Reference (WebView2)

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

The Microsoft Edge WebView2 control enables you to host web content in your application using Microsoft Edge (Chromium) as the rendering engine. For more information, see Overview of Microsoft Edge WebView2) and Getting Started with WebView2. IWebView2WebView is a great place to start learning the details of the API.

Globals

- Globals

Interfaces

- IWebView2Deferral
- IWebView2Environment
- IWebView2Environment2
- IWebView2Environment3
- IWebView2HttpHeadersCollectionIterator
- IWebView2HttpRequestHeaders
- IWebView2HttpResponseHeaders
- IWebView2Settings
- IWebView2Settings2
- IWebView2WebResourceRequest
- IWebView2WebResourceRequestedEventArgs2
- IWebView2WebResourceResponse
- IWebView2WebView
- IWebView2WebView2
- IWebView2WebView3
- IWebView2WebView4
- IWebView2WebView5

Event argument interfaces

- IWebView2AcceleratorKeyPressedEventArgs
- IWebView2DevToolsProtocolEventReceivedEventArgs
- IWebView2DocumentStateChangedEventArgs
- IWebView2MoveFocusRequestedEventArgs
- IWebView2NavigationCompletedEventArgs
- IWebView2NavigationStartingEventArgs
- IWebView2NewVersionAvailableEventArgs
- IWebView2NewWindowRequestedEventArgs
- IWebView2PermissionRequestedEventArgs
- IWebView2ProcessFailedEventArgs
- IWebView2ScriptDialogOpeningEventArgs
- IWebView2WebMessageReceivedEventArgs
- IWebView2WebResourceRequestedEventArgs

Delegate interfaces

- IWebView2AcceleratorKeyPressedEventHandler
- IWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler
- IWebView2CallDevToolsProtocolMethodCompletedHandler
- IWebView2CapturePreviewCompletedHandler
- IWebView2ContainsFullScreenElementChangedEventHandler
- IWebView2CreateWebView2EnvironmentCompletedHandler
- IWebView2CreateWebViewCompletedHandler
- IWebView2DevToolsProtocolEventReceivedEventHandler
- IWebView2DocumentStateChangedEventHandler
- IWebView2DocumentTitleChangedEventHandler
- IWebView2ExecuteScriptCompletedHandler
- IWebView2FocusChangedEventHandler
- IWebView2MoveFocusRequestedEventHandler
- IWebView2NavigationCompletedEventHandler
- IWebView2NavigationStartingEventHandler
- IWebView2NewVersionAvailableEventHandler
- IWebView2NewWindowRequestedEventHandler
- IWebView2PermissionRequestedEventHandler
- IWebView2ProcessFailedEventHandler
- IWebView2ScriptDialogOpeningEventHandler
- IWebView2WebMessageReceivedEventHandler
- IWebView2WebResourceRequestedEventHandler
- IWebView2ZoomFactorChangedEventHandler

interface IWebView2Deferral

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2Deferral
  : public IUnknown
```

This interface is used to complete deferrals on event args that support getting deferrals via their GetDeferral method.

Summary

Members	Descriptions
Complete	Completes the associated deferred event.

Members

Complete

Completes the associated deferred event.

> public HRESULT Complete()

Complete should only be called once for each deferral taken.

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2Environment

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2Environment
  : public IUnknown
```

This represents the WebView2 Environment.

Summary

Members	Descriptions
CreateWebView	Asynchronously create a new IWebView2WebView.
CreateWebResourceResponse	Create a new web resource response object.

WebViews created from an environment run on the Browser process specified with environment parameters and objects created from an environment should be used in the same environment. Using it in different environments are not guaranteed to be compatible and may fail.

Members

CreateWebView

Asynchronously create a new IWebView2WebView.

> public HRESULT CreateWebView(HWND parentWindow,IWebView2CreateWebViewCompletedHandler * handler)

parentWindow is the HWND in which the WebView should be displayed and from which receive input. The WebView will add a child window to the provided window during

WebView creation. Z-order and other things impacted by sibling window order will be affected accordingly.

It is recommended that the application set Application User Model ID for the process or the application window. If none is set, during WebView creation a generated Application User Model ID is set to root window of parentWindow.

C++

```cpp
// Create or recreate the WebView and its environment.
void AppWindow::InitializeWebView(InitializeWebViewFlags webviewInitFlags)
{
    m_lastUsedInitFlags = webviewInitFlags;
    // To ensure browser switches get applied correctly, we need to close
    // the existing WebView. This will result in a new browser process
    // getting created which will apply the browser switches.
    CloseWebView();

    LPCWSTR subFolder = nullptr;
    LPCWSTR additionalBrowserSwitches = nullptr;
    HRESULT hr = CreateWebView2EnvironmentWithDetails(
        subFolder, nullptr, additionalBrowserSwitches,
        Callback<IWebView2CreateWebView2EnvironmentCompletedHandler>(
            this, &AppWindow::OnCreateEnvironmentCompleted)
            .Get());
    if (!SUCCEEDED(hr))
    {
        if (hr == HRESULT_FROM_WIN32(ERROR_FILE_NOT_FOUND))
        {
            MessageBox(
                m_mainWindow,
                L"Couldn't find Edge installation. "
                "Do you have a version installed that's compatible with this
"
                "WebView2 SDK version?",
                nullptr, MB_OK);
        }
        else
        {
            ShowFailure(hr, L"Failed to create webview environment");
        }
    }
}

// This is the callback passed to CreateWebViewEnvironmnetWithDetails.
// Here we simply create the WebView.
HRESULT AppWindow::OnCreateEnvironmentCompleted(
    HRESULT result, IWebView2Environment* environment)
{
    CHECK_FAILURE(result);

    CHECK_FAILURE(environment-
>QueryInterface(IID_PPV_ARGS(&m_webViewEnvironment)));
```

```
        CHECK_FAILURE(m_webViewEnvironment->CreateWebView(
            m_mainWindow, Callback<IWebView2CreateWebViewCompletedHandler>(
                                  this, &AppWindow::OnCreateWebViewCompleted)
                                  .Get()));
    return S_OK;
}
```

It is recommended that the application handles restart manager messages so that it can be restarted gracefully in the case when the app is using Edge for webview from a certain installation and that installation is being uninstalled. For example, if a user installs Edge from Dev channel and opts to use Edge from that channel for testing the app, and then uninstalls Edge from that channel without closing the app, the app will be restarted to allow uninstallation of the dev channel to succeed.

```C++
    case WM_QUERYENDSESSION:
    {
        // yes, we can shut down
        // Register how we might be restarted
        RegisterApplicationRestart(L"--restore", RESTART_NO_CRASH |
RESTART_NO_HANG);
        *result = TRUE;
        return true;
    }
    break;
    case WM_ENDSESSION:
    {
        if (wParam == TRUE)
        {
            // save app state and exit.
            PostQuitMessage(0);
            return true;
        }
    }
    break;
```

CreateWebResourceResponse

Create a new web resource response object.

> public HRESULT CreateWebResourceResponse(IStream * content,int statusCode,LPCWSTR reasonPhrase,LPCWSTR headers,IWebView2WebResourceResponse ** response)

The headers is the raw response header string delimited by newline. It's also possible to create this object with empty headers string and then use the

IWebView2HttpResponseHeaders to construct the headers line by line. For information on other parameters see IWebView2WebResourceResponse.

C++

```
        if (m_blockImages)
        {
            m_webView->AddWebResourceRequestedFilter(L"*",
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
            CHECK_FAILURE(m_webView->add_WebResourceRequested(
                Callback<IWebView2WebResourceRequestedEventHandler>(
                    [this](
                        IWebView2WebView* sender,
                        IWebView2WebResourceRequestedEventArgs* args) {

wil::com_ptr<IWebView2WebResourceRequestedEventArgs2>
                            webResourceEventArgs2;
                        args-
>QueryInterface(IID_PPV_ARGS(&webResourceEventArgs2));
                        WEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
                        CHECK_FAILURE(
                            webResourceEventArgs2-
>get_ResourceContext(&resourceContext));
                        // Ensure that the type is image
                        if (resourceContext !=
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
                        {
                            return E_INVALIDARG;
                        }
                        // Override the response with an empty one to block
the image.
                        // If put_Response is not called, the request will
continue as normal.
                        wil::com_ptr<IWebView2WebResourceResponse> response;
                        CHECK_FAILURE(m_webViewEnvironment-
>CreateWebResourceResponse(
                            nullptr, 403 /*NoContent*/, L"Blocked", L"",
&response));
                        CHECK_FAILURE(args->put_Response(response.get()));
                        return S_OK;
                    })
                    .Get(),
                &m_webResourceRequestedTokenForImageBlocking));
        }
        else
        {
            CHECK_FAILURE(m_webView->remove_WebResourceRequested(
                m_webResourceRequestedTokenForImageBlocking));
        }
```

interface IWebView2Environment2

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2Environment2
  : public IWebView2Environment
```

Additional functionality implemented by the Environment object.

Summary

Members	Descriptions
get_BrowserVersionInfo	The browser version info of the current IWebView2Environment, including channel name if it is not the stable channel.

See the IWebView2Environment interface for more details. You can QueryInterface for this interface from the object that implements IWebView2Environment.

Members

get_BrowserVersionInfo

The browser version info of the current IWebView2Environment, including channel name if it is not the stable channel.

> public HRESULT get_BrowserVersionInfo(LPWSTR * versionInfo)

This matches the format of the GetWebView2BrowserVersionInfo API. Channel names are 'beta', 'dev', and 'canary'.

C++

```
        wil::unique_cotaskmem_string version_info;
        m_webViewEnvironment->get_BrowserVersionInfo(&version_info);
        MessageBox(
            m_mainWindow, version_info.get(), L"Browser Version Info After
WebView Creation",
            MB_OK);
```

Feedback

interface IWebView2Environment3

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2Environment3
  : public IWebView2Environment2
```

Additional functionality implemented by the Environment object.

Summary

Members	Descriptions
add_NewVersionAvailable	The NewVersionAvailable event fires when a newer version of the Edge browser is installed and available to use via WebView2.
remove_NewVersionAvailable	Remove an event handler previously added with add_NewVersionAvailable.

See the IWebView2Environment interface for more details. You can QueryInterface for this interface from the object that implements IWebView2Environment.

Members

add_NewVersionAvailable

The NewVersionAvailable event fires when a newer version of the Edge browser is installed and available to use via WebView2.

> public HRESULT add_NewVersionAvailable(IWebView2NewVersionAvailableEventHandler * eventHandler,EventRegistrationToken * token)

To use the newer version of the browser you must create a new IWebView2Environment and IWebView2WebView. Event will only be fired for new version from the same Edge channel that the code is running from. When not running with installed Edge, no event will be fired.

Because a user data folder can only be used by one browser process at a time, if you want to use the same user data folder in the WebViews using the new version of the browser, you must close the IWebView2Environment and IWebView2WebViews that are using the older version of the browser first. Or simply prompt the user to restart the app.

```C++
    // After the environment is successfully created,
    // register a handler for the NewVersionAvailable event.
    // This handler tells when there is a new Edge version available on the
machine.
    CHECK_FAILURE(m_webViewEnvironment->add_NewVersionAvailable(
        Callback<IWebView2NewVersionAvailableEventHandler>(
            [this](IWebView2Environment* sender,
IWebView2NewVersionAvailableEventArgs* args)
                -> HRESULT {
                // Get the version value from args
                wil::unique_cotaskmem_string newVersion;
                CHECK_FAILURE(args->get_NewVersion(&newVersion));
                std::wstring message = L"We detected there is a new version
for the browser.";
                message += L"\n\nVersion number: ";
                message += newVersion.get();
                message += L"\n\n";
                if (m_webView)
                {
                    message += L"Do you want to restart the app? \n\n";
                    message += L"Click No if you only want to re-create the
webviews. \n";

                    message += L"Click Cancel for no action. \n";
                }
                int response = MessageBox(
                    m_mainWindow, message.c_str(), L"New available version",
                    m_webView ? MB_YESNOCANCEL : MB_OK);

                if (response == IDYES)
                {
                    RestartApp();
                }
                else if (response == IDNO)
                {
                    ReinitializeWebViewWithNewBrowser();
                }
                else
                {
                    // do nothing
                }

                return S_OK;
            })
            .Get(),
        nullptr));
```

remove_NewVersionAvailable

Remove an event handler previously added with add_NewVersionAvailable.

> public HRESULT remove_NewVersionAvailable(EventRegistrationToken token)

---

interface IWebView2HttpHeadersCollectionIterator

Article • 10/14/2020

> Note
>
> This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2HttpHeadersCollectionIterator
  : public IUnknown
```

Iterator for a collection of HTTP headers.

Summary

Members	Descriptions
GetCurrentHeader	Get the name and value of the current HTTP header of the iterator.
MoveNext	Move the iterator to the next HTTP header in the collection.

See IWebView2HttpRequestHeaders and IWebView2HttpResponseHeaders.

Members

GetCurrentHeader

Get the name and value of the current HTTP header of the iterator.

> public HRESULT GetCurrentHeader(LPWSTR * name,LPWSTR * value)

This method will fail if the last call to MoveNext set has_next to FALSE.

MoveNext

Move the iterator to the next HTTP header in the collection.

> public HRESULT MoveNext(BOOL * has_next)

The has_next parameter will be set to FALSE if there are no more HTTP headers. After this occurs the GetCurrentHeader method will fail if called.

---

interface IWebView2HttpRequestHeaders

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2HttpRequestHeaders
  : public IUnknown
```

HTTP request headers.

Summary

Members	Descriptions
GetHeader	Gets the header value matching the name.
Contains	Checks whether the headers contain an entry matching the header name.
SetHeader	Adds or updates header that matches the name.
RemoveHeader	Removes header that matches the name.
GetIterator	Gets an iterator over the collection of request headers.

Used to inspect the HTTP request on WebResourceRequested event and NavigationStarting event. Note, you can modify the HTTP request headers from a WebResourceRequested event, but not from a NavigationStarting event.

Members

GetHeader

Gets the header value matching the name.

> public HRESULT GetHeader(LPCWSTR name,LPWSTR * value)

Contains

Checks whether the headers contain an entry matching the header name.

> public HRESULT Contains(LPCWSTR name,BOOL * contains)

SetHeader

Adds or updates header that matches the name.

> public HRESULT SetHeader(LPCWSTR name,LPCWSTR value)

RemoveHeader

Removes header that matches the name.

> public HRESULT RemoveHeader(LPCWSTR name)

GetIterator

Gets an iterator over the collection of request headers.

> public HRESULT GetIterator(IWebView2HttpHeadersCollectionIterator ** iterator)

Feedback

interface IWebView2HttpResponseHeaders

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2HttpResponseHeaders
  : public IUnknown
```

HTTP response headers.

Summary

Members	Descriptions
AppendHeader	Appends header line with name and value.
Contains	Checks whether the headers contain entries matching the header name.
GetHeaders	Gets the header values matching the name.
GetIterator	Gets an iterator over the collection of entire response headers.

Used to construct a WebResourceResponse for the WebResourceRequested event.

Members

AppendHeader

Appends header line with name and value.

> public HRESULT AppendHeader(LPCWSTR name,LPCWSTR value)

Contains

Checks whether the headers contain entries matching the header name.

> public HRESULT Contains(LPCWSTR name,BOOL * contains)

GetHeaders

Gets the header values matching the name.

> public HRESULT GetHeaders(LPCWSTR name,IWebView2HttpHeadersCollectionIterator ** iterator)

GetIterator

Gets an iterator over the collection of entire response headers.

> public HRESULT GetIterator(IWebView2HttpHeadersCollectionIterator ** iterator)

interface IWebView2Settings

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2Settings
  : public IUnknown
```

Defines properties that enable, disable, or modify WebView features.

Summary

Members	Descriptions
get_IsScriptEnabled	Controls if JavaScript execution is enabled in all future navigations in the WebView.
put_IsScriptEnabled	Set the IsScriptEnabled property.
get_IsWebMessageEnabled	The IsWebMessageEnabled property is used when loading a new HTML document.
put_IsWebMessageEnabled	Set the IsWebMessageEnabled property.
get_AreDefaultScriptDialogsEnabled	AreDefaultScriptDialogsEnabled is used when loading a new HTML document.
put_AreDefaultScriptDialogsEnabled	Set the AreDefaultScriptDialogsEnabled property.
get_IsFullscreenAllowed_deprecated	This setting is deprecated and will always return false.
put_IsFullscreenAllowed_deprecated	This setting is deprecated and will have no effect.
get_IsStatusBarEnabled	IsStatusBarEnabled controls whether the status bar will be displayed.
put_IsStatusBarEnabled	Set the IsStatusBarEnabled property.

Members	Descriptions
get_AreDevToolsEnabled	AreDevToolsEnabled controls whether the user is able to use the context menu or keyboard shortcuts to open the DevTools window.
put_AreDevToolsEnabled	Set the AreDevToolsEnabled property.

Setting changes made after NavigationStarting event will not apply until the next top level navigation.

Members

get_IsScriptEnabled

Controls if JavaScript execution is enabled in all future navigations in the WebView.

> public HRESULT get_IsScriptEnabled(BOOL * isScriptEnabled)

This only affects scripts in the document; scripts injected with ExecuteScript will run even if script is disabled. It is true by default.

```C++
        // Changes to settings will apply at the next navigation, which
includes the
        // navigation after a NavigationStarting event.  We can use this to
change
        // settings according to what site we're visiting.
        if (ShouldBlockScriptForUri(uri.get()))
        {
            m_settings->put_IsScriptEnabled(FALSE);
        }
        else
        {
            m_settings->put_IsScriptEnabled(m_isScriptEnabled);
        }
```

put_IsScriptEnabled

Set the IsScriptEnabled property.

> public HRESULT put_IsScriptEnabled(BOOL isScriptEnabled)

get_IsWebMessageEnabled

The IsWebMessageEnabled property is used when loading a new HTML document.

> public HRESULT get_IsWebMessageEnabled(BOOL * isWebMessageEnabled)

If set to true, communication from the host to the webview's top level HTML document is allowed via PostWebMessageAsJson, PostWebMessageAsString, and window.chrome.webview's message event (see PostWebMessageAsJson documentation for details). Communication from the webview's top level HTML document to the host is allowed via window.chrome.webview's postMessage function and the SetWebMessageReceivedEventHandler method (see the SetWebMessageReceivedEventHandler documentation for details). If set to false, then communication is disallowed. PostWebMessageAsJson and PostWebMessageAsString will fail with E_ACCESSDENIED and window.chrome.webview.postMessage will fail by throwing an instance of an Error object. It is true by default.

```C++
    ComPtr<IWebView2Settings> settings;
    CHECK_FAILURE(m_webView->get_Settings(&settings));

    CHECK_FAILURE(settings->put_IsWebMessageEnabled(TRUE));
```

put_IsWebMessageEnabled

Set the IsWebMessageEnabled property.

> public HRESULT put_IsWebMessageEnabled(BOOL isWebMessageEnabled)

get_AreDefaultScriptDialogsEnabled

AreDefaultScriptDialogsEnabled is used when loading a new HTML document.

> public HRESULT get_AreDefaultScriptDialogsEnabled(BOOL * areDefaultScriptDialogsEnabled)

If set to false, then WebView won't render the default javascript dialog box (Specifically those shown by the javascript alert, confirm, and prompt functions). Instead, if an event handler is set by SetScriptDialogOpeningEventHandler, WebView will send an event that will contain all of the information for the dialog and allow the host app to show its own custom UI.

put_AreDefaultScriptDialogsEnabled

Set the AreDefaultScriptDialogsEnabled property.

> public HRESULT put_AreDefaultScriptDialogsEnabled(BOOL areDefaultScriptDialogsEnabled)

get_IsFullscreenAllowed_deprecated

This setting is deprecated and will always return false.

> public HRESULT get_IsFullscreenAllowed_deprecated(BOOL * isFullscreenAllowed)

That means elements in the WebView will only fill the WebView bounds. This property will then be completely removed. Please listen to the ContainsFullScreenElementChanged event instead.

Controls if fullscreen is allowed for elements in the WebView. When it is allowed, web content such as a video element in the WebView is allowed to be displayed full screen. When it is not allowed, such element will fill the WebView bounds when the element requests full screen. It is true by default.

put_IsFullscreenAllowed_deprecated

This setting is deprecated and will have no effect.

> public HRESULT put_IsFullscreenAllowed_deprecated(BOOL isFullscreenAllowed)

Please listen to the ContainsFullScreenElementChanged event instead.

Set the IsFullscreenAllowed property.

get_IsStatusBarEnabled

IsStatusBarEnabled controls whether the status bar will be displayed.

> public HRESULT get_IsStatusBarEnabled(BOOL * isStatusBarEnabled)

The status bar is usually displayed in the lower left of the WebView and shows things such as the URI of a link when the user hovers over it and other information. It is true by default.

put_IsStatusBarEnabled

Set the IsStatusBarEnabled property.

> public HRESULT put_IsStatusBarEnabled(BOOL isStatusBarEnabled)

get_AreDevToolsEnabled

AreDevToolsEnabled controls whether the user is able to use the context menu or keyboard shortcuts to open the DevTools window.

> public HRESULT get_AreDevToolsEnabled(BOOL * areDevToolsEnabled)

It is true by default.

put_AreDevToolsEnabled

Set the AreDevToolsEnabled property.

> public HRESULT put_AreDevToolsEnabled(BOOL areDevToolsEnabled)

---

interface IWebView2Settings2

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2Settings2
  : public IWebView2Settings
```

Defines properties that enable, disable, or modify WebView features.

Summary

Members	Descriptions
get_AreDefaultContextMenusEnabled	The AreDefaultContextMenusEnabled property is used to prevent default context menus from being shown to user in webview.
put_AreDefaultContextMenusEnabled	Set the AreDefaultContextMenusEnabled property.

Setting changes made after NavigationStarting event will not apply until the next top level navigation.

Members

get_AreDefaultContextMenusEnabled

The AreDefaultContextMenusEnabled property is used to prevent default context menus from being shown to user in webview.

> public HRESULT get_AreDefaultContextMenusEnabled(BOOL * enabled)

Defaults to TRUE.

C++

```
            BOOL allowContextMenus;
            CHECK_FAILURE(m_settings->get_AreDefaultContextMenusEnabled(
                &allowContextMenus));
            if (allowContextMenus) {
                CHECK_FAILURE(m_settings-
>put_AreDefaultContextMenusEnabled(FALSE));
                MessageBox(nullptr,
                L"Context menus will be disabled after the next
navigation.",
                L"Settings change", MB_OK);
            }
            else {
                CHECK_FAILURE(m_settings-
>put_AreDefaultContextMenusEnabled(TRUE));
                MessageBox(nullptr,
                    L"Context menus will be enabled after the next
navigation.",
                    L"Settings change", MB_OK);
            }
```

put_AreDefaultContextMenusEnabled

Set the AreDefaultContextMenusEnabled property.

> public HRESULT put_AreDefaultContextMenusEnabled(BOOL enabled)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebResourceRequest

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebResourceRequest
  : public IUnknown
```

An HTTP request used with the WebResourceRequested event.

Summary

Members	Descriptions
get_Uri	The request URI.
put_Uri	Set the Uri property.
get_Method	The HTTP request method.
put_Method	Set the Method property.
get_Content	The HTTP request message body as stream.
put_Content	Set the Content property.
get_Headers	The mutable HTTP request headers.

Members

get_Uri

The request URI.

> public HRESULT get_Uri(LPWSTR * uri)

put_Uri

Set the Uri property.

> public HRESULT put_Uri(LPCWSTR uri)

get_Method

The HTTP request method.

> public HRESULT get_Method(LPWSTR * method)

put_Method

Set the Method property.

> public HRESULT put_Method(LPCWSTR method)

get_Content

The HTTP request message body as stream.

> public HRESULT get_Content(IStream ** content)

POST data would be here. If a stream is set, which will override the message body, the stream must have all the content data available by the time this response's WebResourceRequested event deferral is completed. Stream should be agile or be created from a background STA to prevent performance impact to the UI thread. Null means no content data. IStream semantics apply (return S_OK to Read calls until all data is exhausted)

put_Content

Set the Content property.

> public HRESULT put_Content(IStream * content)

get_Headers

The mutable HTTP request headers.

> public HRESULT get_Headers(IWebView2HttpRequestHeaders ** headers)

interface IWebView2WebResourceRequestedEventArgs2

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebResourceRequestedEventArgs2
  : public IWebView2WebResourceRequestedEventArgs
```

Event args for the WebResourceRequested event.

Summary

Members	Descriptions
get_ResourceContext	The web resource request contexts.

Members

get_ResourceContext

The web resource request contexts.

> public HRESULT get_ResourceContext(WEBVIEW2_WEB_RESOURCE_CONTEXT * context)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebResourceResponse

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebResourceResponse
  : public IUnknown
```

An HTTP response used with the WebResourceRequested event.

Summary

Members	Descriptions
get_Content	HTTP response content as stream.
put_Content	Set the Content property.
get_Headers	Overridden HTTP response headers.
get_StatusCode	The HTTP response status code.
put_StatusCode	Set the StatusCode property.
get_ReasonPhrase	The HTTP response reason phrase.
put_ReasonPhrase	Set the ReasonPhrase property.

Members

get_Content

HTTP response content as stream.

> public HRESULT get_Content(IStream ** content)

Stream must have all the content data available by the time this response's WebResourceRequested event deferral is completed. Stream should be agile or be created from a background thread to prevent performance impact to the UI thread. Null means no content data. IStream semantics apply (return S_OK to Read calls until all data is exhausted)

put_Content

Set the Content property.

> public HRESULT put_Content(IStream * content)

get_Headers

Overridden HTTP response headers.

> public HRESULT get_Headers(IWebView2HttpResponseHeaders ** headers)

get_StatusCode

The HTTP response status code.

> public HRESULT get_StatusCode(int * statusCode)

put_StatusCode

Set the StatusCode property.

> public HRESULT put_StatusCode(int statusCode)

get_ReasonPhrase

The HTTP response reason phrase.

> public HRESULT get_ReasonPhrase(LPWSTR * reasonPhrase)

put_ReasonPhrase

Set the ReasonPhrase property.

> public HRESULT put_ReasonPhrase(LPCWSTR reasonPhrase)

interface IWebView2WebView

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2WebView
  : public IUnknown
```

WebView2 enables you to host web content using the latest Edge web browser technology.

Summary

Members	Descriptions
get_Settings	The IWebView2Settings object contains various modifiable settings for the running WebView.
get_Source	The URI of the current top level document.
Navigate	Cause a navigation of the top level document to the specified URI.
MoveFocus	Move focus into WebView.
NavigateToString	Initiates a navigation to htmlContent as source HTML of a new document.
add_NavigationStarting	Add an event handler for the NavigationStarting event.
remove_NavigationStarting	Remove an event handler previously added with add_NavigationStarting.
add_DocumentStateChanged	Add an event handler for the DocumentStateChanged event.

Members	Descriptions
remove_DocumentStateChanged	Remove an event handler previously added with add_DocumentStateChanged.
add_NavigationCompleted	Add an event handler for the NavigationCompleted event.
remove_NavigationCompleted	Remove an event handler previously added with add_NavigationCompleted.
add_FrameNavigationStarting	Add an event handler for the FrameNavigationStarting event.
remove_FrameNavigationStarting	Remove an event handler previously added with add_FrameNavigationStarting.
add_MoveFocusRequested	Add an event handler for the MoveFocusRequested event.
remove_MoveFocusRequested	Remove an event handler previously added with add_MoveFocusRequested.
add_GotFocus	Add an event handler for the GotFocus event.
remove_GotFocus	Remove an event handler previously added with add_GotFocus.
add_LostFocus	Add an event handler for the LostFocus event.
remove_LostFocus	Remove an event handler previously added with add_LostFocus.
add_WebResourceRequested_deprecated	This API will be deprecated, please use the new add_WebResourceRequested API.
remove_WebResourceRequested	Remove an event handler previously added with add_WebResourceRequested.
add_ScriptDialogOpening	Add an event handler for the ScriptDialogOpening event.
remove_ScriptDialogOpening	Remove an event handler previously added with add_ScriptDialogOpening.
add_ZoomFactorChanged	Add an event handler for the ZoomFactorChanged event.
remove_ZoomFactorChanged	Remove an event handler previously added with add_ZoomFactorChanged.

Members	Descriptions
add_PermissionRequested	Add an event handler for the PermissionRequested event.
remove_PermissionRequested	Remove an event handler previously added with add_PermissionRequested.
add_ProcessFailed	Add an event handler for the ProcessFailed event.
remove_ProcessFailed	Remove an event handler previously added with add_ProcessFailed.
AddScriptToExecuteOnDocumentCreated	Add the provided JavaScript to a list of scripts that should be executed after the global object has been created, but before the HTML document has been parsed and before any other script included by the HTML document is executed.
RemoveScriptToExecuteOnDocumentCreated	Remove the corresponding JavaScript added via AddScriptToExecuteOnDocumentCreated.
ExecuteScript	Execute JavaScript code from the javascript parameter in the current top level document rendered in the WebView.
CapturePreview	Capture an image of what WebView is displaying.
Reload	Reload the current page.
get_Bounds	The webview bounds.
put_Bounds	Set the Bounds property.
get_ZoomFactor	The zoom factor for the current page in the WebView.
put_ZoomFactor	Set the ZoomFactor property.
get_IsVisible	The IsVisible property determines whether to show or hide the webview.
put_IsVisible	Set the IsVisible property.
PostWebMessageAsJson	Post the specified webMessage to the top level document in this IWebView2WebView.

Members	Descriptions
PostWebMessageAsString	This is a helper for posting a message that is a simple string rather than a JSON string representation of a JavaScript object.
add_WebMessageReceived	This event fires when the IsWebMessageEnabled setting is set and the top level document of the webview calls `window.chrome.webview.postMessage`.
remove_WebMessageReceived	Remove an event handler previously added with add_WebMessageReceived.
Close	Closes the webview and cleans up the underlying browser instance.
CallDevToolsProtocolMethod	Call an asynchronous DevToolsProtocol method.
add_DevToolsProtocolEventReceived	Subscribe to a DevToolsProtocol event.
remove_DevToolsProtocolEventReceived	Remove an event handler previously added with add_DevToolsProtocolEventReceived.
get_BrowserProcessId	The process id of the browser process that hosts the WebView.
get_CanGoBack	Can navigate the webview to the previous page in the navigation history.
get_CanGoForward	Can navigate the webview to the next page in the navigation history.
GoBack	Navigates the webview to the previous page in the navigation history.
GoForward	Navigates the webview to the next page in the navigation history.
WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT	Image format used by the IWebView2WebView::CapturePreview method.
WEBVIEW2_SCRIPT_DIALOG_KIND	Kind of JavaScript dialog used in the IWebView2ScriptDialogOpeningEventHandler interface.
WEBVIEW2_PROCESS_FAILED_KIND	Kind of process failure used in the IWebView2ProcessFailedEventHandler interface.
WEBVIEW2_PERMISSION_TYPE	The type of a permission request.

Members	Descriptions
WEBVIEW2_PERMISSION_STATE	Response to a permission request.
WEBVIEW2_MOVE_FOCUS_REASON	Reason for moving focus.
WEBVIEW2_WEB_ERROR_STATUS	Error status values for web navigations.
WEBVIEW2_WEB_RESOURCE_CONTEXT	Enum for web resource request contexts.

Members

get_Settings

The IWebView2Settings object contains various modifiable settings for the running WebView.

> public HRESULT get_Settings(IWebView2Settings ** settings)

get_Source

The URI of the current top level document.

> public HRESULT get_Source(LPWSTR * uri)

This value potentially changes as a part of the DocumentStateChanged event firing for some cases such as navigating to a different site or fragment navigations. It will remain the same for other types of navigations such as page reloads or history.pushState with the same URL as the current page.

```C++
    // Register a handler for the DocumentStateChanged event.
    // This handler will read the webview's source URI and update
    // the app's address bar.
    CHECK_FAILURE(m_webView->add_DocumentStateChanged(
        Callback<IWebView2DocumentStateChangedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2DocumentStateChangedEventArgs* args)
                -> HRESULT {
                wil::unique_cotaskmem_string uri;
                sender->get_Source(&uri);
                if (wcscmp(uri.get(), L"about:blank") == 0)
                {
                    uri = wil::make_cotaskmem_string(L"");
                }
                SetWindowText(m_toolbar->addressBarWindow, uri.get());

                return S_OK;
            })
            .Get(),
        &m_documentStateChangedToken));
```

Navigate

Cause a navigation of the top level document to the specified URI.

> public HRESULT Navigate(LPCWSTR uri)

See the navigation events for more information. Note that this starts a navigation and the corresponding NavigationStarting event will fire sometime after this Navigate call completes.

```C++
void ControlComponent::NavigateToAddressBar()
{
    WCHAR uri[2048] = L"";
    GetWindowText(m_toolbar->addressBarWindow, uri, ARRAYSIZE(uri));
    CHECK_FAILURE(m_webView->Navigate(uri));
}
```

MoveFocus

Move focus into WebView.

> public HRESULT MoveFocus(WEBVIEW2_MOVE_FOCUS_REASON reason)

WebView will get focus and focus will be set to correspondent element in the page hosted in the WebView. For Programmatic reason, focus is set to previously focused element or the default element if there is no previously focused element. For Next reason, focus is set to the first element. For Previous reason, focus is set to the last element. WebView can also got focus through user interaction like clicking into WebView or Tab into it. For tabbing, the app can call MoveFocus with Next or Previous to align with tab and shift+tab respectively when it decides the WebView is the next tabbable element. Or, the app can call IsDialogMessage as part of its message loop to allow the platform to auto handle tabbing. The platform will rotate through all windows with WS_TABSTOP. When the WebView gets focus from IsDialogMessage, it will internally put the focus on the first or last element for tab and shift+tab respectively.

C++

```cpp
    while (GetMessage(&msg, nullptr, 0, 0))
    {
        if (!TranslateAccelerator(msg.hwnd, hAccelTable, &msg))
        {
            // Calling IsDialogMessage handles Tab traversal automatically.
If the
            // app wants the platform to auto handle tab, then call
IsDialogMessage
            // before calling TranslateMessage/DispatchMessage. If the app
wants to
            // handle tabbing itself, then skip calling IsDialogMessage and
call
            // TranslateMessage/DispatchMessage directly.
            if (!g_autoTabHandle || !IsDialogMessage(GetAncestor(msg.hwnd,
GA_ROOT), &msg))
            {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
            }
        }
    }
```

C++

```cpp
        if (wParam == VK_TAB)
        {
            // Find out if the window is one we've customized for tab
handling
            for (int i = 0; i < m_tabbableWindows.size(); i++)
            {
                if (m_tabbableWindows[i].first == hWnd)
                {
                    if (GetKeyState(VK_SHIFT) < 0)
                    {
                        TabBackwards(i);
                    }
                    else
                    {
                        TabForwards(i);
                    }
                    return true;
                }
            }
        }
```

C++

```
void ControlComponent::TabForwards(int currentIndex)
{
    // Find first enabled window after the active one
    for (int i = currentIndex + 1; i < m_tabbableWindows.size(); i++)
    {
        HWND hwnd = m_tabbableWindows.at(i).first;
        if (IsWindowEnabled(hwnd))
        {
            SetFocus(hwnd);
            return;
        }
    }
    // If this is the last enabled window, tab forwards into the WebView.
    m_webView->MoveFocus(WEBVIEW2_MOVE_FOCUS_REASON_NEXT);
}

void ControlComponent::TabBackwards(int currentIndex)
{
    // Find first enabled window before the active one
    for (int i = currentIndex - 1; i >= 0; i--)
    {
        HWND hwnd = m_tabbableWindows.at(i).first;
        if (IsWindowEnabled(hwnd))
        {
            SetFocus(hwnd);
            return;
        }
    }
    // If this is the last enabled window, tab forwards into the WebView.
    CHECK_FAILURE(m_webView-
>MoveFocus(WEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS));
}
```

NavigateToString

Initiates a navigation to htmlContent as source HTML of a new document.

> public HRESULT NavigateToString(LPCWSTR htmlContent)

The htmlContent parameter may not be larger than 2 MB of characters. The origin of the new page will be about:blank.

```
C++
```

```
                static const PCWSTR htmlContent =
                    L"<h1>Domain Blocked</h1>"
                    L"<p>You've attempted to navigate to a domain in the
blocked "
                    L"sites list. Press back to return to the previous page.
```

```
</p>";
                CHECK_FAILURE(sender->NavigateToString(htmlContent));
```

add_NavigationStarting

Add an event handler for the NavigationStarting event.

> public HRESULT add_NavigationStarting(IWebView2NavigationStartingEventHandler * eventHandler,EventRegistrationToken * token)

NavigationStarting fires when the WebView main frame is requesting permission to navigate to a different URI. This will fire for redirects as well.

```C++
    // Register a handler for the NavigationStarting event.
    // This handler will check the domain being navigated to, and if the
domain
    // matches a list of blocked sites, it will cancel the navigation and
    // possibly display a warning page.  It will also disable JavaScript on
    // selected websites.
    CHECK_FAILURE(m_webView->add_NavigationStarting(
        Callback<IWebView2NavigationStartingEventHandler>(
            [this](IWebView2WebView* sender,
                   IWebView2NavigationStartingEventArgs* args) -> HRESULT
    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Uri(&uri));

        if (ShouldBlockUri(uri.get()))
        {
            CHECK_FAILURE(args->put_Cancel(true));

            // If the user clicked a link to navigate, show a warning page.
            BOOL userInitiated;
            CHECK_FAILURE(args->get_IsUserInitiated(&userInitiated));
            if (userInitiated)
            {
                static const PCWSTR htmlContent =
                    L"<h1>Domain Blocked</h1>"
                    L"<p>You've attempted to navigate to a domain in the
blocked "
                    L"sites list. Press back to return to the previous page.
</p>";
                CHECK_FAILURE(sender->NavigateToString(htmlContent));
            }
        }
        // Changes to settings will apply at the next navigation, which
includes the
        // navigation after a NavigationStarting event.  We can use this to
change
        // settings according to what site we're visiting.
        if (ShouldBlockScriptForUri(uri.get()))
        {
            m_settings->put_IsScriptEnabled(FALSE);
        }
        else
        {
            m_settings->put_IsScriptEnabled(m_isScriptEnabled);
        }
        return S_OK;
    }).Get(), &m_navigationStartingToken));
```

remove_NavigationStarting

Remove an event handler previously added with add_NavigationStarting.

> public HRESULT remove_NavigationStarting(EventRegistrationToken token)

add_DocumentStateChanged

Add an event handler for the DocumentStateChanged event.

> public HRESULT add_DocumentStateChanged(IWebView2DocumentStateChangedEventHandler * eventHandler,EventRegistrationToken * token)

DocumentStateChanged fires when new content has started loading on the webview's main frame or if a same page navigation occurs (such as through fragment navigations or history.pushState navigations). This follows the NavigationStarting event and precedes the NavigationCompleted event.

```C++
    // Register a handler for the DocumentStateChanged event.
    // This handler will read the webview's source URI and update
    // the app's address bar.
    CHECK_FAILURE(m_webView->add_DocumentStateChanged(
        Callback<IWebView2DocumentStateChangedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2DocumentStateChangedEventArgs* args)
                -> HRESULT {
                wil::unique_cotaskmem_string uri;
                sender->get_Source(&uri);
                if (wcscmp(uri.get(), L"about:blank") == 0)
                {
                    uri = wil::make_cotaskmem_string(L"");
                }
                SetWindowText(m_toolbar->addressBarWindow, uri.get());

                return S_OK;
            })
            .Get(),
        &m_documentStateChangedToken));
```

remove_DocumentStateChanged

Remove an event handler previously added with add_DocumentStateChanged.

> public HRESULT remove_DocumentStateChanged(EventRegistrationToken token)

add_NavigationCompleted

Add an event handler for the NavigationCompleted event.

> public HRESULT add_NavigationCompleted(IWebView2NavigationCompletedEventHandler * eventHandler,EventRegistrationToken * token)

NavigationCompleted event fires when the WebView has completely loaded (body.onload has fired) or loading stopped with error.

```C++
    // Register a handler for the NavigationCompleted event.
    // Check whether the navigation succeeded, and if not, do something.
    // Also update the Back, Forward, and Cancel buttons.
    CHECK_FAILURE(m_webView->add_NavigationCompleted(
        Callback<IWebView2NavigationCompletedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2NavigationCompletedEventArgs* args)
                -> HRESULT {
                BOOL success;
                CHECK_FAILURE(args->get_IsSuccess(&success));
                if (!success)
                {
                    WEBVIEW2_WEB_ERROR_STATUS webErrorStatus;
                    CHECK_FAILURE(args-
>get_WebErrorStatus(&webErrorStatus));
                    if (webErrorStatus ==
WEBVIEW2_WEB_ERROR_STATUS_DISCONNECTED)
                    {
                        // Do something here if you want to handle a
specific error case.
                        // In most cases this isn't necessary, because the
WebView will
                        // display its own error page automatically.
                    }
                }

                BOOL canGoBack;
                BOOL canGoForward;
                sender->get_CanGoBack(&canGoBack);
                sender->get_CanGoForward(&canGoForward);
                EnableWindow(m_toolbar->backWindow, canGoBack);
                EnableWindow(m_toolbar->forwardWindow, canGoForward);
                EnableWindow(m_toolbar->cancelWindow, FALSE);
                return S_OK;
            })
            .Get(),
        &m_navigationCompletedToken));
```

remove_NavigationCompleted

Remove an event handler previously added with add_NavigationCompleted.

> public HRESULT remove_NavigationCompleted(EventRegistrationToken token)

add_FrameNavigationStarting

Add an event handler for the FrameNavigationStarting event.

> public HRESULT add_FrameNavigationStarting(IWebView2NavigationStartingEventHandler * eventHandler,EventRegistrationToken * token)

FrameNavigationStarting fires when a child frame in the WebView requesting permission to navigate to a different URI. This will fire for redirects as well.

```C++
    // Register a handler for the FrameNavigationStarting event.
    // This handler will prevent a frame from navigating to a blocked
domain.
    CHECK_FAILURE(m_webView->add_FrameNavigationStarting(
        Callback<IWebView2NavigationStartingEventHandler>(
            [this](IWebView2WebView* sender,
                   IWebView2NavigationStartingEventArgs* args) -> HRESULT
    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Uri(&uri));

        if (ShouldBlockUri(uri.get()))
```

```
            {
                CHECK_FAILURE(args->put_Cancel(true));
            }
            return S_OK;
        }).Get(), &m_frameNavigationStartingToken));
```

remove_FrameNavigationStarting

Remove an event handler previously added with add_FrameNavigationStarting.

> public HRESULT remove_FrameNavigationStarting(EventRegistrationToken token)

add_MoveFocusRequested

Add an event handler for the MoveFocusRequested event.

> public HRESULT add_MoveFocusRequested(IWebView2MoveFocusRequestedEventHandler * eventHandler,EventRegistrationToken * token)

MoveFocusRequested fires when user tries to tab out of the WebView. The WebView's focus has not changed when this event is fired.

```C++
    // Register a handler for the MoveFocusRequested event.
    // This event will be fired when the user tabs out of the webview.
    // The handler will focus another window in the app, depending on which
    // direction the focus is being shifted.
    CHECK_FAILURE(m_webView->add_MoveFocusRequested(
        Callback<IWebView2MoveFocusRequestedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2MoveFocusRequestedEventArgs* args)
                -> HRESULT {
                if (!g_autoTabHandle)
                {
                    WEBVIEW2_MOVE_FOCUS_REASON reason;
                    CHECK_FAILURE(args->get_Reason(&reason));

                    if (reason == WEBVIEW2_MOVE_FOCUS_REASON_NEXT)
                    {
                        TabForwards(-1);
                    }
                    else if (reason == WEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS)
                    {
                        TabBackwards(int(m_tabbableWindows.size()));
                    }
                    CHECK_FAILURE(args->put_Handled(TRUE));
                }
                return S_OK;
            })
            .Get(),
        &m_moveFocusRequestedToken));
```

remove_MoveFocusRequested

Remove an event handler previously added with add_MoveFocusRequested.

> public HRESULT remove_MoveFocusRequested(EventRegistrationToken token)

add_GotFocus

Add an event handler for the GotFocus event.

> public HRESULT add_GotFocus(IWebView2FocusChangedEventHandler * eventHandler,EventRegistrationToken * token)

GotFocus fires when WebView got focus.

remove_GotFocus

Remove an event handler previously added with add_GotFocus.

> public HRESULT remove_GotFocus(EventRegistrationToken token)

add_LostFocus

Add an event handler for the LostFocus event.

> public HRESULT add_LostFocus(IWebView2FocusChangedEventHandler * eventHandler,EventRegistrationToken * token)

LostFocus fires when WebView lost focus. In the case where MoveFocusRequested event is fired, the focus is still on WebView when MoveFocusRequested event fires. Lost focus only fires afterwards when app's code or default action of MoveFocusRequested event set focus away from WebView.

remove_LostFocus

Remove an event handler previously added with add_LostFocus.

> public HRESULT remove_LostFocus(EventRegistrationToken token)

add_WebResourceRequested_deprecated

This API will be deprecated, please use the new add_WebResourceRequested API.

> public HRESULT add_WebResourceRequested_deprecated(LPCWSTR *const urlFilter,WEBVIEW2_WEB_RESOURCE_CONTEXT *const resourceContextFilter,SIZE_T filterLength,IWebView2WebResourceRequestedEventHandler * eventHandler,EventRegistrationToken * token)

Add an event handler for the WebResourceRequested event. Fires when the WebView has performs any HTTP request. Use urlFilter to pass in a list with size filterLength of urls to listen for. Each url entry also supports wildcards: '*' matches zero or more characters, and '?' matches exactly one character. For each urlFilter entry, provide a matching resourceContextFilter representing the types of resources for which WebResourceRequested should fire. If filterLength is 0, the event will fire for all network requests. The supported resource contexts are: Document, Stylesheet, Image, Media, Font, Script, XHR, Fetch.

```
C++

        if (m_blockImages)
        {
            m_webView->AddWebResourceRequestedFilter(L"*",
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
            CHECK_FAILURE(m_webView->add_WebResourceRequested(
                Callback<IWebView2WebResourceRequestedEventHandler>(
                    [this](
                        IWebView2WebView* sender,
                        IWebView2WebResourceRequestedEventArgs* args) {

wil::com_ptr<IWebView2WebResourceRequestedEventArgs2>
                            webResourceEventArgs2;
                        args-
>QueryInterface(IID_PPV_ARGS(&webResourceEventArgs2));
                        WEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
                        CHECK_FAILURE(
                            webResourceEventArgs2-
>get_ResourceContext(&resourceContext));
                        // Ensure that the type is image
                        if (resourceContext !=
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
                        {
                            return E_INVALIDARG;
                        }
                        // Override the response with an empty one to block
the image.
                        // If put_Response is not called, the request will
continue as normal.
                        wil::com_ptr<IWebView2WebResourceResponse> response;
                        CHECK_FAILURE(m_webViewEnvironment-
>CreateWebResourceResponse(
                            nullptr, 403 /*NoContent*/, L"Blocked", L"",
&response));
                        CHECK_FAILURE(args->put_Response(response.get()));
                        return S_OK;
                    })
                    .Get(),
                &m_webResourceRequestedTokenForImageBlocking));
        }
        else
        {
            CHECK_FAILURE(m_webView->remove_WebResourceRequested(
                m_webResourceRequestedTokenForImageBlocking));
        }
```

remove_WebResourceRequested

Remove an event handler previously added with add_WebResourceRequested.

> public HRESULT remove_WebResourceRequested(EventRegistrationToken token)

add_ScriptDialogOpening

Add an event handler for the ScriptDialogOpening event.

> public HRESULT add_ScriptDialogOpening(IWebView2ScriptDialogOpeningEventHandler * eventHandler,EventRegistrationToken * token)

The event fires when a JavaScript dialog (alert, confirm, or prompt) will show for the webview. This event only fires if the IWebView2Settings::AreDefaultScriptDialogsEnabled property is set to false.

C++

```
    // Register a handler for the ScriptDialogOpening event.
    // This handler will set up a custom prompt dialog for the user,
    // and may defer the event if the setting to defer dialogs is enabled.
    CHECK_FAILURE(m_webView->add_ScriptDialogOpening(
        Callback<IWebView2ScriptDialogOpeningEventHandler>(
            [this](
                IWebView2WebView* sender,
                IWebView2ScriptDialogOpeningEventArgs* args) -> HRESULT
    {
        wil::com_ptr<IWebView2ScriptDialogOpeningEventArgs> eventArgs =
args;
        auto showDialog = [this, eventArgs]
        {
            wil::unique_cotaskmem_string uri;
            WEBVIEW2_SCRIPT_DIALOG_KIND type;
            wil::unique_cotaskmem_string message;
            wil::unique_cotaskmem_string defaultText;

            CHECK_FAILURE(eventArgs->get_Uri(&uri));
            CHECK_FAILURE(eventArgs->get_Kind(&type));
            CHECK_FAILURE(eventArgs->get_Message(&message));
            CHECK_FAILURE(eventArgs->get_DefaultText(&defaultText));

            std::wstring promptString = std::wstring(L"The page at '")
                + uri.get() + L"' says:";
            TextInputDialog dialog(
                m_appWindow->GetMainWindow(),
                L"Script Dialog",
                promptString.c_str(),
                message.get(),
                defaultText.get(),
                /* readonly */ type != WEBVIEW2_SCRIPT_DIALOG_KIND_PROMPT);
            if (dialog.confirmed)
            {
                CHECK_FAILURE(eventArgs-
>put_ResultText(dialog.input.c_str()));
                CHECK_FAILURE(eventArgs->Accept());
            }
        };

        if (m_deferScriptDialogs)
        {
            wil::com_ptr<IWebView2Deferral> deferral;
            CHECK_FAILURE(args->GetDeferral(&deferral));
            m_completeDeferredDialog = [showDialog, deferral]
            {
                showDialog();
                CHECK_FAILURE(deferral->Complete());
            };
        }
        else
        {
            showDialog();
        }

        return S_OK;
    }).Get(), &m_scriptDialogOpeningToken));
```

remove_ScriptDialogOpening

Remove an event handler previously added with add_ScriptDialogOpening.

> public HRESULT remove_ScriptDialogOpening(EventRegistrationToken token)

add_ZoomFactorChanged

Add an event handler for the ZoomFactorChanged event.

> public HRESULT add_ZoomFactorChanged(IWebView2ZoomFactorChangedEventHandler * eventHandler,EventRegistrationToken * token)

The event fires when the ZoomFactor property of the WebView changes. The event could fire because the caller modified the ZoomFactor property, or due to the user manually modifying the zoom. When it is modified by the caller via the ZoomFactor property, the internal zoom factor is updated immediately and there will be no ZoomFactorChanged event. WebView associates the last used zoom factor for each site. Therefore, it is possible for the zoom factor to change when navigating to a different page. When the zoom factor changes due to this, the ZoomFactorChanged event fires right after the DocumentStateChanged event.

```C++
    // Register a handler for the ZoomFactorChanged event.
    // This handler just announces the new level of zoom on the window's
title bar.
    CHECK_FAILURE(m_webView->add_ZoomFactorChanged(
        Callback<IWebView2ZoomFactorChangedEventHandler>(
            [this](IWebView2WebView* sender, IUnknown* args) -> HRESULT {
                double zoomFactor;
                CHECK_FAILURE(sender->get_ZoomFactor(&zoomFactor));

                std::wstring message = L"WebView2APISample (Zoom: " +
                                       std::to_wstring(int(zoomFactor *
100)) + L"%)";
                SetWindowText(m_appWindow->GetMainWindow(),
message.c_str());
                return S_OK;
            })
            .Get(),
        &m_zoomFactorChangedToken));
```

remove_ZoomFactorChanged

Remove an event handler previously added with add_ZoomFactorChanged.

> public HRESULT remove_ZoomFactorChanged(EventRegistrationToken token)

add_PermissionRequested

Add an event handler for the PermissionRequested event.

> public HRESULT add_PermissionRequested(IWebView2PermissionRequestedEventHandler * eventHandler,EventRegistrationToken * token)

Fires when content in a WebView requests permission to access some privileged resources.

```C++
    // Register a handler for the PermissionRequested event.
    // This handler prompts the user to allow or deny the request.
    CHECK_FAILURE(m_webView->add_PermissionRequested(
        Callback<IWebView2PermissionRequestedEventHandler>(
            [this](
                IWebView2WebView* sender,
                IWebView2PermissionRequestedEventArgs* args) -> HRESULT
    {
        wil::unique_cotaskmem_string uri;
        WEBVIEW2_PERMISSION_TYPE type =
WEBVIEW2_PERMISSION_TYPE_UNKNOWN_PERMISSION;
        BOOL userInitiated = FALSE;

        CHECK_FAILURE(args->get_Uri(&uri));
        CHECK_FAILURE(args->get_PermissionType(&type));
        CHECK_FAILURE(args->get_IsUserInitiated(&userInitiated));

        std::wstring message = L"Do you want to grant permission for ";
        message += NameOfPermissionType(type);
        message += L" to the website at ";
        message += uri.get();
        message += L"?\n\n";
        message += (userInitiated
            ? L"This request came from a user gesture."
            : L"This request did not come from a user gesture.");

        int response = MessageBox(nullptr, message.c_str(), L"Permission
Request",
                                   MB_YESNOCANCEL | MB_ICONWARNING);

        WEBVIEW2_PERMISSION_STATE state =
              response == IDYES ? WEBVIEW2_PERMISSION_STATE_ALLOW
            : response == IDNO  ? WEBVIEW2_PERMISSION_STATE_DENY
            :                     WEBVIEW2_PERMISSION_STATE_DEFAULT;
        CHECK_FAILURE(args->put_State(state));

        return S_OK;
    }).Get(), &m_permissionRequestedToken));
```

remove_PermissionRequested

Remove an event handler previously added with add_PermissionRequested.

> public HRESULT remove_PermissionRequested(EventRegistrationToken token)

add_ProcessFailed

Add an event handler for the ProcessFailed event.

> public HRESULT add_ProcessFailed(IWebView2ProcessFailedEventHandler * eventHandler,EventRegistrationToken * token)

Fires when a WebView process terminated unexpectedly or become unresponsive.

```C++
    // Register a handler for the ProcessFailed event.
    // This handler checks if the browser process failed, and asks the user
if
    // they want to recreate the webview.
    CHECK_FAILURE(m_webView->add_ProcessFailed(
        Callback<IWebView2ProcessFailedEventHandler>(
            [this](IWebView2WebView* sender,
                IWebView2ProcessFailedEventArgs* args) -> HRESULT
    {
        WEBVIEW2_PROCESS_FAILED_KIND failureType;
        CHECK_FAILURE(args->get_ProcessFailedKind(&failureType));
        if (failureType ==
WEBVIEW2_PROCESS_FAILED_KIND_BROWSER_PROCESS_EXITED)
        {
            int button = MessageBox(
                m_appWindow->GetMainWindow(),
                L"Browser process exited unexpectedly.  Recreate webview?",
                L"Browser process exited",
                MB_YESNO);
            if (button == IDYES)
            {
                m_appWindow->ReinitializeWebView();
            }
        }
```

```
            return S_OK;
    }).Get(), &m_processFailedToken));
```

remove_ProcessFailed

Remove an event handler previously added with add_ProcessFailed.

> public HRESULT remove_ProcessFailed(EventRegistrationToken token)

AddScriptToExecuteOnDocumentCreated

Add the provided JavaScript to a list of scripts that should be executed after the global object has been created, but before the HTML document has been parsed and before any other script included by the HTML document is executed.

> public HRESULT AddScriptToExecuteOnDocumentCreated(LPCWSTR javaScript,IWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler * handler)

The injected script will apply to all future top level document and child frame navigations until removed with RemoveScriptToExecuteOnDocumentCreated. This is applied asynchronously and you must wait for the completion handler to run before you can be sure that the script is ready to execute on future navigations.

Note that if an HTML document has sandboxing of some kind via sandbox properties or the Content-Security-Policy HTTP header this will affect the script run here. So, for example, if the 'allow-modals' keyword is not set then calls to the `alert` function will be ignored.

C++

```
// Prompt the user for some script and register it to execute whenever a new
page loads.
void ScriptComponent::AddInitializeScript()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(),
        L"Add Initialize Script",
        L"Initialization Script:",
        L"Enter the JavaScript code to run as the initialization script that
"
            L"runs before any script in the HTML document.",
    // This example script stops child frames from opening new windows.
Because
    // the initialization script runs before any script in the HTML
document, we
    // can trust the results of our checks on window.parent and window.top.
        L"if (window.parent !== window.top) {\r\n"
        L"    delete window.open;\r\n"
        L"}");
    if (dialog.confirmed)
    {
        m_webView->AddScriptToExecuteOnDocumentCreated(
            dialog.input.c_str(),

Callback<IWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler>(
                [this](HRESULT error, PCWSTR id) -> HRESULT
        {
            m_lastInitializeScriptId = id;
            MessageBox(nullptr, id, L"AddScriptToExecuteOnDocumentCreated
Id", MB_OK);
            return S_OK;
        }).Get());

    }
}
```

RemoveScriptToExecuteOnDocumentCreated

Remove the corresponding JavaScript added via AddScriptToExecuteOnDocumentCreated.

> public HRESULT RemoveScriptToExecuteOnDocumentCreated(LPCWSTR id)

ExecuteScript

Execute JavaScript code from the javascript parameter in the current top level document rendered in the WebView.

> public HRESULT ExecuteScript(LPCWSTR javaScript,IWebView2ExecuteScriptCompletedHandler * handler)

This will execute asynchronously and when complete, if a handler is provided in the ExecuteScriptCompletedHandler parameter, its Invoke method will be called with the result of evaluating the provided JavaScript. The result value is a JSON encoded string. If the result is undefined, contains a reference cycle, or otherwise cannot be encoded into JSON, the JSON null value will be returned as the string 'null'. Note that a function that has no explicit return value returns undefined. If the executed script throws an unhandled exception, then the result is also 'null'. This method is applied asynchronously. If the call is made while the webview is on one document, and a navigation occurs after the call is made but before the JavaScript is executed, then the script will not be executed and the handler will be called with E_FAIL for its errorCode parameter. ExecuteScript will work even if IsScriptEnabled is set to FALSE.

```C++
// Prompt the user for some script and then execute it.
void ScriptComponent::InjectScript()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(),
        L"Inject Script",
        L"Enter script code:",
        L"Enter the JavaScript code to run in the webview.",
        L"window.getComputedStyle(document.body).backgroundColor");
    if (dialog.confirmed)
    {
        m_webView->ExecuteScript(dialog.input.c_str(),
            Callback<IWebView2ExecuteScriptCompletedHandler>(
                [](HRESULT error, PCWSTR result) -> HRESULT
        {
            if (error != S_OK) {
                ShowFailure(error, L"ExecuteScript failed");
            }
            MessageBox(nullptr, result, L"ExecuteScript Result", MB_OK);
            return S_OK;
        }).Get());
    }
}
```

CapturePreview

Capture an image of what WebView is displaying.

> public HRESULT CapturePreview(WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT imageFormat,IStream * imageStream,IWebView2CapturePreviewCompletedHandler * handler)

Specify the format of the image with the imageFormat parameter. The resulting image binary data is written to the provided imageStream parameter. When CapturePreview finishes writing to the stream, the Invoke method on the provided handler parameter is called.

```C++
// Show the user a file selection dialog, then save a screenshot of the
WebView
```

```
// to the selected file.
void FileComponent::SaveScreenshot()
{
    OPENFILENAME openFileName = {};
    openFileName.lStructSize = sizeof(openFileName);
    openFileName.hwndOwner = nullptr;
    openFileName.hInstance = nullptr;
    WCHAR fileName[MAX_PATH] = L"WebView2_Screenshot.png";
    openFileName.lpstrFile = fileName;
    openFileName.lpstrFilter = L"PNG File\0*.png\0";
    openFileName.nMaxFile = ARRAYSIZE(fileName);
    openFileName.Flags = OFN_OVERWRITEPROMPT;

    if (GetSaveFileName(&openFileName))
    {
        wil::com_ptr<IStream> stream;
        CHECK_FAILURE(SHCreateStreamOnFileEx(
            fileName, STGM_READWRITE | STGM_CREATE, FILE_ATTRIBUTE_NORMAL,
TRUE, nullptr,
            &stream));

        HWND mainWindow = m_appWindow->GetMainWindow();

        CHECK_FAILURE(m_webView->CapturePreview(
            WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_PNG, stream.get(),
            Callback<IWebView2CapturePreviewCompletedHandler>(
                [mainWindow](HRESULT error_code) -> HRESULT {
                    CHECK_FAILURE(error_code);

                    MessageBox(mainWindow, L"Preview Captured", L"Preview
Captured", MB_OK);
                    return S_OK;
                })
                .Get()));
    }
}
```

Reload

Reload the current page.

> public HRESULT Reload()

This is similar to navigating to the URI of current top level document including all navigation events firing and respecting any entries in the HTTP cache. But, the back/forward history will not be modified.

get_Bounds

The webview bounds.

> public HRESULT get_Bounds(RECT * bounds)

Bounds are relative to the parent HWND. The app has two ways it can position a WebView:

1. Create a child HWND that is the WebView parent HWND. Position this window where the WebView should be. In this case, use (0, 0) for the WebView's Bound's top left corner (the offset).

2. Use the app's top most window as the WebView parent HWND. Set the WebView's Bound's top left corner so that the WebView is positioned correctly in the app. The Bound's values are in the host's coordinate space.

put_Bounds

Set the Bounds property.

> public HRESULT put_Bounds(RECT bounds)

C++

```cpp
// Update the bounds of the WebView window to fit available space.
void ViewComponent::ResizeWebView()
{
    RECT desiredBounds = m_webViewBounds;
    desiredBounds.bottom = LONG(
        (m_webViewBounds.bottom - m_webViewBounds.top) * m_webViewRatio +
m_webViewBounds.top);
    desiredBounds.right = LONG(
        (m_webViewBounds.right - m_webViewBounds.left) * m_webViewRatio +
m_webViewBounds.left);

    m_webView->put_Bounds(desiredBounds);
}
```

get_ZoomFactor

The zoom factor for the current page in the WebView.

> public HRESULT get_ZoomFactor(double * zoomFactor)

The zoom factor is persisted per site. Note that changing zoom factor could cause `window.innerWidth/innerHeight` and page layout to change. When WebView navigates to a page from a different site, the zoom factor set for the previous page will not be applied. If the app wants to set the zoom factor for a certain page, the earliest place to do it is in the DocumentStateChanged event handler. Note that if it does that, it might receive a ZoomFactorChanged event for the persisted zoom factor before receiving the ZoomFactorChanged event for the specified zoom factor. Specifying a zoomFactor less than or equal to 0 is not allowed. WebView also has an internal supported zoom factor range. When a specified zoom factor is out of that range, it will be normalized to be within the range, and a ZoomFactorChanged event will be fired for the real applied zoom factor. When this range normalization happens, the ZoomFactor property will report the zoom factor specified during the previous modification of the ZoomFactor property until the ZoomFactorChanged event is received after webview applies the normalized zoom factor.

put_ZoomFactor

Set the ZoomFactor property.

> public HRESULT put_ZoomFactor(double zoomFactor)

get_IsVisible

The IsVisible property determines whether to show or hide the webview.

> public HRESULT get_IsVisible(BOOL * isVisible)

If IsVisible is set to false, the webview will be transparent and will not be rendered. However, this will not affect the window containing the webview (the HWND parameter that was passed to CreateWebView). If you want that window to disappear too, call ShowWindow on it directly in addition to modifying the IsVisible property. WebView as a child window won't get window messages when the top window is minimized or restored. For performance reason, developer should set IsVisible property of the WebView to false when the app window is minimized and back to true when app window is restored. App window can do this by handling SC_MINIMIZE and SC_RESTORE command upon receiving WM_SYSCOMMAND message.

```C++
void ViewComponent::ToggleVisibility()
{
    BOOL visible;
```

```
        m_webView->get_IsVisible(&visible);
        m_isVisible = !visible;
        m_webView->put_IsVisible(m_isVisible);
    }
```

put_IsVisible

Set the IsVisible property.

> public HRESULT put_IsVisible(BOOL isVisible)

C++

```cpp
    if (message == WM_SYSCOMMAND)
    {
        if (wParam == SC_MINIMIZE)
        {
            // Hide the webview when the app window is minimized.
            m_webView->put_IsVisible(FALSE);
        }
        else if (wParam == SC_RESTORE)
        {
            // When the app window is restored, show the webview
            // (unless the user has toggle visibility off).
            if (m_isVisible)
            {
                m_webView->put_IsVisible(TRUE);
            }
        }
    }
```

PostWebMessageAsJson

Post the specified webMessage to the top level document in this IWebView2WebView.

> public HRESULT PostWebMessageAsJson(LPCWSTR webMessageAsJson)

The top level document's window.chrome.webview's message event fires. JavaScript in that document may subscribe and unsubscribe to the event via the following:

C++

```cpp
window.chrome.webview.addEventListener('message', handler)
window.chrome.webview.removeEventListener('message', handler)
```

The event args is an instance of `MessageEvent`. The IWebView2Settings::IsWebMessageEnabled setting must be true or this method will fail with E_INVALIDARG. The event arg's data property is the webMessage string parameter parsed as a JSON string into a JavaScript object. The event arg's source property is a reference to the `window.chrome.webview` object. See SetWebMessageReceivedEventHandler for information on sending messages from the HTML document in the webview to the host. This message is sent asynchronously. If a navigation occurs before the message is posted to the page, then the message will not be sent.

```C++
    // Setup the web message received event handler before navigating to
    // ensure we don't miss any messages.
    CHECK_FAILURE(m_webView->add_WebMessageReceived(
        Microsoft::WRL::Callback<IWebView2WebMessageReceivedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2WebMessageReceivedEventArgs* args)
    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Source(&uri));

        // Always validate that the origin of the message is what you
expect.
        if (uri.get() != m_sampleUri)
        {
            return S_OK;
        }
        wil::unique_cotaskmem_string messageRaw;
        CHECK_FAILURE(args->get_WebMessageAsString(&messageRaw));
        std::wstring message = messageRaw.get();

        if (message.compare(0, 13, L"SetTitleText ") == 0)
        {
            m_appWindow->SetTitleText(message.substr(13).c_str());
        }
        else if (message.compare(L"GetWindowBounds") == 0)
        {
            RECT bounds = m_appWindow->GetWindowBounds();
            std::wstring reply =
                L"{\"WindowBounds\":\"Left:" + std::to_wstring(bounds.left)
                + L"\\nTop:" + std::to_wstring(bounds.top)
                + L"\\nRight:" + std::to_wstring(bounds.right)
                + L"\\nBottom:" + std::to_wstring(bounds.bottom)
                + L"\"}";
            CHECK_FAILURE(sender->PostWebMessageAsJson(reply.c_str()));
        }
        return S_OK;
    }).Get(), &m_webMessageReceivedToken));
```

PostWebMessageAsString

This is a helper for posting a message that is a simple string rather than a JSON string representation of a JavaScript object.

> public HRESULT PostWebMessageAsString(LPCWSTR webMessageAsString)

This behaves in exactly the same manner as PostWebMessageAsJson but the `window.chrome.webview` message event arg's data property will be a string with the same value as webMessageAsString. Use this instead of PostWebMessageAsJson if you want to communicate via simple strings rather than JSON objects.

add_WebMessageReceived

This event fires when the IsWebMessageEnabled setting is set and the top level document of the webview calls `window.chrome.webview.postMessage`.

> public HRESULT add_WebMessageReceived(IWebView2WebMessageReceivedEventHandler * handler,EventRegistrationToken * token)

The postMessage function is `void postMessage(object)` where object is any object supported by JSON conversion.

C++

```
        window.chrome.webview.addEventListener('message', arg => {
            if ("SetColor" in arg.data) {
                document.getElementById("colorable").style.color =
arg.data.SetColor;
            }
            if ("WindowBounds" in arg.data) {
                document.getElementById("window-bounds").value =
arg.data.WindowBounds;
            }
        });

        function SetTitleText() {
            let titleText = document.getElementById("title-text");
            window.chrome.webview.postMessage(`SetTitleText
${titleText.value}`);
        }
        function GetWindowBounds() {
            window.chrome.webview.postMessage("GetWindowBounds");
        }
```

When postMessage is called, the IWebView2WebMessageReceivedEventHandler set via this SetWebMessageReceivedEventHandler method will be invoked with the postMessage's object parameter converted to a JSON string.

```C++
    // Setup the web message received event handler before navigating to
    // ensure we don't miss any messages.
    CHECK_FAILURE(m_webView->add_WebMessageReceived(
        Microsoft::WRL::Callback<IWebView2WebMessageReceivedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2WebMessageReceivedEventArgs* args)
    {
        wil::unique_cotaskmem_string uri;
        CHECK_FAILURE(args->get_Source(&uri));

        // Always validate that the origin of the message is what you
expect.
        if (uri.get() != m_sampleUri)
        {
            return S_OK;
        }
        wil::unique_cotaskmem_string messageRaw;
        CHECK_FAILURE(args->get_WebMessageAsString(&messageRaw));
        std::wstring message = messageRaw.get();

        if (message.compare(0, 13, L"SetTitleText ") == 0)
        {
            m_appWindow->SetTitleText(message.substr(13).c_str());
        }
        else if (message.compare(L"GetWindowBounds") == 0)
        {
            RECT bounds = m_appWindow->GetWindowBounds();
            std::wstring reply =
                L"{\"WindowBounds\":\"Left:" + std::to_wstring(bounds.left)
                + L"\\nTop:" + std::to_wstring(bounds.top)
                + L"\\nRight:" + std::to_wstring(bounds.right)
                + L"\\nBottom:" + std::to_wstring(bounds.bottom)
                + L"\"}";
            CHECK_FAILURE(sender->PostWebMessageAsJson(reply.c_str()));
        }
        return S_OK;
    }).Get(), &m_webMessageReceivedToken));
```

remove_WebMessageReceived

Remove an event handler previously added with add_WebMessageReceived.

> public HRESULT remove_WebMessageReceived(EventRegistrationToken token)

Close

Closes the webview and cleans up the underlying browser instance.

> public HRESULT Close()

Cleaning up the browser instace will release the resources powering the webview. The browser instance will be shut down if there are no other webviews using it.

After calling Close, all method calls will fail and event handlers will stop firing. Specifically, the WebView will release its references to its event handlers when Close is called.

Close is implicitly called when the WebView loses its final reference and is destructed. But it is best practice to explicitly call Close to avoid any accidental cycle of references between the WebView and the app code. Specifically, if you capture a reference to the WebView in an event handler you will create a reference cycle between the WebView and the event handler. Close will break this cycle by releasing all event handlers. But to avoid this situation it is best practice to both explicitly call Close on the WebView and to not capture a reference to the WebView to ensure the WebView can be cleaned up correctly.

C++

```cpp
// Close the WebView and deinitialize related state. This doesn't close the
app window.
void AppWindow::CloseWebView()
{
    DeleteAllComponents();
    if (m_webView)
    {
        m_webView->Close();
        m_webView = nullptr;
    }
    m_webViewEnvironment = nullptr;
}
```

CallDevToolsProtocolMethod

Call an asynchronous DevToolsProtocol method.

> public HRESULT CallDevToolsProtocolMethod(LPCWSTR methodName,LPCWSTR parametersAsJson,IWebView2CallDevToolsProtocolMethodCompletedHandler * handler)

See the DevTools Protocol Viewer for a list and description of available methods. The methodName parameter is the full name of the method in the format `{domain}.{method}`. The parametersAsJson parameter is a JSON formatted string containing the parameters for the corresponding method. The handler's Invoke method will be called when the method asynchronously completes. Invoke will be called with the method's return object as a JSON string.

```C++
// Prompt the user for the name and parameters of a CDP method, then call it.
void ScriptComponent::CallCdpMethod()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(),
        L"Call CDP Method",
        L"CDP method name:",
        L"Enter the CDP method name to call, followed by a space,\r\n"
            L"followed by the parameters in JSON format.",
        L"Runtime.evaluate {\"expression\":\"alert(\\\"test\\\")\"}");
    if (dialog.confirmed)
    {
        size_t delimiterPos = dialog.input.find(L' ');
        std::wstring methodName = dialog.input.substr(0, delimiterPos);
        std::wstring methodParams =
            (delimiterPos < dialog.input.size()
                ? dialog.input.substr(delimiterPos + 1)
                : L"{}");

        m_webView->CallDevToolsProtocolMethod(
            methodName.c_str(),
            methodParams.c_str(),
            Callback<IWebView2CallDevToolsProtocolMethodCompletedHandler>(
                [](HRESULT error, PCWSTR resultJson) -> HRESULT
                {
                    MessageBox(nullptr, resultJson, L"CDP Method Result",
MB_OK);
                    return S_OK;
                }).Get());
    }
}
```

add_DevToolsProtocolEventReceived

Subscribe to a DevToolsProtocol event.

> public HRESULT add_DevToolsProtocolEventReceived(LPCWSTR eventName,IWebView2DevToolsProtocolEventReceivedEventHandler * handler,EventRegistrationToken * token)

See the DevTools Protocol Viewer for a list and description of available events. The eventName parameter is the full name of the event in the format `{domain}.{event}`. The handler's Invoke method will be called whenever the corresponding DevToolsProtocol event fires. Invoke will be called with the an event args object containing the CDP event's parameter object as a JSON string.

```C++
// Prompt the user to name a CDP event, and then subscribe to that event.
void ScriptComponent::SubscribeToCdpEvent()
{
    TextInputDialog dialog(
        m_appWindow->GetMainWindow(),
        L"Subscribe to CDP Event",
        L"CDP event name:",
        L"Enter the name of the CDP event to subscribe to.\r\n"
            L"You may also have to call the \"enable\" method of the\r\n"
            L"event's domain to receive events (for example
\"Log.enable\").\r\n",
        L"Log.entryAdded");
    if (dialog.confirmed)
    {
        std::wstring eventName = dialog.input;
        // If we are already subscribed to this event, unsubscribe first.
        auto preexistingToken =
m_devToolsProtocolEventReceivedTokenMap.find(eventName);
        if (preexistingToken !=
m_devToolsProtocolEventReceivedTokenMap.end())
        {
            CHECK_FAILURE(m_webView->remove_DevToolsProtocolEventReceived(
                eventName.c_str(),
                preexistingToken->second));
        }

        CHECK_FAILURE(m_webView->add_DevToolsProtocolEventReceived(
            eventName.c_str(),
            Callback<IWebView2DevToolsProtocolEventReceivedEventHandler>(
                [eventName](IWebView2WebView* sender,
                            IWebView2DevToolsProtocolEventReceivedEventArgs*
args)
                -> HRESULT
        {
            wil::unique_cotaskmem_string parameterObjectAsJson;
            CHECK_FAILURE(args-
>get_ParameterObjectAsJson(&parameterObjectAsJson));
            MessageBox(nullptr, parameterObjectAsJson.get(),
                       (L"CDP Event Fired: " + eventName).c_str(), MB_OK);
            return S_OK;
        }).Get(), &m_devToolsProtocolEventReceivedTokenMap[eventName]));
    }
}
```

remove_DevToolsProtocolEventReceived

Remove an event handler previously added with add_DevToolsProtocolEventReceived.

> public HRESULT remove_DevToolsProtocolEventReceived(LPCWSTR eventName,EventRegistrationToken token)

get_BrowserProcessId

The process id of the browser process that hosts the WebView.

> public HRESULT get_BrowserProcessId(UINT32 * value)

get_CanGoBack

Can navigate the webview to the previous page in the navigation history.

> public HRESULT get_CanGoBack(BOOL * canGoBack)

get_CanGoBack change value with the DocumentStateChanged event.

get_CanGoForward

Can navigate the webview to the next page in the navigation history.

> public HRESULT get_CanGoForward(BOOL * canGoForward)

get_CanGoForward change value with the DocumentStateChanged event.

GoBack

Navigates the webview to the previous page in the navigation history.

> public HRESULT GoBack()

GoForward

Navigates the webview to the next page in the navigation history.

> public HRESULT GoForward()

WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT

Image format used by the IWebView2WebView::CapturePreview method.

> enum WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT

Values	Descriptions
WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_PNG	PNG image format.
WEBVIEW2_CAPTURE_PREVIEW_IMAGE_FORMAT_JPEG	JPEG image format.

WEBVIEW2_SCRIPT_DIALOG_KIND

Kind of JavaScript dialog used in the IWebView2ScriptDialogOpeningEventHandler interface.

> enum WEBVIEW2_SCRIPT_DIALOG_KIND

Values	Descriptions
WEBVIEW2_SCRIPT_DIALOG_KIND_ALERT	A dialog invoked via the window.alert JavaScript function.
WEBVIEW2_SCRIPT_DIALOG_KIND_CONFIRM	A dialog invoked via the window.confirm JavaScript function.
WEBVIEW2_SCRIPT_DIALOG_KIND_PROMPT	A dialog invoked via the window.prompt JavaScript function.

WEBVIEW2_PROCESS_FAILED_KIND

Kind of process failure used in the IWebView2ProcessFailedEventHandler interface.

> enum WEBVIEW2_PROCESS_FAILED_KIND

Values	Descriptions
WEBVIEW2_PROCESS_FAILED_KIND_BROWSER_PROCESS_EXITED	Indicates the browser process terminated unexpectedly.

Values	Descriptions
WEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_EXITED	Indicates the render process terminated unexpectedly.
WEBVIEW2_PROCESS_FAILED_KIND_RENDER_PROCESS_UNRESPONSIVE	Indicates the render process becomes unresponsive.

WEBVIEW2_PERMISSION_TYPE

The type of a permission request.

> enum WEBVIEW2_PERMISSION_TYPE

Values	Descriptions
WEBVIEW2_PERMISSION_TYPE_UNKNOWN_PERMISSION	Unknown permission.
WEBVIEW2_PERMISSION_TYPE_MICROPHONE	Permission to capture audio.
WEBVIEW2_PERMISSION_TYPE_CAMERA	Permission to capture video.
WEBVIEW2_PERMISSION_TYPE_GEOLOCATION	Permission to access geolocation.
WEBVIEW2_PERMISSION_TYPE_NOTIFICATIONS	Permission to send web notifications.
WEBVIEW2_PERMISSION_TYPE_OTHER_SENSORS	Permission to access generic sensor.
WEBVIEW2_PERMISSION_TYPE_CLIPBOARD_READ	Permission to read system clipboard without a user gesture.

WEBVIEW2_PERMISSION_STATE

Response to a permission request.

> enum WEBVIEW2_PERMISSION_STATE

Values	Descriptions
WEBVIEW2_PERMISSION_STATE_DEFAULT	Use default browser behavior, which normally prompt users for decision.
WEBVIEW2_PERMISSION_STATE_ALLOW	Grant the permission request.

Values	Descriptions
WEBVIEW2_PERMISSION_STATE_DENY	Deny the permission request.

WEBVIEW2_MOVE_FOCUS_REASON

Reason for moving focus.

> enum WEBVIEW2_MOVE_FOCUS_REASON

Values	Descriptions
WEBVIEW2_MOVE_FOCUS_REASON_PROGRAMMATIC	Code setting focus into WebView.
WEBVIEW2_MOVE_FOCUS_REASON_NEXT	Moving focus due to Tab traversal forward.
WEBVIEW2_MOVE_FOCUS_REASON_PREVIOUS	Moving focus due to Tab traversal backward.

WEBVIEW2_WEB_ERROR_STATUS

Error status values for web navigations.

> enum WEBVIEW2_WEB_ERROR_STATUS

Values	Descriptions
WEBVIEW2_WEB_ERROR_STATUS_UNKNOWN	An unknown error occurred.
WEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_COMMON_NAME_IS_INCORRECT	The SSL certificate common name does not match the web address.
WEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_EXPIRED	The SSL certificate has expired.

Values	Descriptions
WEBVIEW2_WEB_ERROR_STATUS_CLIENT_CERTIFICATE_CONTAINS_ERRORS	The SSL client certificate contains errors.
WEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_REVOKED	The SSL certificate has been revoked.
WEBVIEW2_WEB_ERROR_STATUS_CERTIFICATE_IS_INVALID	The SSL certificate is invalid.
WEBVIEW2_WEB_ERROR_STATUS_SERVER_UNREACHABLE	The host is unreachable.
WEBVIEW2_WEB_ERROR_STATUS_TIMEOUT	The connection has timed out.
WEBVIEW2_WEB_ERROR_STATUS_ERROR_HTTP_INVALID_SERVER_RESPONSE	The server returned an invalid or unrecognized response.
WEBVIEW2_WEB_ERROR_STATUS_CONNECTION_ABORTED	The connection was aborted.
WEBVIEW2_WEB_ERROR_STATUS_CONNECTION_RESET	The connection was reset.
WEBVIEW2_WEB_ERROR_STATUS_DISCONNECTED	The Internet connection has been lost.
WEBVIEW2_WEB_ERROR_STATUS_CANNOT_CONNECT	Cannot connect to destination.
WEBVIEW2_WEB_ERROR_STATUS_HOST_NAME_NOT_RESOLVED	Could not resolve provided host name.
WEBVIEW2_WEB_ERROR_STATUS_OPERATION_CANCELED	The operation was canceled.

Values	Descriptions
WEBVIEW2_WEB_ERROR_STATUS_REDIRECT_FAILED	The request redirect failed.
WEBVIEW2_WEB_ERROR_STATUS_UNEXPECTED_ERROR	An unexpected error occurred.

WEBVIEW2_WEB_RESOURCE_CONTEXT

Enum for web resource request contexts.

> enum WEBVIEW2_WEB_RESOURCE_CONTEXT

Values	Descriptions
WEBVIEW2_WEB_RESOURCE_CONTEXT_ALL	All resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_DOCUMENT	Document resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_STYLESHEET	CSS resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE	Image resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_MEDIA	Other media resources such as videos.
WEBVIEW2_WEB_RESOURCE_CONTEXT_FONT	Font resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_SCRIPT	Script resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_XML_HTTP_REQUEST	XML HTTP requests.
WEBVIEW2_WEB_RESOURCE_CONTEXT_FETCH	Fetch API communication.
WEBVIEW2_WEB_RESOURCE_CONTEXT_TEXT_TRACK	TextTrack resources.
WEBVIEW2_WEB_RESOURCE_CONTEXT_EVENT_SOURCE	
WEBVIEW2_WEB_RESOURCE_CONTEXT_WEBSOCKET	
WEBVIEW2_WEB_RESOURCE_CONTEXT_MANIFEST	
WEBVIEW2_WEB_RESOURCE_CONTEXT_SIGNED_EXCHANGE	
WEBVIEW2_WEB_RESOURCE_CONTEXT_PING	
WEBVIEW2_WEB_RESOURCE_CONTEXT_CSP_VIOLATION_REPORT	
WEBVIEW2_WEB_RESOURCE_CONTEXT_OTHER	Other resources.

interface IWebView2WebView2

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2WebView2
  : public IUnknown
```

Additional functionality implemented by the primary WebView object.

Summary

Members	Descriptions
Stop	Stop all navigations and pending resource fetches.

You can QueryInterface for this interface from the object that implements IWebView2WebView. See the IWebView2WebView interface for more details.

Members

Stop

Stop all navigations and pending resource fetches.

> public HRESULT Stop()

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebView3

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebView3
  : public IWebView2WebView
```

Additional functionality implemented by the primary WebView object.

Summary

Members	Descriptions
Stop	Stop all navigations and pending resource fetches.
add_NewWindowRequested	Add an event handler for the NewWindowRequested event.
remove_NewWindowRequested	Remove an event handler previously added with add_NewWindowRequested.
add_DocumentTitleChanged	Add an event handler for the DocumentTitleChanged event.
remove_DocumentTitleChanged	Remove an event handler previously added with add_DocumentTitleChanged.
get_DocumentTitle	The title for the current top level document.

You can QueryInterface for this interface from the object that implements IWebView2WebView. See the IWebView2WebView interface for more details.

Members

Stop

Stop all navigations and pending resource fetches.

> public HRESULT Stop()

add_NewWindowRequested

Add an event handler for the NewWindowRequested event.

> public HRESULT add_NewWindowRequested(IWebView2NewWindowRequestedEventHandler * eventHandler,EventRegistrationToken * token)

Fires when content inside the WebView requested to open a new window, such as through window.open. The app can pass a target webview that will be considered the opened window.

C++

```cpp
    // Register a handler for the NewWindowRequested event.
    // This handler will defer the event, create a new app window, and then
once the
    // new window is ready, it'll provide that new window's WebView as the
response to
    // the request.
    CHECK_FAILURE(m_webView->add_NewWindowRequested(
        Callback<IWebView2NewWindowRequestedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2NewWindowRequestedEventArgs* args) {
                wil::com_ptr<IWebView2Deferral> deferral;
                CHECK_FAILURE(args->GetDeferral(&deferral));

                auto newAppWindow = new AppWindow(L"");
                newAppWindow->m_onWebViewFirstInitialized = [args, deferral,
newAppWindow]() {
                    CHECK_FAILURE(args->put_NewWindow(newAppWindow-
>m_webView.get()));
                    CHECK_FAILURE(args->put_Handled(TRUE));
                    CHECK_FAILURE(deferral->Complete());
                };

                return S_OK;
            })
            .Get(),
        nullptr));
```

remove_NewWindowRequested

Remove an event handler previously added with add_NewWindowRequested.

> public HRESULT remove_NewWindowRequested(EventRegistrationToken token)

add_DocumentTitleChanged

Add an event handler for the DocumentTitleChanged event.

> public HRESULT add_DocumentTitleChanged(IWebView2DocumentTitleChangedEventHandler * eventHandler,EventRegistrationToken * token)

The event fires when the DocumentTitle property of the WebView changes and may fire before or after the NavigationCompleted event.

```C++
    // Register a handler for the DocumentTitleChanged event.
    // This handler just announces the new title on the window's title bar.
    CHECK_FAILURE(m_webView->add_DocumentTitleChanged(
        Callback<IWebView2DocumentTitleChangedEventHandler>(
            [this](IWebView2WebView3* sender, IUnknown* args) -> HRESULT {
                wil::unique_cotaskmem_string title;
                CHECK_FAILURE(sender->get_DocumentTitle(&title));
                SetWindowText(m_appWindow->GetMainWindow(), title.get());
                return S_OK;
            })
            .Get(),
        &m_documentTitleChangedToken));
```

remove_DocumentTitleChanged

Remove an event handler previously added with add_DocumentTitleChanged.

> public HRESULT remove_DocumentTitleChanged(EventRegistrationToken token)

get_DocumentTitle

The title for the current top level document.

> public HRESULT get_DocumentTitle(LPWSTR * title)

If the document has no explicit title or is otherwise empty, a default that may or may not match the URI of the document will be used.

interface IWebView2WebView4

Article • 06/03/2021

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2WebView4
  : public IWebView2WebView3
```

Additional functionality implemented by the primary WebView object.

Summary

Members	**Descriptions**
AddRemoteObject	Add the provided host object to script running in the WebView with the specified name.
RemoveRemoteObject	Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the WebView.
OpenDevToolsWindow	Opens the DevTools window for the current document in the WebView.
add_AcceleratorKeyPressed	Add an event handler for the AcceleratorKeyPressed event.
remove_AcceleratorKeyPressed	Remove an event handler previously added with add_AcceleratorKeyPressed.
WEBVIEW2_KEY_EVENT_TYPE	The type of key event that triggered an AcceleratorKeyPressed event.
WEBVIEW2_PHYSICAL_KEY_STATUS	A structure representing the information packed into the LPARAM given to a Win32 key event.

You can QueryInterface for this interface from the object that implements IWebView2WebView. See the IWebView2WebView interface for more details.

Members

AddRemoteObject

Add the provided host object to script running in the WebView with the specified name.

> public HRESULT AddRemoteObject(LPCWSTR name,VARIANT * object)

Host objects are exposed as remote object proxies via `window.chrome.webview.remoteObjects.<name>`. Remote object proxies are promises and will resolve to an object representing the host object. The promise is rejected if the app has not added an object with the name. When JavaScript code access a property or method of the object, a promise is return, which will resolve to the value returned from the host for the property or method, or rejected in case of error such as there is no such property or method on the object or parameters are invalid. For example, when the application code does the following:

```C++
VARIANT object;
object.vt = VT_DISPATCH;
object.pdispVal = appObject;
webview->AddRemoteObject(L"host_object", &host);
```

JavaScript code in the WebView will be able to access appObject as following and then access attributes and methods of appObject:

```JavaScript
let app_object = await window.chrome.webview.remoteObjects.host_object;
let attr1 = await app_object.attr1;
let result = await app_object.method1(parameters);
```

Note that while simple types, IDispatch and array are supported, generic IUnknown, VT_DECIMAL, or VT_RECORD variant is not supported. Remote JavaScript objects like callback functions are represented as an VT_DISPATCH VARIANT with the object implementing IDispatch. The JavaScript callback method may be invoked using DISPID_VALUE for the DISPID. Nested arrays are supported up to a depth of 3. Arrays of by reference types are not supported. VT_EMPTY and VT_NULL are mapped into JavaScript as null. In JavaScript null and undefined are mapped to VT_EMPTY.

Additionally, all remote objects are exposed as `window.chrome.webview.remoteObjects.sync.<name>`. Here the host objects are exposed as synchronous remote object proxies. These are not promises and calls to functions or property access synchronously block running script waiting to communicate cross process for the host code to run. Accordingly this can result in reliability issues and it is recommended that you use the promise based asynchronous `window.chrome.webview.remoteObjects.<name>` API described above.

Synchronous remote object proxies and asynchronous remote object proxies can both proxy the same remote object. Remote changes made by one proxy will be reflected in any other proxy of that same remote object whether the other proxies and synchronous or asynchronous.

While JavaScript is blocked on a synchronous call to native code, that native code is unable to call back to JavaScript. Attempts to do so will fail with HRESULT_FROM_WIN32(ERROR_POSSIBLE_DEADLOCK).

Remote object proxies are JavaScript Proxy objects that intercept all property get, property set, and method invocations. Properties or methods that are a part of the Function or Object prototype are run locally. Additionally any property or method in the array `chrome.webview.remoteObjects.options.forceLocalProperties` will also be run locally. This defaults to including optional methods that have meaning in JavaScript like `toJSON` and `Symbol.toPrimitive`. You can add more to this array as required.

There's a method `chrome.webview.remoteObjects.cleanupSome` that will best effort garbage collect remote object proxies.

Remote object proxies additionally have the following methods which run locally:

- applyRemote, getRemote, setRemote: Perform a method invocation, property get, or property set on the remote object. You can use these to explicitly force a method or property to run remotely if there is a conflicting local method or property. For instance, `proxy.toString()` will run the local toString method on the proxy object. But `proxy.applyRemote('toString')` runs `toString` on the remote proxied object instead.

- getLocal, setLocal: Perform property get, or property set locally. You can use these methods to force getting or setting a property on the remote object proxy itself rather than on the remote object it represents. For instance, `proxy.unknownProperty` will get the property named `unknownProperty` from the remote proxied object. But `proxy.getLocal('unknownProperty')` will get the value of the property `unknownProperty` on the proxy object itself.

- sync: Asynchronous remote object proxies expose a sync method which returns a promise for a synchronous remote object proxy for the same remote object. For example, `chrome.webview.remoteObjects.sample.methodCall()` returns an asynchronous remote object proxy. You can use the `sync` method to obtain a synchronous remote object proxy instead: `const syncProxy = await chrome.webview.remoteObjects.sample.methodCall().sync()`

- async: Synchronous remote object proxies expose an async method which blocks and returns an asynchronous remote object proxy for the same remote object. For example, `chrome.webview.remoteObjects.sync.sample.methodCall()` returns a synchronous remote object proxy. Calling the `async` method on this blocks and then returns an asynchronous remote object proxy for the same remote object: `const asyncProxy = chrome.webview.remoteObjects.sync.sample.methodCall().async()`
- then: Asynchronous remote object proxies have a then method. This allows them to be awaitable. `then` will return a promise that resolves with a representation of the remote object. If the proxy represents a JavaScript literal then a copy of that is returned locally. If the proxy represents a function then a non-awaitable proxy is returned. If the proxy represents a JavaScript object with a mix of literal properties and function properties, then the a copy of the object is returned with some properties as remote object proxies.

All other property and method invocations (other than the above Remote object proxy methods, forceLocalProperties list, and properties on Function and Object prototypes) are run remotely. Asynchronous remote object proxies return a promise representing asynchronous completion of remotely invoking the method, or getting the property. The promise resolves after the remote operations complete and the promises resolve to the resulting value of the operation. Synchronous remote object proxies work similarly but block JavaScript execution and wait for the remote operation to complete.

Setting a property on an asynchronous remote object proxy works slightly differently. The set returns immediately and the return value is the value that will be set. This is a requirement of the JavaScript Proxy object. If you need to asynchronously wait for the property set to complete, use the setRemote method which returns a promise as described above. Synchronous object property set property synchronously blocks until the property is set.

For example, suppose you have a COM object with the following interface

idl

```
    [uuid(3a14c9c0-bc3e-453f-a314-4ce4a0ec81d8), object, local]
    interface IRemoteObjectSample : IUnknown
    {
        // Demonstrate basic method call with some parameters and a return
value.
        HRESULT MethodWithParametersAndReturnValue([in] BSTR
stringParameter, [in] INT integerParameter, [out, retval] BSTR*
stringResult);

        // Demonstrate getting and setting a property.
        [propget] HRESULT Property([out, retval] BSTR* stringResult);
        [propput] HRESULT Property([in] BSTR stringValue);

        // Demonstrate native calling back into JavaScript.
        HRESULT CallCallbackAsynchronously([in] IDispatch*
callbackParameter);
    };
```

We can add an instance of this interface into our JavaScript with `AddRemoteObject`. In this case we name it `sample`:

```C++
            VARIANT remoteObjectAsVariant = {};
            m_remoteObject.query_to<IDispatch>
(&remoteObjectAsVariant.pdispVal);
            remoteObjectAsVariant.vt = VT_DISPATCH;

            // We can call AddRemoteObject multiple times in a row without
            // calling RemoveRemoteObject first. This will replace the
previous object
            // with the new object. In our case this is the same object and
everything
            // is fine.
            CHECK_FAILURE(m_webView->AddRemoteObject(L"sample",
&remoteObjectAsVariant));
            remoteObjectAsVariant.pdispVal->Release();
```

Then in the HTML document we can use this COM object via `chrome.webview.remoteObjects.sample`:

```JavaScript

document.getElementById("getPropertyAsyncButton").addEventListener("click",
async () => {
            const propertyValue = await
chrome.webview.remoteObjects.sample.property;
            document.getElementById("getPropertyAsyncOutput").textContent =
propertyValue;
        });


document.getElementById("getPropertySyncButton").addEventListener("click",
() => {
            const propertyValue =
chrome.webview.remoteObjects.sync.sample.property;
```

```
            document.getElementById("getPropertySyncOutput").textContent =
propertyValue;
        });


document.getElementById("setPropertyAsyncButton").addEventListener("click",
async () => {
            const propertyValue =
document.getElementById("setPropertyAsyncInput").value;
            // The following line will work but it will return immediately
before the property value has actually been set.
            // If you need to set the property and wait for the property to
change value, use the setRemote function.
            chrome.webview.remoteObjects.sample.property = propertyValue;
            document.getElementById("setPropertyAsyncOutput").textContent =
"Set";
        });


document.getElementById("setPropertyExplicitAsyncButton").addEventListener("
click", async () => {
            const propertyValue =
document.getElementById("setPropertyExplicitAsyncInput").value;
            // If you care about waiting until the property has actually
changed value use the setRemote function.
            await chrome.webview.remoteObjects.sample.setRemote("property",
propertyValue);

document.getElementById("setPropertyExplicitAsyncOutput").textContent =
"Set";
        });


document.getElementById("setPropertySyncButton").addEventListener("click",
() => {
            const propertyValue =
document.getElementById("setPropertySyncInput").value;
            chrome.webview.remoteObjects.sync.sample.property =
propertyValue;
            document.getElementById("setPropertySyncOutput").textContent =
"Set";
        });


document.getElementById("invokeMethodAsyncButton").addEventListener("click",
async () => {
            const paramValue1 =
document.getElementById("invokeMethodAsyncParam1").value;
            const paramValue2 =
parseInt(document.getElementById("invokeMethodAsyncParam2").value);
            const resultValue = await
chrome.webview.remoteObjects.sample.MethodWithParametersAndReturnValue(param
Value1, paramValue2);
            document.getElementById("invokeMethodAsyncOutput").textContent =
resultValue;
```

```
        });


document.getElementById("invokeMethodSyncButton").addEventListener("click",
() => {
            const paramValue1 =
document.getElementById("invokeMethodSyncParam1").value;
            const paramValue2 =
parseInt(document.getElementById("invokeMethodSyncParam2").value);
            const resultValue =
chrome.webview.remoteObjects.sync.sample.MethodWithParametersAndReturnValue(
paramValue1, paramValue2);
            document.getElementById("invokeMethodSyncOutput").textContent =
resultValue;
        });

        let callbackCount = 0;

document.getElementById("invokeCallbackButton").addEventListener("click",
async () => {

chrome.webview.remoteObjects.sample.CallCallbackAsynchronously(() => {
                document.getElementById("invokeCallbackOutput").textContent
= "Native object called the callback " + (++callbackCount) + " time(s).";
            });
        });
```

RemoveRemoteObject

Remove the host object specified by the name so that it is no longer accessible from JavaScript code in the WebView.

> public HRESULT RemoveRemoteObject(LPCWSTR name)

While new access attempts will be denied, if the object is already obtained by JavaScript code in the WebView, the JavaScript code will continue to have access to that object. Calling this method for a name that is already removed or never added will fail.

OpenDevToolsWindow

Opens the DevTools window for the current document in the WebView.

> public HRESULT OpenDevToolsWindow()

Does nothing if called when the DevTools window is already open

add_AcceleratorKeyPressed

Add an event handler for the AcceleratorKeyPressed event.

> public HRESULT add_AcceleratorKeyPressed(IWebView2AcceleratorKeyPressedEventHandler * eventHandler,EventRegistrationToken * token)

AcceleratorKeyPressed fires when an accelerator key or key combo is pressed or released while the WebView is focused. A key is considered an accelerator if either:

1. Ctrl or Alt is currently being held, or
2. the pressed key does not map to a character. A few specific keys are never considered accelerators, such as Shift. The Escape key is always considered an accelerator.

Autorepeated key events caused by holding the key down will also fire this event. You can filter these out by checking the event args' KeyEventLParam or PhysicalKeyStatus.

In windowed mode, this event handler is called synchronously. Until you call Handle() on the event args or the event handler returns, the browser process will be blocked and outgoing cross-process COM calls will fail with RPC_E_CANTCALLOUT_ININPUTSYNCCALL. All WebView2 API methods will work, however.

In windowless mode, the event handler is called asynchronously. Further input will not reach the browser until the event handler returns or Handle() is called, but the browser process itself will not be blocked, and outgoing COM calls will work normally.

It is recommended to call Handle(TRUE) as early as you can know that you want to handle the accelerator key.

```C++
    // Register a handler for the AcceleratorKeyPressed event.
    CHECK_FAILURE(m_webView->add_AcceleratorKeyPressed(
        Callback<IWebView2AcceleratorKeyPressedEventHandler>(
            [this](IWebView2WebView* sender,
IWebView2AcceleratorKeyPressedEventArgs* args)
                -> HRESULT {
                WEBVIEW2_KEY_EVENT_TYPE type;
                CHECK_FAILURE(args->get_KeyEventType(&type));
                // We only care about key down events.
                if (type == WEBVIEW2_KEY_EVENT_TYPE_KEY_DOWN ||
                    type == WEBVIEW2_KEY_EVENT_TYPE_SYSTEM_KEY_DOWN)
                {
                    UINT key;
                    CHECK_FAILURE(args->get_VirtualKey(&key));
                    // Check if the key is one we want to handle.
                    if (std::function<void()> action =
                            m_appWindow->GetAcceleratorKeyFunction(key))
                    {
                        // Keep the browser from handling this key, whether
it's autorepeated or
                        // not.
                        CHECK_FAILURE(args->Handle(TRUE));

                        // Filter out autorepeated keys.
                        WEBVIEW2_PHYSICAL_KEY_STATUS status;
                        CHECK_FAILURE(args->get_PhysicalKeyStatus(&status));
                        if (!status.WasKeyDown)
                        {
                            // Perform the action asynchronously to avoid
blocking the
                            // browser process's event queue.
                            m_appWindow->RunAsync(action);
                        }
                    }
                }
                return S_OK;
            })
            .Get(),
        &m_acceleratorKeyPressedToken));
```

remove_AcceleratorKeyPressed

Remove an event handler previously added with add_AcceleratorKeyPressed.

> public HRESULT remove_AcceleratorKeyPressed(EventRegistrationToken token)

WEBVIEW2_KEY_EVENT_TYPE

The type of key event that triggered an AcceleratorKeyPressed event.

> enum WEBVIEW2_KEY_EVENT_TYPE

Values	Descriptions
WEBVIEW2_KEY_EVENT_TYPE_KEY_DOWN	Correspond to window message WM_KEYDOWN.
WEBVIEW2_KEY_EVENT_TYPE_KEY_UP	Correspond to window message WM_KEYUP.

Values	Descriptions
WEBVIEW2_KEY_EVENT_TYPE_SYSTEM_KEY_DOWN	Correspond to window message WM_SYSKEYDOWN.
WEBVIEW2_KEY_EVENT_TYPE_SYSTEM_KEY_UP	Correspond to window message WM_SYSKEYUP.

WEBVIEW2_PHYSICAL_KEY_STATUS

A structure representing the information packed into the LPARAM given to a Win32 key event.

> typedef WEBVIEW2_PHYSICAL_KEY_STATUS

See the documentation for WM_KEYDOWN for details.

interface IWebView2WebView5

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2WebView5
  : public IWebView2WebView4
```

Additional functionality implemented by the primary WebView object.

Summary

Members	Descriptions
add_ContainsFullScreenElementChanged	Notifies when the ContainsFullScreenElement property changes.
remove_ContainsFullScreenElementChanged	Remove an event handler previously added with the corresponding add_ event method.
get_ContainsFullScreenElement	Indicates if the WebView contains a fullscreen HTML element.
add_WebResourceRequested	Add an event handler for the WebResourceRequested event.
AddWebResourceRequestedFilter	Adds a URI and resource context filter to the WebResourceRequested event.
RemoveWebResourceRequestedFilter	Removes a matching WebResource filter that was previously added for the WebResourceRequested event.

You can QueryInterface for this interface from the object that implements IWebView2WebView. See the IWebView2WebView interface for more details.

Members

add_ContainsFullScreenElementChanged

Notifies when the ContainsFullScreenElement property changes.

> public HRESULT add_ContainsFullScreenElementChanged(IWebView2ContainsFullScreenElementChangedEventHandler * eventHandler,EventRegistrationToken * token)

This means that an HTML element inside the WebView is entering fullscreen to the size of the WebView or leaving fullscreen. This event is useful when, for example, a video element requests to go fullscreen. The listener of ContainsFullScreenElementChanged can then resize the WebView in response.

```C++
    // Register a handler for the ContainsFullScreenChanged event.
    CHECK_FAILURE(m_webView->add_ContainsFullScreenElementChanged(
        Callback<IWebView2ContainsFullScreenElementChangedEventHandler>(
            [this](IWebView2WebView5* sender, IUnknown* args) -> HRESULT {
                if (m_fullScreenAllowed)
                {
                    CHECK_FAILURE(sender-
>get_ContainsFullScreenElement(&m_containsFullscreenElement));
                    if (m_containsFullscreenElement)
                    {
                        EnterFullScreen();
                    }
                    else
                    {
                        ExitFullScreen();
                    }
                }
                return S_OK;
            })
            .Get(),
        nullptr));
```

remove_ContainsFullScreenElementChanged

Remove an event handler previously added with the corresponding add_ event method.

> public HRESULT remove_ContainsFullScreenElementChanged(EventRegistrationToken token)

get_ContainsFullScreenElement

Indicates if the WebView contains a fullscreen HTML element.

> public HRESULT get_ContainsFullScreenElement(BOOL * containsFullScreenElement)

add_WebResourceRequested

Add an event handler for the WebResourceRequested event.

> public HRESULT add_WebResourceRequested(IWebView2WebResourceRequestedEventHandler * eventHandler,EventRegistrationToken * token)

Fires when the WebView is performing an HTTP request to a matching URL and resource context filter that was added with AddWebResourceRequestedFilter. At least one filter must be added for the event to fire.

```C++
        if (m_blockImages)
        {
            m_webView->AddWebResourceRequestedFilter(L"*",
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE);
            CHECK_FAILURE(m_webView->add_WebResourceRequested(
                Callback<IWebView2WebResourceRequestedEventHandler>(
                    [this](
                        IWebView2WebView* sender,
                        IWebView2WebResourceRequestedEventArgs* args) {

wil::com_ptr<IWebView2WebResourceRequestedEventArgs2>
                            webResourceEventArgs2;
                        args-
>QueryInterface(IID_PPV_ARGS(&webResourceEventArgs2));
                        WEBVIEW2_WEB_RESOURCE_CONTEXT resourceContext;
                        CHECK_FAILURE(
                            webResourceEventArgs2-
>get_ResourceContext(&resourceContext));
                        // Ensure that the type is image
                        if (resourceContext !=
WEBVIEW2_WEB_RESOURCE_CONTEXT_IMAGE)
                        {
                            return E_INVALIDARG;
                        }
                        // Override the response with an empty one to block
the image.
                        // If put_Response is not called, the request will
continue as normal.
                        wil::com_ptr<IWebView2WebResourceResponse> response;
                        CHECK_FAILURE(m_webViewEnvironment-
>CreateWebResourceResponse(
                                nullptr, 403 /*NoContent*/, L"Blocked", L"",
&response));
                            CHECK_FAILURE(args->put_Response(response.get()));
                            return S_OK;
                        })
                        .Get(),
                    &m_webResourceRequestedTokenForImageBlocking));
        }
        else
        {
            CHECK_FAILURE(m_webView->remove_WebResourceRequested(
                m_webResourceRequestedTokenForImageBlocking));
        }
```

AddWebResourceRequestedFilter

Adds a URI and resource context filter to the WebResourceRequested event.

> public HRESULT AddWebResourceRequestedFilter(LPCWSTR const uri,WEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext)

URI parameter can be a wildcard string ('': zero or more, '?': exactly one). nullptr is equivalent to L"". See WEBVIEW2_WEB_RESOURCE_CONTEXT enum for description of resource context filters.

RemoveWebResourceRequestedFilter

Removes a matching WebResource filter that was previously added for the WebResourceRequested event.

> public HRESULT RemoveWebResourceRequestedFilter(LPCWSTR const uri,WEBVIEW2_WEB_RESOURCE_CONTEXT const resourceContext)

If the same filter was added multiple times, then it will need to be removed as many times as it was added for the removal to be effective. Returns E_INVALIDARG for a filter that was never added.

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

Globals

Article • 12/10/2022

Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

Summary

Expand table

Members	Descriptions
CreateWebView2EnvironmentWithDetails	DLL export to create a WebView2 environment with a custom version of Edge, user data directory and/or additional browser switches.
CreateWebView2Environment	Creates an evergreen WebView2 Environment using the installed Edge version.
GetWebView2BrowserVersionInfo	Get the browser version info including channel name if it is not the stable channel or the Embedded Edge.
CompareBrowserVersions	This method is for anyone want to compare version correctly to determine which version is newer, older or same.

Members

CreateWebView2EnvironmentWithDetails

> public STDAPI CreateWebView2EnvironmentWithDetails(PCWSTR browserExecutableFolder,PCWSTR userDataFolder,PCWSTR additionalBrowserArguments,IWebView2CreateWebView2EnvironmentCompletedHandler * environment_created_handler)

DLL export to create a WebView2 environment with a custom version of Edge, user data directory and/or additional browser switches.

browserExecutableFolder is the relative path to the folder that contains the embedded Edge. The embedded Edge can be obtained by copying the version named folder of an installed Edge, like 73.0.52.0 sub folder of an installed 73.0.52.0 Edge. The folder should have msedge.exe, msedge.dll, etc. Use null or empty string for browserExecutableFolder to create WebView using Edge installed on the machine, in which case the API will try to find a compatible version of Edge installed on the machine according to the channel preference trying to find first per user install and then per machine install.

The default channel search order is stable, beta, dev, and canary. When there is an override WEBVIEW2_RELEASE_CHANNEL_PREFERENCE environment variable or applicable releaseChannelPreference registry value with the value of 1, the channel search order is reversed.

userDataFolder can be specified to change the default user data folder location for WebView2. The path can be an absolute file path or a relative file path that is interpreted as relative to the current process's executable. Otherwise, for UWP apps, the default user data folder will be the app data folder for the package; for non-UWP apps, the default user data folder `{Executable File Name}.WebView2` will be created in the same directory next to the app executable. WebView2 creation can fail if the executable is running in a directory that the process doesn't have permission to create a new folder in. The app is responsible to clean up its user data folder when it is done.

additionalBrowserArguments can be specified to change the behavior of the WebView. These will be passed to the browser process as part of the command line. See Run Chromium with Flags for more information about command line switches to browser process. If the app is launched with a command line switch `--edge-webview-switches=xxx` the value of that switch (xxx in the above example) will also be appended to the browser process command line. Certain switches like `--user-data-dir` are internal and important to WebView. Those switches will be ignored even if specified. If the same switches are specified multiple times, the last one wins. Note that this also applies to switches like `--enable-features`. There is no attempt to merge the different values of the same switch. App process's command line `--edge-webview-switches` value are processed after the additionalBrowserArguments parameter is processed. Also note that as a browser process might be shared among WebViews, the switches are not guaranteed to be applied except for the first WebView that starts the browser process. If parsing failed for the specified switches, they will be ignored. `nullptr` will run browser process with no flags.

environment_created_handler is the handler result to the async operation which will contain the WebView2Environment that got created.

The browserExecutableFolder, userDataFolder and additionalBrowserArguments members of the environmentParams may be overridden by values either specified in environment variables or in the registry.

When creating a WebView2Environment the following environment variables are checked:

```C++
WEBVIEW2_BROWSER_EXECUTABLE_FOLDER
WEBVIEW2_USER_DATA_FOLDER
WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS
WEBVIEW2_RELEASE_CHANNEL_PREFERENCE
```

If an override environment variable is found then we use the browserExecutableFolder, userDataFolder and additionalBrowserArguments values as replacements for the corresponding values in CreateWebView2EnvironmentWithDetails parameters.

While not strictly overrides, there exists additional environment variables that can be set:

```C++
WEBVIEW2_WAIT_FOR_SCRIPT_DEBUGGER
```

When found with a non-empty value, this indicates that the WebView is being launched under a script debugger. In this case, the WebView will issue a `Page.waitForDebugger` CDP command that will cause script execution inside the WebView to pause on launch, until a debugger issues a corresponding `Runtime.runIfWaitingForDebugger` CDP command to resume execution. Note: There is no registry key equivalent of this environment variable.

```C++
WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER
```

When found with a non-empty value, this indicates that the WebView is being launched under a script debugger that also supports host applications that use multiple WebViews. The value is used as the identifier for a named pipe that will be opened and written to when a new WebView is created by the host application. The payload will match that of the remote-debugging-port JSON target and can be used by the external debugger to attach to a specific WebView instance. The format of the pipe created by the debugger should be: `\\.\pipe\WebView2\Debugger\{app_name}\{pipe_name}` where:

- `{app_name}` is the host application exe filename, e.g. WebView2Example.exe
- `{pipe_name}` is the value set for WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER.

To enable debugging of the targets identified by the JSON you will also need to set the WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS environment variable to send `--remote-debugging-port={port_num}` where:

- `{port_num}` is the port on which the CDP server will bind.

Be aware that setting both the WEBVIEW2_PIPE_FOR_SCRIPT_DEBUGGER and WEBVIEW2_ADDITIONAL_BROWSER_ARGUMENTS environment variables will cause the WebViews hosted in your application and their contents to be exposed to 3rd party applications such as debuggers.

Note: There is no registry key equivalent of this environment variable.

If none of those environment variables exist, then the registry is examined next. The following registry keys are checked:

```C++
[{Root}\Software\Policies\Microsoft\EmbeddedBrowserWebView\LoaderOverride\
{AppId}]
"releaseChannelPreference"=dword:00000000
"browserExecutableFolder"=""
"userDataFolder"=""
"additionalBrowserArguments"=""
```

In the unlikely scenario where some instances of WebView are open during a browser update we could end up blocking the deletion of old Edge browsers. To avoid running out of disk space a new WebView creation will fail with the next error if it detects that there are many old versions present.

```C++
ERROR_DISK_FULL
```

The default maximum number of Edge versions allowed is 20.

The maximum number of old Edge versions allowed can be overwritten with the value of the following environment variable.

```C++
```

```
COREWEBVIEW2_MAX_INSTANCES
```

If the Webview depends on an installed Edge and it is uninstalled any subsequent creation will fail with the next error

```C++
ERROR_PRODUCT_UNINSTALLED
```

First we check with Root as HKLM and then HKCU. AppId is first set to the Application User Model ID of the caller's process, then if there's no corresponding registry key the AppId is set to the executable name of the caller's process, or if that isn't a registry key then '*'. If an override registry key is found then we use the browserExecutableFolder, userDataFolder and additionalBrowserArguments registry values as replacements for the corresponding values in CreateWebView2EnvironmentWithDetails parameters. If any of those registry values isn't present, then the parameter passed to CreateWebView2Environment is used.

CreateWebView2Environment

> public STDAPI CreateWebView2Environment(IWebView2CreateWebView2EnvironmentCompletedHandler * environment_created_handler)

Creates an evergreen WebView2 Environment using the installed Edge version.

This is equivalent to calling CreateWebView2EnvironmentWithDetails with nullptr for browserExecutableFolder, userDataFolder, additionalBrowserArguments. See CreateWebView2EnvironmentWithDetails for more details.

GetWebView2BrowserVersionInfo

> public STDAPI GetWebView2BrowserVersionInfo(PCWSTR browserExecutableFolder,LPWSTR * versionInfo)

Get the browser version info including channel name if it is not the stable channel or the Embedded Edge.

Channel names are beta, dev, and canary. If an override exists for the browserExecutableFolder or the channel preference, the override will be used. If there isn't an override, then the parameter passed to GetWebView2BrowserVersionInfo is used.

CompareBrowserVersions

> public STDAPI CompareBrowserVersions(PCWSTR version1,PCWSTR version2,int * result)

This method is for anyone want to compare version correctly to determine which version is newer, older or same.

It can be used to determine whether to use webview2 or certain feature base on version. Sets the value of result to -1, 0 or 1 if version1 is less than, equal or greater than version2 respectively. Returns E_INVALIDARG if it fails to parse any of the version strings or any input parameter is null. Input can directly use the versionInfo obtained from GetWebView2BrowserVersionInfo, channel info will be ignored.

Feedback

interface IWebView2AcceleratorKeyPressedEventArgs

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2AcceleratorKeyPressedEventArgs
  : public IUnknown
```

Event args for the AcceleratorKeyPressed event.

Summary

Members	Descriptions
get_KeyEventType	The key event type that caused the event to be fired.
get_VirtualKey	The Win32 virtual key code of the key that was pressed or released.
get_KeyEventLParam	The LPARAM value that accompanied the window message.
get_PhysicalKeyStatus	A structure representing the information passed in the LPARAM of the window message.
Handle	Calling this will allow the browser process to continue.

Members

get_KeyEventType

The key event type that caused the event to be fired.

> public HRESULT get_KeyEventType(WEBVIEW2_KEY_EVENT_TYPE * keyEventType)

This is one of WEBVIEW2_KEY_EVENT_TYPE_KEY_DOWN, WEBVIEW2_KEY_EVENT_TYPE_KEY_UP, WEBVIEW2_KEY_EVENT_TYPE_SYSTEM_KEY_DOWN, or WEBVIEW2_KEY_EVENT_TYPE_SYSTEM_KEY_UP.

get_VirtualKey

The Win32 virtual key code of the key that was pressed or released.

> public HRESULT get_VirtualKey(UINT * virtualKey)

This will be one of the Win32 virtual key constants such as VK_RETURN or an (uppercase) ASCII value such as 'A'. You can check whether Ctrl or Alt are pressed by calling GetKeyState(VK_CONTROL) or GetKeyState(VK_MENU).

get_KeyEventLParam

The LPARAM value that accompanied the window message.

> public HRESULT get_KeyEventLParam(INT * lParam)

See the documentation for the WM_KEYDOWN and WM_KEYUP messages.

get_PhysicalKeyStatus

A structure representing the information passed in the LPARAM of the window message.

> public HRESULT get_PhysicalKeyStatus(WEBVIEW2_PHYSICAL_KEY_STATUS * physicalKeyStatus)

Handle

Calling this will allow the browser process to continue.

> public HRESULT Handle(BOOL handled)

Passing TRUE will prevent the browser from performing the default action for this accelerator key. If the event handler returns without calling Handle(), it is equivalent to calling Handle(FALSE). Calling Handle() after it has already been called or the event handler has returned will do nothing.

interface IWebView2DevToolsProtocolEventReceivedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2DevToolsProtocolEventReceivedEventArgs
  : public IUnknown
```

Event args for the DevToolsProtocolEventReceived event.

Summary

Members	Descriptions
get_ParameterObjectAsJson	The parameter object of the corresponding DevToolsProtocol event represented as a JSON string.

Members

get_ParameterObjectAsJson

The parameter object of the corresponding DevToolsProtocol event represented as a JSON string.

> public HRESULT get_ParameterObjectAsJson(LPWSTR * parameterObjectAsJson)

Feedback

interface IWebView2DocumentStateChangedEventArgs

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2DocumentStateChangedEventArgs
  : public IUnknown
```

Event args for the DocumentStateChanged event.

Summary

Members	Descriptions
get_IsNewDocument	True if the page being navigated to is a new document.
get_IsErrorPage	True if the loaded content is an error page.

Members

get_IsNewDocument

True if the page being navigated to is a new document.

> public HRESULT get_IsNewDocument(BOOL * isNewDocument)

get_IsErrorPage

True if the loaded content is an error page.

> public HRESULT get_IsErrorPage(BOOL * isErrorPage)

interface IWebView2MoveFocusRequestedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2MoveFocusRequestedEventArgs
  : public IUnknown
```

Event args for the MoveFocusRequested event.

Summary

Members	Descriptions
get_Reason	The reason for WebView to fire the MoveFocus Requested event.
get_Handled	Indicate whether the event has been handled by the app.
put_Handled	Set the Handled property.

Members

get_Reason

The reason for WebView to fire the MoveFocus Requested event.

> public HRESULT get_Reason(WEBVIEW2_MOVE_FOCUS_REASON * value)

get_Handled

Indicate whether the event has been handled by the app.

> public HRESULT get_Handled(BOOL * value)

If the app has moved the focus to its desired location, it should set Handled property to TRUE. When Handled property is false after the event handler returns, default action will be taken. The default action is to try to find the next tab stop child window in the app and try to move focus to that window. If there is no other such window to move focus to, focus will be cycled within the WebView's web content.

put_Handled

Set the Handled property.

> public HRESULT put_Handled(BOOL value)

---

interface IWebView2NavigationCompletedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NavigationCompletedEventArgs
  : public IUnknown
```

Event args for the NavigationCompleted event.

Summary

Members	Descriptions
get_IsSuccess	True when the navigation is successful.
get_WebErrorStatus	The error code if the navigation failed.

Members

get_IsSuccess

True when the navigation is successful.

> public HRESULT get_IsSuccess(BOOL * isSuccess)

This is false for a navigation that ended up in an error page (failures due to no network, DNS lookup failure, HTTP server responds with 4xx), but could also be false for additional things such as window.stop() called on navigated page.

get_WebErrorStatus

The error code if the navigation failed.

> public HRESULT get_WebErrorStatus(WEBVIEW2_WEB_ERROR_STATUS * WEBVIEW2_WEB_ERROR_STATUS)

interface IWebView2NavigationStartingEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NavigationStartingEventArgs
  : public IUnknown
```

Event args for the NavigationStarting event.

Summary

Members	Descriptions
get_Uri	The uri of the requested navigation.
get_IsUserInitiated	True when the navigation was initiated through a user gesture as opposed to programmatic navigation.
get_IsRedirected	True when the navigation is redirected.
get_RequestHeaders	The HTTP request headers for the navigation.
get_Cancel	The host may set this flag to cancel the navigation.
put_Cancel	Set the Cancel property.

Members

get_Uri

The uri of the requested navigation.

> public HRESULT get_Uri(LPWSTR * uri)

get_IsUserInitiated

True when the navigation was initiated through a user gesture as opposed to programmatic navigation.

> public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)

get_IsRedirected

True when the navigation is redirected.

> public HRESULT get_IsRedirected(BOOL * isRedirected)

get_RequestHeaders

The HTTP request headers for the navigation.

> public HRESULT get_RequestHeaders(IWebView2HttpRequestHeaders ** requestHeaders)

Note, you cannot modify the HTTP request headers in a NavigationStarting event.

get_Cancel

The host may set this flag to cancel the navigation.

> public HRESULT get_Cancel(BOOL * cancel)

If set, it will be as if the navigation never happened and the current page's content will be intact. For performance reasons, GET HTTP requests may happen, while the host is responding. This means cookies can be set and used part of a request for the navigation.

put_Cancel

Set the Cancel property.

> public HRESULT put_Cancel(BOOL cancel)

Feedback

interface IWebView2NewVersionAvailableEventArgs

Article • 10/14/2020

> Note
>
> This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NewVersionAvailableEventArgs
  : public IUnknown
```

Event args for the NewVersionAvailable event.

Summary

Members	Descriptions
get_NewVersion	The browser version info of the current IWebView2Environment.

Members

get_NewVersion

The browser version info of the current IWebView2Environment.

> public HRESULT get_NewVersion(LPWSTR * newVersion)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2NewWindowRequestedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NewWindowRequestedEventArgs
  : public IUnknown
```

Event args for the NewWindowRequested event.

Summary

Members	Descriptions
get_Uri	The target uri of the NewWindowRequest.
put_NewWindow	Sets a WebView as a result of the NewWindowRequest.
get_NewWindow	Gets the new window.
put_Handled	Sets whether the NewWindowRequestedEvent is handled by host.
get_Handled	Gets whether the NewWindowRequestedEvent is handled by host.
get_IsUserInitiated	IsUserInitiated is true when the new window request was initiated through a user gesture such as clicking an anchor tag with target.
GetDeferral	Obtain an IWebView2Deferral object and put the event into a deferred state.

The event is fired when content inside webview requested to a open a new window (through window.open() etc.)

Members

get_Uri

The target uri of the NewWindowRequest.

> public HRESULT get_Uri(LPWSTR * uri)

put_NewWindow

Sets a WebView as a result of the NewWindowRequest.

> public HRESULT put_NewWindow(IWebView2WebView * newWindow)

The target webview should not be navigated. If the NewWindow is set, its top level window will return as the opened WindowProxy.

get_NewWindow

Gets the new window.

> public HRESULT get_NewWindow(IWebView2WebView ** newWindow)

put_Handled

Sets whether the NewWindowRequestedEvent is handled by host.

> public HRESULT put_Handled(BOOL handled)

If this is false and no NewWindow is set, the WebView will open a popup window and it will be returned as opened WindowProxy. If set to true and no NewWindow is set for a window.open call, the opened WindowProxy will be for an dummy window object and no window will load. Default is false.

get_Handled

Gets whether the NewWindowRequestedEvent is handled by host.

> public HRESULT get_Handled(BOOL * handled)

get_IsUserInitiated

IsUserInitiated is true when the new window request was initiated through a user gesture such as clicking an anchor tag with target.

> public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)

GetDeferral

Obtain an IWebView2Deferral object and put the event into a deferred state.

> public HRESULT GetDeferral(IWebView2Deferral ** deferral)

You can use the IWebView2Deferral object to complete the window open request at a later time. While this event is deferred the opener window will be returned a WindowProxy to an unnavigated window, which will navigate when the deferral is complete.

interface IWebView2PermissionRequestedEventArgs

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2PermissionRequestedEventArgs
  : public IUnknown
```

Event args for the PermissionRequested event.

Summary

Members	Descriptions
get_Uri	The origin of the web content that requests the permission.
get_PermissionType	The type of the permission that is requested.
get_IsUserInitiated	True when the permission request was initiated through a user gesture.
get_State	The status of a permission request, i.e.
put_State	Set the State property.
GetDeferral	GetDeferral can be called to return an IWebView2Deferral object.

Members

get_Uri

The origin of the web content that requests the permission.

> public HRESULT get_Uri(LPWSTR * uri)

get_PermissionType

The type of the permission that is requested.

> public HRESULT get_PermissionType(WEBVIEW2_PERMISSION_TYPE * value)

get_IsUserInitiated

True when the permission request was initiated through a user gesture.

> public HRESULT get_IsUserInitiated(BOOL * isUserInitiated)

Note that being initiated through a user gesture doesn't mean that user intended to access the associated resource.

get_State

The status of a permission request, i.e.

> public HRESULT get_State(WEBVIEW2_PERMISSION_STATE * value)

whether the request is granted. Default value is WEBVIEW2_PERMISSION_STATE_DEFAULT.

put_State

Set the State property.

> public HRESULT put_State(WEBVIEW2_PERMISSION_STATE value)

GetDeferral

GetDeferral can be called to return an IWebView2Deferral object.

> public HRESULT GetDeferral(IWebView2Deferral ** deferral)

Developer can use the deferral object to make the permission decision at a later time.

interface IWebView2ProcessFailedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ProcessFailedEventArgs
  : public IUnknown
```

Event args for the ProcessFailed event.

Summary

Members	Descriptions
get_ProcessFailedKind	The kind of process failure that has occurred.

Members

get_ProcessFailedKind

The kind of process failure that has occurred.

> public HRESULT get_ProcessFailedKind(WEBVIEW2_PROCESS_FAILED_KIND * processFailedKind)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2ScriptDialogOpeningEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2ScriptDialogOpeningEventArgs
  : public IUnknown
```

Event args for the IWebView2WebView::add_ScriptDialogOpening event.

Summary

Members	Descriptions
get_Uri	The URI of the page that requested the dialog box.
get_Kind	The kind of JavaScript dialog box.
get_Message	The message of the dialog box.
Accept	The host may call this to respond with OK to confirm and prompt dialogs or not call this method to indicate cancel.
get_DefaultText	The second parameter passed to the JavaScript prompt dialog.
get_ResultText	The return value from the JavaScript prompt function if Accept is called.
put_ResultText	Set the ResultText property.
GetDeferral	GetDeferral can be called to return an IWebView2Deferral object.

Members

get_Uri

The URI of the page that requested the dialog box.

> public HRESULT get_Uri(LPWSTR * uri)

get_Kind

The kind of JavaScript dialog box.

> public HRESULT get_Kind(WEBVIEW2_SCRIPT_DIALOG_KIND * kind)

get_Message

The message of the dialog box.

> public HRESULT get_Message(LPWSTR * message)

From JavaScript this is the first parameter passed to alert, confirm, and prompt.

Accept

The host may call this to respond with OK to confirm and prompt dialogs or not call this method to indicate cancel.

> public HRESULT Accept()

From JavaScript this means that the confirm function returns true if Accept is called. And for the prompt function it returns the value of ResultText if Accept is called and returns false otherwise.

get_DefaultText

The second parameter passed to the JavaScript prompt dialog.

> public HRESULT get_DefaultText(LPWSTR * defaultText)

This is the default value to use for the result of the prompt JavaScript function.

get_ResultText

The return value from the JavaScript prompt function if Accept is called.

> public HRESULT get_ResultText(LPWSTR * resultText)

This is ignored for dialog kinds other than prompt. If Accept is not called this value is ignored and false is returned from prompt.

put_ResultText

Set the ResultText property.

> public HRESULT put_ResultText(LPCWSTR resultText)

GetDeferral

GetDeferral can be called to return an IWebView2Deferral object.

> public HRESULT GetDeferral(IWebView2Deferral ** deferral)

You can use this to complete the event at a later time.

interface IWebView2WebMessageReceivedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebMessageReceivedEventArgs
  : public IUnknown
```

Event args for the WebMessageReceived event.

Summary

Members	Descriptions
get_Source	The URI of the document that sent this web message.
get_WebMessageAsJson	The message posted from the webview content to the host converted to a JSON string.
get_WebMessageAsString	If the message posted from the webview content to the host is a string type, this method will return the value of that string.

Members

get_Source

The URI of the document that sent this web message.

> public HRESULT get_Source(LPWSTR * source)

get_WebMessageAsJson

The message posted from the webview content to the host converted to a JSON string.

> public HRESULT get_WebMessageAsJson(LPWSTR * webMessageAsJson)

Use this to communicate via JavaScript objects.

For example the following postMessage calls result in the following WebMessageAsJson values:

```C++
postMessage({'a': 'b'})      L"{\"a\": \"b\"}"
postMessage(1.2)             L"1.2"
postMessage('example')       L"\"example\""
```

get_WebMessageAsString

If the message posted from the webview content to the host is a string type, this method will return the value of that string.

> public HRESULT get_WebMessageAsString(LPWSTR * webMessageAsString)

If the message posted is some other kind of JavaScript type this method will fail with E_INVALIDARG. Use this to communicate via simple strings.

For example the following postMessage calls result in the following WebMessageAsString values:

```C++
postMessage({'a': 'b'})      E_INVALIDARG
postMessage(1.2)             E_INVALIDARG
postMessage('example')       L"example"
```

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebResourceRequestedEventArgs

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebResourceRequestedEventArgs
  : public IUnknown
```

Event args for the WebResourceRequested event.

Summary

Members	Descriptions
get_Request	The HTTP request.
get_Response	The HTTP response.
put_Response	Set the Response property.
GetDeferral	Obtain an IWebView2Deferral object and put the event into a deferred state.

Members

get_Request

The HTTP request.

> public HRESULT get_Request(IWebView2WebResourceRequest ** request)

get_Response

The HTTP response.

> public HRESULT get_Response(IWebView2WebResourceResponse ** response)

put_Response

Set the Response property.

> public HRESULT put_Response(IWebView2WebResourceResponse * response)

GetDeferral

Obtain an IWebView2Deferral object and put the event into a deferred state.

> public HRESULT GetDeferral(IWebView2Deferral ** deferral)

You can use the IWebView2Deferral object to complete the network request at a later time.

interface IWebView2AcceleratorKeyPressedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2AcceleratorKeyPressedEventHandler
  : public IUnknown
```

The caller implements this interface to receive the AcceleratorKeyPressed event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2AcceleratorKeyPressedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2AddScriptToExecuteOnDocumentCreatedCompletedHandler
  : public IUnknown
```

The caller implements this interface to receive the result of the AddScriptToExecuteOnDocumentCreated method.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

Members

Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT errorCode,LPCWSTR id)

Feedback

interface IWebView2CallDevToolsProtocolMethodCompletedHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2CallDevToolsProtocolMethodCompletedHandler
  : public IUnknown
```

The caller implements this interface to receive CallDevToolsProtocolMethod completion results.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

Members

Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT errorCode,LPCWSTR returnObjectAsJson)

Feedback

interface IWebView2CapturePreviewCompletedHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2CapturePreviewCompletedHandler
  : public IUnknown
```

The caller implements this method to receive the result of the CapturePreview method.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status of the corresponding asynchronous method call.

The result is written to the stream provided in the CapturePreview method call.

Members

Invoke

Called to provide the implementer with the completion status of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT result)

Feedback

interface IWebView2ContainsFullScreenElementChangedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ContainsFullScreenElementChangedEventHandler
  : public IUnknown
```

The caller implements this method to receive the ContainsFullScreenElementChanged events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

There are no event args for this event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView5 * webview,IUnknown * args)

There are no event args and the args parameter will be null.

interface IWebView2CreateWebView2EnvironmentCompletedHandler

Article • 10/14/2020

ⓘ **Note**

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2CreateWebView2EnvironmentCompletedHandler
  : public IUnknown
```

The caller implements this interface to receive the WebView2Environment created via CreateWebView2Environment.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

Members

Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT result,IWebView2Environment * webViewEnvironment)

Feedback

interface IWebView2CreateWebViewCompletedHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2CreateWebViewCompletedHandler
  : public IUnknown
```

The caller implements this interface to receive the WebView created via CreateWebView.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

Members

Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT result,IWebView2WebView * webView)

Feedback

interface IWebView2DevToolsProtocolEventReceivedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2DevToolsProtocolEventReceivedEventHandler
  : public IUnknown
```

The caller implements this interface to receive DevToolsProtocolEventReceived events from the IWebView2WebView.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2DevToolsProtocolEventReceivedEventArgs * args)

Feedback

interface IWebView2DocumentStateChangedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2DocumentStateChangedEventHandler
  : public IUnknown
```

The caller implements this interface to receive the DocumentStateChanged event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2DocumentStateChangedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2DocumentTitleChangedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2DocumentTitleChangedEventHandler
  : public IUnknown
```

The caller implements this interface to receive DocumentTitleChanged events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Use the DocumentTitle property to get the modified title.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView3 * webview,IUnknown * args)

There are no event args and the args parameter will be null.

Feedback

interface IWebView2ExecuteScriptCompletedHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ExecuteScriptCompletedHandler
  : public IUnknown
```

The caller implements this interface to receive the result of the ExecuteScript method.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

Members

Invoke

Called to provide the implementer with the completion status and result of the corresponding asynchronous method call.

> public HRESULT Invoke(HRESULT errorCode,LPCWSTR resultObjectAsJson)

Feedback

interface IWebView2FocusChangedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2FocusChangedEventHandler
  : public IUnknown
```

The caller implements this method to receive the GotFocus and LostFocus events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

There are no event args for this event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IUnknown * args)

There are no event args and the args parameter will be null.

Feedback

interface IWebView2MoveFocusRequestedEventHandler

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2MoveFocusRequestedEventHandler
  : public IUnknown
```

The caller implements this method to receive the MoveFocusRequested event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2MoveFocusRequestedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2NavigationCompletedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NavigationCompletedEventHandler
  : public IUnknown
```

The caller implements this interface to receive the NavigationCompleted event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2NavigationCompletedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2NavigationStartingEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NavigationStartingEventHandler
  : public IUnknown
```

The caller implements this interface to receive the NavigationStarting event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2NavigationStartingEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2NewVersionAvailableEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2NewVersionAvailableEventHandler
  : public IUnknown
```

The caller implements this interface to receive NewVersionAvailable events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Use the get_NewVersion method of IWebView2NewVersionAvailableEventArgs to get the new version number.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2Environment * webviewEnvironment,IWebView2NewVersionAvailableEventArgs * args)

Feedback

interface IWebView2NewWindowRequestedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see **WebView2 API Reference**.

```
interface IWebView2NewWindowRequestedEventHandler
  : public IUnknown
```

The caller implements this interface to receive NewWindowRequested events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2NewWindowRequestedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2PermissionRequestedEventHandler

Article • 10/14/2020

Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2PermissionRequestedEventHandler
  : public IUnknown
```

The caller implements this interface to receive the PermissionRequested event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2PermissionRequestedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2ProcessFailedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ProcessFailedEventHandler
  : public IUnknown
```

The caller implements this interface to receive ProcessFailed events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2ProcessFailedEventArgs * args)

Feedback

interface IWebView2ScriptDialogOpeningEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ScriptDialogOpeningEventHandler
  : public IUnknown
```

The caller implements this interface to receive the ScriptDialogOpening event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2ScriptDialogOpeningEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebMessageReceivedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebMessageReceivedEventHandler
  : public IUnknown
```

The caller implements this interface to receive the WebMessageReceived event.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2WebMessageReceivedEventArgs * args)

Feedback

Was this page helpful? Yes No

Get help at Microsoft Q&A

interface IWebView2WebResourceRequestedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2WebResourceRequestedEventHandler
  : public IUnknown
```

Fires when an HTTP request is made in the webview.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

The host can override request, response headers and response content.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IWebView2WebResourceRequestedEventArgs * args)

Feedback

interface IWebView2ZoomFactorChangedEventHandler

Article • 10/14/2020

ⓘ Note

This reference is no longer being maintained. For the latest API reference, see WebView2 API Reference.

```
interface IWebView2ZoomFactorChangedEventHandler
  : public IUnknown
```

The caller implements this interface to receive ZoomFactorChanged events.

Summary

Members	Descriptions
Invoke	Called to provide the implementer with the event args for the corresponding event.

Use the IWebView2WebView.ZoomFactor property to get the modified zoom factor.

Members

Invoke

Called to provide the implementer with the event args for the corresponding event.

> public HRESULT Invoke(IWebView2WebView * webview,IUnknown * args)

There are no event args and the args parameter will be null.

Feedback