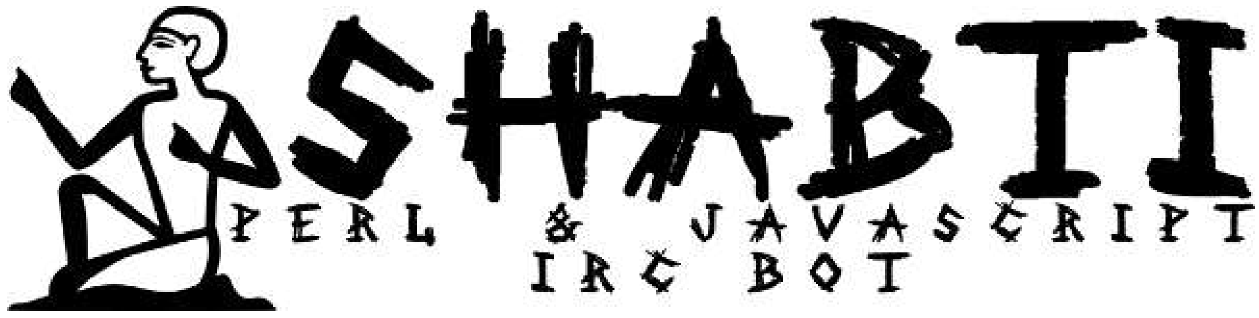

Shabti Javascript Reference Guide

https://github.com/danhetrick/shabti

Table of Contents

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`	
Arguments	`EV_HOST` *The name of the server connected to*
Notes	**Shabti** will call "extra" connect event functions. Their names follow the format `CONNECT_EVENT_x`, with "x" equal to a number; with default settings, **Shabti** will call `CONNECT_EVENT_1` to `CONNECT_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 `CONNECT_EVENT_1` to `CONNECT_EVENT_50` whenever it receives a connect message.
Description	Called when the bot connects to the IRC server.

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

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

`TIME_EVENT`	

Arguments	`EV_WEEKDAY`
	The day of the week.
	`EV_MONTH`
	The month of the year.
	`EV_DAY`
	The numeric day of the month.
	`EV_YEAR`
	The year.
	`EV_HOUR`
	The hour of the day.
	`EV_MINUTE`
	The minute of the hour.
	`EV_SECOND`
	The second of the minute.
	`EV_ZONE`
	Time zone.

Notes	None.
Description	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`	
Arguments	`EV_NICK` *Nick of the message sender.* `EV_USERNAME` *Username of the message sender.* `EV_CHANNEL` *The channel the message was sent from.* `EV_MESSAGE` *The message sent.*
Notes	**Shabti** will call "extra" public message event functions. Their names follow the format `PUBLIC_MESSAGE_EVENT_x`, with "x" equal to a number; with default settings, **Shabti** will call `PUBLIC_MESSAGE_EVENT_1` to `PUBLIC_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 `PUBLIC_MESSAGE_EVENT_1` to `PUBLIC_MESSAGE_EVENT_50` whenever it receives a public message.
Description	Called when the bot receives a public message.

`PRIVATE_MESSAGE_EVENT`	

Arguments	`EV_NICK`
	The day of the week.
	`EV_USERNAME`
	The month of the year.
	`EV_MESSAGE`
	The year.

Notes	**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.
Description	Called when the bot receives a private message.

`ACTION_EVENT`	

Arguments	`EV_NICK`
	Nick of the action sender.
	`EV_USERNAME`
	Username of the action sender.
	`EV_CHANNEL`
	The channel the action was sent from.
	`EV_ACTION`
	The action sent.

Notes	None.
Description	Called when the bot receives a CTCP action message.

`MODE_EVENT`	

Arguments	`EV_NICK`
	Nick of the mode setter.
	`EV_USERNAME`
	Username of the mode setter.
	`EV_TARGET`
	The mode’s target.
	`EV_MODE`
	The mode set.

Notes	None.
Description	Called when the bot receives a mode notification. If the server is the one setting the mode, the `EV_USERNAME` argument will be an empty string.

`JOIN_EVENT`	

Arguments	`EV_NICK`
	Nick of the joiner.
	`EV_USERNAME`
	Username of the joiner.
	`EV_CHANNEL`
	The channel joined.

Notes	**Shabti** will call "extra" join event functions. Their names follow the format `JOIN_EVENT_x`, with "x" equal to a number; with default settings, **Shabti** will call `JOIN_MESSAGE_EVENT_1` to `JOIN_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 `JOIN_MESSAGE_EVENT_1` to `JOIN_MESSAGE_EVENT_50` whenever it receives a join message.
Description	Called when the bot receives a channel join message.

`PART_EVENT`	

Arguments	`EV_NICK`
	Nick of the user leaving.
	`EV_USERNAME`
	Username of the user leaving.
	`EV_CHANNEL`
	The channel being left.
	`EV_MESSAGE`
	Optional parting message

Notes	**Shabti** will call "extra" part event functions. Their names follow the format `PART_EVENT_x`, with "x" equal to a number; with default settings, **Shabti** will call `PART_MESSAGE_EVENT_1` to `PART_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 `PART_MESSAGE_EVENT_1` to `PART_MESSAGE_EVENT_50` whenever it receives a part message.
Description	Called when the bot receives a channel part notification. If the parting user has set a parting message, it will be reflected in the `EV_MESSAGE` argument, which is set to a blank string otherwise.

`IRC_EVENT`	

Arguments	`EV_RAW`
	The unchanges text of the message sent by the server.
	`EV_TYPE`
	The numeric message type, according to RFCs
	`EV_HOST`
	The sending server.
	`EV_NICK`
	The nick the message was sent to.
	`EV_MESSAGE`
	The message content.

Notes	**Shabti** will call "extra" IRC event functions. Their names follow the format `IRC_MESSAGE_EVENT_x`, with "x" equal to a number; with default settings, **Shabti** will call `IRC _MESSAGE_EVENT_1` to `IRC _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 `IRC _MESSAGE_EVENT_1` to `IRC _MESSAGE_EVENT_50` whenever it receives an IRC event.
Description	Called when the bot receives a notification that is not handled by any other event. `EV_RAW` contains the "raw", unchanged notification.

Built-in IRC Functions

`raw`

Arguments	1 (text to send)
Returns	Nothing
Description	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 raw("PRIVMSG Bob :Hello world!").

`set`

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

`login`

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

`nick`

Arguments	1 (new nick)
Returns	Nothing
Description	Changes the bot's nick.

`rnick`

Arguments	1 (new nick)
Returns	Nothing
Description	Changes the bot's nick, adding two numbers to the end of the nick.

join

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

part

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

topic

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

quit

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

message

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

notice

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

action

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

Text Functions

`print`

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

`sprint`

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

`color`

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

`bold`

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

`italic`

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

`underline`

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

File I/O Functions

`read`

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

`write`

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

`swrite`

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

`append`

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

`sappend`

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

`fileexists`

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

direxists

Arguments	1 (directory)
Returns	boolean
Description	Tests if a directory exists or not.

mkdir

Arguments	1 (directory name)
Returns	Nothing
Description	Creates a directory.

rmdir

Arguments	1 (directory name)
Returns	Nothing
Description	Deletes a directory.

delete

Arguments	1 (filename)
Returns	Nothing
Description	Deletes a file.

Miscellaneous Functions

`sha1`

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

`sha256`

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

`require`

Arguments	1 (module name)
Returns	Nothing
Description	Loads a **Shabti** module into memory.

`exit`

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

`tokens`

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