Gnome::Gtk3::Widget

Table of Contents


```
unit class Gnome::Gtk3::Widget;
also is Gnome::Gtk3::GInitiallyUnowned;
```

Widget — Base class for all widgets

Synopsis

```
# create a button and set a tooltip
my Gnome::Gtk3::Button $start-button .= new(:label<Start>);
$start-button.set-tooltip-text('Nooooo don\'t touch that button!!!!!!!');
```

Methods

gtk_widget_destroy

```
method gtk_widget_destroy ( )
```

Destroys a native widget. When a widget is destroyed all references it holds on other objects will be released:

- if the widget is inside a container, it will be removed from its parent
- if the widget is a container, all its children will be destroyed, recursively
- if the widget is a top level, it will be removed from the list of top level widgets that GTK+ maintains internally

It's expected that all references held on the widget will also be released; you should connect to the “destroy” signal if you hold a reference to widget and you wish to remove it when this function is called. It is not necessary to do so if you are implementing a GtkContainer, as you'll be able to use the GtkContainerClass.remove() virtual function for that.

It's important to notice that gtk_widget_destroy() will only cause the widget to be finalized if no additional references, acquired using g_object_ref(), are held on it. In case additional references are in place, the widget will be in an "inert" state after calling this function; widget will still point to valid memory, allowing you to release the references you hold, but you may not query the widget's own state.

You should typically call this function on top level widgets, and rarely on child widgets.

See also: gtk_container_remove()

gtk_widget_show

```
method gtk_widget_show ( )
```

Flags a widget to be displayed. Any widget that isn't shown will not appear on the screen. If you want to show all the widgets in a container, it's easier to call gtk_widget_show_all() on the container, instead of individually showing the widgets.

Remember that you have to show the containers containing a widget, in addition to the widget itself, before it will appear onscreen.

When a toplevel container is shown, it is immediately realized and mapped; other shown widgets are realized and mapped when their toplevel container is realized and mapped.

gtk_widget_hide

```
method gtk_widget_hide ( )
```

Reverses the effects of gtk_widget_show().

[gtk_widget_] show_all

```
method gtk_widget_show_all ( )
```

Recursively shows a widget, and any child widgets (if the widget is a container).

[gtk_widget_] set_name

```
method gtk_widget_set_name ( Str $name )
```

Widgets can be named, which allows you to refer to them from a CSS file. You can apply a style to widgets with a particular name in the CSS file. See the documentation for the CSS syntax (on the same page as the docs for GtkStyleContext).

Note that the CSS syntax has certain special characters to delimit and represent elements in a selector (period, #, >, *...), so using these will make your widget impossible to match by name. Any combination of alphanumeric symbols, dashes and underscores will suffice.

[gtk_widget_] get_name

```
method gtk_widget_get_name ( )
```

Retrieves the name of a widget. See gtk_widget_set_name() for the significance of widget names.

[gtk_widget_] set_sensitive

```
method gtk_widget_set_sensitive ( Int $sensitive )
```

Sets the sensitivity of a widget. A widget is sensitive if the user can interact with it. Insensitive widgets are “grayed out” and the user can’t interact with them. Insensitive widgets are known as “inactive”, “disabled”, or “ghosted” in some other toolkits.

[gtk_widget_] get_sensitive

```
method gtk_widget_get_sensitive ( --> Int )
```

Returns the widget’s sensitivity (in the sense of returning the value that has been set using gtk_widget_set_sensitive()).

The effective sensitivity of a widget is however determined by both its own and its parent widget’s sensitivity. See gtk_widget_is_sensitive().

[gtk_widget_] set_size_request

```
method gtk_widget_set_size_request ( Int $w, Int $h )
```

Sets the minimum size of a widget; that is, the widget’s size request will be at least width by height . You can use this function to force a widget to be larger than it normally would be.

In most cases, gtk_window_set_default_size() is a better choice for toplevel windows than this function; setting the default size will still allow users to shrink the window. Setting the size request will force them to leave the window at least as large as the size request. When dealing with window sizes, gtk_window_set_geometry_hints() can be a useful function as well.

Note the inherent danger of setting any fixed size - themes, translations into other languages, different fonts, and user action can all change the appropriate size for a given widget. So, it's basically impossible to hardcode a size that will always be correct.

The size request of a widget is the smallest size a widget can accept while still functioning well and drawing itself correctly. However in some strange cases a widget may be allocated less than its requested size, and in many cases a widget may be allocated more space than it requested.

If the size request in a given direction is -1 (unset), then the “natural” size request of the widget will be used instead.

The size request set here does not include any margin from the GtkWidget properties margin-left, margin-right, margin-top, and margin-bottom, but it does include pretty much all other padding or border properties set by any subclass of GtkWidget.

[gtk_widget_] set_no_show_all

```
method gtk_widget_set_no_show_all ( Int $no_show_all )
```

Sets the “no-show-all” property, which determines whether calls to gtk_widget_show_all() will affect this widget.

This is mostly for use in constructing widget hierarchies with externally controlled visibility.

[gtk_widget_] get_no_show_all

```
method gtk_widget_get_no_show_all ( --> Int )
```

Returns the current value of the “no-show-all” property, which determines whether calls to gtk_widget_show_all() will affect this widget.

[gtk_widget_] get_allocated_width

```
method gtk_widget_get_allocated_width ( --> Int )
```

Returns the width that has currently been allocated to widget . This function is intended to be used when implementing handlers for the “draw” function.

[gtk_widget_] get_allocated_height

```
method gtk_widget_get_allocated_height ( --> Int )
```

Returns the height that has currently been allocated to widget . This function is intended to be used when implementing handlers for the “draw” function.

[gtk_widget_] queue_draw

```
method gtk_widget_queue_draw ( )
```

Equivalent to calling gtk_widget_queue_draw_area() for the entire area of a widget.

[gtk_widget_] get_display

```
method gtk_widget_get_display ( )
```

Get the GdkDisplay for the toplevel window associated with this widget. This function can only be called after the widget has been added to a widget hierarchy with a GtkWindow at the top.

In general, you should only create display specific resources when a widget has been realized, and you should free those resources when the widget is unrealized.

[gtk_widget_] set_direction

Sets the reading direction on a particular widget. This direction controls the primary direction for widgets containing text, and also the direction in which the children of a container are packed. The ability to set the direction is present in order so that correct localization into languages with right-to-left reading directions can be done. Generally, applications will let the default reading direction present, except for containers where the containers are arranged in an order that is explicitly visual rather than logical (such as buttons for text justification).

If the direction is set to `GTK_TEXT_DIR_NONE`, then the value set by `gtk_widget_set_default_direction()` will be used.

```
method gtk_widget_set_direction ( Int $direction )
```

- Int $direction; the new direction. This is a GtkTextDirection enum type defined in GtkEnums.

[gtk_widget_] get_direction

Gets the reading direction for a particular widget.

```
method gtk_widget_get_direction ( --> GtkTextDirection )
```

Returns the current text direction. This is a GtkTextDirection enum type defined in GtkEnums.

[gtk_widget_] set_default_direction

Sets the default reading direction on a particular widget.

```
method gtk_widget_set_default_direction ( GtkTextDirection $direction )
```

- $direction; the default direction.

[gtk_widget_] get_default_direction

Gets the default reading direction for a particular widget.

```
method gtk_widget_get_default_direction ( --> GtkTextDirection )
```

Returns the default text direction.

[gtk_widget_] set_tooltip_text

```
method gtk_widget_set_tooltip_text ( Str $text )
```

Sets text as the contents of the tooltip. This function will take care of setting “has-tooltip” to TRUE and of the default handler for the “query-tooltip” signal.

[gtk_widget_] get_tooltip_text

```
method gtk_widget_get_tooltip_text ( --> Str )
```

Sets text as the contents of the tooltip. This function will take care of setting “has-tooltip” to TRUE and of the default handler for the “query-tooltip” signal.

[gtk_widget_] get_window

```
method gtk_widget_get_window ( --> N-GObject )
```

Returns the widget’s window (is a GtkWindow) if it is realized, NULL otherwise.

```
my Gtk::V3::Gtk::GtkButton $b .= new(:build-id<startButton>);
my Gtk::V3::Gdk::GdkWindow $w .= new(:widget($b.get-window));
```

[gtk_widget_] set_visible

```
method gtk_widget_set_visible ( Int $visible )
```

Sets the visibility state of widget. Note that setting this to TRUE doesn’t mean the widget is actually viewable, see gtk_widget_get_visible().

This function simply calls gtk_widget_show() or gtk_widget_hide() but is nicer to use when the visibility of the widget depends on some condition.

[gtk_widget_] get_visible

```
method gtk_widget_get_visible ( --> Int )
```

Determines whether the widget is visible. If you want to take into account whether the widget’s parent is also marked as visible, use gtk_widget_is_visible() instead.

This function does not check if the widget is obscured in any way.

[gtk_widget_] get_has_window

```
method gtk_widget_get_has_window ( --> Int )
```

Specifies whether widget has a GdkWindow of its own. Note that all realized widgets have a non-NULL “window” pointer (gtk_widget_get_window() never returns a NULL window when a widget is realized), but for many of them it’s actually the GdkWindow of one of its parent widgets. Widgets that do not create a window for themselves in “realize” must announce this by calling this function with has_window = FALSE.

This function should only be called by widget implementations, and they should call it in their init() function.