JavaScript Jitterbit and Common Functions

Jitterbit JavaScripts can access—in addition to the standard JavaScript functions that are part of ECMA-262 v5.1—these Jitterbit-specific functions. As ECMA-262 v5.1 is an older version of JavaScript than what is commonly available in browsers and other JavaScript engines, we've included tables of the common and standard JavaScript functions that are available in Jitterbit JavaScripts.

NOTE: A maximum of 50,000 loop iterations is allowed for each JavaScript script. To increase the allowed number of iterations per script, see Loop Iterations under JavaScript.

Search in This Topic

Related Articles

- JavaScript
- JavaScript Standard Properties and Functions

Related Topics

- Cloud Studio
- Functions
- Scripts

Last updated: Jun 05, 2019

Jitterbit Functions

Jitterbit.DbExecute

Declaration

```
array Jitterbit.DbExecute(string databaseId, string sql)

int Jitterbit.DbExecute(string databaseId, string sql, string
outputVariable,...)
```

Syntax

```
Jitterbit.DbExecute(<databaseId>, <sql>)

Jitterbit.DbExecute(<databaseId>, <sql>, <outputVariable>,...])
```

Required Parameters

- **`databaseId`:** A string reference path to a database connection in the current project
- **`sql`:** The SQL command to be executed against the database
- **`outputVariable`:** (Second form) An output parameter that is matched to fields returned in the SQL command. Additional arguments can be specified as required.

Description

Executes a SQL statement on a database and returns the results. See the Jitterbit Script `DBExecute()` function for details.

The database used in this function call must be defined as a database connection in the current project. For more information, see the instructions on inserting endpoints under Endpoints in JavaScript.

[Back to Top]

Jitterbit.DbLookup

Declaration

```
string Jitterbit.DbLookup(string databaseId, string sql)
```

Syntax

```
Jitterbit.DbLookup(<databaseId>, <sql>)
```

Required Parameters

- **`databaseId`:** A string reference path to a database connection in the current project

- **`sql`:** The SQL command to be executed against the database

Description

Executes a SQL statement on a database and returns the first field of the first result matching the specified criteria. See the Jitterbit Script `DBLookup()` function for details.

The database used in this function call must be defined as a database connection in the current project. For more information, see the instructions on inserting endpoints under Endpoints in JavaScript.

[Back to Top]

Jitterbit.GetVar

Declaration

```
string Jitterbit.GetVar(string jitterbitVariableName)
```

Syntax

```
Jitterbit.GetVar(<jitterbitVariableName>)
```

Required Parameters

- **`jitterbitVariableName`:** The name of a Jitterbit variable or global variable

Description

Returns the value of either a Jitterbit variable (the predefined global variables that are built into Jitterbit Harmony and begin with "`$jitterbit.`") or another global variable. The dollar symbol is optional and can be omitted. The available variables can be seen in Cloud Studio and are documented under Jitterbit Variables. Global variables defined within the current project can also be retrieved by the same mechanism.

Examples

```
// Retrieves the value of the Jitterbit variable "jitterbit.api.request"
var request = Jitterbit.GetVar("$jitterbit.api.request");

// Retrieves the value of the global variable "email"
var email = Jitterbit.GetVar("email");

// Another method for retrieving a global variable
var email = Jitterbit.GetVar("$email");
```

[Back to Top]

Jitterbit.ReadFile

Declaration

```
string Jitterbit.ReadFile(string sourceId[, string fileFilter])
```

Syntax

```
Jitterbit.ReadFile(<sourceId>[, <fileFilter>])
```

Required Parameters

- **`sourceId:`** A string reference path to an activity associated with a file-type endpoint in the current project

Optional Parameters

- **`fileFilter:`** File filter or filename to override the activity configuration

Description

Reads the contents of a file from a source.

The source used in this function call must be defined as an activity associated with a file-type endpoint in the current project. These include configured file share, FTP, HTTP, local storage, and temporary storage activities. For more information, see the instructions on inserting endpoints under Endpoints in JavaScript.

The method returns the contents of the file pointed to by the source. If the source filter selects more than one file, the first one will be used. It is recommended to specify a source that uniquely identifies a single file.

The parameter `fileFilter` is optional and can be used to override the file filter used in the activity configuration. A filename can be used. Alternatively, a global variable can be used to override the file filter in the activity configuration. Global variables are referenced as `[de_name]` in the activity configuration.

If the `ReadFile()` function fails, the operation does not fail. A script will abort, a warning added to the operation log, and the operation will continue.

This method can be used to read data from an HTTP source. In that case, all Jitterbit `$jitterbit.source.http.*` variables will be populated.

> **WARNING:** This function doesn't work reliably with files that have binary content, as it will usually read only a portion of the file. If the file has binary content, use the Jitterbit Script function `Base64EncodeFile()` instead to read the entire file contents.

Examples

```
// Reads the first file retrieved from a source
var fileContents = Jitterbit.ReadFile("<TAG>activity:ftp/FTP Endpoint
/ftp_read/FTP Files</TAG>");

// Use the filename "test.txt" instead of what is defined in the source
var fileContents = Jitterbit.ReadFile("<TAG>activity:ftp/FTP Endpoint
/ftp_read/FTP Files</TAG>", "test.txt");
```

[Back to Top]

Jitterbit.SetVar

Declaration

```
Jitterbit.SetVar(string jitterbitVariableName, string value)
```

Syntax

```
Jitterbit.SetVar(<jitterbitVariableName>, <value>)
```

Required Parameters

- **`jitterbitVariableName:`** The name of a Jitterbit variable or global variable
- **`value:`** A value to be assigned to the variable

Description

Sets the value of a Jitterbit variable (the predefined global variables that are built into Jitterbit and begin with "`$jitterbit.`") or another global variable. The dollar symbol is optional and can be omitted. The available variables can be seen in Cloud Studio and are documented under Jitterbit Variables. Global variables defined within the current project can also be retrieved by the same mechanism.

Examples

```
// Sets a Jitterbit variable and a global variable
Jitterbit.SetVar("$jitterbit.scripting.db.max_rows", 1000);
Jitterbit.SetVar("$email", "first.lastname@example.com");
```

[Back to Top]

Jitterbit.WriteFile

Declaration

```
void Jitterbit.WriteFile(string targetId, type fileContents[, string
filename])
```

Syntax

```
Jitterbit.WriteFile(<targetId>, <fileContents>[, <filename>])
```

Required Parameters

- **`targetId:`** A string reference path to an activity associated with a file-type endpoint in the current project

Optional Parameters

- **`fileContents:`** Data to be written to the file
- **`filename:`** Filename to override the activity configuration

Description

Writes the `fileContents` to the target specified by `targetId`. If `fileContents` is of type binary, the binary data is written to the file. In all other cases, a string representation of the data is written.

The target used in this function call must be defined as an activity associated with a file-type endpoint in the current project. These include configured file share, FTP, HTTP, local storage, and temporary storage activities. For more information, see the instructions on inserting endpoints under Endpoints in JavaScript.

The third parameter, `filename`, is optional and can be used to override the filename used in the activity configuration. Alternatively, a global variable can be used to override the filename in the activity configuration. Global variables are referenced as `[de_name]` in the activity configuration.

This method can also be used to write/post data to an HTTP target. In that case, `$jitterbit.target.http.*` variables will be populated.

If the `WriteFile()` function fails, the operation does not fail. A script will abort, a warning added to the operation log, and the operation will continue.

Examples

```
//  Writes the value of $contents to the file defined by a target
Jitterbit.WriteFile("<TAG>activity:ftp/FTP Endpoint/ftp_write/FTP Files<
/TAG>", $contents);

// Use the filename "test.txt" instead of what is defined in the target
Jitterbit.WriteFile("<TAG>activity:ftp/FTP Endpoint/ftp_write/FTP Files<
/TAG>", $contents, "test.txt");
```

[Back to Top]

SetScriptOutput

Declaration

```
void SetScriptOutput(string data)
```

Syntax

```
SetScriptOutput(<data>)
```

Required Parameters

- **`data:`** Data to be returned by the script

Description

Sets the value returned by a script.

The return value of a script can be accessed as the return value of the `RunScript()` function.

Examples

```
// In the calling Jitterbit Script script:
$result = RunScript("<TAG>script:CalculateSomething</TAG>",
value_to_be_calculated);

// In the Jitterbit JavaScript script "CalculateSomething":
var calculated_value = ...;
...
SetScriptOutput(calculated_value);
```

[Back to Top]

SetScriptResult

Declaration

```
void SetScriptResult(string data)
```

Syntax

```
SetScriptResult(<data>)
```

Required Parameters

- **`data:`** Data to be returned by the script

Description

An alias for the `SetScriptOutput()` function. See `SetScriptOutput()` for details.

[Back to Top]

WriteToOperationLog

Declaration

```
string WriteToOperationLog(string message)
```

Syntax

```
WriteToOperationLog(<message>)
```

Required Parameters

- **`message:`** A string message

Description

Writes a message to the operation log.

Examples

```
// Writes a message to the operation log
WriteToOperationLog("The source field 'Price' has an invalid value.");
```

[Back to Top]

JavaScript Common Functions

These common JavaScript functions—part of ECMA-262 v5.1—are supported in Jitterbit JavaScripts.

Common Property	Description
`Array`	See the JavaScript Array properties and functions
`Date`	See the JavaScript Date properties and functions
`decodeURI()`	Decodes a URI
`decodeURIComponent()`	Decodes a URI component
`encodeURI()`	Encodes a URI
`encodeURIComponent()`	Encodes a URI component
`eval()`	Evaluates a string and executes it as if it were JavaScript code
`isFinite()`	Tests if a value is a finite and legal number
`isNaN()`	Tests if a value is an illegal number
`JSON.parse()`	Parses a JSON string and returns a JavaScript object
`JSON.stringify()`	Converts a JavaScript object into a JSON string
`Math`	See the JavaScript Math properties and functions
`Number()`	Converts an object's value into a number
`parseFloat()`	Parses a string and returns a floating point number
`parseInt()`	Parses a string and returns an integer
`String`	See the JavaScript String properties and functions

[Back to Top]