

# GTK::V3::Gdk::GdkEventTypes

Event Structures — Data structures specific to each type of event

## Table of Contents

- 1 [Synopsis](#)
- 2 [class GTK::V3::Gdk::GdkEventTypes](#)
- 3 [Enums, Structs and Unions](#)
- 3.1 [Enum GdkEventType Specifies the type of the event.](#)
- 3.2 [class GdkEventAny](#)
- 3.3 [class GdkEventKey](#)
- 3.4 [class GdkEventButton](#)
- 3.5 [class GdkEventTouch Used for touch events. type field will be one of GDK\\_TOUCH\\_BEGIN, GDK\\_TOUCH\\_UPDATE, GDK\\_TOUCH\\_END or GDK\\_TOUCH\\_CANCEL.](#)
- 3.6 [GdkEvent](#)

## Synopsis

```
my GTK::V3::Gtk::GtkWindow $top-window .= new(:empty);
$top-window.set-title('Hello GTK!');
# ... etcetera ...

# Register a signal handler for a window event
$top-window.register-signal( self, 'handle-keypress', 'key-press-event');

method handle-keypress ( :$widget, GdkEvent :$event ) {
    if $event.event-any.type ~~ GDK_KEY_PRESS and
        $event.event-key.keyval eq 's' {

        # key 's' pressed, stop process ...
    }
}
```

The handler signature can also be defined as

```
method handle-keypress ( :$widget, GdkEventKey :$event ) {
    if $event.type ~~ GDK_KEY_PRESS and $event.keyval eq 's' {

        # key 's' pressed, stop process ...
    }
}
```

## class GTK::V3::Gdk::GdkEventTypes

## Enums, Structs and Unions

### Enum GdkEventType Specifies the type of the event.

Do not confuse these events with the signals that GTK+ widgets emit. Although many of these events result in corresponding signals being emitted, the events are often transformed or filtered along the way.

In some language bindings, the values GDK\_2BUTTON\_PRESS and GDK\_3BUTTON\_PRESS would translate into something syntactically invalid (eg `Gdk.EventType.2ButtonPress`, where a symbol is not allowed to start with a number). In that case, the aliases GDK\_DOUBLE\_BUTTON\_PRESS and GDK\_TRIPLE\_BUTTON\_PRESS can be used instead.

- GDK\_NOTHING; a special code to indicate a null event.
- GDK\_DELETE; the window manager has requested that the toplevel window be hidden or destroyed, usually when the user clicks on a special icon in the title bar.
- GDK\_DESTROY; the window has been destroyed.
- GDK\_EXPOSE; all or part of the window has become visible and needs to be redrawn.

- GDK\_MOTION\_NOTIFY; the pointer (usually a mouse) has moved.
- GDK\_BUTTON\_PRESS; a mouse button has been pressed.
- GDK\_2BUTTON\_PRESS; a mouse button has been double-clicked (clicked twice within a short period of time). Note that each click also generates a GDK\_BUTTON\_PRESS event.
- GDK\_DOUBLE\_BUTTON\_PRESS; alias for GDK\_2BUTTON\_PRESS, added in 3.6.
- GDK\_3BUTTON\_PRESS; a mouse button has been clicked 3 times in a short period of time. Note that each click also generates a GDK\_BUTTON\_PRESS event.
- GDK\_TRIPLE\_BUTTON\_PRESS; alias for GDK\_3BUTTON\_PRESS, added in 3.6.
- GDK\_BUTTON\_RELEASE; a mouse button has been released.
- GDK\_KEY\_PRESS; a key has been pressed.
- GDK\_KEY\_RELEASE; a key has been released.
- GDK\_ENTER\_NOTIFY; the pointer has entered the window.
- GDK\_LEAVE\_NOTIFY; the pointer has left the window.
- GDK\_FOCUS\_CHANGE; the keyboard focus has entered or left the window.
- GDK\_CONFIGURE; the size, position or stacking order of the window has changed. Note that GTK+ discards these events for GDK\_WINDOW\_CHILD windows.
- GDK\_MAP; the window has been mapped.
- GDK\_UNMAP; the window has been unmapped.
- GDK\_PROPERTY\_NOTIFY; a property on the window has been changed or deleted.
- GDK\_SELECTION\_CLEAR; the application has lost ownership of a selection.
- GDK\_SELECTION\_REQUEST; another application has requested a selection.
- GDK\_SELECTION\_NOTIFY; a selection has been received.
- GDK\_PROXIMITY\_IN; an input device has moved into contact with a sensing surface (e.g. a touchscreen or graphics tablet).
- GDK\_PROXIMITY\_OUT; an input device has moved out of contact with a sensing surface.
- GDK\_DRAG\_ENTER; the mouse has entered the window while a drag is in progress.
- GDK\_DRAG\_LEAVE; the mouse has left the window while a drag is in progress.
- GDK\_DRAG\_MOTION; the mouse has moved in the window while a drag is in progress.
- GDK\_DRAG\_STATUS; the status of the drag operation initiated by the window has changed.
- GDK\_DROP\_START; a drop operation onto the window has started.
- GDK\_DROP\_FINISHED; the drop operation initiated by the window has completed.
- GDK\_CLIENT\_EVENT; a message has been received from another application.
- GDK\_VISIBILITY\_NOTIFY; the window visibility status has changed.
- GDK\_SCROLL; the scroll wheel was turned.
- GDK\_WINDOW\_STATE; the state of a window has changed. See GdkWindowState for the possible window states.
- GDK\_SETTING; a setting has been modified.
- GDK\_OWNER\_CHANGE; the owner of a selection has changed. This event type was added in 2.6
- GDK\_GRAB\_BROKEN; a pointer or keyboard grab was broken. This event type was added in 2.8.
- GDK\_DAMAGE; the content of the window has been changed. This event type was added in 2.14.

- GDK\_TOUCH\_BEGIN; A new touch event sequence has just started. This event type was added in 3.4.
- GDK\_TOUCH\_UPDATE; A touch event sequence has been updated. This event type was added in 3.4.
- GDK\_TOUCH\_END; A touch event sequence has finished. This event type was added in 3.4.
- GDK\_TOUCH\_CANCEL; A touch event sequence has been canceled. This event type was added in 3.4.
- GDK\_TOUCHPAD\_SWIPE; A touchpad swipe gesture event, the current state is determined by its phase field. This event type was added in 3.18.
- GDK\_TOUCHPAD\_PINCH; A touchpad pinch gesture event, the current state is determined by its phase field. This event type was added in 3.18.
- GDK\_PAD\_BUTTON\_PRESS; A tablet pad button press event. This event type was added in 3.22.
- GDK\_PAD\_BUTTON\_RELEASE; A tablet pad button release event. This event type was added in 3.22.
- GDK\_PAD\_RING; A tablet pad axis event from a "ring". This event type was added in 3.22.
- GDK\_PAD\_STRIP; A tablet pad axis event from a "strip". This event type was added in 3.22.
- GDK\_PAD\_GROUP\_MODE; A tablet pad group mode change. This event type was added in 3.22.
- GDK\_EVENT\_LAST; Marks the end of the GdkEventType enumeration. Added in 2.18

## class GdkEventAny

Contains the fields which are common to all event classes. This comes in handy to check its type for instance.

- GdkEventType \$.type; the type of the event.
- N-GObject \$.window; the window which received the event.
- Int \$.send\_event; TRUE if the event was sent explicitly.

## class GdkEventKey

Describes a key press or key release event. The type of the event will be one of GDK\_KEY\_PRESS or GDK\_KEY\_RELEASE.

- GdkEventType \$.type
- N-GObject \$.window
- Int \$.send\_event
- UInt \$.time; the time of the event in milliseconds.
- UInt \$.state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType. [type GdkModifierType].
- UInt \$.keyval; the key that was pressed or released. See the gdk/gdkkeysyms.h header file for a complete list of GDK key codes.
- Int \$.length; the length of string.
- Str \$.string; deprecated.
- UInt \$.hardware\_keycode; the raw code of the key that was pressed or released.
- UInt \$.group; the keyboard group.
- UInt \$.is\_modifier; a flag that indicates if hardware\_keycode is mapped to a modifier. Since 2.10

## class GdkEventButton

Used for mouse button press and button release events. The type will be one of GDK\_BUTTON\_PRESS, GDK\_2BUTTON\_PRESS, GDK\_3BUTTON\_PRESS or GDK\_BUTTON\_RELEASE,

Double and triple-clicks result in a sequence of events being received. For double-clicks the order of events will be:

GDK\_BUTTON\_PRESS, GDK\_BUTTON\_RELEASE, GDK\_BUTTON\_PRESS, GDK\_2BUTTON\_PRESS and GDK\_BUTTON\_RELEASE.

Note that the first click is received just like a normal button press, while the second click results in a GDK\_2BUTTON\_PRESS being received just after the GDK\_BUTTON\_PRESS.

Triple-clicks are very similar to double-clicks, except that GDK\_3BUTTON\_PRESS is inserted after the third click. The order of the events is: GDK\_BUTTON\_PRESS, GDK\_BUTTON\_RELEASE, GDK\_BUTTON\_PRESS, GDK\_2BUTTON\_PRESS, GDK\_BUTTON\_RELEASE, GDK\_BUTTON\_PRESS, GDK\_3BUTTON\_PRESS and GDK\_BUTTON\_RELEASE.

For a double click to occur, the second button press must occur within 1/4 of a second of the first. For a triple click to occur, the third button press must also occur within 1/2 second of the first button press.

To handle e.g. a triple mouse button presses, all events can be ignored except GDK\_3BUTTON\_PRESS

```
method handle-keypress ( :$widget, GdkEventButton :$event ) {  
  # check if left mouse button was pressed three times  
  if $event.type == GDK_3BUTTON_PRESS and $event.button == 1 {  
    ...  
  }  
}
```

- GdkEventType \$.type;
- N-Object \$.window;
- Int \$.send\_event;
- UInt \$.time; the time of the event in milliseconds.
- Num \$.x; the x coordinate of the pointer relative to the window.
- Num \$.y; the y coordinate of the pointer relative to the window.
- Pointer[Num] \$.axes; x , y translated to the axes of device , or NULL if device is the mouse.
- UInt \$.state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType.
- UInt \$.button; the button which was pressed or released, numbered from 1 to 5. Normally button 1 is the left mouse button, 2 is the middle button, and 3 is the right button. On 2-button mice, the middle button can often be simulated by pressing both mouse buttons together.
- N-Object \$.device; the master device that the event originated from. Use gdk\_event\_get\_source\_device() to get the slave device.
- Num \$.x\_root; the x coordinate of the pointer relative to the root of the screen.
- Num \$.y\_root; the y coordinate of the pointer relative to the root of the screen.

**class GdkEventTouch Used for touch events. type field will be one of GDK\_TOUCH\_BEGIN, GDK\_TOUCH\_UPDATE, GDK\_TOUCH\_END or GDK\_TOUCH\_CANCEL.**

Touch events are grouped into sequences by means of the sequence field, which can also be obtained with gdk\_event\_get\_event\_sequence(). Each sequence begins with a GDK\_TOUCH\_BEGIN event, followed by any number of GDK\_TOUCH\_UPDATE events, and ends with a GDK\_TOUCH\_END (or GDK\_TOUCH\_CANCEL) event. With multitouch devices, there may be several active sequences at the same time.

- GdkEventType \$.type; the type of the event (GDK\_TOUCH\_BEGIN, GDK\_TOUCH\_UPDATE, GDK\_TOUCH\_END, GDK\_TOUCH\_CANCEL)
- N-Object \$.window;
- Int \$.send\_event;
- uint32 \$.time; the time of the event in milliseconds.
- num64 \$.x; the x coordinate of the pointer relative to the window
- num64 \$.y; the y coordinate of the pointer relative to the window

- Pointer[num64] \$.axes; x , y translated to the axes of device , or NULL if device is the mouse
- uint32 state; a bit-mask representing the state of the modifier keys (e.g. Control, Shift and Alt) and the pointer buttons. See GdkModifierType.
- Pointer \$.sequence; the event sequence that the event belongs to
- int32 emulating\_pointer; whether the event should be used for emulating pointer event (0 or 1)
- GObject \$.device; the master device that the event originated from. Use gdk\_event\_get\_source\_device() to get the slave device.
- num64 \$.x\_root; the x coordinate of the pointer relative to the root of the screen
- num64 \$.y\_root; the y coordinate of the pointer relative to the root of the screen

## GdkEvent

The event structures contain data specific to each type of event in GDK. The type is a union of all structures explained above.