



# GTK::V3::Glib::GSignal

## Table of Contents

- 0.1 [GSignal — A means for customization of object behaviour and a general purpose notification mechanism](#)
- 1 [Synopsis](#)
- 2 [Methods](#)
  - 2.1 [\[g\\_signal\\_\] connect\\_object](#)
  - 2.2 [\[g\\_signal\\_\] emit\\_by\\_name](#)
  - 2.3 [\[g\\_signal\\_\] handler\\_disconnect](#)

```
unit class GTK::V3::Glib::GSignal;
```

## **GSignal — A means for customization of object behaviour and a general purpose notification mechanism**

### **Synopsis**

```
# define method
method mouse-event ( :widget($w), :event($e)) { ... }

# get the window object
my GTK::V3::Gtk::GtkWindow $w .= new( ... );

# define proper handler. you must study the GTK developer guides. you will
# then notice that C<connect-object> is a bit different than the real mcCoy.
my Callable $handler;
$handler = -> N-GObject $ignore-w, GdkEvent $e, OpaquePointer $ignore-d {
    self.mouse-event( :widget($w), :event($e) );
}

# connect signal to the handler
$w.connect-object( 'button-press-event', $handler);
```

It will be easier to use the [register-signal\(\)](#) method defined in [GTK::V3::Glib::GObject](#).

```
# define method
method mouse-event ( :widget($w), :event($e), :$time) { ... }

# get the window object
my GTK::V3::Gtk::GtkWindow $w .= new( ... );

# then register
$w.register-signal( self, 'mouse-event', 'button-press-event', :time(now));
```

## Methods

### [g\_signal\_] connect\_object

Connects a callback function to a signal for a particular object.

```
method g_signal_connect_object( Str $signal, Callable $handler --> uint64 )
```

- \$signal; a string of the form `signal-name::detail`.
- \$handler; the callback to connect.

### [g\_signal\_] emit\_by\_name

Emits a signal.

Note that `g_signal_emit_by_name()` resets the return value to the default if no handlers are connected.

```
g_signal_emit_by_name ( Str $signal, N-GObject $widget )
```

- \$signal; a string of the form "signal-name::detail".
- \$widget; widget to pass to the handler.

### [g\_signal\_] handler\_disconnect

Disconnects a handler from an instance so it will not be called during any future or currently ongoing emissions of the signal it has been connected to. The `handler_id` becomes invalid and may be reused.

The `handler_id` has to be a valid signal handler id, connected to a signal of instance .

```
g_signal_handler_disconnect( int32 $handler_id )
```

- `$handler_id`; Handler id of the handler to be disconnected.