NAME

AnyEvent::I3 – communicate with the i3 window manager

VERSION

Version 0.17

SYNOPSIS

This module connects to the i3 window manager using the UNIX socket based IPC interface it provides (if enabled in the configuration file). You can then subscribe to events or send messages and receive their replies.

```
use AnyEvent::I3 qw(:all);

my $i3 = i3();

$i3->connect->recv or die "Error connecting";
say "Connected to i3";

my $workspaces = $i3->message(TYPE_GET_WORKSPACES)->recv;
say "Currently, you use " . @{$workspaces} . " workspaces";
```

...or, using the sugar methods:

```
use AnyEvent::I3;

my $workspaces = i3->get_workspaces->recv;
say "Currently, you use " . @{$workspaces} . " workspaces";
```

A somewhat more involved example which dumps the i3 layout tree whenever there is a workspace event:

```
use Data::Dumper;
use AnyEvent;
use AnyEvent::I3;

my $i3 = i3();

$i3->connect->recv or die "Error connecting to i3";

$i3->subscribe({
    workspace => sub {
        $i3->get_tree->cb(sub {
            my ($tree) = @_;
            say "tree: " . Dumper($tree);
        });
    }
})->recv->{success} or die "Error subscribing to events";

AE::cv->recv
```

EXPORT

`$i3` **= i3([** `$path` **]);**

Creates a new `AnyEvent::I3` object and returns it.

`path` is an optional path of the UNIX socket to connect to. It is strongly advised to NOT specify this unless you're absolutely sure you need it. `AnyEvent::I3` will automatically figure it out by querying the running i3 instance on the current DISPLAY which is almost always what you want.

SUBROUTINES/METHODS

`$i3` = AnyEvent::I3−>new([`$path`])

Creates a new `AnyEvent::I3` object and returns it.

`path` is an optional path of the UNIX socket to connect to. It is strongly advised to NOT specify this unless you're absolutely sure you need it. `AnyEvent::I3` will automatically figure it out by querying the running i3 instance on the current DISPLAY which is almost always what you want.

`$i3`−>connect

Establishes the connection to i3. Returns an `AnyEvent::CondVar` which will be triggered with a boolean (true if the connection was established) as soon as the connection has been established.

```
if ($i3->connect->recv) {
    say "Connected to i3";
}
```

`$i3`−>subscribe(\%callbacks)

Subscribes to the given event types. This function awaits a hashref with the key being the name of the event and the value being a callback.

```
my %callbacks = (
    workspace => sub { say "Workspaces changed" }
);

if ($i3->subscribe(\%callbacks)->recv->{success}) {
    say "Successfully subscribed";
}
```

The special callback with name `_error` is called when the connection to i3 is killed (because of a crash, exit or restart of i3 most likely). You can use it to print an appropriate message and exit cleanly or to try to reconnect.

```
my %callbacks = (
    _error => sub {
        my ($msg) = @_;
        say "I am sorry. I am so sorry: $msg";
        exit 1;
    }
);

$i3->subscribe(\%callbacks)->recv;
```

`$i3`−>message($type, `$content`)

Sends a message of the specified `type` to i3, possibly containing the data structure `content` (or `content`, encoded as utf8, if `content` is a scalar), if specified.

```
my $reply = $i3->message(TYPE_COMMAND, "reload")->recv;
if ($reply->{success}) {
    say "Configuration successfully reloaded";
}
```

SUGAR METHODS

These methods intend to make your scripts as beautiful as possible. All of them automatically establish a connection to i3 blockingly (if it does not already exist).

get_workspaces

Gets the current workspaces from i3.

```
my $ws = i3->get_workspaces->recv;
say Dumper($ws);
```

get_outputs

Gets the current outputs from i3.

```
my $outs = i3->get_outputs->recv;
say Dumper($outs);
```

get_tree

Gets the layout tree from i3 (>= v4.0).

```
my $tree = i3->get_tree->recv;
say Dumper($tree);
```

get_marks

Gets all the window identifier marks from i3 (>= v4.1).

```
my $marks = i3->get_marks->recv;
say Dumper($marks);
```

get_bar_config

Gets the bar configuration for the specific bar id from i3 (>= v4.1).

```
my $config = i3->get_bar_config($id)->recv;
say Dumper($config);
```

get_version

Gets the i3 version via IPC, with a fall-back that parses the output of i3 −−version (for i3 < v4.3).

```
my $version = i3->get_version()->recv;
say "major: " . $version->{major} . ", minor = " . $version->{minor};
```

command($content)

Makes i3 execute the given command

```
my $reply = i3->command("reload")->recv;
die "command failed" unless $reply->{success};
```

AUTHOR

Michael Stapelberg, `<michael at i3wm.org>`

BUGS

Please report any bugs or feature requests to `bug-anyevent-i3 at rt.cpan.org`, or through the web interface at <http://rt.cpan.org/NoAuth/ReportBug.html?Queue=AnyEvent−I3>. I will be notified, and then you'll automatically be notified of progress on your bug as I make changes.

SUPPORT

You can find documentation for this module with the perldoc command.

```
perldoc AnyEvent::I3
```

You can also look for information at:

- RT: CPAN's request tracker

  <http://rt.cpan.org/NoAuth/Bugs.html?Dist=AnyEvent−I3>

- The i3 window manager website

  <http://i3wm.org>

ACKNOWLEDGEMENTS

LICENSE AND COPYRIGHT