

class FB JSAPIAuto

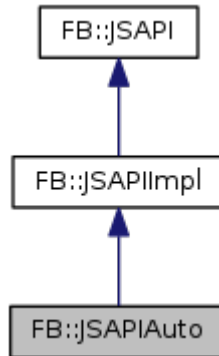
FB::JSAPIAuto Class Reference

Public Member Functions | List of all members

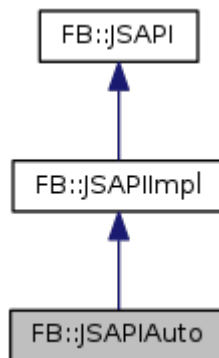
JSAPI class with automatic argument type enforcement. [More...](#)

```
#include "JSAPIAuto.h"
```

Inheritance diagram for FB::JSAPIAuto:



Collaboration diagram for FB::JSAPIAuto:



Public Member Functions

JSAPIAuto (const std::string &description="<JSAPI-Auto Javascript Object>")
Description is used by **ToString()**. [More...](#)

virtual void **registerAttribute** (const std::string &name, const **FB::variant** &value, bool readonly=false)
Registers an attribute *name* and sets the value to *_value*. Optionally read-only. [More...](#)

virtual void **setReserved** (const std::string &name)
Prevents attributes from being created from JavaScript with the specified name. [More...](#)

bool **isReserved** (const std::string &propertyName) const
Returns true if the specified propertyName is a reserved attribute. [More...](#)

virtual void **getMemberNames** (std::vector< std::string > &nameVector) const
Called by the browser to enumerate the members of this **JSAPI** object. [More...](#)

virtual size_t **getMemberCount** () const
Gets the member count. [More...](#)

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

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

virtual **JSAPIPtr** **GetMethodObject** (const std::string &methodObjName)

Gets a method object (**JSAPI** object that has a default method) [More...](#)

virtual void **unregisterMethod** (const std::string &name)
Unregisters a method that has been exposed to javascript. [More...](#)

virtual void **registerMethod** (const std::string &name, const **CallMethodFunc** &func)
Registers a method to be exposed to javascript. [More...](#)

virtual bool **HasMethod** (const std::string &methodName) const
Query if the **JSAPI** object has the 'methodName' method. [More...](#)

virtual bool **HasMethodObject** (const std::string &methodObjName) const
Query if 'methodObjName' is a valid methodObj. [More...](#)

virtual bool **HasProperty** (const std::string &propertyName) const
Query if 'propertyName' is a valid property. [More...](#)

virtual bool **HasProperty** (int idx) const
Query if the property at "idx" exists. [More...](#)

virtual void **registerProperty** (const std::string &name, const **PropertyFuncs** &propFuncs)
Register property to be exposed to javascript. [More...](#)

virtual void **unregisterProperty** (const std::string &name)
Unregisters a property that has been exposed to javascript. [More...](#)

virtual **variant** **GetProperty** (const std::string &propertyName)
Gets a property value. [More...](#)

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

virtual void **RemoveProperty** (const std::string &propertyName)
Removes a property. [More...](#)

virtual **variant** **GetProperty** (int idx)
Gets the value of an indexed property. [More...](#)

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

virtual void **RemoveProperty** (int idx)
Removes an indexed property. [More...](#)

virtual void **FireJSEvent** (const std::string &eventName, const **FB::VariantMap** &members, const **FB::VariantList** &arguments)
Fires an event into javascript asynchronously using a W3C-compliant event parameter. [More...](#)

virtual std::string **ToString** ()
Default method called when a string value is requested for the scriptable object. [More...](#)

virtual bool **get_valid** ()
Property exposed by default to javascript useful for checking to make sure that the **JSAPI** is working. [More...](#)

virtual **FB::variant** **getAttribute** (const std::string &name)
Returns the attribute with the given name, empty variant if none. [More...](#)

virtual void **setAttribute** (const std::string &name, const **FB::variant** &value)
Assigns a value to the specified attribute, if it is not reserved or read-only. [More...](#)

► Public Member Functions inherited from **FB::JSAPIImpl**

JSAPIImpl (void)
Default constructor. [More...](#)

virtual **~JSAPIImpl** (void)

Finaliser. [More...](#)

void **invalidate** ()
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...](#)

virtual **SecurityZone** **getDefaultZone** () const
Gets the default zone (the zone the class operates on before a push) [More...](#)

virtual **SecurityZone** **getZone** () const
Gets the currently active zone. [More...](#)

virtual void **registerEvent** (const std::string &name)
Register event so that event listeners can be added/attached from javascript. [More...](#)

virtual void **registerEvent** (const std::wstring &name)

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...](#)

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

JSAPI (void)
Default constructor. [More...](#)

virtual **~JSAPI** (void)
Finaliser. [More...](#)

virtual bool **HasMethod** (const std::wstring &methodName) const

virtual bool **HasMethodObject** (const std::wstring &methodObjName) const

virtual bool **HasProperty** (const std::wstring &propertyName) const

virtual **JSAPIPtr** **GetMethodObject** (const std::wstring &methodObjName)

virtual **variant** **GetProperty** (const std::wstring &propertyName)

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

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

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

Additional Inherited Members

► Protected Member Functions inherited from **FB::JSAPIImpl**

```
virtual void FireEvent (const std::wstring &eventName, const std::vector< variant > &args)
```

```
virtual void FireEvent (const std::string &eventName, const std::vector< variant > &args)  
Fires an event into javascript asynchronously. More...
```

```
virtual void FireJSEvent (const std::string &eventName, const FB::VariantMap &params)
```

```
virtual void FireJSEvent (const std::string &eventName, const FB::VariantList &arguments)
```

Detailed Description

JSAPI class with automatic argument type enforcement.

Most of the time when implementing a **JSAPI** class for your plugin you will want to extend **JSAPIAuto**. **JSAPIAuto** implements all abstract functions from **JSAPI** and provides an automatic function map and type conversion for methods that you create.

For example, to provide a property called "IsFinished" that javascript can access, you need two methods:

```
bool get_IsFinished();  
void set_IsFinished(bool newVal);
```

Then in the constructor of your class that extends **JSAPIAuto** (we'll call it MyPluginAPI), you register it like so:

```
registerProperty("IsFinished",  
make\_property(this,  
&MyPluginAPI::get_IsFinished,  
&MyPluginAPI::set_IsFinished));
```

To register a property that is read-only, use the second form:

```
registerProperty("IsFinished", make\_property(this, &MyPluginAPI::get_IsFinished));
```

Similarly, to provide a method called "add" that accepts two integers and returns an integer, you need one method on your class:

```
int add(int a, int b);
```

Then in the constructor of your class that extends **JSAPIAuto** (still calling it MyPluginAPI), you register it like so:

```
registerMethod("add", make\_method(this, &MyPluginAPI::add));
```

If arguments are passed that cannot be converted to an int, a javascript exception will be thrown.

See Also

[JSAPI](#)
[PluginCore](#)

Author

Georg Fritzsche

Definition at line 94 of file [JSAPIAuto.h](#).

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

- [JSAPIAuto.h](#)
- [JSAPIAuto.cpp](#)