Embracing Standard C++ for the Windows Runtime

Kenny Kerr

Microsoft Windows

@KennyKerr

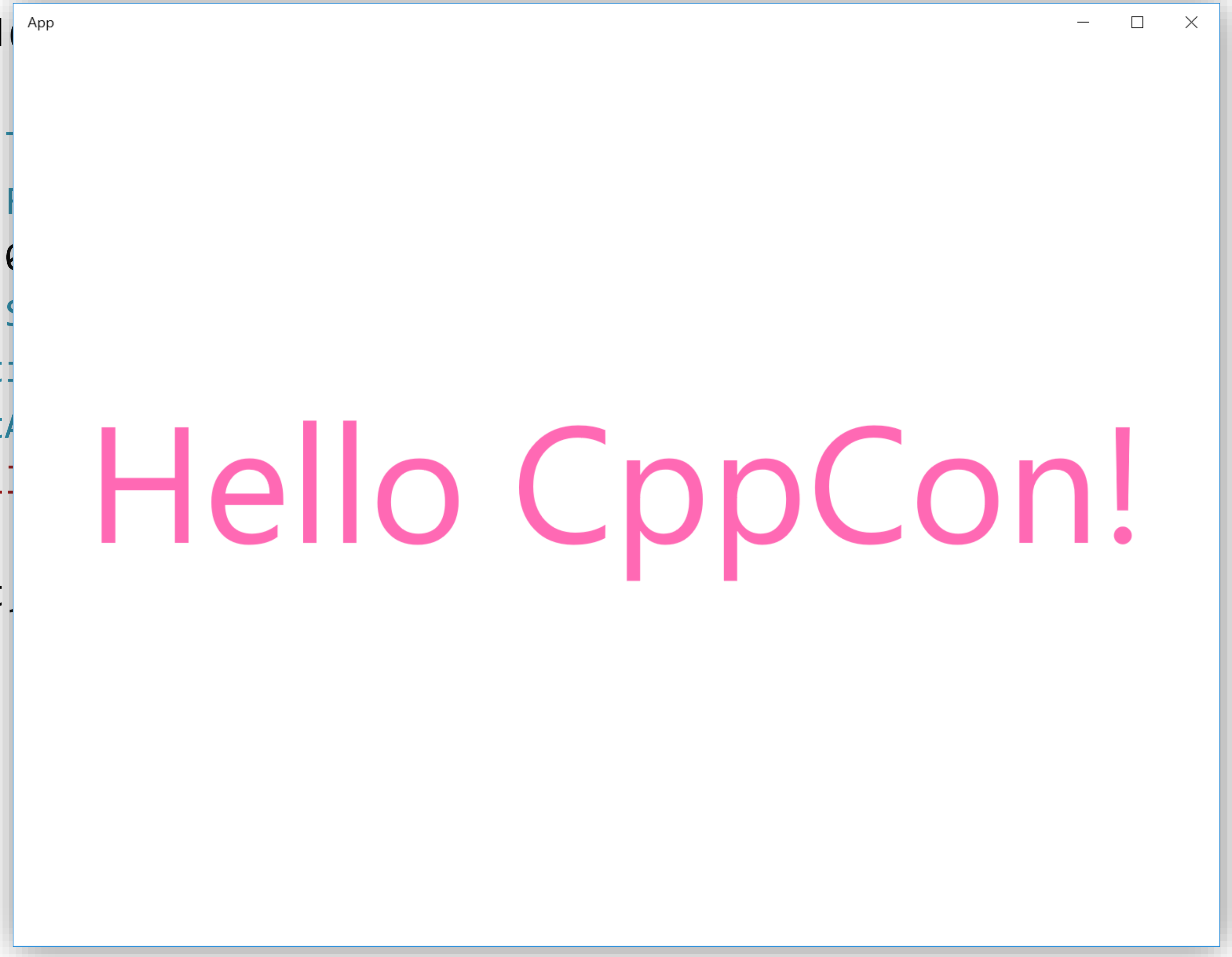
James McNellis

Microsoft Visual C++

@JamesMcNellis

“Hello, World!”

```
sealed partial class App : Application
{
    protected override void OnLaunched(
    {
        TextBlock block              = new T
        block.FontFamily             = new F
        block.FontSize               = 140.0
        block.Foreground             = new S
        block.VerticalAlignment = Verti
        block.TextAlignment          = TextA
        block.Text                   = "Hell

        Window window = Window.Current
        window.Content = block;
        window.Activate();
    }
}
```



```
sealed partial class App : Application
{
    protected override void OnLaunched(LaunchActivatedEventArgs e)
    {
        TextBlock block           = new TextBlock();
        block.FontFamily          = new FontFamily("Segoe UI Semibold");
        block.FontSize            = 140.0;
        block.Foreground          = new SolidColorBrush(Colors.HotPink);
        block.VerticalAlignment = VerticalAlignment.Center;
        block.TextAlignment       = TextAlignment.Center;
        block.Text                = "Hello CppCon!";

        Window window = Window.Current;
        window.Content = block;
        window.Activate();
    }
}
```

C++ “Hello, World!”

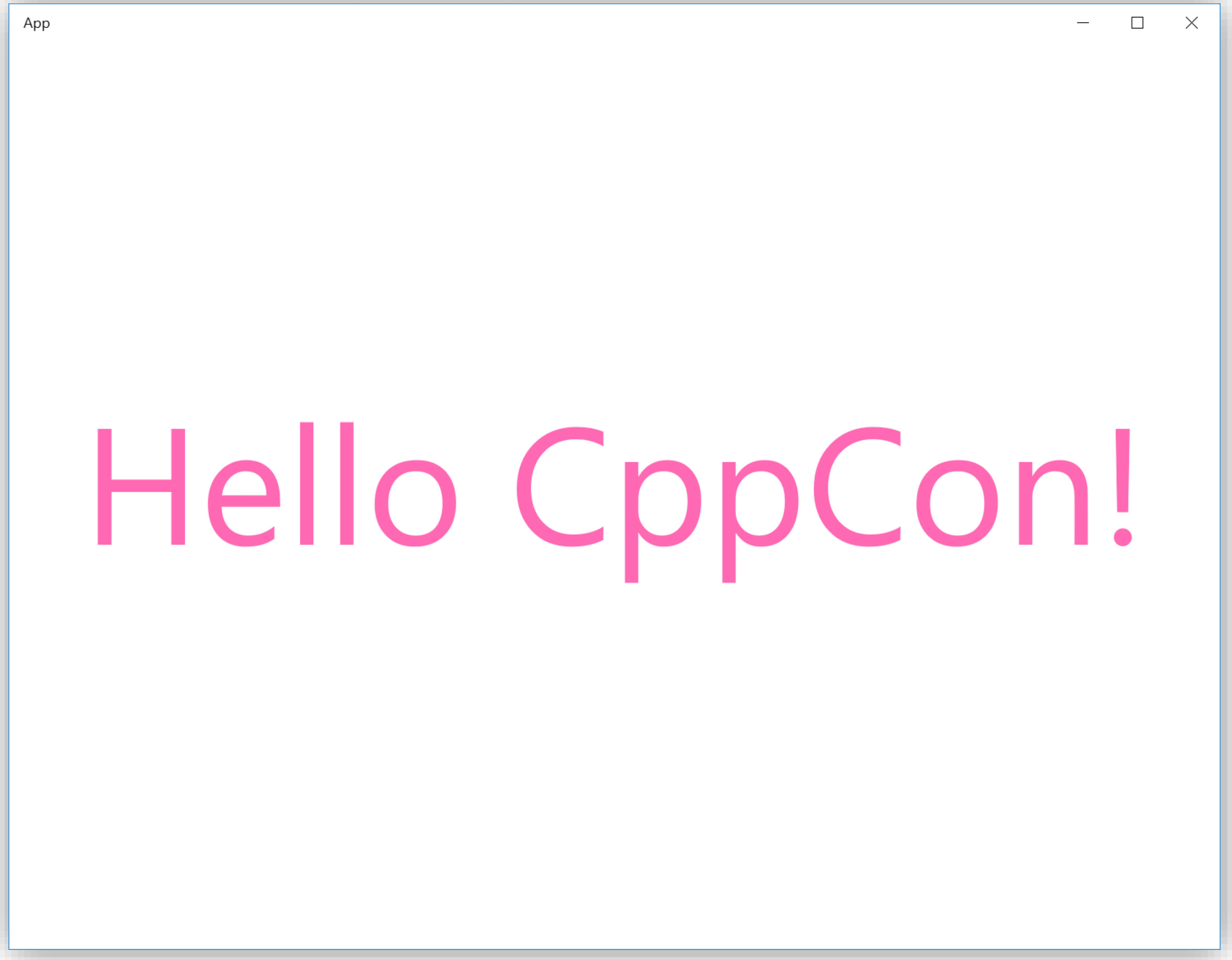
```
// The Windows Runtime is a set of C and COM APIs, so we need a little helper
void check_hresult(HRESULT const hr)
{
    if (hr != S_OK)
    {
        std::terminate();
    }
}
```

```cpp
// sealed partial class App : Application
class App : public RuntimeClass<IApplicationOverrides, ComposableBase<IApplicationFactory>>
{
public:
    App()
    {
        ComPtr<IApplicationFactory> factory;
        check_hresult(GetActivationFactory(
            HStringReference(RuntimeClass_Windows_UI_Xaml_Application).Get(),
            factory.GetAddressOf()));

        ComPtr<IInspectable> inner_inspectable;
        ComPtr<IApplication> inner_instance;
        check_hresult(factory->CreateInstance(
            this,
            inner_inspectable.GetAddressOf(),
            inner_instance.GetAddressOf()));

        check_hresult(SetComposableBasePointers(inner_inspectable.Get(), factory.Get()));
    }
    // ...
// IApplicationOverrides has these virtual functions:
virtual HRESULT __stdcall OnActivated(IActivatedEventArgs*);
virtual HRESULT __stdcall OnFileActivated(IFileActivatedEventArgs*);
virtual HRESULT __stdcall OnSearchActivated(ISearchActivatedEventArgs*);
virtual HRESULT __stdcall OnShareTargetActivated(IShareTargetActivatedEventArgs*);
virtual HRESULT __stdcall OnFileOpenPickerActivated(IFileOpenPickerActivatedEventArgs*);
virtual HRESULT __stdcall OnFileSavePickerActivated(IFileSavePickerActivatedEventArgs*);
virtual HRESULT __stdcall OnCachedFileUpdaterActivated(ICachedFileUpdaterActivatedEventArgs*);
virtual HRESULT __stdcall OnWindowCreated(IWindowCreatedEventArgs*);
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*);

// We have to define them all, but we can just return success.  E.g.,
virtual HRESULT __stdcall OnWindowCreated(IWindowCreatedEventArgs*)
{
    return S_OK;
}
```

```
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // TextBlock block = new TextBlock();
    ComPtr<IInspectable> block_inspectable;
    check_hresult(RoActivateInstance(
        HStringReference(RuntimeClass_Windows_UI_Xaml_Controls_TextBlock).Get(),
        block_inspectable.GetAddressOf()));

    ComPtr<ITextBlock> block;
    check_hresult(block_inspectable.As(&block));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // block.FontFamily = new FontFamily("Segoe UI Semibold");
    ComPtr<IFontFamilyFactory> font_family_factory;
    check_hresult(GetActivationFactory(
        HStringReference(RuntimeClass_Windows_UI_Xaml_Media_FontFamily).Get(),
        font_family_factory.GetAddressOf()));

    ComPtr<IFontFamily> font_family;
    check_hresult(font_family_factory->CreateInstanceWithName(
        HStringReference(L"Segoe UI Semibold").Get(),
        nullptr,
        nullptr,
        font_family.GetAddressOf()));

    check_hresult(block->put_FontFamily(font_family.Get()));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // block.FontSize = 140.0;
    check_hresult(block->put_FontSize(140.00));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // block.Foreground = new SolidColorBrush(Colors.HotPink); (Part 1)
    ComPtr<IColorsStatics> colors_statics;
    check_hresult(GetActivationFactory(
        HStringReference(RuntimeClass_Windows_UI_Colors).Get(),
        colors_statics.GetAddressOf()));

    Color hot_pink;
    check_hresult(colors_statics->get_HotPink(&hot_pink));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // block.Foreground = new SolidColorBrush(Colors.HotPink); (Part 2)
    ComPtr<ISolidColorBrushFactory> brush_factory;
    check_hresult(GetActivationFactory(
        HStringReference(RuntimeClass_Windows_UI_Xaml_Media_SolidColorBrush).Get(),
        brush_factory.GetAddressOf()));

    ComPtr<ISolidColorBrush> hot_pink_brush;
    check_hresult(brush_factory->CreateInstanceWithColor(
        hot_pink,
        hot_pink_brush.GetAddressOf()));

    ComPtr<IBrush> foreground_brush;
    check_hresult(hot_pink_brush.As(&foreground_brush));
    check_hresult(block->put_Foreground(foreground_brush.Get()));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // block.VerticalAlignment = VerticalAlignment.Center;
    ComPtr<IFrameworkElement> block_framework_element;
    check_hresult(block.As(&block_framework_element));
    check_hresult(block_framework_element->put_VerticalAlignment(VerticalAlignment_Center));

    // block.TextAlignment = TextAlignment.Center;
    check_hresult(block->put_TextAlignment(TextAlignment_Center));

    // block.Text = "Hello CppCon!";
    check_hresult(block->put_Text(HStringReference(L"Hello CppCon!").Get()));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // Window window = Window.Current;
    ComPtr<IWindowStatics> window_statics;
    check_hresult(GetActivationFactory(
        HStringReference(RuntimeClass_Windows_UI_Xaml_Window).Get(),
        window_statics.GetAddressOf()));

    ComPtr<IWindow> window;
    check_hresult(window_statics->get_Current(window.GetAddressOf()));

    // ...
// protected override void OnLaunched(LaunchActivatedEventArgs e)
virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
{
    // ...

    // window.Content = block;
    ComPtr<IUIElement> block_ui_element;
    check_hresult(block.As(&block_ui_element));
    check_hresult(window->put_Content(block_ui_element.Get()));

    // window.Activate();
    check_hresult(window->Activate());

    return S_OK;
}
```

```
class App : public RuntimeClass<IApplicationOverrides, ComposableBase<IApplicationFactory>>
{
public:
    App()
    {
        ComPtr<IApplicationFactory> factory;
        check_hr(GetActivationFactory(
            HStringReference(RuntimeClass_Windows_UI_Xaml_Application).Get(),
            factory.GetAddressOf()));

        ComPtr<IInspectable> inner_inspectable;
        ComPtr<IApplication> inner_instance;
        check_hr(factory->CreateInstance(
            this,
            inner_inspectable.GetAddressOf(),
            inner_instance.GetAddressOf()));

        check_hr(SetComposableBasePointers(inner_inspectable.Get(), factory.Get()));
    }

    virtual HRESULT __stdcall OnActivated(IActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnFileActivated(IFileActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnSearchActivated(ISearchActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnShareTargetActivated(IShareTargetActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnFileOpenPickerActivated(IFileOpenPickerActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnFileSavePickerActivated(IFileSavePickerActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnCachedFileUpdaterActivated(ICachedFileUpdaterActivatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnWindowCreated(IWindowCreatedEventArgs*)
    {
        return S_OK;
    }

    virtual HRESULT __stdcall OnLaunched(ILaunchActivatedEventArgs*)
    {
        ComPtr<IInspectable> block_inspectable;
        check_hresult(RoActivateInstance(
            HStringReference(RuntimeClass_Windows_UI_Xaml_Controls_TextBlock).Get(),
```

```
            block_inspectable.GetAddressOf()));

        ComPtr<ITextBlock> block;
        check_hresult(block_inspectable.As(&block));

        ComPtr<IFontFamilyFactory> font_family_factory;
```



C++/CX
in a Nutshell

```
sealed partial class App         :         Application
{

    protected override void OnLaunched(LaunchActivatedEventArgs  e)
    {
        TextBlock  block            =     new TextBlock();
        block. FontFamily           =     new FontFamily("Segoe UI Semibold");
        block. FontSize             = 140.0;
        block. Foreground           =     new SolidColorBrush(Colors. HotPink);
        block. VerticalAlignment = VerticalAlignment. Center;
        block. TextAlignment        = TextAlignment. Center;
        block. Text                 = "Hello CppCon!";

        Window  window  = Window. Current;
        window. Content = block;
        window. Activate();
    }
}
```

C#

```
ref class           App sealed : public Application
{
protected:
    virtual void            OnLaunched(LaunchActivatedEventArgs^ e) override
    {
        TextBlock^ block          = ref new TextBlock();
        block->FontFamily         = ref new FontFamily("Segoe UI Semibold");
        block->FontSize           = 140.0;
        block->Foreground         = ref new SolidColorBrush(Colors::HotPink);
        block->VerticalAlignment = VerticalAlignment::Center;
        block->TextAlignment      = TextAlignment::Center;
        block->Text               = "Hello CppCon!";

        Window^ window  = Window::Current;
        window->Content = block;
        window->Activate();
    }
};
```

C++/CX

Challenges with C++/CX

- Memory management not customizable
- Code bloat in exception/HRESULT translation
- Interop between standard and WinRT types
- Arrays perform poorly by default
- No visibility into abstractions
- Debuggability
- Syntax differences
- And more…

C++/WinRT in a Nutshell

C++/WinRT

Standard C++

Header-only library

Classy type system

Natural, productive, safe

Best performance, smallest binaries

Language projection for the systems programmer

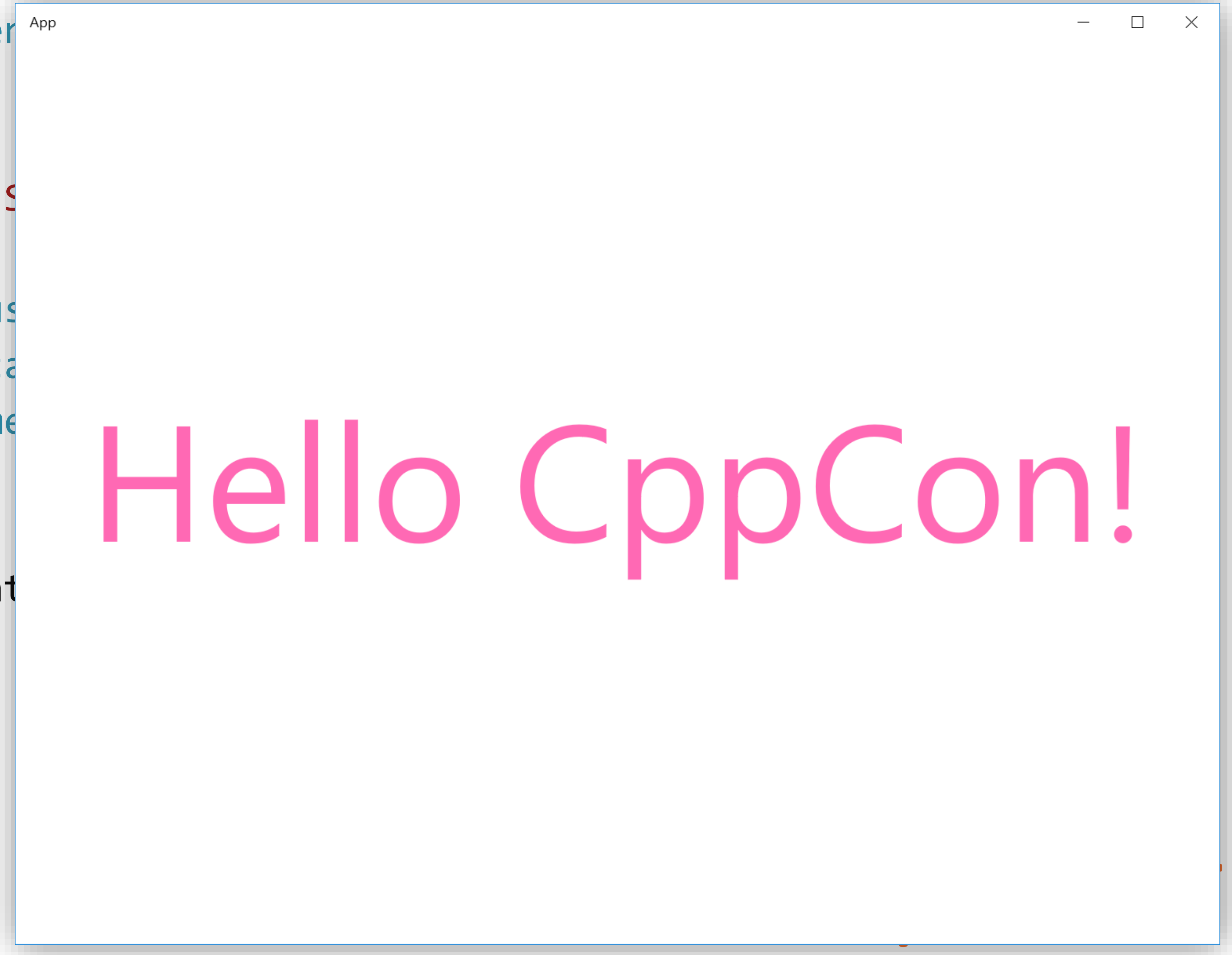
…but also for app developers and other programmers!

Succinct

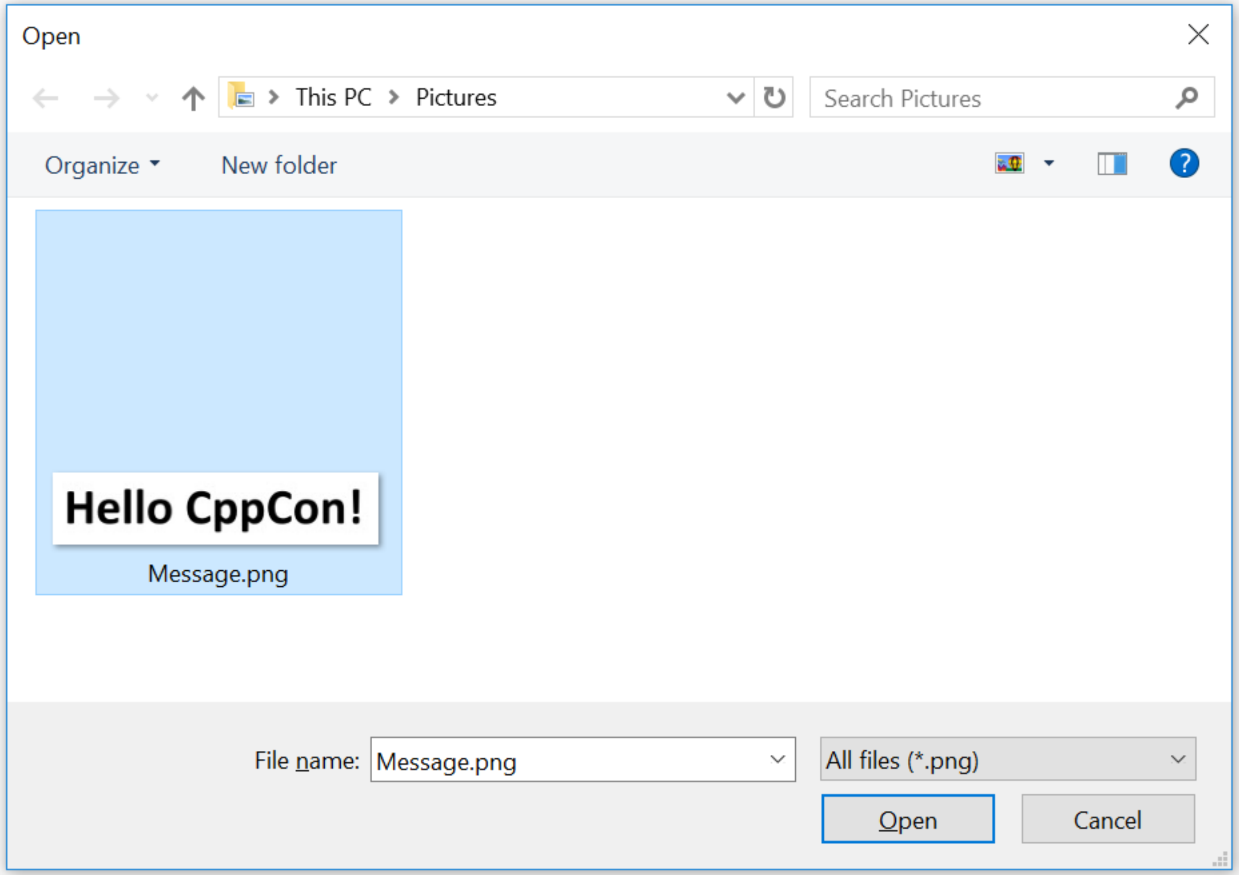
```
sealed partial class App : Application
{
    protected override void OnLaunched(LaunchActivatedEventArgs e)
    {
        TextBlock block            = new TextBlock();
        block.FontFamily           = new FontFamily("Segoe UI Semibold");
        block.FontSize             = 140.0;
        block.Foreground           = new SolidColorBrush(Colors.HotPink);
        block.VerticalAlignment = VerticalAlignment.Center;
        block.TextAlignment        = TextAlignment.Center;
        block.Text                 = "Hello CppCon!";

        Window window  = Window.Current;
        window.Content = block;
        window.Activate();
    }
}
```

C#

```
struct App : ApplicationT<App>
{
    void OnLaunched(LaunchActivatedEver
    {
        TextBlock block;
        block.FontFamily(FontFamily(L"S
        block.FontSize(140.0);
        block.Foreground(SolidColorBrus
        block.VerticalAlignment(Vertica
        block.TextAlignment(TextAlignme
        block.Text(L"Hello CppCon!");

        Window window = Window::Current
        window.Content(block);
        window.Activate();
    }
};
```

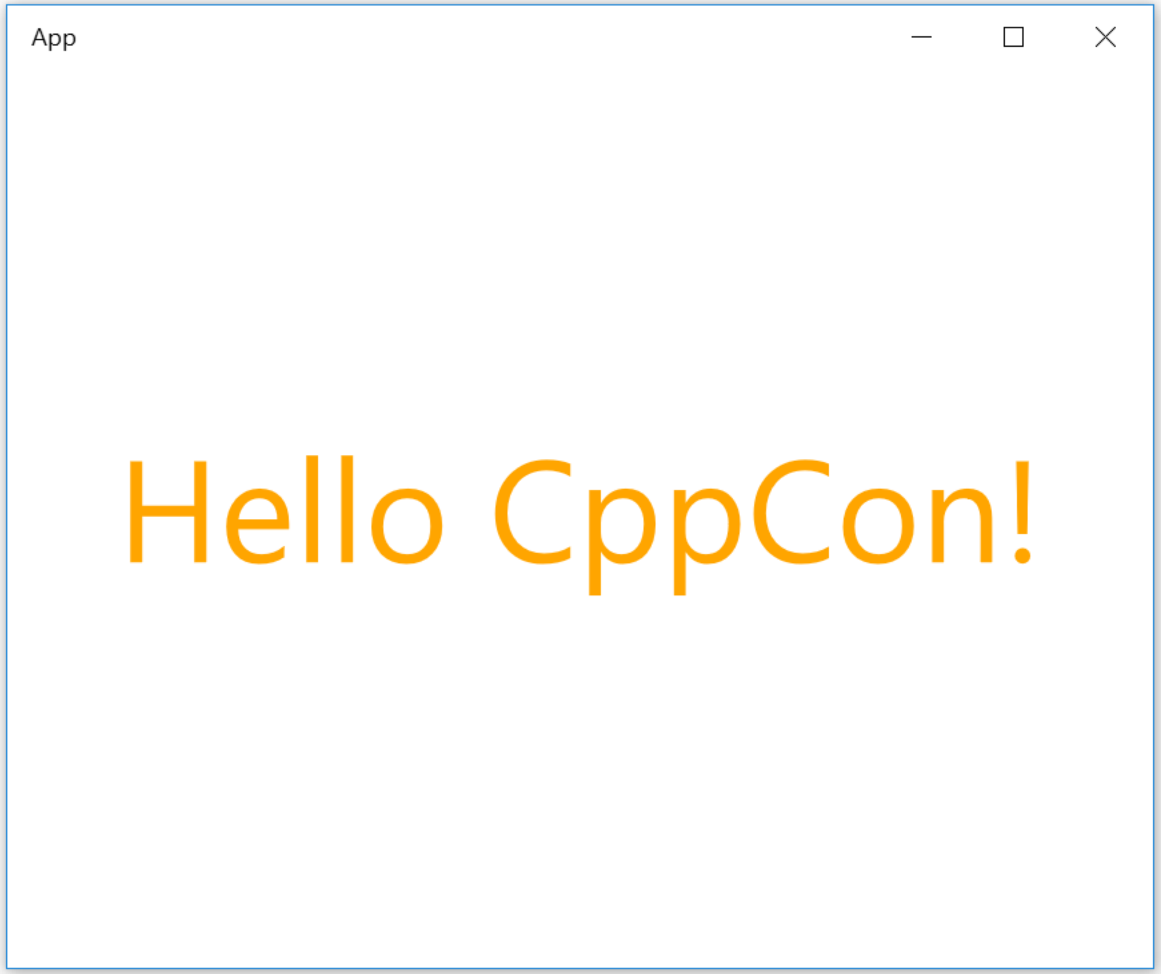


Let’s Build Something a Bit More Interesting…



File name: Message.png

All files (*.png)



```
class App : Application
{
    protected override void OnLaunched(LaunchActivatedEventArgs args)
    { ... }

    async void ForegroundAsync(TextBlock block)
    { ... }

    IAsyncOperation<string> BackgroundAsync(StorageFile file)
    { ... }

    static void Main()
    {
        Application.Start((param) => { new App(); });
    }
}
```

Warning: C#

```
protected override void OnLaunched(LaunchActivatedEventArgs args)
{
    TextBlock block = new TextBlock();

    block.FontFamily = new FontFamily("Segoe UI Semibold");
    block.FontSize = 72.0;
    block.Foreground = new SolidColorBrush(Colors.Orange);
    block.VerticalAlignment = VerticalAlignment.Center;
    block.TextAlignment = TextAlignment.Center;
    block.TextWrapping = TextWrapping.Wrap;

    Window window = Window.Current;
    window.Content = block;
    window.Activate();

    ForegroundAsync(block);
}
```

1/3 OnLaunched in C#

```
async void ForegroundAsync(TextBlock block)
{
    FileOpenPicker picker = new FileOpenPicker();
    picker.FileTypeFilter.Add(".png");
    picker.SuggestedStartLocation = PickerLocationId.PicturesLibrary;
    var file = await picker.PickSingleFileAsync();

    if (file == null)
    {
        return;
    }

    block.Text = await BackgroundAsync(file);
}
```

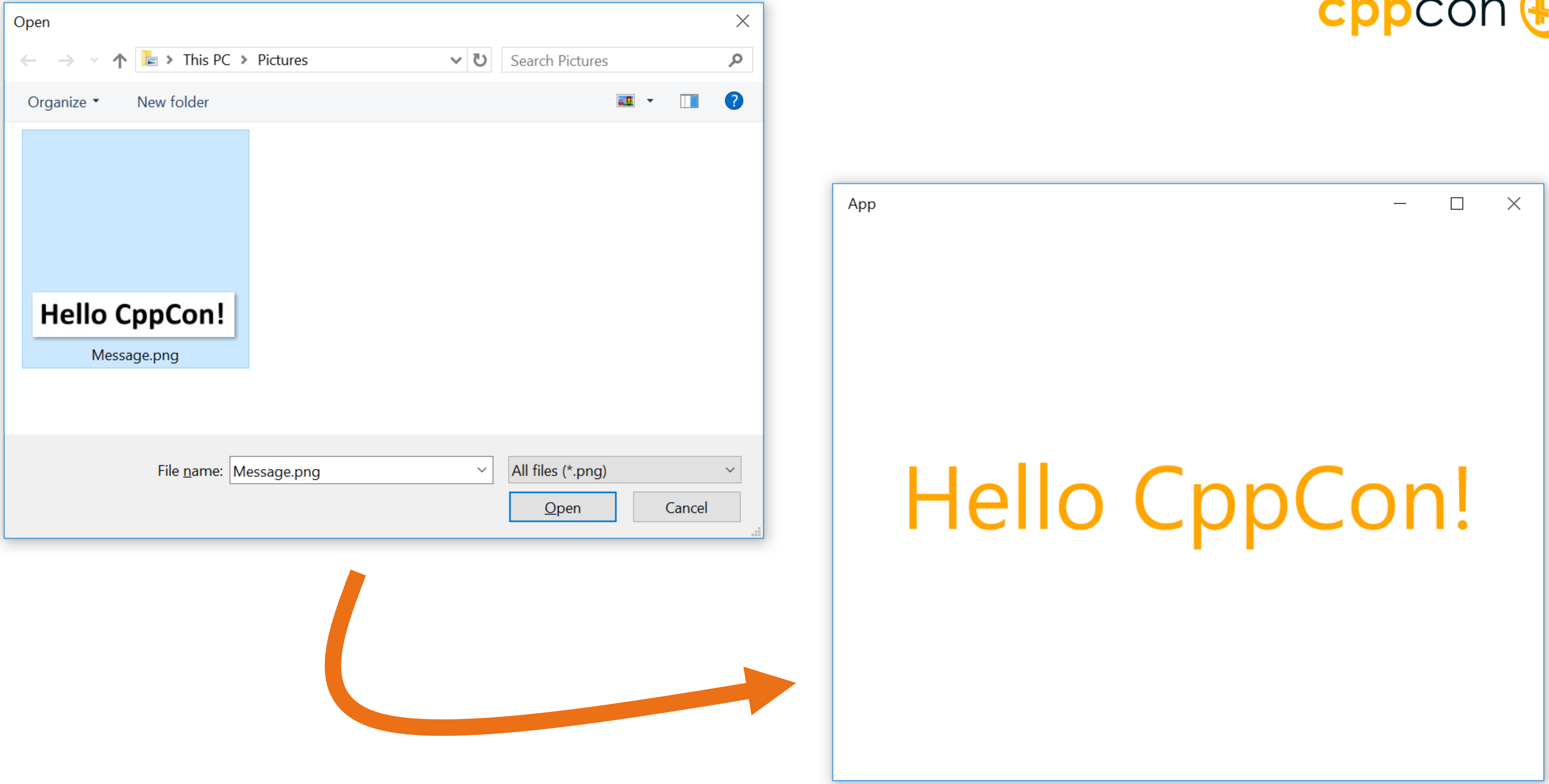
2/3 ForegroundAsync in C#

```
IAsyncOperation<string> BackgroundAsync(StorageFile file)
{
    return Task<string>.Run(async () =>
    {
        var stream = await file.OpenAsync(FileAccessMode.Read);
        var decoder = await BitmapDecoder.CreateAsync(stream);
        var bitmap = await decoder.GetSoftwareBitmapAsync();

        var engine = OcrEngine.TryCreateFromUserProfileLanguages();
        var result = await engine.RecognizeAsync(bitmap);
        return result.Text;
    })
    .AsAsyncOperation<string>();
}
```

1. Get work on thread pool

2. Get IAsync... representing task

3/3 BackgroundAsync in C#



Hello CppCon!

Message.png

File name: Message.png

All files (*.png)

Hello CppCon!

```
ref class App : Application
{
protected:
    void OnLaunched(LaunchActivatedEventArgs ^) override;

private:
    task<void> ForegroundAsync(TextBlock ^ block);
    IAsyncOperation<String ^> ^ BackgroundAsync(StorageFile ^ file);
};

int main(Array<String ^> ^)
{
    Application::Start(ref new ApplicationInitializationCallback([](auto &&)
    {
        ref new App;
    }));
}
```

Warning: C++/CX

```
void OnLaunched(LaunchActivatedEventArgs ^) override
{
    TextBlock ^ block = ref new TextBlock();

    block->FontFamily = ref new FontFamily("Segoe UI Semibold");
    block->FontSize = 72.0;
    block->Foreground = ref new SolidColorBrush(Colors::Orange);
    block->VerticalAlignment = VerticalAlignment::Center;
    block->TextAlignment = TextAlignment::Center;
    block->TextWrapping = TextWrapping::Wrap;

    Window ^ window = Window::Current;
    window->Content = block;
    window->Activate();

    ForegroundAsync(block);
}
```

1/3 OnLaunched in C++/CX

```
task<void> ForegroundAsync(TextBlock ^ block)
{
    FileOpenPicker ^ picker = ref new FileOpenPicker();
    picker->FileTypeFilter->Append(".png");
    picker->SuggestedStartLocation = PickerLocationId::PicturesLibrary;
    auto file = co_await picker->PickSingleFileAsync();

    if (file == nullptr)
    {
        return;
    }

    block->Text = co_await BackgroundAsync(file);
}
```

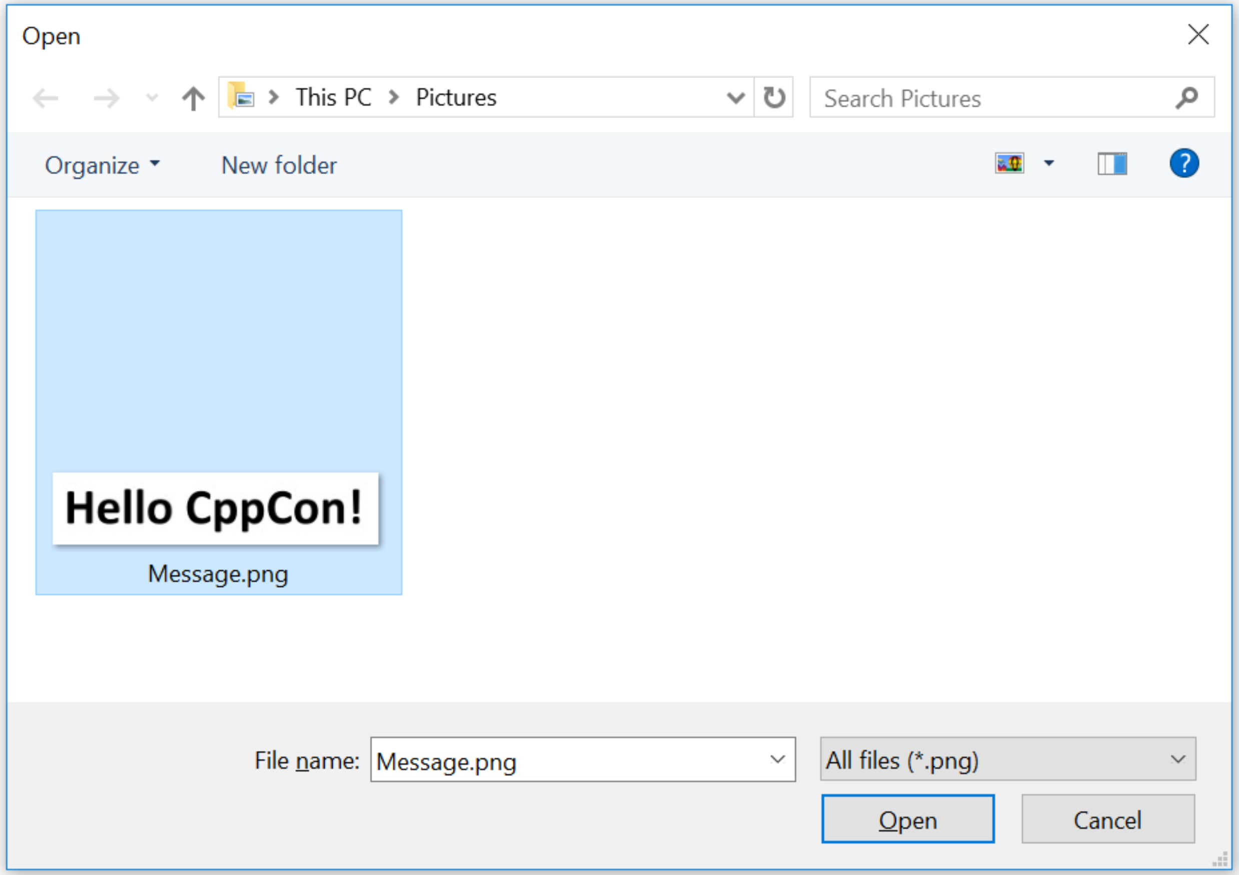
2/3 ForegroundAsync in C++/CX

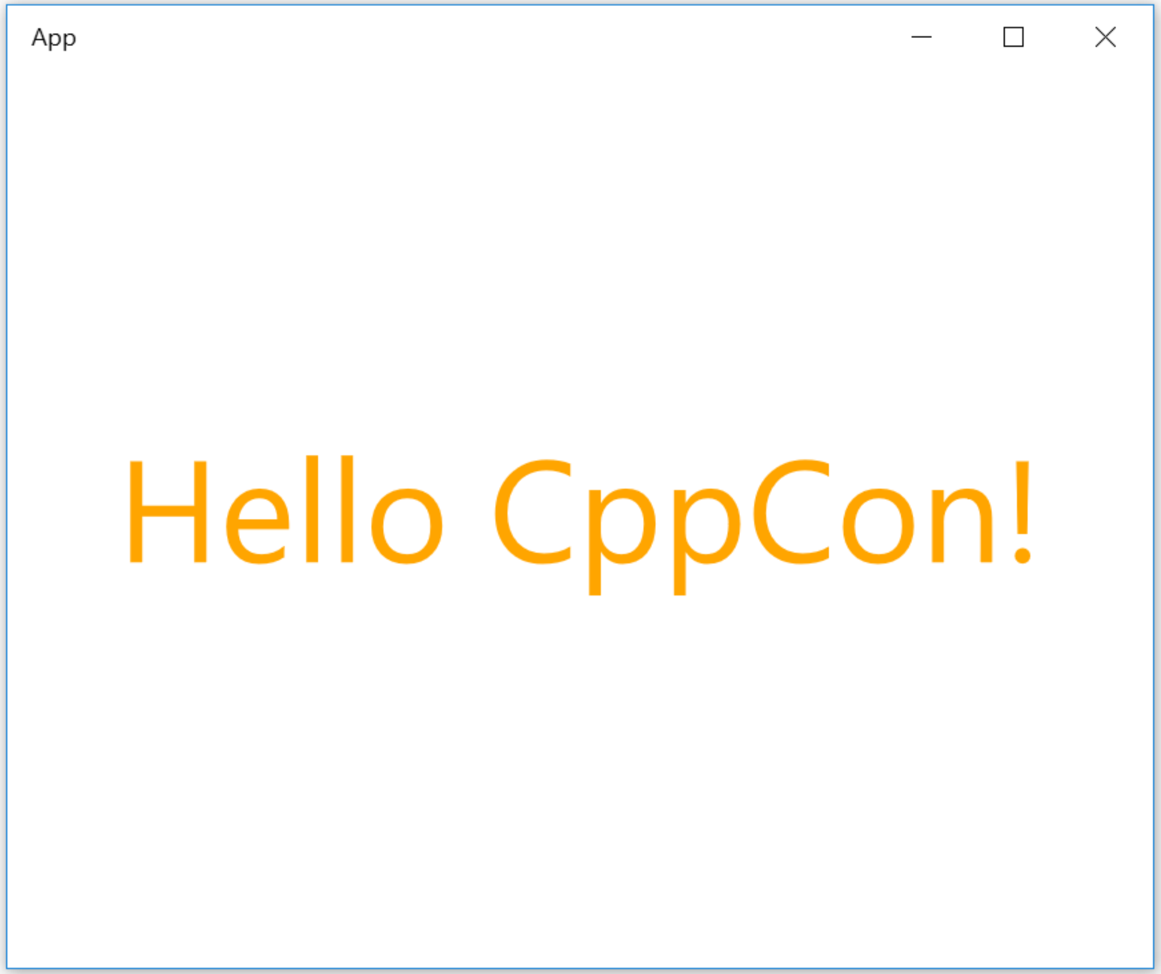
```
IAsyncOperation<String ^> ^ BackgroundAsync(StorageFile ^ file)
{
    return create_async([=]
    {
        return create_task([=] () -> task<String ^>
        {
            auto stream = co_await file->OpenAsync(FileAccessMode::Read);
            auto decoder = co_await BitmapDecoder::CreateAsync(stream);
            auto bitmap = co_await decoder->GetSoftwareBitmapAsync();

            auto engine = OcrEngine::TryCreateFromUserProfileLanguages();
            auto result = co_await engine->RecognizeAsync(bitmap);
            return result->Text;
        });
    });
}
```

1. Get work on thread pool

2. Get IAsync... representing task

3/3 BackgroundAsync in C++/CX


Open
This PC > Pictures
Search Pictures
Organize
New folder
Hello CppCon!
Message.png
File name: Message.png
All files (*.png)
Open
Cancel


App
Hello CppCon!

There’s a better way!

```
struct App : ApplicationT<App>
{
    void OnLaunched(LaunchActivatedEventArgs const &);

    fire_and_forget ForegroundAsync(TextBlock block);

    IAsyncOperation<hstring> BackgroundAsync(StorageFile file);
};

int __stdcall wWinMain(HINSTANCE, HINSTANCE, PWSTR, int)
{
    Application::Start([](auto &&) { make<App>(); });
}
```

C++/WinRT

```cpp
void OnLaunched(LaunchActivatedEventArgs const &)
{
    TextBlock block;

    block.FontFamily(FontFamily(L"Segoe UI Semibold"));
    block.FontSize(72.0);
    block.Foreground(SolidColorBrush(Colors::Orange()));
    block.VerticalAlignment(VerticalAlignment::Center);
    block.TextAlignment(TextAlignment::Center);
    block.TextWrapping(TextWrapping::Wrap);

    Window window = Window::Current();
    window.Content(block);
    window.Activate();

    ForegroundAsync(block);
}
```

1/3 OnLaunched in C++/WinRT

```
fire_and_forget ForegroundAsync(TextBlock block)
{
    FileOpenPicker picker;
    picker.FileTypeFilter().Append(L".png");
    picker.SuggestedStartLocation(PickerLocationId::PicturesLibrary);
    auto file = co_await picker.PickSingleFileAsync();

    if (file == nullptr)
    {
        return;
    }

    block.Text(co_await BackgroundAsync(file));
}
```

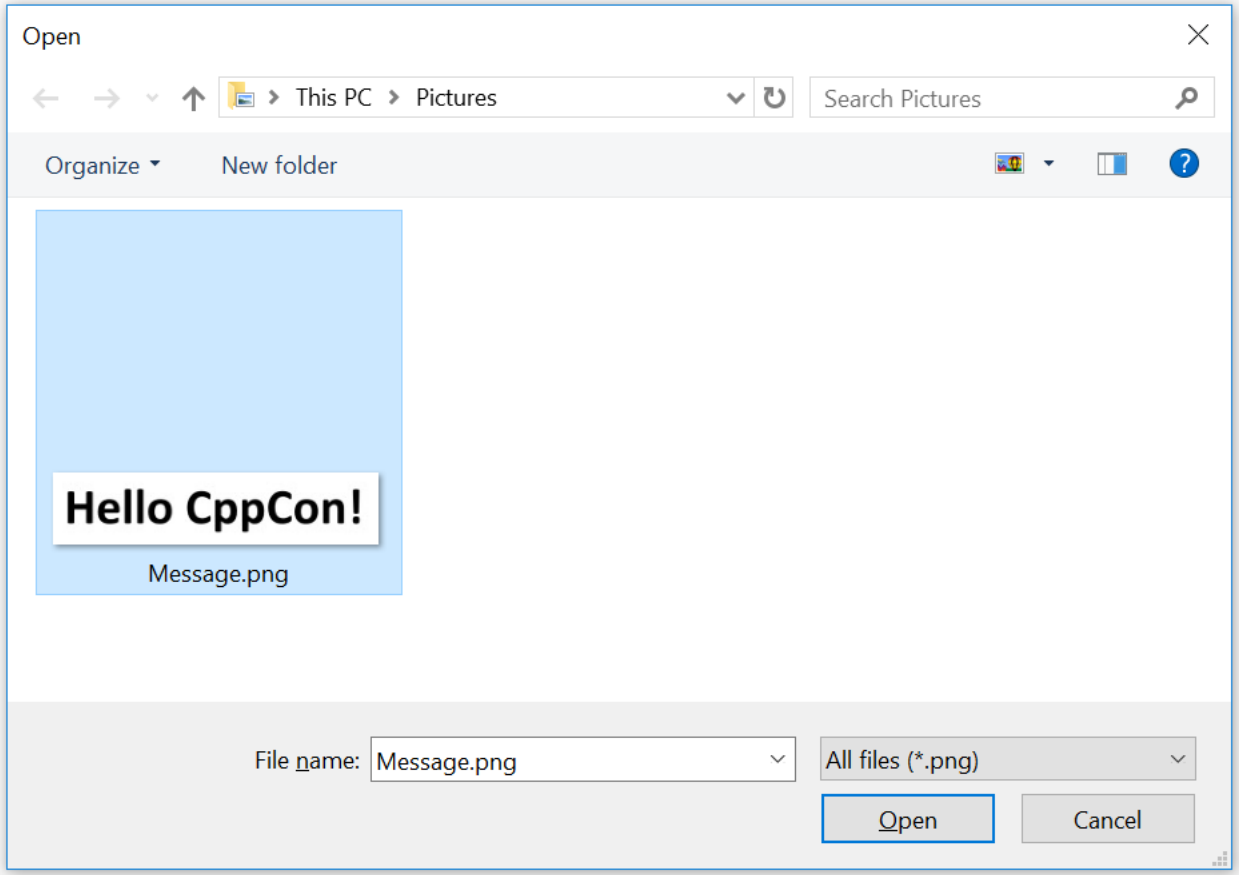
2/3 ForegroundAsync in C++/WinRT

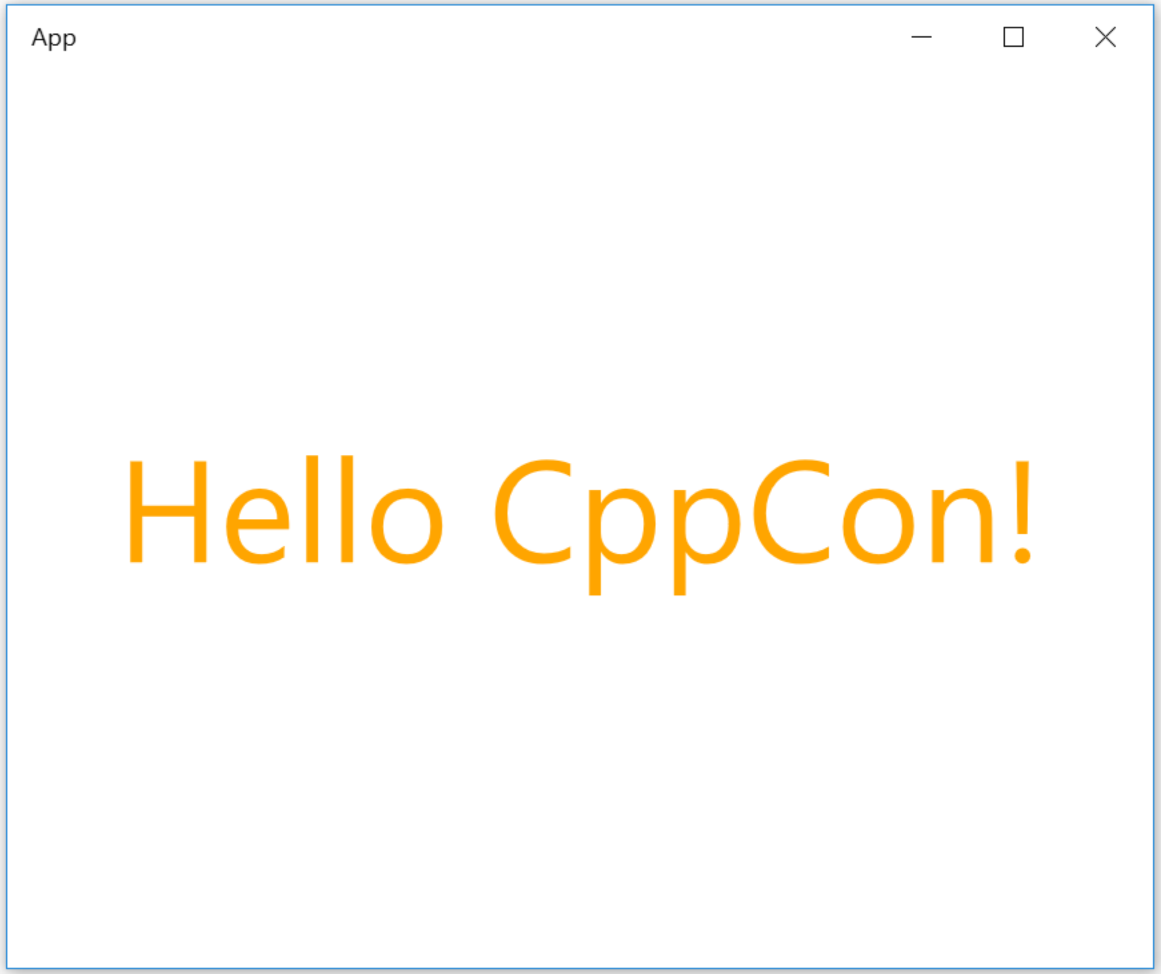
```
IAsyncOperation<hstring> BackgroundAsync(StorageFile file)
{
    co_await resume_background();

    auto stream = co_await file.OpenAsync(FileAccessMode::Read);
    auto decoder = co_await BitmapDecoder::CreateAsync(stream);
    auto bitmap = co_await decoder.GetSoftwareBitmapAsync();

    auto engine = OcrEngine::TryCreateFromUserProfileLanguages();
    auto result = co_await engine.RecognizeAsync(bitmap);
    return result.Text();
}
```

2. Resume on thread pool

1. Produce IAsync…

3/3 BackgroundAsync in C++/WinRT


Open
This PC > Pictures
Search Pictures
Organize
New folder
Hello CppCon!
Message.png
File name:
Message.png
All files (*.png)
Open
Cancel


App
Hello CppCon!

Interfaces

```
struct IUnknown
{
    virtual HRESULT QueryInterface(GUID const & id,
                                   void ** object) = 0;
    virtual uint32_t AddRef() = 0;
    virtual uint32_t Release() = 0;
};



struct IInspectable : IUnknown
{
    virtual HRESULT GetIids(uint32_t * count, GUID ** iids) = 0;
    virtual HRESULT GetRuntimeClassName(HSTRING * className) = 0;
    virtual HRESULT GetTrustLevel(TrustLevel * trustLevel) = 0;
};
```

IUnknown & IInspectable…

```
struct ITextBlock : IInspectable
{
    virtual HRESULT get_FontSize(double * value) = 0;
    virtual HRESULT put_FontSize(double value) = 0;

    // ...
}
```

ITextBlock

```cpp
ITextBlock * block = ...

HRESULT hr = block->put_FontSize(72.0);

if (hr != S_OK)
{
    // pain and suffering...
}

block->Release();
```

ITextBlock with raw pointers

```
ComPtr<ITextBlock> block = ...

HRESULT hr = block->put_FontSize(72.0);

if (hr != S_OK)
{
    // pain and suffering...
}
```

ITextBlock with smart pointers

```
ITextBlock block = ...

block.FontSize(72.0);
```

ITextBlock with C++/WinRT

```
using namespace Windows::UI::Xaml::Controls;

TextBlock block;




ITextBlock block = TextBlock(); // This works... but don't do this :)
```

Interfaces at the heart of classes...

Move

Construction

```
void Scope()
{
    ITextBlock block = TextBlock();

    ITextBlock block2 = block;
}
```

AddRef

Release x 2

Reference counting is automatic…

```
void Scope()
{
    ITextBlock block = TextBlock();

    ITextBlock block2 = std::move(block);
}
```

Move

Construction

Move

Release x 1

Moves are cheaper…

```
using namespace Windows::Storage;

IStorageFile file = ...

IInspectable in = file;

IUnknown un = in;
```

AddRef

AddRef

Classic inheritance good but shallow…

```cpp
using namespace Windows::Storage;

IStorageFile file = ...

IInspectable const & in = file;

IUnknown const & un = in;
```

No AddRef

Good for synchronous parameters…

```
IUnknown un = ...

IInspectable in = un.as<IInspectable>();

IStorageFile file = in.as<IStorageFile>();
```

QueryInterface

Explicit queries are explicit…

$$\begin{tikzcd}\textbf{IStorageFile} \arrow[r, bend left, "\textbf{Requires}"] & \textbf{IStorageItem}\end{tikzcd}$$

FileType	RenameAsync
ContentType	DeleteAsync
OpenAsync	GetBasicPropertiesAsync
OpenTransactedWriteAsync	Name
CopyAsync	Path
CopyAndReplaceAsync	Attributes
MoveAsync	DateCreated
MoveAndReplaceAsync	IsOfType

```
IStorageFile file = ...

IInspectable in = file;

IStorageItem item = file;
```

AddRef

QueryInterface

Implicit queries are implicit…

```
IStorageFile file = ...

IStorageItem item = file;

hstring in = item.Name();

hstring fn = file.Name();

hstring ft = file.FileType();

hstring it = item.FileType();
```

QueryInterface

v-call only

QueryInterface & v-call

v-call only

error: 'FileType': is not a member of 'IStorageItem'

Implicit queries lead to implicit methods…

Calling Methods

```
struct IStorageItem : IInspectable
{
    abi<IStorageItem> * operator->() const noexcept;

    hstring Name() const
    {
        hstring value;
        check_hresult((*this)->get_Name(put(value)));
        return value;
    }
};
```

Returns vptr (pointer to vtable)

Which vptr?

Looks good & almost works…

```
struct IStorageFile : IInspectable
{
    abi<IStorageFile> * operator->() const noexcept;

    operator IStorageItem() const
    {
        return as<IStorageItem>();
    }

    hstring Name() const
    {
        return as<IStorageItem>().Name();
    }
};
```

x Number of interfaces

x Number of methods

But then this happens…

```
template <typename D, typename I = D>
struct consume;

template <typename D, typename I>
struct produce;


template <typename D>
struct consume<D, Windows::Storage::IStorageItem>
{
    hstring Name() const;
    // ...
};

template <typename D>
struct produce<D, Windows::Storage::IStorageItem> // ...
```

Consuming and producing interfaces…

```
struct IStorageItem
{
    hstring Name() const;
    hstring Path() const;
    IAsyncAction RenameAsync(hstring_ref desiredName) const;

    // ...

struct MyStorage : implements<MyStorage, IStorageItem, IStorageItem2>
{
    hstring Name() const { return L"Hello world.txt"; }
    hstring Path() const { return L"C:\\CppCon";       }

    IAsyncAction RenameAsync(hstring_ref desiredName) const
    {
        co_await ...
    }

    // ...
```

Symmetry…

```
template <typename D, typename I = D>
struct consume;

template <typename D>
struct consume<D, Windows::Storage::IStorageItem>
{ /* shims */ }

template <typename D>
struct impl_IStorageItem
{ /* shims */ }

template <> struct traits<Windows::Storage::IStorageItem>
{
    template <typename D> using consume = Windows::Storage::impl_IStorageItem<D>;
};

template <typename D, typename I = D>
using consume = typename traits<I>::template consume<D>;
```

Simple :)

Not so simple :(

Consuming in the real world…

```
template <typename D>
struct impl_IStorageItem
{
    hstring Name() const
    {
        hstring value;

        check_hresult(
            static_cast<const D &>(*this)
                ->get_Name(put(value)));

        return value;
    }
};
```

1. CRTP

2. v-call

Shims that almost works…

```
template <typename D>
struct impl_IStorageItem
{
    hstring Name() const
    {
        hstring value;

        check_hresult(
            static_cast<const IStorageItem &>(
                static_cast<const D &>(*this))
                    ->get_Name(put(value)));

        return value;
    }
};
```

2. Redundant?

1. CRTP

3. v-call

Touch of compile-time indirection…

```cpp
struct IStorageFile :
    IInspectable,
    impl_IStorageFile<IStorageFile>,
    impl_IStorageItem<IStorageFile>
{
    abi<IStorageFile> * operator->() const noexcept;

    operator IStorageItem() const
    {
        return as<IStorageItem>();
    }
};
```

Ownership

Default

Required…

x Number of interfaces

Rough assembly…

```cpp
template <typename D, typename I = D>
using consume = typename traits<I>::template consume<D>;

template <typename D, typename I>
struct require_one : consume<D, I>
{
    operator I() const
    {
        return static_cast<const D *>(this)->template as<I>();
    }
};

template <typename D, typename ... I>
struct require : require_one<D, I> ... {};
```

Glue

Code generator

Variadic scaffolding...

```cpp
struct IStorageFile :
    IInspectable,
    consume<IStorageFile>,
    require<IStorageFile, IStorageItem,
                          IInputStreamReference,
                          IRandomAccessStreamReference>
{
    abi<IStorageFile> * operator->() const noexcept;
};
```

Variadic (and elegant) assembly

Runtime Classes

```
struct StorageFile :
    IStorageFile,
    require<StorageFile, IStorageItem2,
                         IStorageItemProperties,
                         IStorageItemProperties2,
                         IStorageItemPropertiesWithProvider,
                         IStorageFilePropertiesWithAvailability,
                         IStorageFile2>
{
};
```

Ownership & default interface

CRTP

Additional interfaces

Class assembly

```cpp
using namespace Windows::Storage::Pickers;

FileOpenPicker picker;



FileOpenPicker::FileOpenPicker() :
    FileOpenPicker(activate_instance<FileOpenPicker>())
{}
```

Default constructor

“RoActivateInstance”

Delegating constructor

Behind default constructors…

```
using namespace Windows::Networking;

HostName name(L"moderncpp.com");
```

```
HostName::HostName(hstring_ref hostName) :
    HostName(get_activation_factory<HostName, IHostNameFactory>().
                                            CreateHostName(hostName))
{}
```

“RoGetActivationFactory”

Behind constructors with params…

```
// StorageFile file;

StorageFile file =
    activate_instance<StorageFile>();
```

Default interface

```
IStorageFile2 file =
    activate_instance<StorageFile, IStorageFile2>();
```

Request alternative interface

You can do this yourself!

```
using namespace Windows::Networking;

// HostName name(L"moderncpp.com");

auto factory = get_activation_factory<HostName,
                                      IHostNameFactory>();

HostName name = factory.CreateHostName(L"moderncpp.com");
```

You can do this too!

```
using namespace Windows::Networking;

// HostName name(L"moderncpp.com");

auto factory = get_activation_factory<HostName,
                                      IHostNameFactory>();

// HostName name = factory.CreateHostName(L"moderncpp.com");

HostName name = nullptr;

HRESULT hr = factory->abi_CreateHostName(get(hstring_ref(L"moderncpp.com")), put(name));
```

You can go deeper still!

```
template <> struct traits<Windows::Storage::IStorageFile>
{
    using abi = ABI::Windows::Storage::IStorageFile;

    template <typename D> using consume =
        Windows::Storage::impl_IStorageFile<D>;
};

template <> struct traits<Windows::Storage::StorageFile>
{
    using abi = ABI::Windows::Storage::StorageFile;

    static constexpr wchar_t const * name() noexcept
    {
        return L"Windows.Storage.StorageFile";
    }
};
```

Metadata as traits

```cpp
template <typename C, typename I = C>
I activate_instance()
{
    return get_activation_factory<C>().
                ActivateInstance().
                    template as<I>();
}
```

1. Get factory

2. Default activation

3. Query for desired interface

Default activation...

```
template <typename C, typename I = IActivationFactory>
I get_activation_factory()
{
    static I factory = impl::get_agile_activation_factory<C, I>();

    if (!factory)
    {
        return impl::get_activation_factory<C, I>();
    }

    return factory;
}
```

1. Try get agile factory

2. Fallback to non-agile

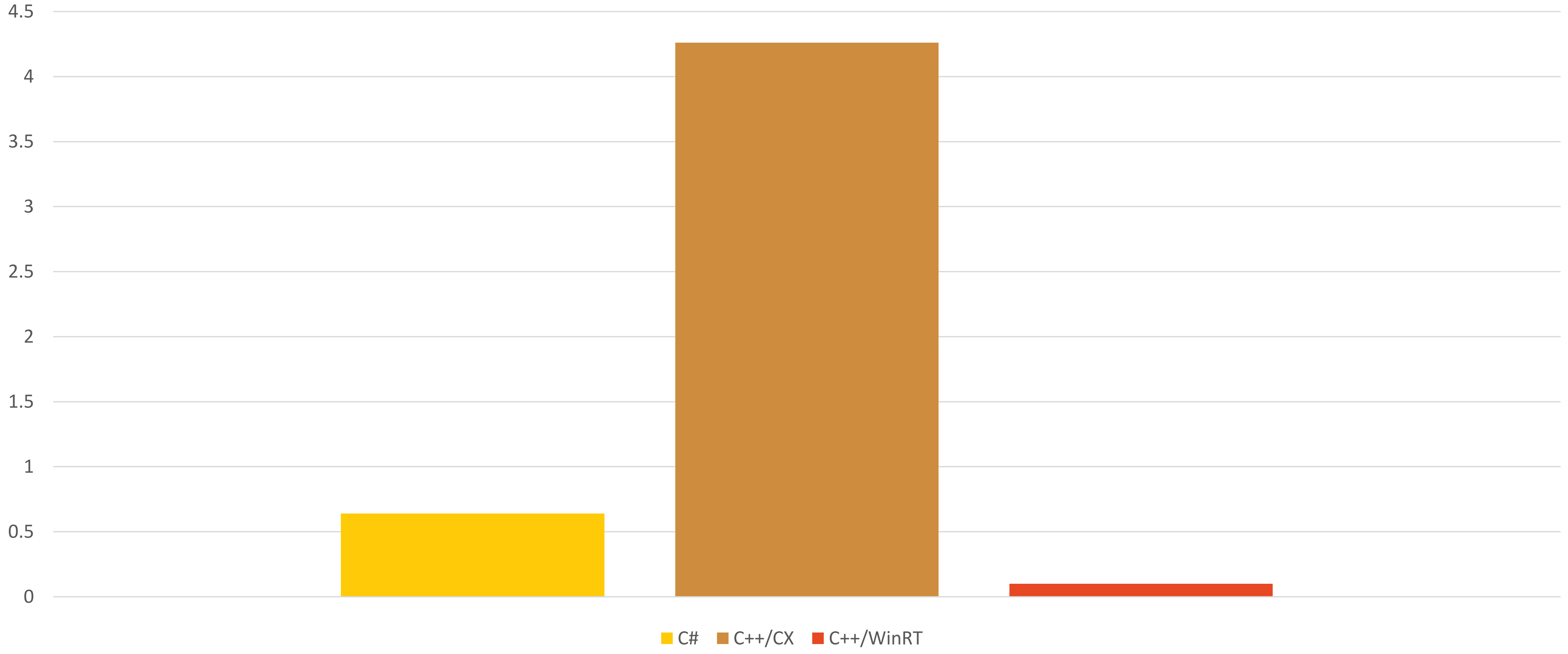
Get activation factory…

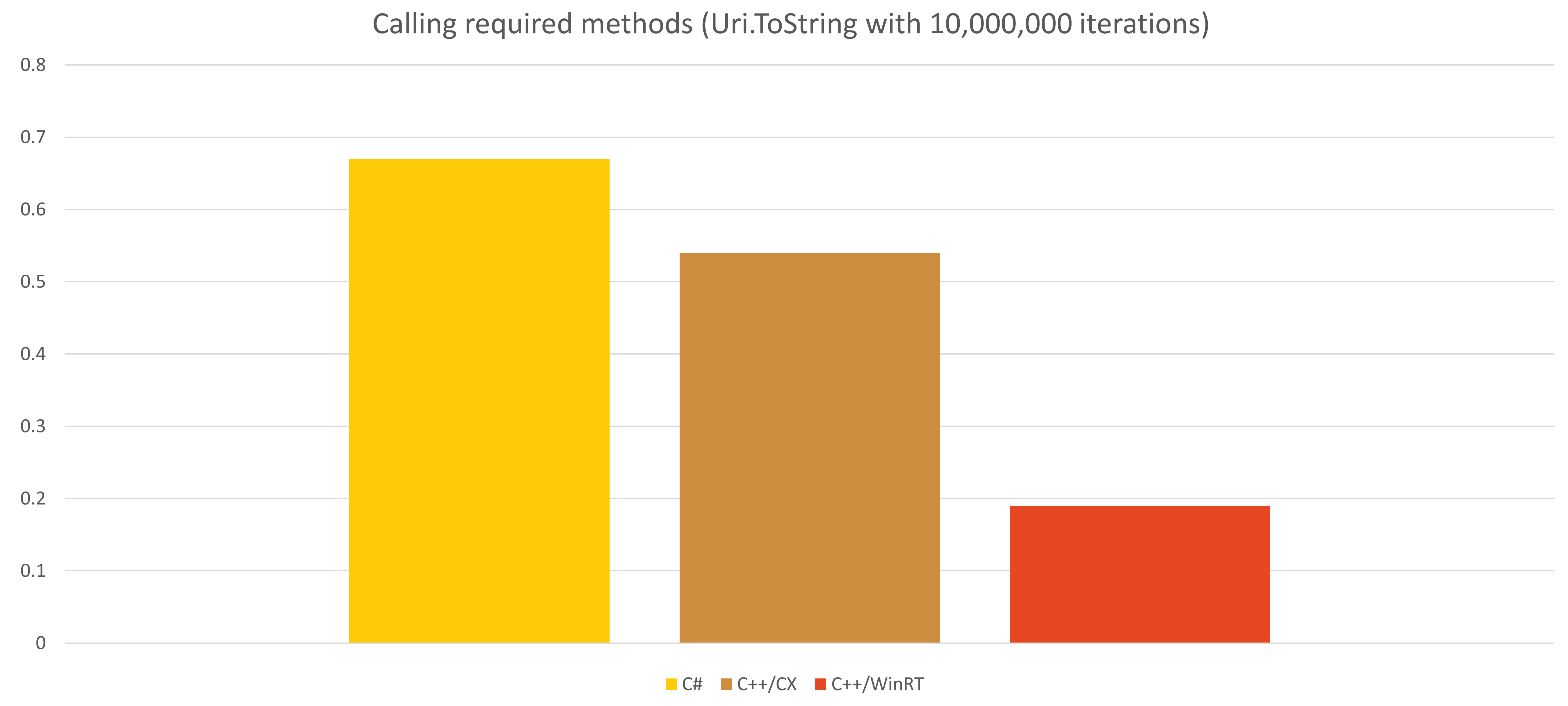
Performance

	C++/WinRT	C++/CX	C#
Smallest binary	53 KB + 594 KB	86 KB + 594 KB	261 KB + 3.31 MB

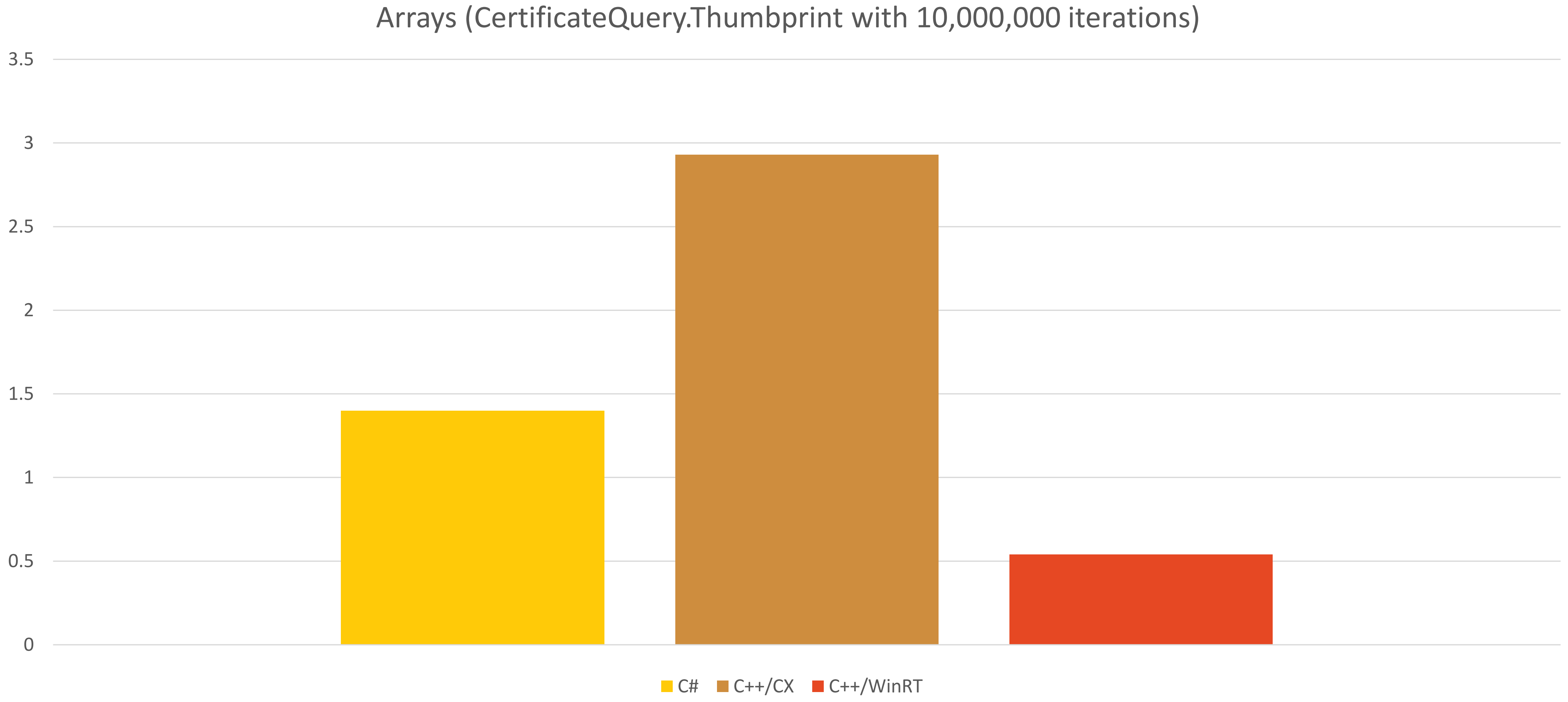
It’s not just about syntax

Calling static methods (PropertyValue::CreateEmpty with 4,000,000 iterations)

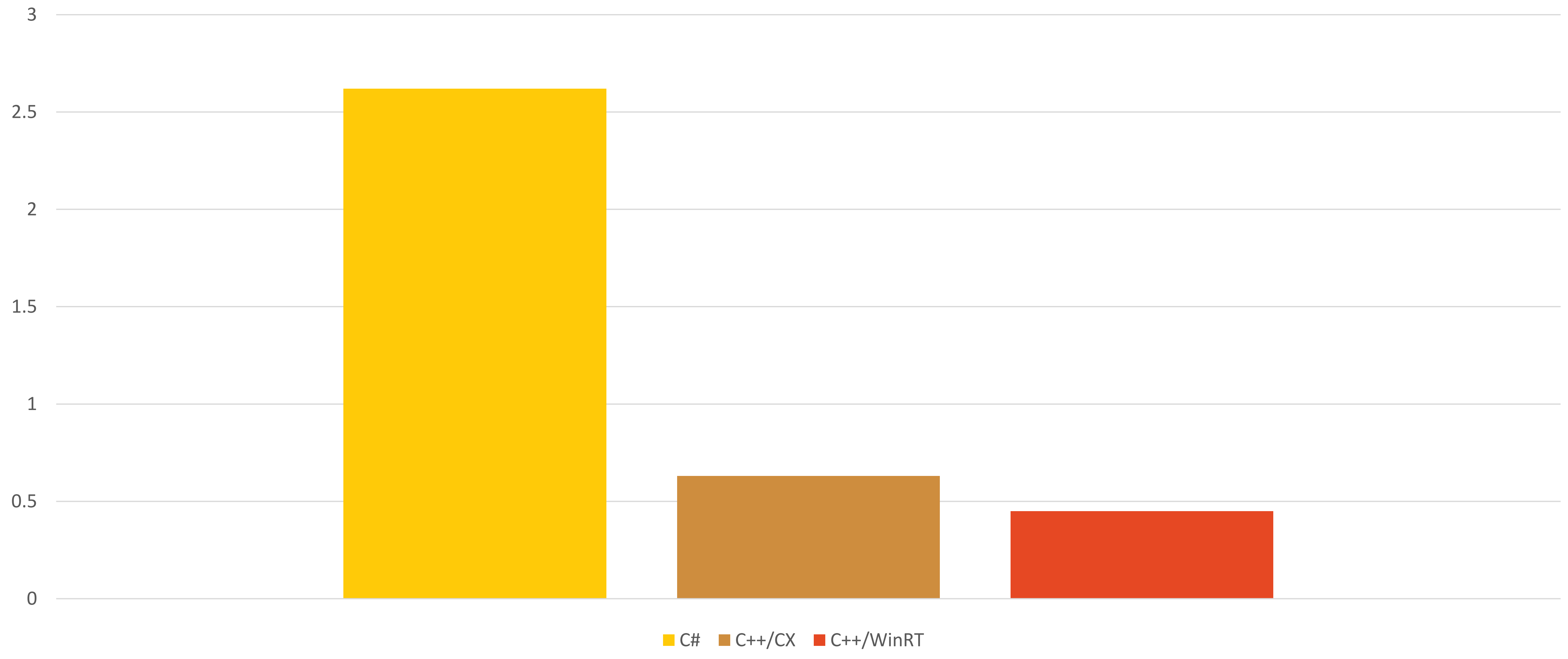


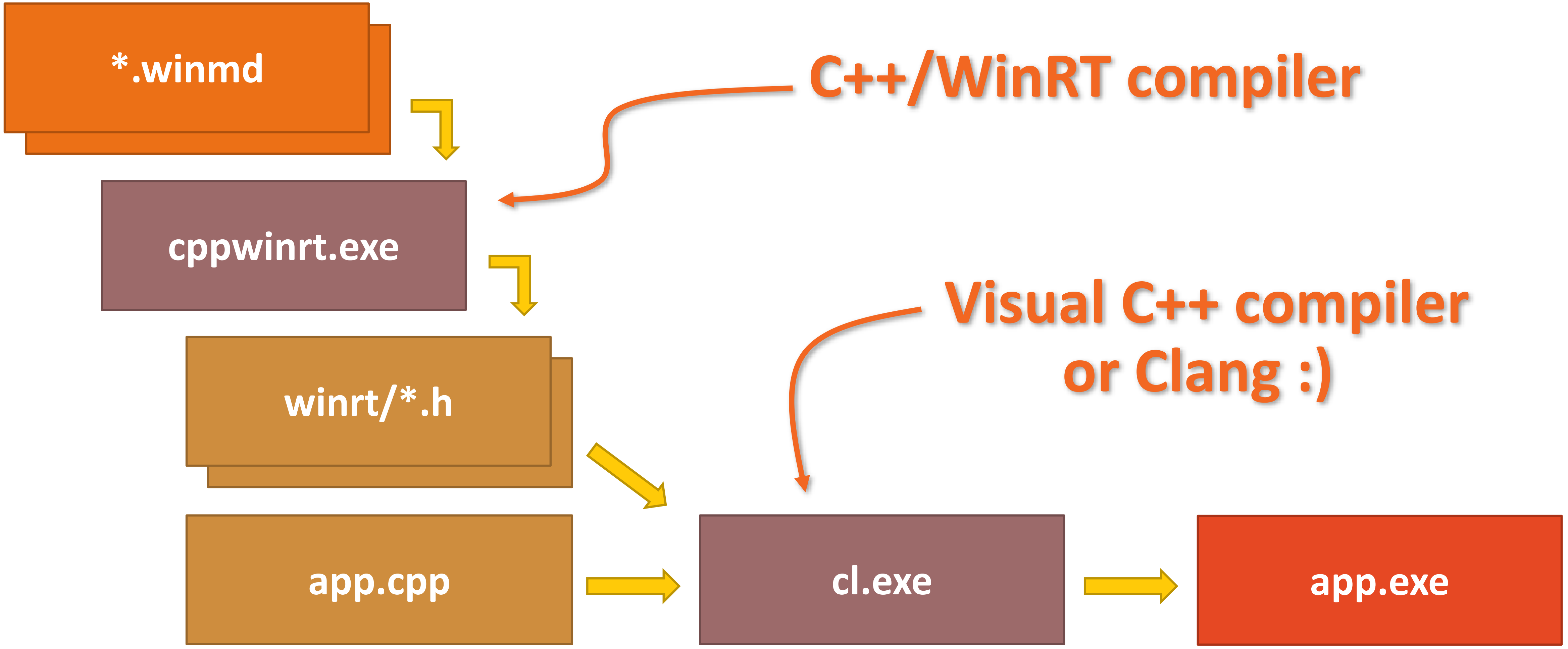


Calling required methods (Uri.ToString with 10,000,000 iterations)
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
C#
C++/CX
C++/WinRT


Arrays (CertificateQuery.Thumbprint with 10,000,000 iterations)
3.5
3
2.5
2
1.5
1
0.5
0
C#
C++/CX
C++/WinRT

Collections (IVectorView<hstring> with 10,000,000 elements)





Compilers & metadata

Visual C++ optimizations

Empty base classes

strlen/wcslen

Magic statics

Pure functions

Coroutines

Modules

More information

Come to CoroutineCon tomorrow!

- 9:00am: An Introduction to C++ Coroutines
- 2:00pm: C++ Coroutines: Under the covers
- 3:15pm: Putting Coroutines to Work with the Windows Runtime

Web: https://moderncpp.com

Email: kenny.kerr@microsoft.com

Twitter: @KennyKerr & @JamesMcNellis

\0