

Classes and Structures

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

- ▼ **N** **FB** Primary location of FireBreath classes and utility functions
 - ▼ **N** **Npapi**
 - C** **Npa** Provides a **FB::BrowserHost** implementation for Npapi
 - piBrowserHost**
 - C** **NPO** Provides a **FB::JSObject** implementation that wraps a **NPObjct***
 - bjectAPI**
 - C** **Npa** Provides a **D3d10AsyncDrawService** implementation for NPAPI
 - piAsyncDrawService**
 - ▼ **N** **DOM** These are convenience objects used to access and manipulate **DOM** objects. They should never be created directly; instead, get the **Window** object or the **Document** object from the **BrowserHost** or create it with **DOM::Element::create(obj)** (or similar)
 - C** **Doc** Abstraction for accessing and manipulating a **DOM Document**
 - ument**
 - C** **Ele** **DOM Element** wrapper
 - ment**
 - C** **Node** **DOM Node** wrapper
 - ode**
 - C** **Win** **DOM Window** abstraction for manipulating and accessing the javascript window object that the plugin is contained in
 - indow**
 - ▼ **N** **ActiveX**
 - ▼ **N** **AX**
 - DOM**
 - C** **D** ActiveX specific implementation of **DOM::Document**
 - ocument**
 - C** **E** ActiveX specific implementation of **DOM::Element**
 - lement**
 - C** **N** Provides an ActiveX specific implementation of **DOM::Node**
 - ode**
 - C** **W** ActiveX specific implementation of **DOM::Window**
 - indow**
 - C** **Act** Provides a **D3d10AsyncDrawService** implementation for ActiveX
 - iveXAsyncDrawService**
 - C** **Act** Provides a **BrowserHost** implementation for ActiveX
 - iveXBrowserHost**
 - C** **Idi** Provide a **JSObject** implementation to wrap a **IDispatch** ActiveX object
 - spatchAPI**
 - C** **Async** Asynchronous drawing service base class
 - DrawService**
 - C** **Brows** Browser-specific plugin base class
 - erPlugin**

▼ **C** **Brows** This is the abstract base class (interface class) for a browser stream
 erStream

C **Ran** Specifies the range for a read range request (start to end) in bytes
 ge

C **Brows** Information about an HTTP request to be made
 erStreamReque
 st

C **Defau** Simple implementation of a stream event handler from which you can derive your own stream
 ltBrowserStre
 amHandler events handler

C **Facto** This is the base factory for a plugin. Every plugin *must* implement this class and override at
 ryBase least the `createPlugin()` method. To further customize your plugin, you can override other
 methods to replace the `PluginWindow` or the `NpapiPlugin` class

C **Plugi** Base class for all FireBreath Plugins
 nCore

C **Plugi** Plugin event base class
 nEvent

C **Attac** Fired when a `PluginEventSink` is attached to a `PluginEventSource` (such as a `PluginCore`
 hedEvent derived plugin object being attached to a `PluginWindow` to get events)

C **Detac** Fired when a `PluginEventSink` is detached from a `PluginEventSource` (such as a `PluginCo`
 hedEvent re derived plugin object being detached from a `PluginWindow` to get events)

C **Chang** Fired when a `PluginEventSink` has changed in some fundamental way that the plugin needs
 edEvent to know about, such as a different drawing context being provided by the browser to a `PluginW`
 indow

C **Resiz** Fired when the plugin window is resized
 edEvent

C **ClipC** Fired when the clipping of the plugin drawing area changes
 hangedEvent

C **Refre** Fired when the plugin should repaint itself (such as on windows when `WM_PAINT` is received)
 shEvent

C **Quick** Mac QuickDraw Draw event
 DrawDraw

C **CoreG** Mac CoreGraphics Draw event
 raphicsDraw

C **Focus** Fired when the focus changes
 ChangedEvent

C **Timer** Fired when a timer event is received; currently this only works on windows and is fired when
 Event `WM_TIMER` is received

C **KeyEv** Fired for a key event
 ent

C **KeyUp** Fired for a key up event
 Event

C **KeyDo** Fired for a key down event
 wnEvent

C **TextE** Used at least on Mac, possibly elsewhere; gives us text
 vent

C **MacEv** Carbon Mac event
 entCarbon

C **MacEv** Cocoa mac event
 entCocoa

C **Mouse** Fired when a mouse event occurs
Event

C **Mouse** Fired when the mouse moves
MoveEvent

C **Mouse** Fired when a mouse button event occurs
ButtonEvent

C **Mouse** Fired when a mouse down event occurs
DownEvent

C **Mouse** Fired when a mouse double click event occurs
DoubleClickEvent

C **Mouse** Fired when a mouse up event occurs
UpEvent

C **Mouse** Fired when the user moves the scrollwheel
ScrollEvent

C **Mouse** Fired when the user moves mouse over the plugin
EnteredEvent

C **Mouse** Fired when the user moves mouse away from the plugin
ExitedEvent

C **Stream** Base class for all stream events
mEvent

C **Stream** This event is fired when the given stream was created
mCreatedEvent

C **Stream** This event is fired when a stream failed to open, e.g. the url was invalid or a seekable stream was requested while the server provided only a non-seekable stream
mFailedOpenEvent

C **Stream** This event is fired when a stream has completed downloading
mCompletedEvent

C **Window** Generic windows event. All windows events going through the winproc get first fired as a **WindowEvent** and then fired as the specific type, allowing you to handle them either way
WindowEvent

C **X11Event** Generic X11 event
X11Event

C **X11NativeGdkEvent** Class encapsulation for native Gdk event
X11NativeGdkEvent

C **X11NativeGdkExposeEvent** Class encapsulation for native Gdk expose event
X11NativeGdkExposeEvent

C **PluginEventSink** Plugin event sink; all objects that can receive **PluginEvent**s should inherit from this class. Most notably, **PluginCore** extends this class
PluginEventSink

C **PluginEventSource** Base class for any object, such as **BrowserStream** or **PluginWindow**, that needs to fire events to a **PluginEventSink** object (such as a **PluginCore** derived plugin class)
PluginEventSource

C **PluginWindow** **PluginWindow** is the base class for all **PluginWindow** objects
PluginWindow

C **HttpStreamResponse** Data structure to hold the response to an HTTP Get request
HttpStreamResponse

C **SimpleStreamHelper** Helper class for making simple HTTP requests
SimpleStreamHelper

C **Timer** **Timer** Utility
Timer

C **Timer** `TimerService` Utility Service

C **WinMe** Creates a message window. Don't touch this if you don't understand what you are doing
`MessageWindow`

C **Catch** When used as a parameter on a `JSAPIAuto` function this matches 0 or more variants – in other words, all other parameters from this point on regardless of type
All

C **Metho** Used by `FB::JSAPISimple` to store information about a method
`dInfo`

C **Prope** Used by `FB::JSAPISimple` to store information about a property
`rtyInfo`

C **Prope** Used by `FB::JSAPIAuto` to store property implementation details, created by `FB::`
`rtyFuncutors` `make_property()`

C **Async** This class is used by `BrowserHost` for the `BrowserHost::AsyncHtmlLog` method
`LogRequest`

C **Brows** Browser host base class
`erHost`

C **JSAPI** JavaScript API class – provides a javascript interface that can be exposed to the browser

C **scope** Provides a helper class for locking
`d_zonelock`

C **JSAPI** `JSAPI` class with automatic argument type enforcement
Auto

C **JSAPI** JavaScript API base class implementation – provides basic functionality for C++ `JSAPI` objects
Impl

C **JSAPI** JavaScript API Wrapper – this can wrap any type of `JSAPI` object and be passed back to any browser that
Proxy

C **JSAPI** Simple `JSAPI` implementation for those who for whatever reason don't want to use `JSAPIAuto`
Simple

C **scrip** Exception type; when thrown in a `JSAPI` method, a javascript exception will be thrown
`t_error`

C **inval** Thrown by a `JSAPI` object when the argument(s) provided to a `SetProperty` or `Invoke` call are found to be invalid. `JSAPIAuto` will throw this automatically if the argument cannot be `convert_cast` to the type expected by the function
`id_arguments`

C **objec** Thrown by a `JSAPI` object when a call is made on it after the object has been invalidated
`t_invalidated`

C **inval** Thrown when an `Invoke`, `SetProperty`, or `GetProperty` call is made for a member that is invalid (does not exist, not accessible, only supports `Get` or `Set`, etc)
`id_member`

C **JSObj** Wraps a Javascript Object
`ect`

C **SafeQ** Basic thread-safe queue class
`ueue`

C **TypeI** Bidirectional map between an identifier and a variant
DMap

C **URI** Data structure for dealing with `URI` strings

C **bad_v** Thrown when `variant::cast<type>` or `variant::convert_cast<type>` fails because the type of the value stored in the variant is not compatible with the operation
`ariant_cast`

C **varia** Accepts any datatype, used in all interactions with javascript. Provides tools for getting back out the type you put in or for coercing that type into another type (if possible)
nt

C **Plugin**
nEventMacCarbon Mac OS X Carbon specific implementation of **PluginEventManager**

C **Plugin**
nEventMacCocoa Mac OS X Cocoa specific implementation of **PluginEventManager**

C **Plugin**
nWindowlessWin Windows specific implementation of **PluginWindow**

C **Plugin**
nWindowWin Windows specific implementation of **PluginWindow**

C **Plugin**
nWindowX11 X11 specific implementation of **PluginWindow**

C **Content**
Fired when DPI settings change (e.g. when moving webbrowser window between Retina and non-Retina displays)