_ON_FIRE;
🚫	- return std::nullopt;
🚫	- return std::unexpected(…);
🚫	- return in_range_value;
✓	- **Absolutely nothing**

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

But what does it actually DO?

```
void handle_contract_violation( std::contracts::contract_violation );
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

😍

- C++ Co
- stays in
- unlocks

```
✅ - email_function_author();
✅ - log();
✅ - terminate();
✅ - throw std::logic_error();
✅ - throw some_exception();
🚫 - return -1;
🚫 - return E_PRINTER_ON_FIRE;
🚫 - return std::nullopt;
🚫 - return std::unexpected(…);
🚫 - return in_range_value;

✅ - Absolutely nothing
```

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

But what does it actually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

```
static int g;
int f()
   pre(g++ >= 0);
```

```
✅ - email_function_author();
✅ - log();
     e();
     d::logic_error();
     me_exception();
     1;
     _PRINTER_ON_FIRE;
🚫 - return std::nullopt;
🚫 - return std::unexpected(…);
🚫 - return in_range_value;

✅ - Absolutely nothing
```

- stays in
- unlocks

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

But what does it actually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

```
✓ - email_function_author();
✓ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing
```

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✓	✓	🚫
quickenforce	✓	🚫	✓
enforce	✓	✓	✓ *

Custom...

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented

tually DO?

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

```
 ✓ - email_function_author();
 ✓ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing
```

tually DO?

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✓	✓	🚫
quickenforce	✓	🚫	✓
enforce	✓	✓	✓ *

Custom...

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented
the rest (eval) -

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

✅ **- email_function_author();**

✅ - log();

e();

d::logic_error();

me_exception();

1;

_PRINTER_ON_FIRE;

td::nullopt;

td::unexpected(…);

n_range_value;

ly nothing

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented

the rest (eval) -

– g *probably* incremented

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

```
✅ - email_function_author();
✅ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing
```

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

```
static int g;
int f()
   pre(g++ >= 0);
```

```
ignore (non eval) – g not incremented
the rest (eval) -
- g probably incremented
- g maybe incremented MORE THAN ONCE
```

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✓	✓	🚫
quickenforce	✓	🚫	✓
enforce	✓	✓	✓ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented
the rest (eval) -
- g *probably* incremented
- g *maybe* incremented MORE THAN ONCE
- g *maybe* NEVER incremented

```
✓ - email_function_author();
✓ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;
```

ly nothing

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✓	✓	🚫
quickenforce	✓	🚫	✓
enforce	✓	✓	✓ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented
the rest (eval) -
- g *probably* incremented
- g *maybe* incremented MORE THAN ONCE
- g *maybe* NEVER incremented
    - if compiler can prove predicate
      (ie no other code decreases g)

✓ - **email_function_author();**
✓ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✓	✓	🚫
quickenforce	✓	🚫	✓
enforce	✓	✓	✓ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
✓ - email_function_author();
✓ - log();
                   e();
                   d::logic_error();
                   me_exception();
                   1;
                   _PRINTER_ON_FIRE;
                   td::nullopt;
                   td::unexpected(…);
                   n_range_value;

                   ly nothing
```

```
static int g;
int f()
   pre(g++ >= 0);
```

ignore (non eval) – g not incremented
the rest (eval) -
– g *probably* incremented
- g *maybe* incremented MORE THAN ONCE
- g *maybe* NEVER incremented
    - if compiler can prove predicate
      (ie no other code decreases g)

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
✅ - email_function_author();
✅ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing
```

```
static int g;
int f()
   pre(const_cast<int&>(g)++ >= 0);
```

ignore (non eval) – g not incremented
the rest (eval) -
- g *probably* incremented
- g *maybe* incremented MORE THAN ONCE
- g *maybe* NEVER incremented
  - if compiler can prove predicate (ie no other code decreases g)

tually DO?

```
void handle_contract_violation(contract_violation);
```

Contracts - C++

Contract Evaluation Semantics

Setting	eval	handler	terminate*
ignore	🚫	🚫	🚫
observe	✅	✅	🚫
quickenforce	✅	🚫	✅
enforce	✅	✅	✅ *

Custom...

```
//
// does some thing
int someFunction(int x, Foo * ptr)
   pre(x > 0)
   pre(ptr != nullptr)
   pre(is_valid(thing))
   post(r : is_highest_prime(r,x));
```

```
✅ - email_function_author();
✅ - log();
e();
d::logic_error();
me_exception();
1;
_PRINTER_ON_FIRE;
td::nullopt;
td::unexpected(…);
n_range_value;

ly nothing
```

```
static int g;
int f()
   pre(const_cast<int&>(g)++ >= 0);
```

😬

ignore (non eval) – g not incremented
the rest (eval) -
- g *probably* incremented
- g *maybe* incremented MORE THAN ONCE
- g *maybe* NEVER incremented
  - if compiler can prove predicate
    (ie no other code decreases g)

ctually DO?

```
void handle_contract_violation(contract_violation);
```

“terminate” – contract-terminate
terminate() or abort() or just stop
Enforce ✅* – doesn’t terminate if handler throws

“It Depends”

So

“Should I check for null here?”

```
int someFunction(Foo * ptr)
{
    if (ptr == nullptr)
        …

 ...
```

~~“It Depends”~~

No

```
int someFunction(Foo * ptr)
{
    if (ptr == nullptr)
        …

 ...
```

~~“It Depends”~~

No

← Check here

```
//
// does some thing
int someFunction(Foo * ptr)
   pre(ptr != nullptr);
```

```
int someFunction(Foo * ptr)
{



   ...
```

The End

The End

“It Depends”

The End

“It Depends”

The End

“It Depends”

The End

“It Depends”

The End

“It Depends”

The End

“It Depends”

“It Depends”

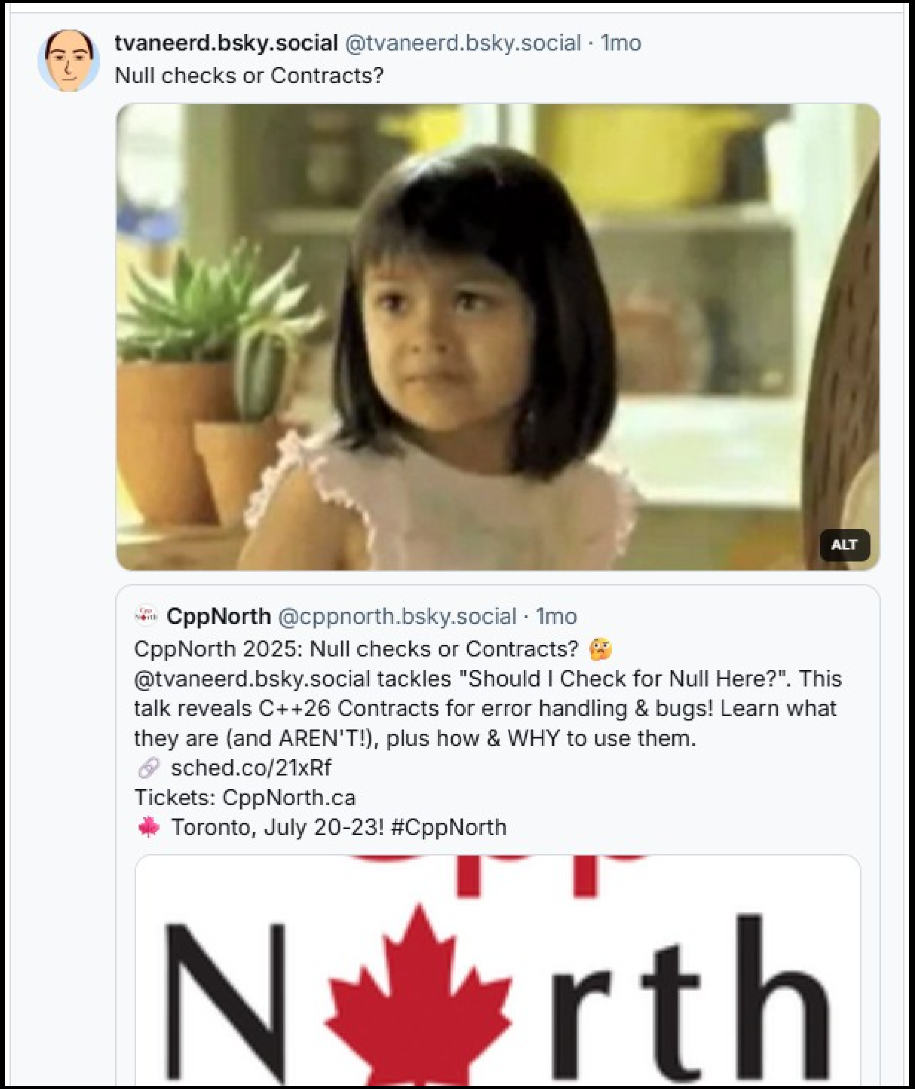
"It Depends"

```
//
// does some thing
int someFunction(Foo * ptr)
   pre(ptr != nullptr);
```

```
int someFunction(Foo * ptr)
{
    if (ptr == nullptr)
        ...


   ...
```

Why
Not
Both?


tvaneerd.bsky.social @tvaneerd.bsky.social · 1mo
Null checks or Contracts?

CppNorth @cppnorth.bsky.social · 1mo
CppNorth 2025: Null checks or Contracts? 🫣
@tvaneerd.bsky.social tackles "Should I Check for Null Here?". This talk reveals C++26 Contracts for error handling & bugs! Learn what they are (and AREN'T!), plus how & WHY to use them.
🔗 sched.co/21xRf
Tickets: CppNorth.ca
🍁 Toronto, July 20-23! #CppNorth

“It Depends”

```
//
// does some thing
int someFunction(Foo * ptr)
   pre(ptr != nullptr);
```

```
int someFunction(Foo * ptr)
{
    if (ptr == nullptr)
        ...


   ...
```

tvaneerd.bsky.social @tvaneerd.bsky.social · 1mo
Null checks or Contracts?

th @cppnorth.bsky.social · 1mo
2025: Null checks or Contracts? 😮
.bsky.social tackles "Should I Check for Null Here?". This
C++26 Contracts for error handling & bugs! Learn what
nd AREN'T!), plus how & WHY to use them.
:o/21xRf
pNorth.ca
, July 20-23! #CppNorth

tvaneerd.bsky.social @tvaneerd.bsky.social · now

But Hyrum's Law

Any observable behavior of a software system, even if undocumented and unintended, will be relied upon by users if the system has a sufficient number of users.

```
//
// does some thing
int someFunction(Foo * ptr)
   pre(ptr != nullptr);
```

```
int someFunction(Foo * ptr)
{
    if (ptr == nullptr)
        load-bearing-code();


   ...
```

tvaneerd.bsky.social @tvaneerd.bsky.social · 1mo
Null checks or Contracts?

th @cppnorth.bsky.social · 1mo
2025: Null checks or Contracts? 🫣
.bsky.social tackles "Should I Check for Null Here?". This
C++26 Contracts for error handling & bugs! Learn what
nd AREN'T!), plus how & WHY to use them.
:o/21xRf
pNorth.ca
, July 20-23! #CppNorth

tvaneerd.bsky.social @tvaneerd.bsky.social · now

“It Depends”

“It Depends”

(The End)

“It Depends”

(The End)

(for real)

Questions?

“It Depends”

Extra

- Unit tests
- Provability

Should I Check for Null Here?

(a talk about Contracts, shhhh)

Tony Van Eerd

CppNorth 2025