



Shabti Javascript Reference Guide

<https://github.com/danhetrick/shabti>

Table of Contents

IRC Events

CONNECT_EVENT	5
NICK_TAKEN_EVENT	5
PING_EVENT	5
TIME_EVENT	6
PUBLIC_MESSAGE_EVENT	7
PRIVATE_MESSAGE_EVENT	8
ACTION_EVENT	8
JOIN_EVENT	9
PART_EVENT	9
IRC_EVENT	11

<i>Built-In Variables</i>	4
----------------------------------	---

Built-In Functions

raw	12
set	12
login	12
nick	12
rnick	12
join	13
part	13
topic	13
quit	13
notice	13
message	13

action	13
users	14
print	15
sprint	15
color	15
bold	15
underline	15
read	16
write	16
swrite	16
append	16
sappend	16
fileexists	16
direxists	17
mkdir	17
rmdir	17
delete	17
sha1	18
sha256	18
exit	18
tokens	18

Built-In Variables

SV_SERVER	The name/host of the IRC server connected to.
SV_PORT	The IRC server port connected to.
SV_NICK	The bot's nick.
SV_USER	The bot's username.
SV_IRCNAME	The bot's IRCname.
SV_TIME	Server time.
SV_DATE	Server date.
SV_BOT	The bot's software name.
SV_VERSION	The bot's software version.
SV_LOCAL_DIRECTORY	The directory where Shabti is installed.
SV_CONFIG_DIRECTORY	The configuration directory Shabti is using.
WHITE	White color *
BLACK	Black color *
BLUE	Blue color *
GREEN	Green color *
RED	Red color *
BROWN	Brown color *
PURPLE	Purple color *
ORANGE	Orange color *
YELLOW	Yellow color *
LIGHT_GREEN	Light green color *
TEAL	Teal color *
CYAN	Cyan color *
LIGHT_BLUE	Light blue color *
PINK	Pink color *
GREY	Grey color *
LIGHT_GREY	Light color *

**** For use with the “color” function***

IRC Events

CONNECT_EVENT	
<i>Arguments</i>	<code>EV_HOST</code> <i>The name of the server connected to</i>
<i>Notes</i>	Shabti will call "extra" connect event functions. Their names follow the format <code>CONNECT_EVENT_x</code> , with "x" equal to a number; with default settings, Shabti will call <code>CONNECT_EVENT_1</code> to <code>CONNECT_EVENT_10</code> . How many "extra" functions are called can be set with the command line option <code>--extra</code> ; for example, if you started Shabti with <code>perl shabti.pl --extra 50</code> , it would then call event functions <code>CONNECT_EVENT_1</code> to <code>CONNECT_EVENT_50</code> whenever it receives a connect message.
<i>Description</i>	Called when the bot connects to the IRC server.

NICK_TAKEN_EVENT	
<i>Arguments</i>	None.
<i>Notes</i>	None.
<i>Description</i>	Called when the bot is notified by the server that their requested nick is taken. By default, this is <i>not</i> handled automatically, and must be handled by the script. The script that comes with a default Shabti installation calls the built-in function <code>rnick</code> to handle this problem.

PING_EVENT	
<i>Arguments</i>	None.
<i>Notes</i>	None.
<i>Description</i>	Called when the bot receives a "PING?" request from the server. The necessary response to maintain connectivity is handled automatically.

TIME_EVENT	
<i>Arguments</i>	<p>EV_WEEKDAY <i>The day of the week.</i></p> <p>EV_MONTH <i>The month of the year.</i></p> <p>EV_DAY <i>The numeric day of the month.</i></p> <p>EV_YEAR <i>The year.</i></p> <p>EV_HOUR <i>The hour of the day.</i></p> <p>EV_MINUTE <i>The minute of the hour.</i></p> <p>EV_SECOND <i>The second of the minute.</i></p> <p>EV_ZONE <i>Time zone.</i></p>
<i>Notes</i>	None.
<i>Description</i>	Called when the bot receives a RPL_TIME message from the server. Shabti can ask the server to send a RPL_TIME message by using the built-in function raw with the argument "TIME" (raw("TIME") ;).

PUBLIC_MESSAGE_EVENT	
<i>Arguments</i>	<p><code>EV_NICK</code> <i>Nick of the message sender.</i></p> <p><code>EV_USERNAME</code> <i>Username of the message sender.</i></p> <p><code>EV_CHANNEL</code> <i>The channel the message was sent from.</i></p> <p><code>EV_MESSAGE</code> <i>The message sent.</i></p>
<i>Notes</i>	<p>Shabti will call "extra" public message event functions. Their names follow the format <code>PUBLIC_MESSAGE_EVENT_x</code>, with "x" equal to a number; with default settings, Shabti will call <code>PUBLIC_MESSAGE_EVENT_1</code> to <code>PUBLIC_MESSAGE_EVENT_10</code>. How many "extra" functions are called can be set with the command line option <code>--extra</code>; for example, if you started Shabti with <code>perl shabti.pl --extra 50</code>, it would then call event functions <code>PUBLIC_MESSAGE_EVENT_1</code> to <code>PUBLIC_MESSAGE_EVENT_50</code> whenever it receives a public message.</p>
<i>Description</i>	Called when the bot receives a public message.

PRIVATE_MESSAGE_EVENT	
<i>Arguments</i>	<p>EV_NICK</p> <p><i>The day of the week.</i></p> <p>EV_USERNAME</p> <p><i>The month of the year.</i></p> <p>EV_MESSAGE</p> <p><i>The year.</i></p>
<i>Notes</i>	<p>Shabti will call "extra" private message event functions. Their names follow the format PRIVATE_MESSAGE_EVENT_x, with "x" equal to a number; with default settings, Shabti will call PRIVATE_MESSAGE_EVENT_1 to PRIVATE_MESSAGE_EVENT_10. How many "extra" functions are called can be set with the command line option --extra; for example, if you started Shabti with perl shabti.pl --extra 50, it would then call event functions PRIVATE_MESSAGE_EVENT_1 to PRIVATE_MESSAGE_EVENT_50 whenever it receives a public message.</p>
<i>Description</i>	Called when the bot receives a private message.

ACTION_EVENT	
<i>Arguments</i>	<p>EV_NICK</p> <p><i>Nick of the action sender.</i></p> <p>EV_USERNAME</p> <p><i>Username of the action sender.</i></p> <p>EV_CHANNEL</p> <p><i>The channel the action was sent from.</i></p> <p>EV_ACTION</p> <p><i>The action sent.</i></p>
<i>Notes</i>	None.
<i>Description</i>	Called when the bot receives a CTCP action message.

MODE_EVENT	
<i>Arguments</i>	<p><code>EV_NICK</code> <i>Nick of the mode setter.</i></p> <p><code>EV_USERNAME</code> <i>Username of the mode setter.</i></p> <p><code>EV_TARGET</code> <i>The mode's target.</i></p> <p><code>EV_MODE</code> <i>The mode set.</i></p>
<i>Notes</i>	None.
<i>Description</i>	Called when the bot receives a mode notification. If the server is the one setting the mode, the <code>EV_USERNAME</code> argument will be an empty string.

JOIN_EVENT	
<i>Arguments</i>	<p><code>EV_NICK</code> <i>Nick of the joiner.</i></p> <p><code>EV_USERNAME</code> <i>Username of the joiner.</i></p> <p><code>EV_CHANNEL</code> <i>The channel joined.</i></p>
<i>Notes</i>	Shabti will call "extra" join event functions. Their names follow the format <code>JOIN_EVENT_x</code> , with "x" equal to a number; with default settings, Shabti will call <code>JOIN_MESSAGE_EVENT_1</code> to <code>JOIN_MESSAGE_EVENT_10</code> . How many "extra" functions are called can be set with the command line option <code>--extra</code> ; for example, if you started Shabti with <code>perl shabti.pl --extra 50</code> , it would then call event functions <code>JOIN_MESSAGE_EVENT_1</code> to <code>JOIN_MESSAGE_EVENT_50</code> whenever it receives a join message.
<i>Description</i>	Called when the bot receives a channel join message.

<code>PART_EVENT</code>	
<i>Arguments</i>	<code>EV_NICK</code> <i>Nick of the user leaving.</i> <code>EV_USERNAME</code> <i>Username of the user leaving.</i> <code>EV_CHANNEL</code> <i>The channel being left.</i> <code>EV_MESSAGE</code> <i>Optional parting message</i>
<i>Notes</i>	Shabti will call "extra" part event functions. Their names follow the format <code>PART_EVENT_x</code> , with "x" equal to a number; with default settings, Shabti will call <code>PART_MESSAGE_EVENT_1</code> to <code>PART_MESSAGE_EVENT_10</code> . How many "extra" functions are called can be set with the command line option <code>--extra</code> ; for example, if you started Shabti with <code>perl shabti.pl --extra 50</code> , it would then call event functions <code>PART_MESSAGE_EVENT_1</code> to <code>PART_MESSAGE_EVENT_50</code> whenever it receives a part message.
<i>Description</i>	Called when the bot receives a channel part notification. If the parting user has set a parting message, it will be reflected in the <code>EV_MESSAGE</code> argument, which is set to a blank string otherwise.

IRC_EVENT	
<i>Arguments</i>	<p><code>EV_RAW</code></p> <p><i>The unchanges text of the message sent by the server.</i></p> <p><code>EV_TYPE</code></p> <p><i>The numeric message type, according to RFCs</i></p> <p><code>EV_HOST</code></p> <p><i>The sending server.</i></p> <p><code>EV_NICK</code></p> <p><i>The nick the message was sent to.</i></p> <p><code>EV_MESSAGE</code></p> <p><i>The message content.</i></p>
<i>Notes</i>	<p>Shabti will call "extra" IRC event functions. Their names follow the format <code>IRC_MESSAGE_EVENT_x</code>, with "x" equal to a number; with default settings, Shabti will call <code>IRC_MESSAGE_EVENT_1</code> to <code>IRC_MESSAGE_EVENT_10</code>. How many "extra" functions are called can be set with the command line option <code>--extra</code>; for example, if you started Shabti with <code>perl shabti.pl --extra 50</code>, it would then call event functions <code>IRC_MESSAGE_EVENT_1</code> to <code>IRC_MESSAGE_EVENT_50</code> whenever it receives an IRC event.</p>
<i>Description</i>	<p>Called when the bot receives a notification that is not handled by any other event. <code>EV_RAW</code> contains the "raw", unchanged notification.</p>

Built-in IRC Functions

raw

<i>Arguments</i>	1 (text to send)
<i>Returns</i>	Nothing
<i>Description</i>	Sends “raw” text to the IRC server; that is, the bot will send the server this text without any modification. This can be used to send IRC commands that don’t have Shabti built-in functions to perform. For example, to send a private message to Bob, you could use <code>raw("PRIVMSG Bob :Hello world!")</code> .

set

<i>Arguments</i>	2+ (targets, flags, optional arguments)
<i>Returns</i>	Nothing
<i>Description</i>	Sets a mode on the server. For example, to give channel operator status to Bob in channel “#foo”, you could use <code>set("#foo", "+o", "Bob")</code> .

login

<i>Arguments</i>	2 (username, password)
<i>Returns</i>	Nothing
<i>Description</i>	Logs into an IRCop account.

nick

<i>Arguments</i>	1 (new nick)
<i>Returns</i>	Nothing
<i>Description</i>	Changes the bot’s nick.

rnick

<i>Arguments</i>	1 (new nick)
<i>Returns</i>	Nothing
<i>Description</i>	Changes the bot’s nick, adding two numbers to the end of the nick.

join

<i>Arguments</i>	1+ (channel to join, optional password)
<i>Returns</i>	Nothing
<i>Description</i>	Joins a channel.

part

<i>Arguments</i>	1+ (channel to part, optional parting message)
<i>Returns</i>	Nothing
<i>Description</i>	Parts a channel.

topic

<i>Arguments</i>	2 (channel, new topic)
<i>Returns</i>	Nothing
<i>Description</i>	Sets a channel's topic.

quit

<i>Arguments</i>	0+ (optional quit message)
<i>Returns</i>	Nothing
<i>Description</i>	Quits the IRC server.

message

<i>Arguments</i>	2 (target user or channel, message)
<i>Returns</i>	Nothing
<i>Description</i>	Sends a message to the target user or channel. An identical version of this command named <code>msg</code> can alternately used.

notice

<i>Arguments</i>	2 (target user or channel, message)
<i>Returns</i>	Nothing
<i>Description</i>	Sends a notice to the target user or channel.

action

<i>Arguments</i>	2 (channel, action)
<i>Returns</i>	Nothing
<i>Description</i>	Sends an action message to a channel.

users

<i>Arguments</i>	1 (channel)
<i>Returns</i>	Array
<i>Description</i>	Returns an array of users in a channel. The bot must be in the channel to return a list; otherwise, an empty array is returned.

Text Functions

print

<i>Arguments</i>	1+ (text to print)
<i>Returns</i>	Nothing
<i>Description</i>	Prints text to the console, followed by a carriage return.

sprint

<i>Arguments</i>	1+ (text to print)
<i>Returns</i>	Nothing
<i>Description</i>	Prints text to the console; a trailing carriage return is <i>not</i> printed.

color

<i>Arguments</i>	3 (foreground color, background color, text)
<i>Returns</i>	string
<i>Description</i>	Formats text using IRC color codes, and returns it.

bold

<i>Arguments</i>	1 (text)
<i>Returns</i>	string
<i>Description</i>	Formats text using IRC bold code, and returns it.

italic

<i>Arguments</i>	1 (text)
<i>Returns</i>	string
<i>Description</i>	Formats text using IRC italic code, and returns it.

underline

<i>Arguments</i>	1 (text)
<i>Returns</i>	string
<i>Description</i>	Formats text using IRC underline code, and returns it.

File I/O Functions

read

<i>Arguments</i>	1 (file to read)
<i>Returns</i>	string
<i>Description</i>	Reads data from a file and returns it.

write

<i>Arguments</i>	2 (filename, contents)
<i>Returns</i>	Nothing
<i>Description</i>	Writes data to a file, followed by a carriage return.

swrite

<i>Arguments</i>	2 (filename, contents)
<i>Returns</i>	Nothing
<i>Description</i>	Writes data to a file; a trailing carriage return is <i>not</i> written.

append

<i>Arguments</i>	2 (filename, contents)
<i>Returns</i>	Nothing
<i>Description</i>	Appends data to a file, followed by a carriage return.

sappend

<i>Arguments</i>	2 (filename, contents)
<i>Returns</i>	Nothing
<i>Description</i>	Appends data to a file; a trailing carriage return is <i>not</i> written.

fileexists

<i>Arguments</i>	1 (filename)
<i>Returns</i>	boolean
<i>Description</i>	Tests if a file exists or not.

direxists

<i>Arguments</i>	1 (directory)
<i>Returns</i>	boolean
<i>Description</i>	Tests if a directory exists or not.

mkdir

<i>Arguments</i>	1 (directory name)
<i>Returns</i>	Nothing
<i>Description</i>	Creates a directory.

rmdir

<i>Arguments</i>	1 (directory name)
<i>Returns</i>	Nothing
<i>Description</i>	Deletes a directory.

delete

<i>Arguments</i>	1 (filename)
<i>Returns</i>	Nothing
<i>Description</i>	Deletes a file.

Miscellaneous Functions

sha1

<i>Arguments</i>	1 (data)
<i>Returns</i>	string
<i>Description</i>	Calculates a SHA1 hash and returns it.

sha256

<i>Arguments</i>	1 (data)
<i>Returns</i>	string
<i>Description</i>	Calculates a SHA256 hash and returns it.

require

<i>Arguments</i>	1 (module name)
<i>Returns</i>	Nothing
<i>Description</i>	Loads a Shabti module into memory.

exit

<i>Arguments</i>	0, 1 (message), or 2 (message, exit code)
<i>Returns</i>	Nothing
<i>Description</i>	Exits out of Shabti . Optionally, can display a message on exit, or an exit code (which <i>must</i> be 0 or 1).

tokens

<i>Arguments</i>	1 (string)
<i>Returns</i>	Array
<i>Description</i>	Tokenizes a string into an array, using space(s) as a delimiter. Quotes can be used to set a token containing whitespace.

