JSClient

Constants Summary

String	CALLBACK_EVENT Constant that is returned as a JSEvent type when in the callback method when it executed normally.
String	CALLBACK_EXCEPTION_EVENT Constant that is returned as a JSEvent type when in the callback method when an exception occurred.

Method Summary

String	getClientID() Gets the id of the client.
Object	getDataProviderValue(contextName, dataprovider) Get a data-provider value.
Object	getDataProviderValue(contextName, dataprovider, methodName) Get a data-provider value.
Boolean	isValid() returns true if this client is still valid/usable.
void	queueMethod(contextName, methodName, args, notifyCallBackMethod) Queues a method call on the remote server.
Object	setDataProviderValue(contextName, dataprovider, value) Set a data-provider value.
Object	setDataProviderValue(contextName, dataprovider, value, methodName) Set a data-provider value.
void	shutdown() closes the client.
void	shutdown(force) closes the client.

Constants Details

CALLBACK_EVENT

Constant that is returned as a JSEvent type when in the callback method when it executed normally.

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);
}
```

CALLBACK_EXCEPTION_EVENT

Constant that is returned as a JSEvent type when in the callback method when an exception occurred.

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);
}
```

Method Details

getClientID

String **getClientID** ()

Gets the id of the client.

This client id can be used to find the client from the headless client plugin.
Note that this client id is not the same id as the id displayed on the Aplicationb Server admin page.

Returns

String

Sample

```
var headlessClient = plugins.headlessclient.createClient("someSolution", "user", "pass", null);
var clientID = headlessClient.getClientID()
....
headlessClient = plugins.headlessclient.getClient(clientID);
if (headlessClient != null && headlessClient.isValid()) {
         headlessClient.queueMethod(null, "someRemoteMethod", null, callback);
}
```

getDataProviderValue

Object **getDataProviderValue** (contextName, dataprovider)

Get a data-provider value.

Parameters

{String} contextName - The context of the given method, null if it is global method or a form name for a form method
{String} dataprovider - the data-provider name as seen in Servoy

Returns

Object - the value for the data-provider.

Sample

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

getDataProviderValue

Object **getDataProviderValue** (contextName, dataprovider, methodName)

Get a data-provider value.

Parameters

{String} contextName - The context of the given method; null if it is global method or a form name for a form method.
{String} dataprovider - the data-provider name as seen in Servoy.
{String} methodName - if this is specified, the data-provider's value will only be returned if the specified method is running in this headless client because the currently running client requested it to. Otherwise undefined is returned.

Returns

Object - the value of the data-provider.

Sample

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

Queues a method call on the remote server. The callback method will be called when the method is executed on the server
and the return value is given as the JSEvent.data object with the JSEvent.getType() value of JSClient.CALLBACK_EVENT.
If an exception is thrown somewhere then the callback method will be called with
the exception as the JSEvent data object with the JSEvent.getType() value of JSClient.CALLBACK_EXCEPTION_EVENT
The second argument that is give back is the JSClient instance that did the call.

Parameters

{String} contextName - The context of the given method, null if it is global method or a form name for a form method.
{String} methodName - The method name.
{Object[]} args - The arguments that should be passed to the method.
{Function} notifyCallBackMethod - The callback method that is called when the execution is finished.

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** (contextName, dataprovider, value)

Set a data-provider value.

Parameters

{String} contextName - The context of the given method, null if it is global method or a form name for a form method.
{String} dataprovider - the data-provider name as seen in Servoy.
{Object} value - the value to set.

Returns

Object - the old value or null if no change.

Sample

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

setDataProviderValue

Object **setDataProviderValue** (contextName, dataprovider, value, methodName)

Set a data-provider value.

Parameters

{String} contextName - The context of the given method, null if it is global method or a form name for a form method
{String} dataprovider - the data-provider name as seen in Servoy
{Object} value - the value to set
{String} methodName - if this is specified, the data-provider's value will only be set if the specified method is running in this headless client because the currently running client requested it to. Otherwise the value is not set into the data-provider and undefined is returned.

Returns

Object - the old value or null if no change

Sample

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

shutdown

void **shutdown** ()

closes the client.

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);
}
```

shutdown

void **shutdown** (force)

closes the client.

Parameters

{Boolean} 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);
}
```