Gnome::Gtk3::Window

Window — Toplevel which can contain other widgets

Table of Contents

```
unit class Gnome::Gtk3::Window;
also is Gnome::Gtk3::Bin;
```

Methods

new

```
multi method new (
  Bool :$empty!, GtkWindowType :$window-type = GTK_WINDOW_TOPLEVEL
)
```

Create an empty top level window or popup.

```
multi method new (
  Bool :$title!, GtkWindowType :$window-type = GTK_WINDOW_TOPLEVEL
)
```

Create a top level window or popup with title set.

```
multi method new ( :$widget! )
```

Create a button using a native object from elsewhere. See also Gnome::GObject::Object.

```
multi method new ( Str :$build-id! )
```

Create a button using a native object from a builder. See also Gnome::GObject::Object.

gtk_window_new

```
method gtk_window_new ( int32 $type )
```

Creates a new GtkWindow, which is a toplevel window that can contain other widgets. Nearly always, the type of the window should be GTK_WINDOW_TOPLEVEL. If you're implementing something like a popup menu from scratch (which is a bad idea, just use GtkMenu), you might use GTK_WINDOW_POPUP. GTK_WINDOW_POPUP is not for dialogs, though in some other toolkits dialogs are called “popups”. In GTK+, GTK_WINDOW_POPUP means a pop-up menu or pop-up tooltip. On X11, popup windows are not controlled by the window manager.

If you simply want an undecorated window (no window borders), use gtk_window_set_decorated(), don't use GTK_WINDOW_POPUP.

All top-level windows created by gtk_window_new() are stored in an internal top-level window list. This list can be obtained from gtk_window_list_toplevels(). Due to Gtk+ keeping a reference to the window internally, gtk_window_new() does not return a reference to the caller.

To delete a GtkWindow, call gtk_widget_destroy().

[gtk_window_] set_title

```
method gtk_window_set_title ( Str $title )
```

Sets the title of the GtkWindow. The title of a window will be displayed in its title bar; on the X Window System, the title bar is rendered by the window manager, so exactly how the title appears to users may vary according to a user’s exact configuration. The title should help a user distinguish this window from other windows they may have open. A good title might include the application name and current document filename, for example.

[gtk_window_] set_default_size

```
method gtk_window_set_default_size ( Int $width, Int $height )
```

Sets the default size of a window. See also the developer docs.

[gtk_window_] set_modal

```
method gtk_window_set_modal ( Int $modal )
```

Sets a window modal or non-modal. Modal windows prevent interaction with other windows in the same application. To keep modal dialogs on top of main application windows, use gtk_window_set_transient_for() to make the dialog transient for the parent; most window managers will then disallow lowering the dialog below the parent.

[gtk_window_] set_position

```
method gtk_window_set_position ( Int $position )
```

Sets a position constraint for this window. If the old or new constraint is GTK_WIN_POS_CENTER_ALWAYS, this will also cause the window to be repositioned to satisfy the new constraint.

[gtk_window_] set_transient_for

```
method gtk_window_set_transient_for ( Gnome::GObject::Object $main-window )
```

Dialog windows should be set transient for the main application window they were spawned from. This allows window managers to e.g. keep the dialog on top of the main window, or center the dialog over the main window.

Types

GtkWindowPosition

Window placement can be influenced using this enumeration. Note that using GTK_WIN_POS_CENTER_ALWAYS is almost always a bad idea. It won't necessarily work well with all window managers or on all windowing systems.

- GTK_WIN_POS_NONE. No influence is made on placement.
- GTK_WIN_POS_CENTER. Windows should be placed in the center of the screen.
- GTK_WIN_POS_MOUSE. Windows should be placed at the current mouse position.
- GTK_WIN_POS_CENTER_ALWAYS. Keep window centered as it changes size, etc.
- GTK_WIN_POS_CENTER_PARENT. Center the window on its transient parent.

GtkWindowType

A GtkWindow can be one of these types. Most things you'd consider a "window" should have type GTK_WINDOW_TOPLEVEL; windows with this type are managed by the window manager and have a frame by default (call gtk_window_set_decorated() to toggle the frame). Windows with type GTK_WINDOW_POPUP are ignored by the window manager; window manager keybindings won't work on them, the window manager won't decorate the window with a frame, many GTK+ features that rely on the window manager will not work (e.g. resize grips and maximization/minimization). GTK_WINDOW_POPUP is used to implement widgets such as GtkMenu or tooltips that you normally don't think of as windows per se. Nearly all windows should be GTK_WINDOW_TOPLEVEL. In particular, do not use GTK_WINDOW_POPUP just to turn off the window borders; use gtk_window_set_decorated() for that.

- GTK_WINDOW_TOPLEVEL. A regular window, such as a dialog.
- GTK_WINDOW_POPUP. A special window such as a tooltip.

Signals

Supported signals

activate-default

The `activate-default` signal is a keybinding signal which gets emitted when the user activates the default widget of window.

Signal Handler Signature

```
method handler (
  Gnome::GObject::Object :$widget, :$user-option1, ..., $user-optionN
)
```

activate-focus

The `activate-focus` signal is a keybinding signal which gets emitted when the user activates the currently focused widget of window.

Signal Handler Signature

```
method handler (
  Gnome::GObject::Object :$widget, :$user-option1, ..., $user-optionN
)
```

keys-changed

The `keys-changed` signal gets emitted when the set of accelerators or mnemonics that are associated with window changes.

Signal Handler Signature

```
method handler (
  Gnome::GObject::Object :$widget, :$user-option1, ..., $user-optionN
)
```

set-focus

Signal Handler Signature

```
method handler (
  Gnome::GObject::Object :$widget, N-GObject :$nativewidget,
  :$user-option1, ..., :$user-optionN
)
```

Not yet supported signals

enable-debugging

The `enable-debugging` signal is a keybinding signal which gets emitted when the user enables or disables interactive debugging. When toggle is 1, interactive debugging is toggled on or off, when it is 0, the debugger will be pointed at the widget under the pointer.

The default bindings for this signal are `Ctrl-Shift-I` and `Ctrl-Shift-D`.

Return: 1 if the key binding was handled

Handler Method Arguments

- $widget; This can be any perl6 widget with `Gnome::GObject::Object` as the top parent class e.g. `Gnome::Gtk3::Button`.
- $event; A structure defined in `Gnome::Gdk::EventTypes`.
- $nativewidget; A native widget which can be turned into a perl6 widget using `.new(:widget())` on the appropriate class.
- $user-option*; Any extra options given by the user when registering the signal.