class FB JSObject

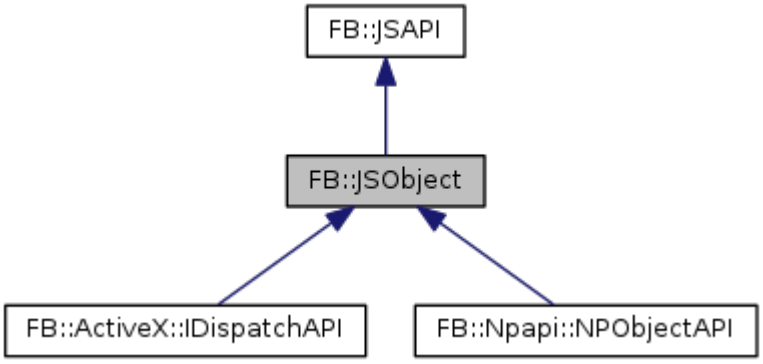
FB::JSObject Class Referenceabstract

Public Member Functions | Static Public Member Functions | List of all members

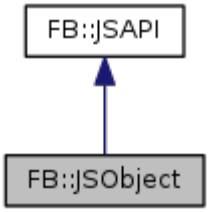
Wraps a Javascript Object. More...

`#include "JSObject.h"`

Inheritance diagram for FB::JSObject:



Collaboration diagram for FB::JSObject:



Public Member Functions

	JSObject (const **BrowserHostPtr** &h) Constructor. More...
virtual	**~JSObject** () Finaliser. More...
virtual void	**InvokeAsync** (const std::string &methodName, const std::vector< **variant** > &args) Just like Invoke, but works asynchronously. Useful for javascript callbacks and events. Can be safely called from any thread. More...
virtual void	**SetPropertyAsync** (const std::string &propertyName, const **variant** &value) Just like SetProperty, but works asynchronously. Useful if you are running on another thread and don't need to wait to be sure it worked. More...
virtual **JSAPIPtr**	**getJSAPI** () const =0 Get associated **FB::JSAPI**. More...
BrowserHostPtr	**getHost** () Get the associated **FB::BrowserHost**; may throw std::bad_cast. More...

Public Member Functions inherited from **FB::JSAPI**

	JSAPI (void) Default constructor. More...
virtual	**~JSAPI** (void) Finaliser. More...
virtual void	**invalidate** ()=0 Invalidates this object. More...
virtual void	**shutdown** ()

	Called to notify the object that the plugin is shutting down. More...
virtual void	**pushZone** (const **SecurityZone** &securityLevel) Pushes a new security level and locks a mutex (for every Push there *must* be a Pop!) More...
virtual void	**popZone** () Pops off a security level and unlocks the mutex (for every Push there *must* be a Pop!) More...
virtual void	**setDefaultZone** (const **SecurityZone** &securityLevel) Sets the default zone (the zone the class operates on before a push) More...
tual **SecurityZone**	**getDefaultZone** () const Gets the default zone (the zone the class operates on before a push) More...
tual **SecurityZone**	**getZone** () const Gets the currently active zone. More...
virtual void	**registerEventMethod** (const std::string &name, **JSObjectPtr** &event) Called by the browser to register an event handler method. More...
virtual void	**registerEventMethod** (const std::wstring &name, **JSObjectPtr** &event)
virtual void	**unregisterEventMethod** (const std::string &name, **JSObjectPtr** &event) Called by the browser to unregister an event handler method. More...
virtual void	**unregisterEventMethod** (const std::wstring &name, **JSObjectPtr** &event)
virtual void	**registerEventInterface** (const **JSObjectPtr** &event) Called by the browser to register a **JSObject** interface that handles events. This is primarily used by IE. Objects provided to this method are called when events are fired by calling a method of the event name on the event interface. More...
virtual void	**unregisterEventInterface** (const **JSObjectPtr** &event) Called by the browser to unregister a **JSObject** interface that handles events. More...
virtual void	**getMemberNames** (std::vector< std::string > &nameVector) const =0 Called by the browser to enumerate the members of this **JSAPI** object. More...
virtual size_t	**getMemberCount** () const =0 Gets the member count. More...
virtual bool	**HasMethod** (const std::wstring &methodName) const
virtual bool	**HasMethod** (const std::string &methodName) const =0 Query if the **JSAPI** object has the 'methodName' method. More...
virtual bool	**HasMethodObject** (const std::wstring &methodObjName) const
virtual bool	**HasMethodObject** (const std::string &methodObjName) const Query if 'methodObjName' is a valid methodObj. More...
virtual bool	**HasProperty** (const std::wstring &propertyName) const
virtual bool	**HasProperty** (const std::string &propertyName) const =0 Query if 'propertyName' is a valid property. More...
virtual bool	**HasProperty** (int idx) const =0 Query if the property at "idx" exists. More...
virtual **JSAPIPtr**	**GetMethodObject** (const std::wstring &methodObjName)
virtual **JSAPIPtr**	**GetMethodObject** (const std::string &methodObjName) Gets a method object (**JSAPI** object that has a default method) More...
virtual **variant**	**GetProperty** (const std::wstring &propertyName)
virtual **variant**	**GetProperty** (const std::string &propertyName)=0

Gets a property value. More...

virtual void **SetProperty** (const std::wstring &propertyName, const **variant** &value)

virtual void **SetProperty** (const std::string &propertyName, const **variant** &value)=0
Sets the value of a property. More...

virtual **variant** **GetProperty** (int idx)=0
Gets the value of an indexed property. More...

virtual void **SetProperty** (int idx, const **variant** &value)=0
Sets the value of an indexed property. More...

virtual void **RemoveProperty** (const std::wstring &propertyName)

virtual void **RemoveProperty** (const std::string &propertyName)=0
Removes a property. More...

virtual void **RemoveProperty** (int idx)=0
Removes an indexed property. More...

virtual **variant** **Invoke** (const std::wstring &methodName, const std::vector< **variant** > &args)

virtual **variant** **Invoke** (const std::string &methodName, const std::vector< **variant** > &args)=0
Called by the browser to invoke a method on the **JSAPI** object. More...

virtual **variant** **Construct** (const std::vector< **variant** > &args)=0
Called by the browser to construct the **JSAPI** object. More...

Static Public Member Functions

template<class Cont >
static void **GetArrayValues** (const **FB::JSObjectPtr** &src, Cont &dst)
Gets Array values out of src and adds them to the STL container dst. More...

template<class Dict >
static void **GetObjectValues** (const **FB::JSObjectPtr** &src, Dict &dst)
Gets object values out of the javscript object src and adds them to the STL Dict container dst. More...

Detailed Description

Wraps a Javascript Object.

Whenever you access a javascript object on a page, it will be through this object. Because **JSObject** extends **JSAPI**, the API will be the same as a **JSAPI** object other than a few additional methods provided on this object for convenience.

Implementations of **JSObject** are expected to be threadsafe as of 1.3.0. What that usually means is that if you call a method from another thread that it can't be safely run on it will use **BrowserHost::CallOnMainThread** to make the call on the main thread and wait for it to complete. There may be performance issues with this that should be taken into consideration.

See Also
JSAPI
NPObjectAPI
IDispatchAPI

Definition at line **46** of file **JSObject.h**.

The documentation for this class was generated from the following file:

- **JSObject.h**