JSClient

Constants Summary

String #CALLBACK_EVENT
String #CALLBACK_EXCEPTION_EVENT

Method Summary

String	#getClientID() gets the id of the client
Object	#getDataProviderValue(context, variableName, [currentMethodName]) get a dataprovider value from the client
Boolean	#isValid() returns true if this client is still valid/usable
void	#queueMethod(context, methodName, args, callbackFunction) queue a method on the client, calling the method name specified on the context, the callback method will get a JSEvent as the first and a JSClient (the this of the client that did the call) as the second parameter
Object	#setDataProviderValue(context, variableName, value, [currentMethodName]) set a dataprovider value on the client
void	#shutdown(force) shutdown

Constants Details

CALLBACK_EVENT
Replace with description
Returns
String

CALLBACK_EXCEPTION_EVENT
Replace with description
Returns
String

Method Details

getClientID
String **getClientID**()
gets the id of the client
Returns
String
Sample

```
if (jsclient && jsclient.isValid())
{
	/*Queue a method where the callback can do something like this
	if (event.getType() == JSClient.CALLBACK_EVENT)
	{
		application.output("callback data, name: " + event.data);
	}
	else if (event.getType() == JSClient.CALLBACK_EXCEPTION_EVENT)
	{
		application.output("exception callback, name: " + event.data);
	}*/
	var x = new Object();
	x.name = 'remote1';
	x.number = 10;
	// this calls a 'remoteMethod' on the server as a global method, because the context (first argument is
set to null), you can use a formname to call a form method
	jsclient.queueMethod(null, "remoteMethod", [x], callback);
}
```

getDataProviderValue

Object **getDataProviderValue**(context, variableName, [currentMethodName])
get a dataprovider value from the client

Parameters
context
variableName
[currentMethodName]

Returns
Object

Sample

```
if (jsclient && jsclient.isValid())
{
	// only gets the globals.media when the 'remoteMethod' is currently executing for this client
	var value = jsclient.getDataProviderValue(null, "globals.number", 'remoteMethod');
	if (value != null)
	{
		application.output("value get from globals.number :: "+ value);
		globals.value = value+10;
		var returnValue = jsclient.setDataProviderValue(null, "globals.number", globals.value,
'remoteMethod');
		application.output("value set to globals.number previous value "+ returnValue);
	}
	else
	{
		application.output("value get from globals.number :: " + null);
	}
}
```

isValid

Boolean **isValid**()
returns true if this client is still valid/usable

Returns
Boolean

Sample

```
if (jsclient && jsclient.isValid())
{
	/*Queue a method where the callback can do something like this
	if (event.getType() == JSClient.CALLBACK_EVENT)
	{
		application.output("callback data, name: " + event.data);
	}
	else if (event.getType() == JSClient.CALLBACK_EXCEPTION_EVENT)
	{
		application.output("exception callback, name: " + event.data);
	}*/
	var x = new Object();
	x.name = 'remote1';
	x.number = 10;
	// this calls a 'remoteMethod' on the server as a global method, because the context (first argument is
set to null), you can use a formname to call a form method
	jsclient.queueMethod(null, "remoteMethod", [x], callback);
}
```

queueMethod

void **queueMethod**(context, methodName, args, callbackFunction)
queue a method on the client, calling the method name specified on the context, the callback method will get a JSEvent as the first and a JSClient (the this of the client that did the call) as the second parameter

Parameters
context
methodName
args
callbackFunction

Returns
void

Sample

```
if (jsclient && jsclient.isValid())
{
	/*Queue a method where the callback can do something like this
	if (event.getType() == JSClient.CALLBACK_EVENT)
	{
		application.output("callback data, name: " + event.data);
	}
	else if (event.getType() == JSClient.CALLBACK_EXCEPTION_EVENT)
	{
		application.output("exception callback, name: " + event.data);
	}*/
	var x = new Object();
	x.name = 'remote1';
	x.number = 10;
	// this calls a 'remoteMethod' on the server as a global method, because the context (first argument is
set to null), you can use a formname to call a form method
	jsclient.queueMethod(null, "remoteMethod", [x], callback);
}
```

setDataProviderValue

Object **setDataProviderValue**(context, variableName, value, [currentMethodName])

set a dataprovider value on the client

Parameters

context
variableName
value
[currentMethodName]

Returns

Object

Sample

```
if (jsclient && jsclient.isValid())
{
	// only gets the globals.media when the 'remoteMethod' is currently executing for this client
	var value = jsclient.getDataProviderValue(null, "globals.number", 'remoteMethod');
	if (value != null)
	{
		application.output("value get from globals.number :: "+ value);
		globals.value = value+10;
		var returnValue = jsclient.setDataProviderValue(null, "globals.number", globals.value,
'remoteMethod');
		application.output("value set to globals.number previous value "+ returnValue);
	}
	else
	{
		application.output("value get from globals.number :: " + null);
	}
}
```

shutdown

void **shutdown**(force)

shutdown

Parameters

force

Returns

void

Sample

```
if (jsclient && jsclient.isValid())
{
	/*Queue a method where the callback can do something like this
	if (event.getType() == JSClient.CALLBACK_EVENT)
	{
		application.output("callback data, name: " + event.data);
	}
	else if (event.getType() == JSClient.CALLBACK_EXCEPTION_EVENT)
	{
		application.output("exception callback, name: " + event.data);
	}*/
	var x = new Object();
	x.name = 'remote1';
	x.number = 10;
	// this calls a 'remoteMethod' on the server as a global method, because the context (first argument is
set to null), you can use a formname to call a form method
	jsclient.queueMethod(null, "remoteMethod", [x], callback);
}
```