# asynctwitch Documentation

***Release 4.1.0***

**martmists**

**Oct 06, 2017**

# Contents

Contents:

CHAPTER 1

# Bots

**class** asynctwitch.**Bot**(***kwargs*)

A basic Bot. All others inherit from this.

**Parameters**

- **oauth** (*str*) – The oauth code for your account
- **user** (*str*) – Your username
- **prefix** (*Optional[str]*) – The prefix for the bot to listen to. (default: *!*)
- **channel** (*Optional[str, list]*) – The channel(s) to serve. (default: *twitch*)
- **client_id** (*Optional[str]*) – The application Client ID for the kraken API.
- **cache** (*Optional[int]*) – The amount of messages to cache. (default: 100)
- **admins** (*Optional[list]*) – The usernames with full access to the bot.
- **allow_streams** (*Optional[bool]*) – Allow music to play continuous streams

**load**(*path*)

Load settings from a config file.

**Parameters** **path** (*str*) – path to the config file

**override**(*coro*)

Decorator function to override events.

```
@bot.override
async def event_message(message):
    print(message.content)
```

**start**(*tasked=False*)

Starts the bot.

**Parameters** **tasked** (*Optional[bool]*) – Creates a task on the bot loop if True. (default: False)

**colour** (*colour*)
See *bot.color*

**event_notice** (*tags*)
Called on NOTICE events (when commands are called).

**event_clear** (*channel*)
Called when chat is cleared by someone else.

**event_subscribe** (*message*, *tags*)
Called when someone (re-)subscribes.

**event_host_start** (*channel*, *hosted_channel*, *viewer_count*)
Called when the streamer starts hosting.

**event_host_stop** (*channel*, *viewercount*)
Called when the streamer stops hosting.

**event_ban** (*user*, *tags*)
Called when a user is banned.

**event_timeout** (*user*, *tags*)
Called when a user is timed out.

**event_roomstate** (*channel*, *tags*)
Triggered when a channel's chat settings change.

**event_userstate** (*user*)
Triggered when the bot sends a message.

**raw_event** (*data*)
Called on all events after event_ready.

**event_user_join** (*user*)
Called when a user joins a channel.

**event_user_leave** (*user*)
Called when a user leaves a channel.

**event_user_deop** (*user*)
Called when a user is de-opped.

**event_user_op** (*user*)
Called when a user is opped.

**event_private_message** (*message*)
Called when the bot receives a private message.

**event_message** (*message*)
Called when a message is sent by someone in chat.

**stop** (*exit=False*)
Stops the bot and disables using it again.

**Parameters** **exit** (*Optional[bool]*) – If True, this will close the event loop and raise SystemExit. (default: False)

**play_file** (*file*)
Plays an audio file For this to work, ffplay, ffmpeg and ffprobe are required. These are downloadable from the ffmpeg website, and have to be in the same folder as the bot OR added to path.

**Parameters** **file** (*str*) – Filename of the file to play.

**play_ytdl**(*query*, *, *filename='song.mp3'*, *options={}*, *play=True*)
Play a song using youtube_dl

This requires youtube_dl to be installed *pip install youtube_dl*

**Parameters**

- **query** (*str*) – The text to search for or the url to play
- **filename** (*Optional[str]*) – The temporary filename to use. This file will be removed once done playing. (default: "song.mp3")
- **options** (*Optional[dict]*) – The arguments to pass to the YoutubeDL constructor.
- **play** (*Optional[bool]*) – Automatically plays the song if True. If False, this will return a Song object. (default: True)

**parse_error**(*e*)
Called when something errors.

**class** asynctwitch.**CommandBot**(**args*, ***kwargs*)
Allows the usage of Commands more easily

**event_message**(*m*)
If you override this function, make sure to yield from/await *CommandBot.parse_commands*

**parse_commands**(*rm*)
The command parser. It is not recommended to override this.

**command**(**args*, ***kwargs*)
A decorator to add a command. see Command for usage.

**add_timer**(*channel*, *message*, *time=60*)
Send a message on a timer.

**Parameters**

- **channel** (*str*) – The channel to send the message to.
- **message** (*str*) – The message to send.
- **time** (*Optional[int]*) – The interval to send the message. (default: 60)

**class** asynctwitch.**CurrencyBot**(**args*, *points_database='points.db'*, *currency='gold'*, ***kwargs*)
A Bot with support for currency

**check_user_currency**(*user*)
Check if the user is already in the database

**class** asynctwitch.**ViewTimeBot**(**args*, *time_database='time.db'*, ***kwargs*)
A Bot to track view time

**check_user_time**(*user*)
Check if the user is already in the database

**class** asynctwitch.**RankedBot**(**args*, *ranks_database='ranks.db'*, *points_per_minute=1*, ***kwargs*)
A Bot with ranks

**check_user_rank**(*user*, *rank*)
Check if the user is already in the database

# CHAPTER 2

## Dataclasses

**class** asynctwitch.**Object**(***kwargs*)
An object that may be created as substitute for functions.

**class** asynctwitch.**Message**(*m*, *a*, *channel*, *tags*)
Custom message object to combine message, author and timestamp

**class** asynctwitch.**User**(*a*, *channel*, *tags=None*)
Custom user class

**class** asynctwitch.**Emote**(*id*, *loc*)
A class to hold emote data

**id**
*int* – The ID of the emote.

**location**
*str* – The location of the emote in the message.

**url**
*str* – The url of the emote.

**class** asynctwitch.**Badge**(*name*, *value*)
A class to hold badge data.

**name**
*str* – Name of the badge.

**value**
*str* – Variant of the badge.

**classmethod from_str**(*s*)
e.g. Moderator/1

**class** asynctwitch.**Color**(*value*)
Available colors for non-turbo users when using Bot.color

Conversions are not working perfectly:

```
>>> str( Color.blue() ) #0000FF
'#0000ff'

#0000FF to and from yiq
>>> str( Color.from_yiq( *Color.blue().to_yiq() ) )
'#0000fe'

#0000FF to and from hsv
>>> str( Color.from_hsv( *Color.blue().to_hsv() ) )
'#00ffff'
```

**to_rgb**()
Returns an (r, g, b) tuple of the color

**to_yiq**()
Returns a (y, i, q) tuple of the color

**to_hsv**()
Returns a (h, s, v) tuple of the color

**classmethod from_rgb**(*r*, *g*, *b*)
(0,0,0) to (255,255,255)

asynctwitch.**Colour**
alias of *Color*

CHAPTER 3

# Command API

**class** asynctwitch.**Command**(*bot*, *comm*, *desc=''*, *alias=[]*, *admin=False*, *unprefixed=False*, *listed=True*)

A command class to provide methods we can use with it

**subcommand**(**args*, ***kwargs*)

Create subcommands

**run**(*message*)

Does type checking for command arguments

**class** asynctwitch.**SubCommand**(*parent*, *comm*, *desc*, **alias*)

Subcommand class

CHAPTER 4

# Indices and tables

- genindex
- modindex
- search

# Index

## T



## U



## V