Evaluation Topics

Chromium based Edge usage in WPF
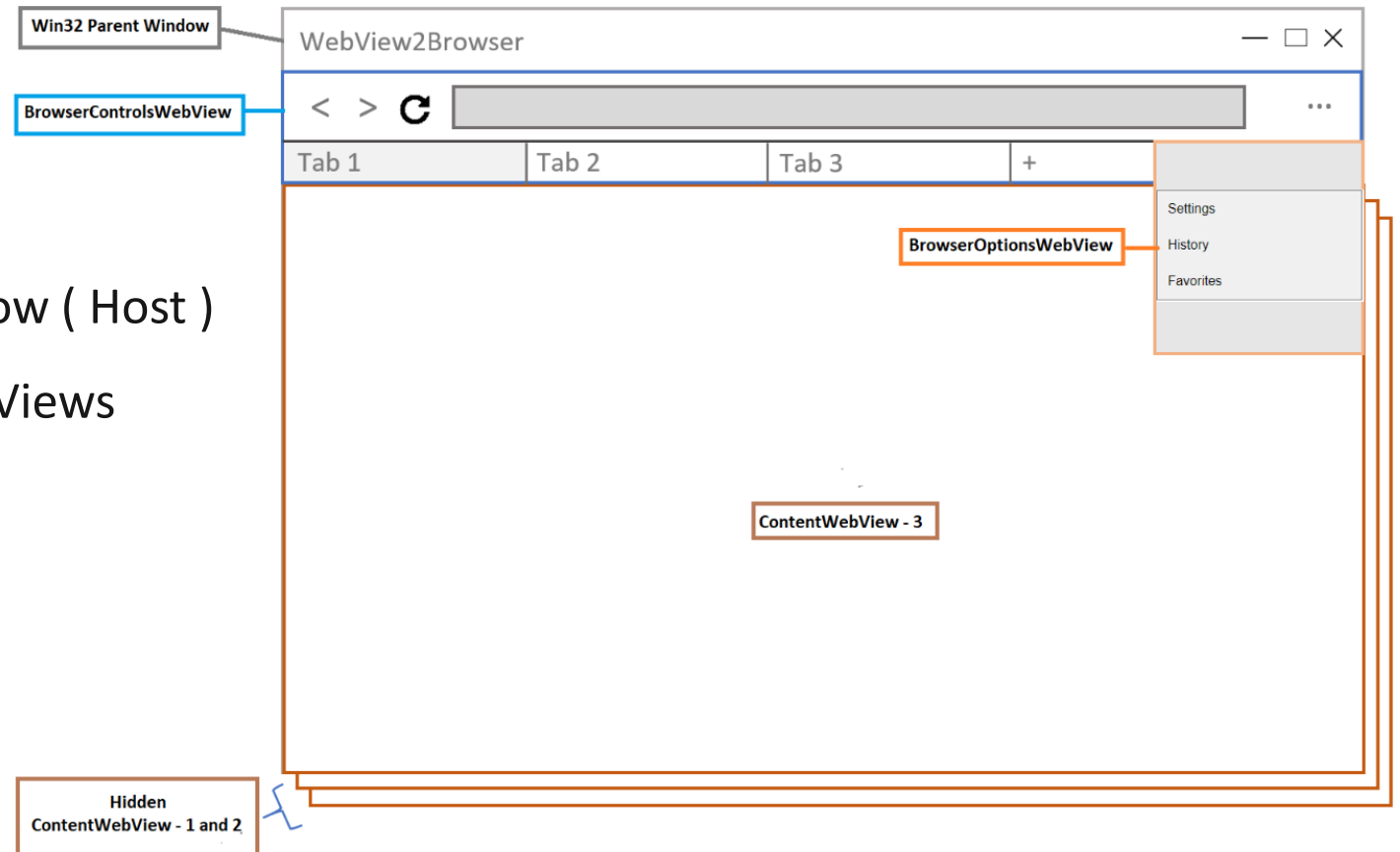
Chromium based Edge Control - Webview2

Current state from Microsoft

Developer preview is first available for Win32 C++ on Windows 10.
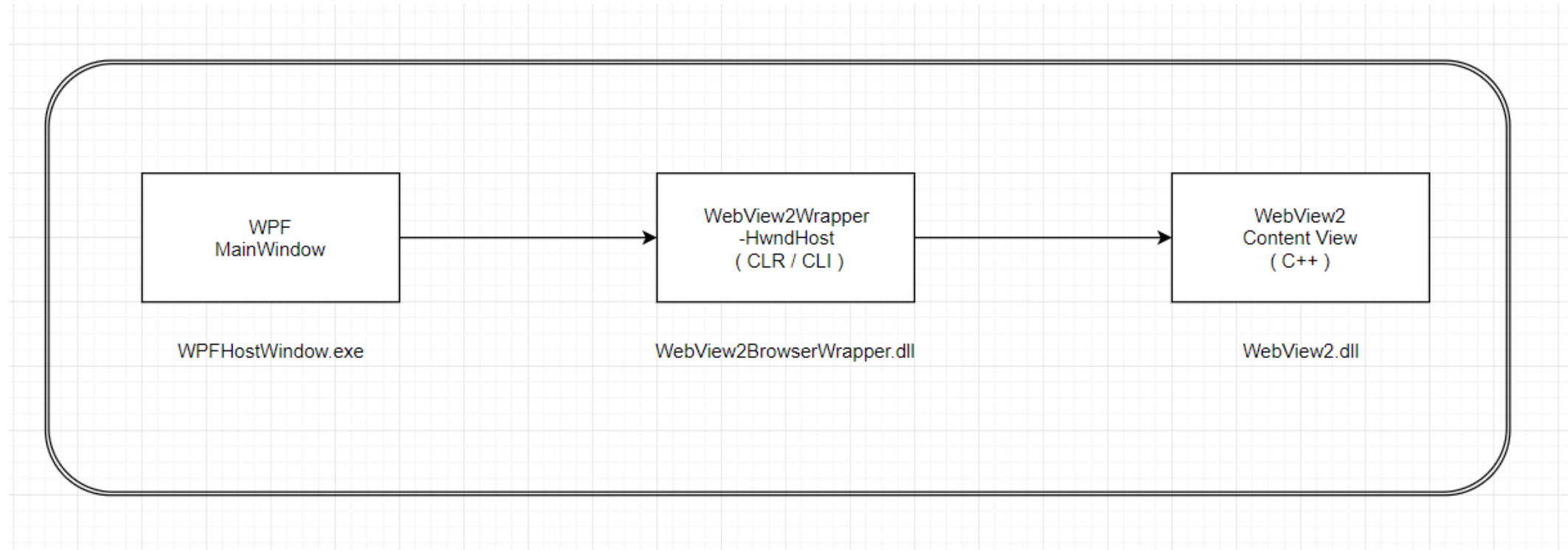
WebView2 Developer preview:

- A single window win32 app

- Multiple WebViews within Parent Window (Host)

- Communication between host and WebViews

- Navigate API in WebView2 to Load Page
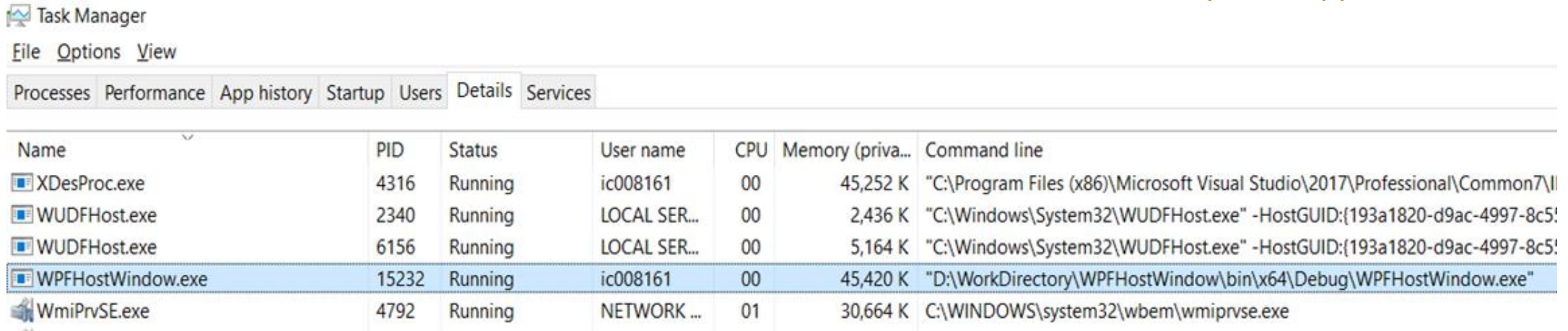
  (IWebView2WebView::Navigate)

Link:

https://blogs.windows.com/msedgedev/2019/08/15/webview2browse-sample-for-webview2/#PYmmq2UhbdOyjRPV.97

Hosting Of Webview2 in WPF Window

```
┌─────────────────┐        ┌─────────────────┐        ┌─────────────────┐
│                 │        │  WebView2Wrapper│        │    WebView2     │
│      WPF        │───────▶│   -HwndHost     │───────▶│   Content View  │
│   MainWindow    │        │   ( CLR / CLI ) │        │    ( C++ )      │
│                 │        │                 │        │                 │
└─────────────────┘        └─────────────────┘        └─────────────────┘

  WPFHostWindow.exe         WebView2BrowserWrapper.dll      WebView2.dll
```

Process View – Before & After WebView2 Hosting in WPF

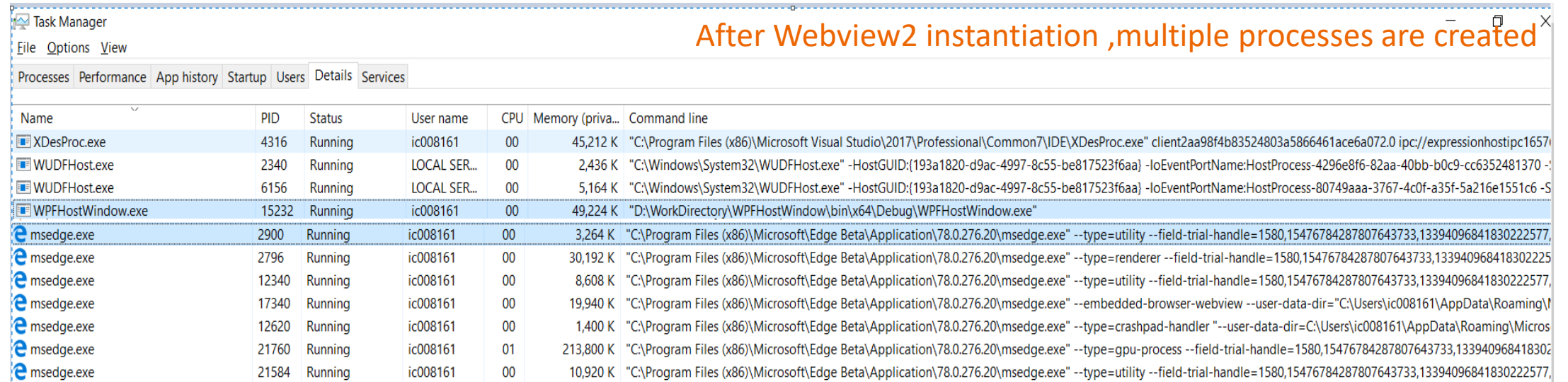Before Webview2 instantiation, only WPF Application exists

After Webview2 instantiation ,multiple processes are created

Planned Activities

- WebView2 hosted WPF Vs Electron desktop app Performance Evaluation

    (with Table Data in various scenarios)

- WebView2 Resize handling

- Key Input handling (Ex: Tab)

- Demo - WebOrganizer integrated to WPF

- Demo - SHUI Image Segment integrated to WPF

- Communication from WPF to WebView2 (Using ExecuteScript API)

- WebView2 Events

- Enabling / Disabling DevTools by Context Menu or Shortcuts

- WebView2 Zoomfactor

- Kiosk mode behavior

Performance Evaluation

Performance Values are measured for

- WPF Data Grid (100 Rows)
- WPF – WebView2
    - Static HTML Table (100 Rows)
    - Angular 9 HTML Table (100 Rows)
    - Angular 9 Material Table (100 Rows)
- Electron
    - Static HTML Table (100 Rows)
    - Angular 9 HTML Table (100 Rows)
    - Angular 9 Material Table (100 Rows)

Performance Evaluation Scenarios are,

- Application Startup + Content Rendering
- WebView2 Navigate API (Upon User Interaction)

Performance Evaluation Setup

1. WebView2 C++ Layer, C# Wrapper and all Applications are built on Debug x64
2. Angular 9 Served with Production Build (AOT Enabled => Ahead-Of-Time)
3. Angular 9 Ivy compilation Enabled and Disabled scenarios are considered.

Demo - Snapshots

Performance Evaluation Results

Scenario – I Application Startup + Content Rendering	WPF DataGrid (With 100 Rows)	WebView2 With Static HTML Table (With 100 Rows) Application Loading	WebView2 With Angular HTML Table (With 100 Rows) Application Loading		WebView2 With Angular Material Table (With 100 Rows) Application Loading	
			Ivy Disabled	Ivy Enabled	Ivy Disabled	Ivy Enabled
Cold Startup	150 ms	360 ms	1028 ms	850 ms	1615 ms	1503 ms
Warm Startup	147 ms	345 ms	850 ms	850 ms	1523 ms	1502 ms
		Electron With Static HTML Table (With 100 Rows)	Electron With Angular HTML Table (With 100 Rows)		Electron With Angular Material Table (With 100 Rows)	
			Ivy Disabled	Ivy Enabled	Ivy Disabled	Ivy Enabled
Cold Startup		291 ms	1369 ms	963 ms	1670 ms	1544 ms
Scenario – II WebView2 Navigate API (Upon User Interaction)		WebView2 With Static HTML Table (With 100 Rows) WebView2 - Navigate API	WebView2 With Angular HTML Table (With 100 Rows) WebView2 - Navigate API		WebView2 With Angular Material Table (With 100 Rows) WebView2 - Navigate API	
			Ivy Disabled	Ivy Enabled	Ivy Disabled	Ivy Enabled
Cold Startup		96 ms	337 ms	290 ms	368 ms	268 ms
Warm Startup		92 ms	300 ms	272 ms	300 ms	232 ms

**Based on the values WebView2 Initialization consume considerable amount of time during application startup.

WebView2 Resize Handling

To make Resize proper as expected, WM_SIZE message has to be handled in C# Wrapper.

In the Wrapper (inheriting from HwndHost) we have to override WndProc method and resize WebView2 by API provided by Microsoft.

```cpp
System::IntPtr WndProc(System::IntPtr hwnd, int msg, System::IntPtr wParam, System::IntPtr lParam, bool% handled) override
{
    handled = false;
    switch (msg)
    {
        case WM_SIZE:
        {
            if (myBrowserWindow != nullptr)
            {
                RECT bounds;
                GetClientRect((HWND)hwnd.ToPointer(), &bounds);
                int newWidth = LOWORD((DWORD)lParam.ToPointer());
                int newHeight = HIWORD((DWORD)lParam.ToPointer());
                MoveWindow((HWND)myWin32Window.ToPointer(), bounds.left + 5, bounds.top + 5, newWidth, newHeight, TRUE);
                myBrowserWindow->ResizeTabWebViews();
            }
        }
        break;
    }
    return System::IntPtr::Zero;
}
```

Key Input Handling

To make Tab proper as expected, TabInto method has to be handled in C# Wrapper.

In the Wrapper (inheriting from HwndHost), we have to override **TabInto** method and set focus WebView2 by API provided by Microsoft.

```
virtual bool TabInto(TraversalRequest^ request) override
{
    if (request->FocusNavigationDirection == FocusNavigationDirection::First)
    {
        myBrowserWindow->SetFocus();
    }

    return true;
}
```

Communication-WPF->WebView2(Using ExecuteScript API)

Code Snippet to Inject JavaScript to WebView2 as per previous Demo.

```
1 reference
private void OnPostScriptClick(object sender, RoutedEventArgs e)
{
    var script1 = "var canvas = document.getElementById(\"ImageCanvas1\");" +
                  "var ctx = canvas.getContext(\"2d\");" +
                  "var img = document.getElementById(\"source1\");" +
                  "ctx.drawImage(img, 0, 0);";

    var script2 = "var canvas = document.getElementById(\"ImageCanvas2\");" +
                  "var ctx = canvas.getContext(\"2d\");" +
                  "var img = document.getElementById(\"source2\");" +
                  "ctx.drawImage(img, 0, 0);";

    myWebView2BrowserWrapper1.PostScript(script1);
    myWebView2BrowserWrapper1.PostScript(script2);
}
```

PS:

If multiple instance of WebView2 is hosted in a WPF window, we can use ExecuteScript as a communication mechanism across WebView2 instances via WPF.

Enabling / Disabling DevTools by Context Menu or Keyboard Shortcuts

WebView2 provides API to control the Visibilty of DevTools(either by Context menu or Keyboard Shortcuts) by interface IWebView2Settings.

You can disable as shown in below code snippet in our end C++ layer (Dll).

```
wil::com_ptr<IWebView2Settings> settings;
m_TabStates.at(m_ContentWebViewId)->m_contentWebView.Get()->get_Settings(&settings);
settings->put_AreDevToolsEnabled(FALSE);
```

PS:

Unfortunately disabling of DevTools is not working in the version of WebView2 which we are currently working. We have to test it in latest WebView2 SDK's.