Part IV. Theming in GTK+: GTK+ 3 Reference Manual

Table of Contents

Filesystem utilities

Filesystem utilities — Functions for working with GIO

Functions

GMountOperation *	gtk_mount_operation_new ()
gboolean	gtk_mount_operation_is_showing ()
void	gtk_mount_operation_set_parent ()
GtkWindow *	gtk_mount_operation_get_parent ()
void	gtk_mount_operation_set_screen ()
GdkScreen *	gtk_mount_operation_get_screen ()
gboolean	gtk_show_uri ()
gboolean	gtk_show_uri_on_window ()

Properties

gboolean	is-showing	Read
GtkWindow *	parent	Read / Write
GdkScreen *	screen	Read / Write

Types and Values

struct	GtkMountOperation
struct	GtkMountOperationClass

Object Hierarchy

```
GObject
└── GMountOperation
    └── GtkMountOperation
```

Includes

```
#include <gtk/gtk.h>
```

Description

The functions and objects described here make working with GTK+ and GIO more convenient.

GtkMountOperation is needed when mounting volumes: It is an implementation of GMountOperation that can be used with GIO functions for mounting volumes such as `g_file_mount_enclosing_volume()`, `g_file_mount_mountable()`, `g_volume_mount()`, `g_mount_unmount_with_operation()` and others.

When necessary, GtkMountOperation shows dialogs to ask for passwords, questions or show processes

blocking unmount.

gtk_show_uri_on_window() is a convenient way to launch applications for URIs.

Another object that is worth mentioning in this context is GdkAppLaunchContext, which provides visual feedback when lauching applications.

Functions

gtk_mount_operation_new ()

```
GMountOperation *
gtk_mount_operation_new (GtkWindow *parent);
```

Creates a new GtkMountOperation

Parameters

parent	transient parent of the window, or `NULL`.	[allow-none]

Returns

a new GtkMountOperation

Since: 2.14

---

gtk_mount_operation_is_showing ()

```
gboolean
gtk_mount_operation_is_showing (GtkMountOperation *op);
```

Returns whether the GtkMountOperation is currently displaying a window.

Parameters

op	a GtkMountOperation

Returns

`TRUE` if `op` is currently displaying a window

Since: 2.14

---

gtk_mount_operation_set_parent ()

```
void
gtk_mount_operation_set_parent (GtkMountOperation *op,
                                GtkWindow *parent);
```

Sets the transient parent for windows shown by the GtkMountOperation.

Parameters

op	a GtkMountOperation	
parent	transient parent of the window, or `NULL`.	[allow-none]

Since: 2.14

---

gtk_mount_operation_get_parent ()

```
GtkWindow *
gtk_mount_operation_get_parent (GtkMountOperation *op);
```

Gets the transient parent used by the GtkMountOperation

Parameters

op	a GtkMountOperation

Returns

the transient parent for windows shown by `op` .

[transfer none]

Since: 2.14

---

gtk_mount_operation_set_screen ()

```
void
gtk_mount_operation_set_screen (GtkMountOperation *op,
                                GdkScreen *screen);
```

Sets the screen to show windows of the GtkMountOperation on.

Parameters

op	a GtkMountOperation
screen	a GdkScreen

Since: 2.14

---

gtk_mount_operation_get_screen ()

```
GdkScreen *
gtk_mount_operation_get_screen (GtkMountOperation *op);
```

Gets the screen on which windows of the GtkMountOperation will be shown.

Parameters

op	a GtkMountOperation	

Returns

the screen on which windows of op are shown.

[transfer none]

Since: 2.14

---

gtk_show_uri ()

```
gboolean
gtk_show_uri (GdkScreen *screen,
              const gchar *uri,
              guint32 timestamp,
              GError **error);
```

`gtk_show_uri` is deprecated and should not be used in newly-written code.

A convenience function for launching the default application to show the uri. Like `gtk_show_uri_on_window()`, but takes a screen as transient parent instead of a window.

Note that this function is deprecated as it does not pass the necessary information for helpers to parent their dialog properly, when run from sandboxed applications for example.

Parameters

screen	screen to show the uri on or `NULL` for the default screen.	[allow-none]
uri	the uri to show	
timestamp	a timestamp to prevent focus stealing	
error	a GError that is returned in case of errors	

Returns

`TRUE` on success, `FALSE` on error

Since: 2.14

gtk_show_uri_on_window ()

```
gboolean
gtk_show_uri_on_window (GtkWindow *parent,
                        const char *uri,
                        guint32 timestamp,
                        GError **error);
```

This is a convenience function for launching the default application to show the uri. The uri must be of a form understood by GIO (i.e. you need to install gvfs to get support for uri schemes such as http:// or ftp://, as only local files are handled by GIO itself). Typical examples are

- `file:///home/gnome/pict.jpg`
- `http://www.gnome.org`
- `mailto:me@gnome.org`

Ideally the timestamp is taken from the event triggering the `gtk_show_uri()` call. If timestamp is not known you can take `GDK_CURRENT_TIME`.

This is the recommended call to be used as it passes information necessary for sandbox helpers to parent their dialogs properly.

Parameters

parent	parent window.	[allow-none]
uri	the uri to show	
timestamp	a timestamp to prevent focus stealing	
error	a GError that is returned in case of errors	

Returns

`TRUE` on success, `FALSE` on error

Since: 3.22

Types and Values

struct GtkMountOperation

```
struct GtkMountOperation;
```

This should not be accessed directly. Use the accessor functions below.

struct GtkMountOperationClass

```
struct GtkMountOperationClass {
  GMountOperationClass parent_class;
};
```

Members

Property Details

The “is-showing” property

```
  “is-showing”                gboolean
```

Are we showing a dialog.

Flags: Read

Default value: FALSE

The “parent” property

```
  “parent”                    GtkWindow *
```

The parent window.

Flags: Read / Write

The “screen” property

```
  “screen”                    GdkScreen *
```

The screen where this window will be displayed.

Flags: Read / Write

GTK+ CSS Overview

GTK+ CSS Overview — Overview of CSS in GTK+

Overview of CSS in GTK+

This chapter describes in detail how GTK+ uses CSS for styling and layout.

We loosely follow the CSS value definition specification in the formatting of syntax productions.

Nonterminals are enclosed in angle backets (⟨⟩), all other strings that are not listed here are literals
Juxtaposition means all components must occur, in the given order
A double ampersand (&&) means all components must occur, in any order
A double bar (||) means one or more of the components must occur, in any order
A single bar (|) indicates an alternative; exactly one of the components must occur
Brackets ([]) are used for grouping
A question mark (?) means that the preceding component is optional
An asterisk (*) means zero or more copies of the preceding component
A plus (+) means one or more copies of the preceding component
A number in curly braces ({n}) means that the preceding component occurs exactly n times
Two numbers in curly braces ({m,n}) mean that the preceding component occurs at least m times and at most n times

CSS nodes

GTK+ applies the style information found in style sheets by matching the selectors against a tree of nodes. Each node in the tree has a name, a state and possibly style classes. The children of each node are linearly ordered.

Every widget has one or more of these CSS nodes, and determines their name, state, style classes and how they are layed out as children and siblings in the overall node tree. The documentation for each widget explains what CSS nodes it has.

Example 6. The CSS nodes of a GtkScale

```
scale[.fine-tune]
├── marks.top
│   ├── mark
┆   ┆
│   └── mark
├── trough
│   ├── slider
│   ├── [highlight]
│   └── [fill]
└── marks.bottom
    ├── mark
    ┆
    └── mark
```

Style sheets

The basic structure of the style sheets understood by GTK+ is a series of statements, which are either rule sets or “@-rules”, separated by whitespace.

A rule set consists of a selector and a declaration block, which is a series of declarations enclosed in curly braces. The declarations are separated by semicolons. Multiple selectors can share the same declaration block, by putting all the separators in front of the block, separated by commas.

Example 7. A rule set with two selectors

```
button, entry {
  color: #ff00ea;
  font: 12px "Comic Sans";
}
```

Importing style sheets

GTK+ supports the CSS @import rule, in order to load another style sheet in addition to the currently parsed one.

The syntax for @import rules is as follows:

```
⟨import rule⟩ = @import [ ⟨url⟩ | ⟨string⟩ ]
⟨url⟩ = url( ⟨string⟩ )
```

Example 8. An example for using the @import rule

```
@import url("path/to/common.css");
```

To learn more about the @import rule, you can read the Cascading module of the CSS specification.

Selectors

Selectors work very similar to the way they do in CSS.

All widgets have one or more CSS nodes with element names and style classes. When style classes are used in selectors, they have to be prefixed with a period. Widget names can be used in selectors like IDs. When used in a selector, widget names must be prefixed with a # character.

In more complicated situations, selectors can be combined in various ways. To require that a node satisfies several conditions, combine several selectors into one by concatenating them. To only match a node when it occurs inside some other node, write the two selectors after each other, separated by whitespace. To restrict the match to direct children of the parent node, insert a > character between the two selectors.

Example 9. Theme labels that are descendants of a window

```
window label {
  background-color: #898989;
}
```

Example 10. Theme notebooks, and anything within

```
notebook {
  background-color: #a939f0;
}
```

Example 11. Theme combo boxes, and entries that are direct children of a notebook

```
combobox,
notebook > entry {
  color: @fg_color;
  background-color: #1209a2;
}
```

Example 12. Theme any widget within a GtkBox

```
box * {
  font: 20px Sans;
}
```

Example 13. Theme a label named title-label

```
label#title-label {
  font: 15px Sans;
}
```

Example 14. Theme any widget named main-entry

```
#main-entry {
  background-color: #f0a810;
}
```

Example 15. Theme all widgets with the style class entry

```
.entry {
  color: #39f1f9;
}
```

Example 16. Theme the entry of a GtkSpinButton

```
spinbutton entry {
  color: #900185;
}
```

It is possible to select CSS nodes depending on their position amongst their siblings by applying pseudo-classes to the selector, like :first-child, :last-child or :nth-child(even). When used in selectors, pseudo-classes must be prefixed with a : character.

Example 17. Theme labels in the first notebook tab

```
notebook tab:first-child label {
  color: #89d012;
}
```

Another use of pseudo-classes is to match widgets depending on their state. The available pseudo-classes for widget states are :active, :hover :disabled, :selected, :focus, :indeterminate, :checked and :backdrop. In addition, the following pseudo-classes don't have a direct equivalent as a widget state: :dir(ltr) and :dir(rtl) (for text direction), :link and :visited (for links) and :drop(active) (for highlighting drop targets). Widget state pseudo-classes may only apply to the last element in a selector.

Example 18. Theme pressed buttons

```
button:active {
  background-color: #0274d9;
}
```

Example 19. Theme buttons with the mouse pointer over it

```
button:hover {
  background-color: #3085a9;
}
```

Example 20. Theme insensitive widgets

```
*:disabled {
  background-color: #320a91;
}
```

Example 21. Theme checkbuttons that are checked

```
checkbutton:checked {
```

```
2                                                   background-color: #56f9a0;
3                                                 }
```

Example 22. Theme focused labels

```
1                                                 label:focus {
2                                                   background-color: #b4940f;
3                                                 }
```

Example 23. Theme inconsistent checkbuttons

```
1                                                 checkbutton:indeterminate {
2                                                   background-color: #20395a;
3                                                 }
```

To determine the effective style for a widget, all the matching rule sets are merged. As in CSS, rules apply by specificity, so the rules whose selectors more closely match a node will take precedence over the others.

The full syntax for selectors understood by GTK+ can be found in the table below. The main difference to CSS is that GTK+ does not currently support attribute selectors.

Table 1. Selector syntax

Pattern	Matches	Reference	Notes
*	any node	CSS	
E	any node with name E	CSS	
E.class	any E node with the given style class	CSS	
E#id	any E node with the given ID	CSS	GTK+ uses the widget name as ID
E:nth-child(⟨nth-child⟩)	any E node which is the n-th child of its parent node	CSS	
E:nth-last-child(⟨nth-child⟩)	any E node which is the n-th child of its parent node, counting from the end	CSS	
E:first-child	any E node which is the first child of its parent node	CSS	
E:last-child	any E node which is the last child of its parent node	CSS	
E:only-child	any E node which is the only child of its parent node	CSS	Equivalent to E:first-child:last-child
E:link, E:visited	any E node which represents a hyperlink, not yet visited (:link) or already visited (:visited)	CSS	Corresponds to GTK_STATE_FLAG_LINK and GTK_STATE_FLAGS_VISITED
E:active, E:hover, E:focus	any E node which is part of a widget	CSS	Corresponds to

Pattern	Matches	Reference	Notes
	with the corresponding state		GTK_STATE_FLAG_ACTIVE, GTK_STATE_FLAG_PRELIGHT and GTK_STATE_FLAGS_FOCUSED; GTK+ also allows E:prelight and E:focused
E:disabled	any E node which is part of a widget which is disabled	CSS	Corresponds to GTK_STATE_FLAG_INSENSITIVE; GTK+ also allows E:insensitive
E:checked	any E node which is part of a widget (e.g. radio- or checkbuttons) which is checked	CSS	Corresponds to GTK_STATE_FLAG_CHECKED
E:indeterminate	any E node which is part of a widget (e.g. radio- or checkbuttons) which is in an indeterminate state	CSS3, CSS4	Corresponds to GTK_STATE_FLAG_INCONSISTENT; GTK+ also allows E:inconsistent
E:backdrop, E:selected	any E node which is part of a widget with the corresponding state		Corresponds to GTK_STATE_FLAG_BACKDROP, GTK_STATE_FLAG_SELECTED
E:not(⟨selector⟩)	any E node which does not match the simple selector ⟨selector⟩	CSS	
E:dir(ltr), E:dir(rtl)	any E node that has the corresponding text direction	CSS4	
E:drop(active)	any E node that is an active drop target for a current DND operation	CSS4	
E F	any F node which is a descendent of an E node	CSS	
E > F	any F node which is a child of an E node	CSS	
E ~ F	any F node which is preceded by an E node	CSS	
E + F	any F node which is immediately preceded by an E node	CSS	

```
⟨nth-child⟩ = even | odd | ⟨integer⟩ | ⟨integer⟩n | ⟨integer⟩n [ + | - ] ⟨integer⟩
```

To learn more about selectors in CSS, read the Selectors module of the CSS specification.

Colors

CSS allows to specify colors in various ways, using numeric values or names from a predefined list of colors.

```
⟨color⟩ = currentColor | transparent | ⟨color name⟩ | ⟨rgb color⟩ | ⟨rgba color⟩ | ⟨hex co
lor⟩ | ⟨gtk color⟩
⟨rgb color⟩ = rgb( ⟨number⟩, ⟨number⟩, ⟨number⟩ ) | rgb( ⟨percentage⟩, ⟨percentage⟩, ⟨perc
entage⟩ )
⟨rgba color⟩ = rgba( ⟨number⟩, ⟨number⟩, ⟨number⟩, ⟨alpha value⟩ ) | rgba( ⟨percentage⟩, ⟨
percentage⟩, ⟨percentage⟩, ⟨alpha value⟩ )
⟨hex color⟩ = #⟨hex digits⟩
```

⟨alpha value⟩ = ⟨number⟩, clamped to values between 0 and 1

The keyword currentColor resolves to the current value of the color property when used in another property, and to the inherited value of the color property when used in the color property itself.

The keyword transparent can be considered a shorthand for rgba(0,0,0,0).

For a list of valid color names and for more background on colors in CSS, see the Color module of the CSS specification.

Example 24. Specifying colors in various ways

```
1                                                color: transparent;
2                                                background-color: red;
3                                                border-top-color: rgb(128,57,0);
4                                                border-left-color: rgba(10%,20%,30%,0.5);
5                                                border-right-color: #ff00cc;
6                                                border-bottom-color: #ffff0000cccc;
```

GTK+ adds several additional ways to specify colors.

```
⟨gtk color⟩ = ⟨symbolic color⟩ | ⟨color expression⟩ | ⟨win32 color⟩
```

The first is a reference to a color defined via a @define-color rule. The syntax for @define-color rules is as follows:

```
⟨define color rule⟩ = @define-color ⟨name⟩ ⟨color⟩
```

To refer to the color defined by a @define-color rule, use the name from the rule, prefixed with @.

```
⟨symbolic color⟩ = @⟨name⟩
```

Example 25. An example for defining colors

```
1                                                @define-color bg_color #f9a039;
2
3                                                * {
4                                                  background-color: @bg_color;
5                                                }
```

GTK+ also supports color expressions, which allow colors to be transformed to new ones and can be nested, providing a rich language to define colors. Color expressions resemble functions, taking 1 or more colors and in some cases a number as arguments.

shade() leaves the color unchanged when the number is 1 and transforms it to black or white as the number approaches 0 or 2 respectively. For mix(), 0 or 1 return the unaltered 1st or 2nd color respectively; numbers between 0 and 1 return blends of the two; and numbers below 0 or above 1 intensify the RGB components of the 1st or 2nd color respectively. alpha() takes a number from 0 to 1 and applies that as the opacity of the supplied color.

```
⟨color expression⟩ = lighter( ⟨color⟩ ) | darker( ⟨color⟩ ) | shade( ⟨color⟩, ⟨number⟩ ) |
                     alpha( ⟨color⟩, ⟨number⟩ ) | mix( ⟨color⟩, ⟨color⟩, ⟨number⟩ )
```

On Windows, GTK+ allows to refer to system colors, as follows:

```
⟨win32 color⟩ = -gtk-win32-color( ⟨name⟩, ⟨integer⟩ )
```

Images

CSS allows to specify images in various ways, for backgrounds and borders.

```
⟨image⟩ = ⟨url⟩ | ⟨crossfade⟩ | ⟨alternatives⟩ | ⟨gradient⟩ | ⟨gtk image⟩
⟨crossfade⟩ = cross-fade( ⟨percentage⟩, ⟨image⟩, ⟨image⟩ )
⟨alternatives⟩ = image([ ⟨image⟩, ]* [ ⟨image⟩ | ⟨color⟩ ] )
⟨gradient⟩ = ⟨linear gradient⟩ | ⟨radial gradient⟩
⟨linear gradient⟩ = [ linear-gradient | repeating-linear-gradient ] (
                      [ [ ⟨angle⟩ | to ⟨side or corner⟩ ] , ]?
                      ⟨color stops⟩ )
⟨radial gradient⟩ = [ radial-gradient | repeating-radial-gradient ] (
                      [ [ ⟨shape⟩ || ⟨size⟩ ] [ at ⟨position⟩ ]? , | at ⟨position⟩, ]?
                      ⟨color stops⟩ )
⟨side or corner⟩ = [ left | right ] || [ top | bottom ]
⟨color stops⟩ =  ⟨color stop⟩ [ , ⟨color stop⟩ ]+
⟨color stop⟩ = ⟨color⟩ [ ⟨percentage⟩ | ⟨length⟩ ]?
⟨shape⟩ = circle | ellipse
⟨size⟩ = ⟨extent keyword⟩ | ⟨length⟩ | [ ⟨length⟩ | ⟨percentage⟩ ]{1,2}
⟨extent keyword⟩ = closest-size | farthest-side | closest-corner | farthest-corner
```

The simplest way to specify an image in CSS is to load an image file from a URL. CSS does not specify anything about supported file formats; within GTK+, you can expect at least PNG, JPEG and SVG to work. The full list of supported image formats is determined by the available gdk-pixbuf image loaders and may vary between systems.

Example 26. Loading an image file

```
1
2
3
```

```
button {
  background-image: url("water-lily.png");
}
```

A crossfade lets you specify an image as an intermediate between two images. Crossfades are specified in the draft of the level 4 Image module of the CSS specification.

Example 27. Crossfading two images

```
button {
  background-image: cross-fade(50%,
url("water-lily.png"), url("buffalo.jpg"));
}
```

The image() syntax provides a way to specify fallbacks in case an image format may not be supported. Multiple fallback images can be specified, and will be tried in turn until one can be loaded successfully. The last fallback may be a color, which will be rendered as a solid color image.

Example 28. Image fallback

```
button {
  background-image: image(url("fancy.svg"),
url("plain.png"), green);
}
```

Gradients are images that smoothly fades from one color to another. CSS provides ways to specify repeating and non-repeating linear and radial gradients. Radial gradients can be circular, or axis-aligned ellipses. In addition to CSS gradients, GTK+ has its own -gtk-gradient extensions.

A linear gradient is created by specifying a gradient line and then several colors placed along that line. The gradient line may be specified using an angle, or by using direction keywords.

Example 29. Linear gradients

```
button {
  background-image: linear-gradient(45deg, yellow, blue);
}
label {
  background-image: linear-gradient(to top right, blue 20%, #f0f 80%);
}
```

A radial gradient is created by specifying a center point and one or two radii. The radii may be given explicitly as lengths or percentages or indirectly, by keywords that specify how the end circle or ellipsis should be positioned relative to the area it is derawn in.

Example 30. Radial gradients

```
button {
  background-image: radial-gradient(ellipse at center, yellow 0%, green 100%);
}
```

```
4   label {
5     background-image: radial-gradient(circle farthest-side at left bottom, red, yellow
6   50px, green);
    }
```

To learn more about gradients in CSS, including details of how color stops are placed on the gradient line and keywords for specifying radial sizes, you can read the Image module of the CSS specification.

GTK+ extends the CSS syntax for images and also uses it for specifying icons.

```
⟨gtk image⟩ = ⟨gtk gradient⟩ | ⟨themed icon⟩ | ⟨scaled image⟩ | ⟨recolored image⟩ | ⟨win32
theme part⟩
```

GTK+ supports an alternative syntax for linear and radial gradients (which was implemented before CSS gradients were supported).

```
⟨gtk gradient⟩ = ⟨gtk linear gradient⟩ | ⟨gtk radial gradient⟩
⟨gtk linear gradient⟩ = -gtk-gradient(linear,
                          [ ⟨x position⟩ ⟨y position⟩ , ]{2}
                          ⟨gtk color stops⟩ )
⟨gtk radial gradient⟩ = -gtk-gradient(radial,
                          [ ⟨x position⟩ ⟨y position⟩ , ⟨radius⟩ , ]{2}
                          ⟨gtk color stops⟩ )
⟨x position⟩ = left | right | center | ⟨number⟩
⟨y position⟩ = top | bottom | center | ⟨number⟩
⟨radius ⟩ = ⟨number⟩
⟨gtk color stops⟩ = ⟨gtk color stop⟩ [ , ⟨gtk color stop⟩ ]+
⟨gtk color stop⟩ = color-stop( ⟨number⟩ , ⟨color⟩ ) | from( ⟨color⟩ ) | to( ⟨color⟩ )
```

The numbers used to specify x and y positions, radii, as well as the positions of color stops, must be between 0 and 1. The keywords for for x and y positions (left, right, top, bottom, center), map to numeric values of 0, 1 and 0.5 in the obvious way. Color stops using the from() and to() syntax are abbreviations for color-stop with numeric positions of 0 and 1, respectively.

Example 31. Linear gradients

```
1   button {
2     background-image: -gtk-gradient (linear,
3                                      left top, right bottom,
4                                      from(@yellow), to(@blue));
5   }
6   label {
7     background-image: -gtk-gradient (linear,
8                                      0 0, 0 1,
9                                      color-stop(0, @yellow),
10                                     color-stop(0.2, @blue),
11                                     color-stop(1, #0f0));
12  }
```

Example 32. Radial gradients

```
button {
  background-image: -gtk-gradient (radial,
                                   center center, 0,
                                   center center, 1,
                                   from(@yellow), to(@green));
}
label {
  background-image: -gtk-gradient (radial,
                                   0.4 0.4, 0.1,
                                   0.6 0.6, 0.7,
                                   color-stop(0, #f00),
                                   color-stop(0.1, $a0f),
                                   color-stop(0.2, @yellow),
                                   color-stop(1, @green));
}
```

GTK+ has extensive support for loading icons from icon themes. It is accessible from CSS with the -gtk-icontheme syntax.

```
⟨themed icon⟩ = -gtk-icontheme( ⟨icon name⟩ )
```

The specified icon name is used to look up a themed icon, while taking into account the values of the -gtk-icon-theme and -gtk-icon-palette properties. This kind of image is mainly used as value of the -gtk-icon-source property.

Example 33. Using themed icons in CSS

```
spinner {
  -gtk-icon-source: -gtk-icontheme('process-working-symbolic');
  -gtk-icon-palette: success blue, warning #fc3, error magenta;
}
arrow.fancy {
  -gtk-icon-source: -gtk-icontheme('pan-down');
  -gtk-icon-theme: 'Oxygen';
}
```

GTK+ supports scaled rendering on hi-resolution displays. This works best if images can specify normal and hi-resolution variants. From CSS, this can be done with the -gtk-scaled syntax.

```
⟨scaled image⟩ = -gtk-scaled( ⟨image⟩[ , ⟨image⟩ ]* )
```

While -gtk-scaled accepts multiple higher-resolution variants, in practice, it will mostly be used to specify a regular image and one variant for scale 2.

Example 34. Scaled images in CSS

```
arrow {
  -gtk-icon-source: -gtk-scaled(url('my-arrow.png'),
                                url('my-arrow@2.png'));
}
```

```
⟨recolored image⟩ = -gtk-recolor( ⟨url⟩ [ , ⟨color palette⟩ ] )
```

Symbolic icons from the icon theme are recolored according to the -gtk-icon-palette property. The recoloring is sometimes needed for images that are not part of an icon theme, and the -gtk-recolor syntax makes this available. -gtk-recolor requires a url as first argument. The remaining arguments specify the color palette to use. If the palette is not explicitly specified, the current value of the -gtk-icon-palette property is used.

Example 35. Recoloring an image

```
1   arrow {
2     -gtk-icon-source: -gtk-recolor(url('check.svg'), success blue, error rgb(255,0,0));
3   }
```

On Windows, GTK+ allows to refer to system theme parts as images, as follows:

```
⟨win32 theme part⟩ = -gtk-win32-theme-part( ⟨name⟩, ⟨integer⟩ ⟨integer⟩
                                                [ , [ over( ⟨integer⟩ ⟨integer⟩ [ , ⟨alpha v
alue⟩ ]? ) | margins( ⟨integer⟩{1,4} ) ] ]* )
```

Transitions

CSS defines a mechanism by which changes in CSS property values can be made to take effect gradually, instead of all at once. GTK+ supports these transitions as well.

To enable a transition for a property when a rule set takes effect, it needs to be listed in the transition-property property in that rule set. Only animatable properties can be listed in the transition-property.

The details of a transition can modified with the transition-duration, transition-timing-function and transition-delay properties.

To learn more about transitions, you can read the Transitions module of the CSS specification.

Animations

In addition to transitions, which are triggered by changes of the underlying node tree, CSS also supports defined animations. While transitions specify how property values change from one value to a new value, animations explicitly define intermediate property values in keyframes.

Keyframes are defined with an @-rule which contains one or more of rule sets with special selectors. Property declarations for nonanimatable properties are ignored in these rule sets (with the exception of animation properties).

```
⟨keyframe rule⟩ = @keyframes ⟨name⟩ { ⟨animation rule⟩ }
⟨animation rule⟩ = ⟨animation selector⟩ { ⟨declaration⟩* }
⟨animation selector⟩ = ⟨single animation selector⟩ [ , ⟨single animation selector⟩ ]*
```

```
⟨single animation selector⟩ = from | to | ⟨percentage⟩
```

To enable an animation, the name of the keyframes must be set as the value of the animation-name property. The details of the animation can modified with the animation-duration, animation-timing-function, animation-iteration-count and other animation properties.

Example 36. A CSS animation

```
@keyframes spin {
  to { -gtk-icon-transform: rotate(1turn); }
}

spinner {
  animation-name: spin;
  animation-duration: 1s;
  animation-timing-function: linear;
  animation-iteration-count: infinite;
}
```

To learn more about animations, you can read the Animations module of the CSS specification.

Key bindings

In order to extend key bindings affecting different widgets, GTK+ supports the @binding-set rule to parse a set of bind/unbind directives. Note that in order to take effect, the binding sets defined in this way must be associated with rule sets by setting the -gtk-key-bindings property.

The syntax for @binding-set rules is as follows:

```
⟨binding set rule⟩ = @binding-set ⟨binding name⟩ { [ [ ⟨binding⟩ | ⟨unbinding⟩ ] ; ]* }
⟨binding⟩ = bind "⟨accelerator⟩" { ⟨signal emission⟩* }
⟨signal emission⟩ = "⟨signal name⟩" ( [ ⟨argument⟩ [ , ⟨argument⟩ ]* ]? }
⟨unbinding⟩ = unbind "⟨accelerator⟩"
```

where ⟨accelerator⟩ is a string that can be parsed by gtk_accelerator_parse(), ⟨signal name⟩ is the name of a keybinding signal of the widget in question, and the ⟨argument⟩ list must be according to the signals declaration.

Example 37. An example for using the @binding-set rule

```
@binding-set binding-set1 {
  bind "<alt>Left" { "move-cursor" (visual-positions, -3, 0) };
  unbind "End";
};

@binding-set binding-set2 {
  bind "<alt>Right" { "move-cursor" (visual-positions, 3, 0) };
  bind "<alt>KP_space" { "delete-from-cursor" (whitespace, 1)
                         "insert-at-cursor" (" ") };
};
```

```
11
12   entry {
13     -gtk-key-bindings: binding-set1, binding-set2;
14   }
```

GTK+ CSS Properties

GTK+ CSS Properties — CSS Properties in GTK+

Supported CSS Properties

GTK+ supports CSS properties and shorthands as far as they can be applied in the context of widgets, and adds its own properties only when needed. All GTK+-specific properties have a -gtk prefix.

All properties support the following keywords: inherit, initial, unset, with the same meaning as in CSS.

The following basic datatypes are used throughout:

```
⟨length⟩ = ⟨number⟩ [ px | pt | em | ex |rem | pc | in | cm | mm ] | ⟨calc expression⟩
⟨percentage⟩ = ⟨number⟩ % | ⟨calc expression⟩
⟨angle⟩ = ⟨number⟩ [ deg | grad | turn ] | ⟨calc expression⟩
⟨time⟩ = ⟨number⟩ [ s | ms ] | ⟨calc expression⟩
```

Length values with the em or ex units are resolved using the font size value, unless they occur in setting the font-size itself, in which case they are resolved using the inherited font size value.

The rem unit is resolved using the initial font size value, which is not quite the same as the CSS definition of rem.

Whereever a number is allowed, GTK+ also accepts a Windows-specific theme size:

```
⟨win32 theme size⟩ = ⟨win32 size⟩ | ⟨win32 part size⟩
⟨win32 size⟩ = -gtk-win32-size ( ⟨theme name⟩, ⟨metric id⟩ )
⟨win32 part size⟩ = [ -gtk-win32-part-width | -gtk-win32-part-height |
                      -gtk-win32-part-border-top | -gtk-win32-part-border-right |
                      -gtk-win32-part-border-bottom | -gtk-win32-part-border-left ]
                    ( ⟨theme name⟩ , ⟨integer⟩ , ⟨integer⟩ )
```

```
⟨calc expression⟩ = calc( ⟨calc sum⟩ )
⟨calc sum⟩ = ⟨calc product⟩ [ [ + | - ] ⟨calc product⟩ ]*
⟨calc product⟩ = ⟨calc value⟩ [ * ⟨calc value⟩ | / ⟨number⟩ ]*
⟨calc value⟩ = ⟨number⟩ | ⟨length⟩ | ⟨percentage⟩ | ⟨angle⟩ | ⟨time⟩ | ( ⟨calc sum⟩ )
```

The calc() notation adds considerable expressive power. There are limits on what types can be combined in such an expression (e.g. it does not make sense to add a number and a time). For the full details, see the CSS3 Values and Units spec.

A common pattern among shorthand properties (called 'four sides') is one where one to four values can be specified, to determine a value for each side of an area. In this case, the specified values are interpreted as follows:

4 values:	top right bottom left
3 values:	top horizontal bottom
2 values:	vertical horizontal
1 value:	all

Table 2. Color Properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
color	⟨color⟩	rgba(1,1,1,1)	✓	✓	CSS2, CSS3	
opacity	⟨alpha value⟩	1		✓	CSS3	

The color property specifies the color to use for text, icons and other foreground rendering. The opacity property specifies the opacity that is used to composite the widget onto its parent widget.

Table 3. Font Properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
font-family	⟨family name⟩ [, ⟨family name⟩]*	gtk-font-name setting	✓		CSS2, CSS3	
font-size	⟨absolute size⟩ | ⟨relative size⟩ | ⟨length⟩ | ⟨percentage⟩	gtk-font-name setting	✓	✓	CSS2, CSS3	
font-style	normal | oblique | italic	normal	✓		CSS2, CSS3	
font-variant	normal | small-caps	normal	✓		CSS2, CSS3	only CSS2 values supported
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900	normal	✓	✓	CSS2, CSS3	normal is synonymous with 400, bold with 700
font-stretch	ultra-condensed | extra-condensed | condensed | semi-condensed | normal | semi-expanded | expanded | extra-expanded | ultra-expanded	normal	✓		CSS3	
-gtk-dpi	⟨number⟩	screen resolution	✓	✓		

Shorthand	Value	Initial	Reference	Notes
font	`[ ⟨font-style⟩ || ⟨font-variant⟩ || ⟨font-weight⟩ || ⟨font-stretch⟩ ]? ⟨font-size⟩ ⟨font-family⟩`	see individual properties	CSS2, CSS3	CSS allows line-height, etc

```
⟨absolute size⟩ = xx-small | x-small | small | medium | large | x-large | xx-large
⟨relative size⟩ = larger | smaller
```

The font properties determine the font to use for rendering text. Relative font sizes and weights are resolved relative to the inherited value for these properties.

Table 4. Text caret properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
caret-color	`⟨color⟩`	`currentColor`	✓	✓	CSS3	CSS allows an auto value
-gtk-secondary-caret-color	`⟨color⟩`	`currentColor`	✓	✓		Used for the secondary caret in bidirectional text

The caret properties provide a way to change the appearance of the insertion caret in editable text.

Table 5. Text decoration properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
letter-spacing	`⟨length⟩`	`0px`	✓	✓	CSS3	
text-decoration-line	`none | underline | line-through`	`none`			CSS2, CSS3	CSS allows overline
text-decoration-color	`⟨color⟩`	`currentColor`		✓	CSS3	
text-decoration-style	`solid | double | wavy`	`solid`			CSS3	CSS allows dashed and dotted
text-shadow	`none | ⟨shadow⟩`	`none`	✓	✓	CSS3	

Shorthand	Value	Initial	Reference	Notes
text-decoration	`⟨text-decoration-line⟩ || ⟨text-decoration-style⟩ || ⟨text-decoration-color⟩`	see individual properties	CSS3	

```
⟨shadow⟩ = ⟨length⟩ ⟨length⟩ ⟨color⟩?
```

The text decoration properties determine whether to apply extra decorations when rendering text.

Table 6. Icon Properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
-gtk-icon-source	`builtin | ⟨image⟩ | none`	`builtin`		✓		
-gtk-icon-transform	`none | ⟨transform⟩+`	`none`		✓		
-gtk-icon-style	`requested | regular | symbolic`	`requested`	✓			Determines the preferred style for application-loaded icons
-gtk-icon-theme	`⟨name⟩`	current icon theme	✓			The icon theme to use with -gtk-icontheme(). Since 3.20
-gtk-icon-palette	`⟨color palette⟩`	`default`	✓	✓		Used to recolor symbolic icons (both application-loaded and from -gtk-icontheme()). Since 3.20
-gtk-icon-shadow	`none | ⟨shadow⟩`	`none`	✓	✓		
-gtk-icon-effect	`none | highlight | dim`	`none`	✓			

```
⟨transform⟩ = matrix( ⟨number⟩ [ , ⟨number⟩ ]{5} ) | translate( ⟨length⟩, ⟨length⟩ ) | tra
nslateX( ⟨length⟩ ) | translateY( ⟨length⟩ ) |
                scale( ⟨number⟩ [ , ⟨number⟩ ]? ) | scaleX( ⟨number⟩ ) | scaleY( ⟨number⟩
) | rotate( ⟨angle⟩ ) | skew( ⟨angle⟩ [ , ⟨angle⟩ ]? ) |
                skewX( ⟨angle⟩ ) |  skewY( ⟨angle⟩ )
⟨color palette⟩ = default | ⟨name⟩ ⟨color⟩ [ , ⟨name⟩ ⟨color⟩ ]*
```

The -gtk-icon-source property is used by widgets that are rendering 'built-in' icons, such as arrows, expanders, spinners, checks or radios.

The -gtk-icon-style property determines the preferred style for application-provided icons.

The -gtk-icon-transform and -gtk-icon-shadow properties affect the rendering of both built-in and application-provided icons.

-gtk-icon-palette defines a color palette for recoloring symbolic icons. The recognized names for colors in symbolic icons are error, warning and success. The default palette maps these three names to symbolic colors with the names @error_color, @warning_color and @success_color respectively.

Table 7. Box properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
min-width	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages
min-height	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages
margin-top	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages or auto
margin-right	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages or auto
margin-bottom	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages or auto
margin-left	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages or auto
padding-top	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages
padding-right	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages
padding-bottom	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages
padding-left	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows percentages

Shorthand	Value	Initial	Reference	Notes
margin	⟨length⟩{1,4}	see individual properties	CSS2, CSS3	a 'four sides' shorthand
padding	⟨length⟩{1,4}	see individual properties	CSS2, CSS3	a 'four sides' shorthand

Table 8. Border properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
border-top-width	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows other values
border-right-	⟨length⟩	0px		✓	CSS2, CSS3	CSS allows

Name	Value	Initial	Inh.	Ani.	Reference	Notes
border-bottom-width	⟨length⟩	0px		✓	CSS2, CSS3	other values CSS allows
border-left-width	⟨length⟩	0px		✓	CSS2, CSS3	other values CSS allows
border-top-style	⟨border style⟩	none			CSS2, CSS3	other values
border-right-style	⟨border style⟩	none			CSS2, CSS3	
border-bottom-style	⟨border style⟩	none			CSS2, CSS3	
border-left-style	⟨border style⟩	none			CSS2, CSS3	
border-top-right-radius	⟨corner radius⟩	0		✓	CSS2, CSS3	
border-bottom-right-radius	⟨corner radius⟩	0		✓	CSS2, CSS3	
border-bottom-left-radius	⟨corner radius⟩	0		✓	CSS2, CSS3	
border-top-left-radius	⟨corner radius⟩	0		✓	CSS2, CSS3	
border-top-color	⟨color⟩	currentColor		✓	CSS2, CSS3	
border-right-color	⟨color⟩	currentColor		✓	CSS2, CSS3	
border-bottom-color	⟨color⟩	currentColor		✓	CSS2, CSS3	
border-left-color	⟨color⟩	currentColor		✓	CSS2, CSS3	
border-image-source	⟨image⟩ | none	none		✓	CSS3	
border-image-repeat	⟨border repeat⟩{1,2}	stretch			CSS3	
border-image-slice	[⟨number⟩ | ⟨percentage⟩]{1,4} && fill?	100%			CSS3	a 'four sides' shorthand
border-image-width	[⟨length⟩ | ⟨number⟩ | ⟨percentage⟩ | auto]{1,4}	1			CSS3	a 'four sides' shorthand

Shorthand	Value	Initial	Reference	Notes
border-width	⟨length⟩{1,4}	see individual properties	CSS2, CSS3	a 'four sides' shorthand
border-style	⟨border style⟩{1,4}	see individual properties	CSS2, CSS3	a 'four sides' shorthand
border-color	⟨color⟩{1,4}	see individual	CSS3	a 'four sides'

Shorthand	Value	Initial	Reference	Notes
		properties		shorthand
border-top	⟨length⟩ || ⟨border style⟩ || ⟨color⟩	see individual properties	CSS2, CSS3	
border-right	⟨length⟩ || ⟨border style⟩ || ⟨color⟩	see individual properties	CSS2, CSS3	
border-bottom	⟨length⟩ || ⟨border style⟩ || ⟨color⟩	see individual properties	CSS2, CSS3	
border-left	⟨length⟩ || ⟨border style⟩ || ⟨color⟩	see individual properties	CSS2, CSS3	
border	⟨length⟩ || ⟨border style⟩ || ⟨color⟩	see individual properties	CSS2, CSS3	
border-radius	[⟨length⟩ | ⟨percentage⟩]{1,4} [/ [⟨length⟩ | ⟨percentage⟩]{1,4}]?	see individual properties	CSS3	
border-image	⟨border-image-source⟩ || ⟨border-image-slice⟩ [/ ⟨border-image-width⟩ | / ⟨border-image-width⟩? / ⟨border-image-outset⟩]? || ⟨border-image-repeat⟩	see individual properties	CSS3	

```
⟨border style⟩ = none | solid | inset | outset | hidden | dotted | dashed | double | groov
e | ridge
⟨corner radius⟩ = [ ⟨length⟩ | ⟨percentage⟩ ]{1,2}
```

Table 9. Outline properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
outline-style	none | solid | inset | outset | hidden | dotted | dashed | double | groove | ridge	none			CSS2, CSS3	
outline-width	⟨length⟩	0px		✓	CSS2, CSS3	
outline-color	⟨color⟩	currentColor		✓	CSS2, CSS3	
outline-offset	⟨length⟩	0px			CSS3	

Name	Value	Initial	Inh.	Ani.	Reference	Notes
-gtk-outline-top-left-radius	⟨corner radius⟩	0		✓		
-gtk-outline-top-right-radius	⟨corner radius⟩	0		✓		
-gtk-outline-bottom-right-radius	⟨corner radius⟩	0		✓		
-gtk-outline-bottom-left-radius	⟨corner radius⟩	0		✓		

Shorthand	Value	Initial	Reference	Notes
outline	⟨outline-color⟩ || ⟨outline-style⟩ || ⟨outline-width⟩	see individual properties	CSS2, CSS3	
-gtk-outline-radius	[⟨length⟩ | ⟨percentage⟩]{1,4} [/ [⟨length⟩ | ⟨percentage⟩]{1,4}]?	see individual properties		

GTK+ uses the CSS outline properties to render the 'focus rectangle'.

Table 10. Background properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
background-color	⟨color⟩	rgba(0,0,0,0)		✓	CSS2, CSS3	
background-clip	⟨box⟩ [, ⟨box⟩]*	border-box			CSS3	
background-origin	⟨box⟩ [, ⟨box⟩]*	padding-box			CSS3	
background-size	⟨bg-size⟩ [, ⟨bg-size⟩]*	auto		✓	CSS3	
background-position	⟨position⟩ [, ⟨position⟩]*	0		✓	CSS2, CSS3	
background-repeat	⟨bg-repeat⟩ [, ⟨bg-repeat⟩]*	repeat			CSS2, CSS3	
background-image	⟨bg-image⟩ [, ⟨bg-image⟩]*	none		✓	CSS2, CSS3	not supported: urls without quotes, CSS radial gradients, colors in crossfades

Name	Value	Initial	Inh.	Ani.	Reference	Notes
background-blend-mode	⟨blend-mode⟩ [, ⟨blend-mode⟩]*	normal				only affects multiple backgrounds
box-shadow	none | ⟨box shadow⟩ [, ⟨box shadow⟩]*	none		✓	CSS3	

Shorthand	Value	Initial	Reference	Notes
background	[⟨bg-layer⟩ ,]* ⟨final-bg-layer⟩	see individual properties	CSS2, CSS3	

```
⟨box⟩ = border-box | padding-box | content-box
⟨bg-size⟩ = [ ⟨length⟩ | ⟨percentage⟩ | auto ]{1,2} | cover | contain
⟨position⟩ = [ left | right | center | top | bottom | ⟨percentage⟩ | ⟨length⟩ ]{1,2,3,4}
⟨bg-repeat⟩ = repeat-x | repeat-y | [ no-repeat | repeat | round | space ]{1,2}
⟨bg-image⟩ = ⟨image⟩ | none
⟨bg-layer⟩ = ⟨bg-image⟩ || ⟨position⟩ [ / ⟨bg-size⟩ ]? || ⟨bg-repeat⟩ || ⟨box⟩ || ⟨box⟩
⟨final-bg-layer⟩ = ⟨bg-image⟩ || ⟨position⟩ [ / ⟨bg-size⟩ ]? || ⟨bg-repeat⟩ || ⟨box⟩ || ⟨b
ox⟩ || ⟨color⟩
⟨blend-mode⟩ = color || color-burn || color-dodge || darken || difference || exclusion ||
               hard-light || hue || lighten || luminosity || multiply || normal || overlay
||
               saturate || screen || soft-light
⟨box shadow⟩ = inset? && ⟨length⟩{2,4}? && ⟨color⟩?
```

As in CSS, the background color is rendered underneath all the background image layers, so it will only be visible if background images are absent or have transparency.

Alternatively, multiple backgrounds can be blended using the `background-blend-mode` property.

Table 11. Transition properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
transition-property	none | all | ⟨property name⟩ [, ⟨property name⟩]*	all			CSS3	
transition-duration	⟨time⟩ [, ⟨time⟩]*	0s			CSS3	
transition-timing-function	⟨single-timing-function⟩ [, ⟨single-timing-function⟩]*	ease			CSS3	
transition-delay	⟨time⟩ [, ⟨time⟩]*	0s			CSS3	

Shorthand	Value	Initial	Reference	Notes
transition	⟨single-transition⟩ [, ⟨single-	see individual properties	CSS3	

Shorthand	Value	Initial	Reference	Notes
	transition〉]*			

```
〈single-timing-function〉 = ease | linear | ease-in | ease-out | ease-in-out |
                            step-start | step-end | steps( 〈integer〉 [ , [ start | end ] ]?
) |
                            cubic-bezier( 〈number〉, 〈number〉, 〈number〉, 〈number〉 )
〈single-transition〉 = [ none | 〈property name〉 ] || 〈time〉 || 〈single-transition-timing-
function〉 || 〈time〉
```

Table 12. Animation properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
animation-name	〈single-animation-name〉 [, 〈single-animation-name〉]*	none			CSS3	
animation-duration	〈time〉 [, 〈time〉]*	0s			CSS3	
animation-timing-function	〈single-timing-function〉 [, 〈single-timing-function〉]*	ease			CSS3	
animation-iteration-count	〈single-animation-iteration-count〉 [, 〈single-animation-iteration-count〉]*	1			CSS3	
animation-direction	〈single-animation-direction〉 [, 〈single-animation-direction〉]*	normal			CSS3	
animation-play-state	〈single-animation-play-state〉 [, 〈single-animation-play-state〉]*	running			CSS3	
animation-delay	〈time〉 [, 〈time〉]*	0s			CSS3	
animation-fill-mode	〈single-animation-fill-mode〉 [, 〈single-	none			CSS3	

Name	Value	Initial	Inh.	Ani.	Reference	Notes
	animation-fill-mode⟩]*					

Shorthand	Value	Initial	Reference	Notes
animation	⟨single-animation⟩ [, ⟨single-animation⟩]*	see individual properties	CSS3	

```
⟨single-animation-name⟩ = none | ⟨property name⟩
⟨single-animation-iteration-count⟩ = infinite | ⟨number⟩
⟨single-animation-direction⟩ = normal | reverse | alternate | alternate-reverse
⟨single-animation-play-state⟩ = running | paused
⟨single-animation-fill-mode⟩ = none | forwards | backwards | both
⟨single-animation⟩ = ⟨single-animation-name⟩ || ⟨time⟩ || ⟨single-timing-function⟩ || ⟨time⟩ ||
                     ⟨single-animation-iteration-count⟩ || ⟨single-animation-direction⟩ ||
                     ⟨single-animation-play-state⟩ || ⟨single-animation-fill-mode⟩
```

Table 13. Key binding properties

Name	Value	Initial	Inh.	Ani.	Reference	Notes
-gtk-key-bindings	none | ⟨binding name⟩ [, ⟨binding name⟩]*	none				

⟨binding name⟩ must have been assigned to a binding set with a @binding-set rule.

GtkStyleContext

GtkStyleContext — Rendering UI elements

Functions

GtkStyleContext *	gtk_style_context_new ()
void	gtk_style_context_add_provider ()
void	gtk_style_context_add_provider_for_screen ()
void	gtk_style_context_get ()
GtkTextDirection	gtk_style_context_get_direction ()

GtkJunctionSides	gtk_style_context_get_junction_sides ()
GtkStyleContext *	gtk_style_context_get_parent ()
const GtkWidgetPath *	gtk_style_context_get_path ()
void	gtk_style_context_get_property ()
GdkScreen *	gtk_style_context_get_screen ()
GdkFrameClock *	gtk_style_context_get_frame_clock ()
GtkStateFlags	gtk_style_context_get_state ()
void	gtk_style_context_get_style ()
void	gtk_style_context_get_style_property ()
void	gtk_style_context_get_style_valist ()
void	gtk_style_context_get_valist ()
GtkCssSection *	gtk_style_context_get_section ()
void	gtk_style_context_get_color ()
void	gtk_style_context_get_background_color ()
void	gtk_style_context_get_border_color ()
void	gtk_style_context_get_border ()
void	gtk_style_context_get_padding ()
void	gtk_style_context_get_margin ()
const PangoFontDescription *	gtk_style_context_get_font ()
void	gtk_style_context_invalidate ()
gboolean	gtk_style_context_state_is_running ()
gboolean	gtk_style_context_lookup_color ()
GtkIconSet *	gtk_style_context_lookup_icon_set ()
void	gtk_style_context_notify_state_change ()
void	gtk_style_context_pop_animatable_region ()
void	gtk_style_context_push_animatable_region ()
void	gtk_style_context_cancel_animations ()
void	gtk_style_context_scroll_animations ()
void	gtk_style_context_remove_provider ()
void	gtk_style_context_remove_provider_for_screen ()
void	gtk_style_context_reset_widgets ()
void	gtk_style_context_set_background ()
void	gtk_style_context_restore ()
void	gtk_style_context_save ()
void	gtk_style_context_set_direction ()
void	gtk_style_context_set_junction_sides ()
void	gtk_style_context_set_parent ()
void	gtk_style_context_set_path ()
void	gtk_style_context_add_class ()
void	gtk_style_context_remove_class ()
gboolean	gtk_style_context_has_class ()
GList *	gtk_style_context_list_classes ()
void	gtk_style_context_add_region ()
void	gtk_style_context_remove_region ()
gboolean	gtk_style_context_has_region ()
GList *	gtk_style_context_list_regions ()
void	gtk_style_context_set_screen ()
void	gtk_style_context_set_frame_clock ()
void	gtk_style_context_set_state ()
void	gtk_style_context_set_scale ()

gint	gtk_style_context_get_scale ()
char *	gtk_style_context_to_string ()
GtkBorder *	gtk_border_new ()
GtkBorder *	gtk_border_copy ()
void	gtk_border_free ()
void	gtk_render_arrow ()
void	gtk_render_background ()
void	gtk_render_background_get_clip ()
void	gtk_render_check ()
void	gtk_render_expander ()
void	gtk_render_extension ()
void	gtk_render_focus ()
void	gtk_render_frame ()
void	gtk_render_frame_gap ()
void	gtk_render_handle ()
void	gtk_render_layout ()
void	gtk_render_line ()
void	gtk_render_option ()
void	gtk_render_slider ()
void	gtk_render_activity ()
GdkPixbuf *	gtk_render_icon_pixbuf ()
void	gtk_render_icon_surface ()
void	gtk_render_icon ()
void	gtk_render_insertion_cursor ()

Properties

GtkTextDirection	direction	Read / Write
GdkFrameClock *	paint-clock	Read / Write
GtkStyleContext *	parent	Read / Write
GdkScreen *	screen	Read / Write

Signals

void	changed	Run First

Types and Values

#define	GTK_STYLE_PROPERTY_BACKGROUND_COLOR
#define	GTK_STYLE_PROPERTY_COLOR
#define	GTK_STYLE_PROPERTY_FONT
#define	GTK_STYLE_PROPERTY_MARGIN
#define	GTK_STYLE_PROPERTY_PADDING
#define	GTK_STYLE_PROPERTY_BORDER_WIDTH
#define	GTK_STYLE_PROPERTY_BORDER_RADIUS
#define	GTK_STYLE_PROPERTY_BORDER_STYLE

#define GTK_STYLE_PROPERTY_BORDER_COLOR
#define GTK_STYLE_PROPERTY_BACKGROUND_IMAGE

enum GtkBorderStyle
#define GTK_STYLE_CLASS_ACCELERATOR
#define GTK_STYLE_CLASS_ARROW
#define GTK_STYLE_CLASS_BACKGROUND
#define GTK_STYLE_CLASS_BOTTOM
#define GTK_STYLE_CLASS_BUTTON
#define GTK_STYLE_CLASS_CALENDAR
#define GTK_STYLE_CLASS_CELL
#define GTK_STYLE_CLASS_COMBOBOX_ENTRY
#define GTK_STYLE_CLASS_CONTEXT_MENU
#define GTK_STYLE_CLASS_CHECK
#define GTK_STYLE_CLASS_CSD
#define GTK_STYLE_CLASS_CURSOR_HANDLE
#define GTK_STYLE_CLASS_DEFAULT
#define GTK_STYLE_CLASS_DESTRUCTIVE_ACTION
#define GTK_STYLE_CLASS_DIM_LABEL
#define GTK_STYLE_CLASS_DND
#define GTK_STYLE_CLASS_DOCK
#define GTK_STYLE_CLASS_ENTRY
#define GTK_STYLE_CLASS_ERROR
#define GTK_STYLE_CLASS_EXPANDER
#define GTK_STYLE_CLASS_FRAME
#define GTK_STYLE_CLASS_FLAT
#define GTK_STYLE_CLASS_GRIP
#define GTK_STYLE_CLASS_HEADER
#define GTK_STYLE_CLASS_HIGHLIGHT
#define GTK_STYLE_CLASS_HORIZONTAL
#define GTK_STYLE_CLASS_IMAGE
#define GTK_STYLE_CLASS_INFO
#define GTK_STYLE_CLASS_INLINE_TOOLBAR
#define GTK_STYLE_CLASS_INSERTION_CURSOR
#define GTK_STYLE_CLASS_LABEL
#define GTK_STYLE_CLASS_LEFT
#define GTK_STYLE_CLASS_LEVEL_BAR
#define GTK_STYLE_CLASS_LINKED
#define GTK_STYLE_CLASS_LIST
#define GTK_STYLE_CLASS_LIST_ROW
#define GTK_STYLE_CLASS_MARK
#define GTK_STYLE_CLASS_MENU
#define GTK_STYLE_CLASS_MENUBAR
#define GTK_STYLE_CLASS_MENUITEM
#define GTK_STYLE_CLASS_MESSAGE_DIALOG
#define GTK_STYLE_CLASS_MONOSPACE
#define GTK_STYLE_CLASS_NEEDS_ATTENTION
#define GTK_STYLE_CLASS_NOTEBOOK
#define GTK_STYLE_CLASS_OSD
#define GTK_STYLE_CLASS_OVERSHOOT

#define GTK_STYLE_CLASS_PANE_SEPARATOR
#define GTK_STYLE_CLASS_PAPER
#define GTK_STYLE_CLASS_POPUP
#define GTK_STYLE_CLASS_POPOVER
#define GTK_STYLE_CLASS_PRIMARY_TOOLBAR
#define GTK_STYLE_CLASS_PROGRESSBAR
#define GTK_STYLE_CLASS_PULSE
#define GTK_STYLE_CLASS_QUESTION
#define GTK_STYLE_CLASS_RADIO
#define GTK_STYLE_CLASS_RAISED
#define GTK_STYLE_CLASS_READ_ONLY
#define GTK_STYLE_CLASS_RIGHT
#define GTK_STYLE_CLASS_RUBBERBAND
#define GTK_STYLE_CLASS_SCALE
#define GTK_STYLE_CLASS_SCALE_HAS_MARKS_ABOVE
#define GTK_STYLE_CLASS_SCALE_HAS_MARKS_BELOW
#define GTK_STYLE_CLASS_SCROLLBAR
#define GTK_STYLE_CLASS_SCROLLBARS_JUNCTION
#define GTK_STYLE_CLASS_SEPARATOR
#define GTK_STYLE_CLASS_SIDEBAR
#define GTK_STYLE_CLASS_SLIDER
#define GTK_STYLE_CLASS_SPINBUTTON
#define GTK_STYLE_CLASS_SPINNER
#define GTK_STYLE_CLASS_STATUSBAR
#define GTK_STYLE_CLASS_SUBTITLE
#define GTK_STYLE_CLASS_SUGGESTED_ACTION
#define GTK_STYLE_CLASS_TITLE
#define GTK_STYLE_CLASS_TITLEBAR
#define GTK_STYLE_CLASS_TOOLBAR
#define GTK_STYLE_CLASS_TOOLTIP
#define GTK_STYLE_CLASS_TOUCH_SELECTION
#define GTK_STYLE_CLASS_TOP
#define GTK_STYLE_CLASS_TROUGH
#define GTK_STYLE_CLASS_UNDERSHOOT
#define GTK_STYLE_CLASS_VERTICAL
#define GTK_STYLE_CLASS_VIEW
#define GTK_STYLE_CLASS_WARNING
#define GTK_STYLE_CLASS_WIDE
#define GTK_STYLE_REGION_COLUMN
#define GTK_STYLE_REGION_COLUMN_HEADER
#define GTK_STYLE_REGION_ROW
#define GTK_STYLE_REGION_TAB
GtkStyleContext

enum GtkJunctionSides
enum GtkRegionFlags
enum GtkStyleContextPrintFlags
struct GtkBorder

Object Hierarchy

```
GBoxed
└── GtkBorder
GObject
└── GtkStyleContext
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkStyleContext is an object that stores styling information affecting a widget defined by GtkWidgetPath.

In order to construct the final style information, GtkStyleContext queries information from all attached GtkStyleProviders. Style providers can be either attached explicitly to the context through `gtk_style_context_add_provider()`, or to the screen through `gtk_style_context_add_provider_for_screen()`. The resulting style is a combination of all providers' information in priority order.

For GTK+ widgets, any GtkStyleContext returned by `gtk_widget_get_style_context()` will already have a GtkWidgetPath, a GdkScreen and RTL/LTR information set. The style context will also be updated automatically if any of these settings change on the widget.

If you are using the theming layer standalone, you will need to set a widget path and a screen yourself to the created style context through `gtk_style_context_set_path()` and possibly `gtk_style_context_set_screen()`. See the "Foreign drawing" example in gtk3-demo.

Style Classes

Widgets can add style classes to their context, which can be used to associate different styles by class. The documentation for individual widgets lists which style classes it uses itself, and which style classes may be added by applications to affect their appearance.

GTK+ defines macros for a number of style classes.

Style Regions

Widgets can also add regions with flags to their context. This feature is deprecated and will be removed in a future GTK+ update. Please use style classes instead.

GTK+ defines macros for a number of style regions.

Custom styling in UI libraries and applications

If you are developing a library with custom GtkWidgets that render differently than standard components, you may need to add a GtkStyleProvider yourself with the `GTK_STYLE_PROVIDER_PRIORITY_FALLBACK` priority, either a GtkCssProvider or a custom object implementing the GtkStyleProvider interface. This way themes may still attempt to style your UI elements in a different way if needed so.

If you are using custom styling on an applications, you probably want then to make your style information prevail to the theme's, so you must use a GtkStyleProvider with the `GTK_STYLE_PROVIDER_PRIORITY_APPLICATION` priority, keep in mind that the user settings in `XDG_CONFIG_HOME/gtk-3.0/gtk.css` will still take precedence over your changes, as it uses the `GTK_STYLE_PROVIDER_PRIORITY_USER` priority.

Functions

gtk_style_context_new ()

```
GtkStyleContext *
gtk_style_context_new (void);
```

Creates a standalone GtkStyleContext, this style context won't be attached to any widget, so you may want to call `gtk_style_context_set_path()` yourself.

This function is only useful when using the theming layer separated from GTK+, if you are using GtkStyleContext to theme GtkWidgets, use `gtk_widget_get_style_context()` in order to get a style context ready to theme the widget.

Returns

A newly created GtkStyleContext.

---

gtk_style_context_add_provider ()

```
void
gtk_style_context_add_provider (GtkStyleContext *context,
                                GtkStyleProvider *provider,
                                guint priority);
```

Adds a style provider to `context` , to be used in style construction. Note that a style provider added by this function only affects the style of the widget to which `context` belongs. If you want to affect the style of all widgets, use `gtk_style_context_add_provider_for_screen()`.

Note: If both priorities are the same, a GtkStyleProvider added through this function takes precedence over another added through `gtk_style_context_add_provider_for_screen()`.

Parameters

context	a GtkStyleContext
provider	a GtkStyleProvider
priority	the priority of the style provider. The lower it is, the earlier it will be used in the style construction. Typically this will be in the range between GTK_STYLE_PROVIDER_PRIORITY_FALLBACK and GTK_STYLE_PROVIDER_PRIORITY_USER

Since: 3.0

---

gtk_style_context_add_provider_for_screen ()

```
void
gtk_style_context_add_provider_for_screen
                                (GdkScreen *screen,
                                 GtkStyleProvider *provider,
                                 guint priority);
```

Adds a global style provider to `screen` , which will be used in style construction for all GtkStyleContexts under `screen` .

GTK+ uses this to make styling information from GtkSettings available.

Note: If both priorities are the same, A GtkStyleProvider added through `gtk_style_context_add_provider()` takes precedence over another added through this function.

Parameters

screen	a GdkScreen
provider	a GtkStyleProvider
priority	the priority of the style provider. The lower it is, the earlier it will be used in the style construction. Typically this will be in the range between GTK_STYLE_PROVIDER_PRIORITY_FALLBACK and GTK_STYLE_PROVIDER_PRIORITY_USER

Since: 3.0

---

gtk_style_context_get ()

```
void
gtk_style_context_get (GtkStyleContext *context,
                       GtkStateFlags state,
                       ...);
```

Retrieves several style property values from `context` for a given state.

See `gtk_style_context_get_property()` for details.

Parameters

context	a GtkStyleContext
state	state to retrieve the property values for

... property name /return value pairs, followed by `NULL`

Since: 3.0

---

gtk_style_context_get_direction ()

```
GtkTextDirection
gtk_style_context_get_direction (GtkStyleContext *context);
```

`gtk_style_context_get_direction` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `gtk_style_context_get_state()` and check for GTK_STATE_FLAG_DIR_LTR and GTK_STATE_FLAG_DIR_RTL instead.

Returns the widget direction used for rendering.

Parameters

context	a GtkStyleContext

Returns

the widget direction

Since: 3.0

---

gtk_style_context_get_junction_sides ()

```
GtkJunctionSides
gtk_style_context_get_junction_sides (GtkStyleContext *context);
```

Returns the sides where rendered elements connect visually with others.

Parameters

context	a GtkStyleContext

Returns

the junction sides

Since: 3.0

---

gtk_style_context_get_parent ()

```
GtkStyleContext *
gtk_style_context_get_parent (GtkStyleContext *context);
```

Gets the parent context set via `gtk_style_context_set_parent()`. See that function for details.

Parameters

context	a GtkStyleContext

Returns

the parent context or `NULL`.

[nullable][transfer none]

Since: 3.4

---

gtk_style_context_get_path ()

```
const GtkWidgetPath *
gtk_style_context_get_path (GtkStyleContext *context);
```

Returns the widget path used for style matching.

Parameters

context	a GtkStyleContext

Returns

A GtkWidgetPath.

[transfer none]

Since: 3.0

---

gtk_style_context_get_property ()

```
void
gtk_style_context_get_property (GtkStyleContext *context,
                                const gchar *property,
                                GtkStateFlags state,
                                GValue *value);
```

Gets a style property from `context` for the given state.

Note that not all CSS properties that are supported by GTK+ can be retrieved in this way, since they may not be representable as GValue. GTK+ defines macros for a number of properties that can be used with this function.

Note that passing a state other than the current state of `context` is not recommended unless the style context has been saved with `gtk_style_context_save()`.

When `value` is no longer needed, `g_value_unset()` must be called to free any allocated memory.

Parameters

context	a GtkStyleContext	
property	style property name	
state	state to retrieve the property value for	
value	return location for the style property value.	[out][transfer full]

Since: 3.0

---

gtk_style_context_get_screen ()

```
GdkScreen *
gtk_style_context_get_screen (GtkStyleContext *context);
```

Returns the GdkScreen to which `context` is attached.

Parameters

context	a GtkStyleContext

Returns

a GdkScreen.

[transfer none]

---

gtk_style_context_get_frame_clock ()

```
GdkFrameClock *
gtk_style_context_get_frame_clock (GtkStyleContext *context);
```

Returns the GdkFrameClock to which `context` is attached.

Parameters

context	a GtkStyleContext

Returns

a GdkFrameClock, or `NULL` if `context` does not have an attached frame clock.

[nullable][transfer none]

Since: 3.8

---

gtk_style_context_get_state ()

```
GtkStateFlags
gtk_style_context_get_state (GtkStyleContext *context);
```

Returns the state used for style matching.

This method should only be used to retrieve the GtkStateFlags to pass to GtkStyleContext methods, like `gtk_style_context_get_padding()`. If you need to retrieve the current state of a GtkWidget, use `gtk_widget_get_state_flags()`.

Parameters

context	a GtkStyleContext

Returns

the state flags

Since: 3.0

---

gtk_style_context_get_style ()

```
void
gtk_style_context_get_style (GtkStyleContext *context,
                             ...);
```

Retrieves several widget style properties from `context` according to the current style.

Parameters

context	a GtkStyleContext
...	property name /return value pairs, followed by `NULL`

Since: 3.0

---

gtk_style_context_get_style_property ()

```
void
gtk_style_context_get_style_property (GtkStyleContext *context,
                                      const gchar *property_name,
                                      GValue *value);
```

Gets the value for a widget style property.

When `value` is no longer needed, `g_value_unset()` must be called to free any allocated memory.

Parameters

context	a GtkStyleContext
property_name	the name of the widget style property
value	Return location for the property value

---

gtk_style_context_get_style_valist ()

```
void
gtk_style_context_get_style_valist (GtkStyleContext *context,
                                    va_list args);
```

Retrieves several widget style properties from `context` according to the current style.

Parameters

context	a GtkStyleContext
args	va_list of property name/return location pairs, followed by `NULL`

Since: 3.0

---

gtk_style_context_get_valist ()

```
void
gtk_style_context_get_valist (GtkStyleContext *context,
                              GtkStateFlags state,
                              va_list args);
```

Retrieves several style property values from `context` for a given state.

See `gtk_style_context_get_property()` for details.

Parameters

context	a GtkStyleContext
state	state to retrieve the property values for
args	va_list of property name/return location pairs, followed by `NULL`

Since: 3.0

---

gtk_style_context_get_section ()

```
GtkCssSection *
gtk_style_context_get_section (GtkStyleContext *context,
                               const gchar *property);
```

Queries the location in the CSS where `property` was defined for the current `context` . Note that the state to be queried is taken from `gtk_style_context_get_state()`.

If the location is not available, `NULL` will be returned. The location might not be available for various reasons, such as the property being overridden, `property` not naming a supported CSS property or tracking of definitions being disabled for performance reasons.

Shorthand CSS properties cannot be queried for a location and will always return `NULL`.

Parameters

context	a GtkStyleContext
property	style property name

Returns

`NULL` or the section where a value for `property` was defined.

[nullable][transfer none]

---

gtk_style_context_get_color ()

```
void
gtk_style_context_get_color (GtkStyleContext *context,
                             GtkStateFlags state,
                             GdkRGBA *color);
```

Gets the foreground color for a given state.

See `gtk_style_context_get_property()` and GTK_STYLE_PROPERTY_COLOR for details.

Parameters

context	a GtkStyleContext	
state	state to retrieve the color for	
color	return value for the foreground color.	[out]

Since: 3.0

---

gtk_style_context_get_background_color ()

```
void
gtk_style_context_get_background_color
                                (GtkStyleContext *context,
                                 GtkStateFlags state,
                                 GdkRGBA *color);
```

`gtk_style_context_get_background_color` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `gtk_render_background()` instead.

Gets the background color for a given state.

This function is far less useful than it seems, and it should not be used in newly written code. CSS has no concept of "background color", as a background can be an image, or a gradient, or any other pattern including solid colors.

The only reason why you would call `gtk_style_context_get_background_color()` is to use the returned value to draw the background with it; the correct way to achieve this result is to use `gtk_render_background()` instead, along with CSS style classes to modify the color to be rendered.

Parameters

context	a GtkStyleContext	
state	state to retrieve the color for	
color	return value for the background color.	[out]

Since: 3.0

---

gtk_style_context_get_border_color ()

```
void
gtk_style_context_get_border_color (GtkStyleContext *context,
                                    GtkStateFlags state,
                                    GdkRGBA *color);
```

`gtk_style_context_get_border_color` has been deprecated since version 3.16 and should not be used in newly-written code.

Use `gtk_render_frame()` instead.

Gets the border color for a given state.

Parameters

context	a GtkStyleContext	
state	state to retrieve the color for	
color	return value for the border color.	[out]

Since: 3.0

---

gtk_style_context_get_border ()

```
void
gtk_style_context_get_border (GtkStyleContext *context,
                              GtkStateFlags state,
                              GtkBorder *border);
```

Gets the border for a given state as a GtkBorder.

See `gtk_style_context_get_property()` and GTK_STYLE_PROPERTY_BORDER_WIDTH for details.

Parameters

context	a GtkStyleContext	
state	state to retrieve the border for	
border	return value for the border settings.	[out]

Since: 3.0

---

gtk_style_context_get_padding ()

```
void
gtk_style_context_get_padding (GtkStyleContext *context,
                               GtkStateFlags state,
                               GtkBorder *padding);
```

Gets the padding for a given state as a GtkBorder. See `gtk_style_context_get()` and GTK_STYLE_PROPERTY_PADDING for details.

Parameters

context	a GtkStyleContext	
state	state to retrieve the padding for	
padding	return value for the padding settings.	[out]

Since: 3.0

---

gtk_style_context_get_margin ()

```
void
gtk_style_context_get_margin (GtkStyleContext *context,
                              GtkStateFlags state,
                              GtkBorder *margin);
```

Gets the margin for a given state as a GtkBorder. See `gtk_style_property_get()` and GTK_STYLE_PROPERTY_MARGIN for details.

Parameters

context	a GtkStyleContext
state	state to retrieve the border for
margin	return value for the margin settings. [out]

Since: 3.0

---

gtk_style_context_get_font ()

```
const PangoFontDescription *
gtk_style_context_get_font (GtkStyleContext *context,
                            GtkStateFlags state);
```

`gtk_style_context_get_font` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `gtk_style_context_get()` for "font" or subproperties instead.

Returns the font description for a given state. The returned object is const and will remain valid until the "changed" signal happens.

Parameters

context	a GtkStyleContext
state	state to retrieve the font for

Returns

the PangoFontDescription for the given state. This object is owned by GTK+ and should not be freed.

[transfer none]

Since: 3.0

---

gtk_style_context_invalidate ()

```
void
gtk_style_context_invalidate (GtkStyleContext *context);
```

`gtk_style_context_invalidate` has been deprecated since version 3.12 and should not be used in newly-written code.

Style contexts are invalidated automatically.

Invalidates `context` style information, so it will be reconstructed again. It is useful if you modify the `context` and need the new information immediately.

Parameters

context	a GtkStyleContext.

Since: 3.0

---

gtk_style_context_state_is_running ()

```
gboolean
gtk_style_context_state_is_running (GtkStyleContext *context,
                                    GtkStateType state,
                                    gdouble *progress);
```

`gtk_style_context_state_is_running` has been deprecated since version 3.6 and should not be used in newly-written code.

This function always returns `FALSE`

Returns `TRUE` if there is a transition animation running for the current region (see `gtk_style_context_push_animatable_region()`).

If `progress` is not `NULL`, the animation progress will be returned there, 0.0 means the state is closest to being unset, while 1.0 means it's closest to being set. This means transition animation will run from 0 to 1 when `state` is being set and from 1 to 0 when it's being unset.

Parameters

context	a GtkStyleContext	
state	a widget state	
progress	return location for the transition progress.	[out]

Returns

`TRUE` if there is a running transition animation for `state` .

Since: 3.0

---

gtk_style_context_lookup_color ()

```
gboolean
gtk_style_context_lookup_color (GtkStyleContext *context,
                                const gchar *color_name,
                                GdkRGBA *color);
```

Looks up and resolves a color name in the `context` color map.

Parameters

context	a GtkStyleContext

color_name	color name to lookup	
color	Return location for the looked up color.	[out]

Returns

`TRUE` if `color_name` was found and resolved, `FALSE` otherwise

---

gtk_style_context_lookup_icon_set ()

```
GtkIconSet *
gtk_style_context_lookup_icon_set (GtkStyleContext *context,
                                   const gchar *stock_id);
```

`gtk_style_context_lookup_icon_set` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_icon_theme_lookup_icon()` instead.

Looks up `stock_id` in the icon factories associated to `context` and the default icon factory, returning an icon set if found, otherwise `NULL`.

Parameters

context	a GtkStyleContext
stock_id	an icon name

Returns

The looked up `GtkIconSet`, or `NULL`.

[nullable][transfer none]

---

gtk_style_context_notify_state_change ()

```
void
gtk_style_context_notify_state_change (GtkStyleContext *context,
                                       GdkWindow *window,
                                       gpointer region_id,
                                       GtkStateType state,
                                       gboolean state_value);
```

`gtk_style_context_notify_state_change` has been deprecated since version 3.6 and should not be used in newly-written code.

This function does nothing.

Notifies a state change on `context` , so if the current style makes use of transition animations, one will be started so all rendered elements under `region_id` are animated for state `state` being set to value

state_value .

The window parameter is used in order to invalidate the rendered area as the animation runs, so make sure it is the same window that is being rendered on by the gtk_render_*() functions.

If region_id is NULL, all rendered elements using context will be affected by this state transition.

As a practical example, a GtkButton notifying a state transition on the prelight state:

```
1   gtk_style_context_notify_state_change (context,
2                                          gtk_widget_get_window (widget),
3                                          NULL,
4                                          GTK_STATE_PRELIGHT,
5                                          button->in_button);
```

Can be handled in the CSS file like this:

```
1   button {
2       background-color: #f00
3   }
4
5   button:hover {
6       background-color: #fff;
7       transition: 200ms linear
8   }
```

This combination will animate the button background from red to white if a pointer enters the button, and back to red if the pointer leaves the button.

Note that state is used when finding the transition parameters, which is why the style places the transition under the :hover pseudo-class.

Parameters

context	a GtkStyleContext	
window	a GdkWindow	
region_id	animatable region to notify on, or NULL. See gtk_style_context_push_animatable_region().	[allow-none]
state	state to trigger transition for	
state_value	TRUE if state is the state we are changing to, FALSE if we are changing away from it	

Since: 3.0

---

gtk_style_context_pop_animatable_region ()

```
void
gtk_style_context_pop_animatable_region
                                (GtkStyleContext *context);
```

gtk_style_context_pop_animatable_region has been deprecated since version 3.6 and should not be used in newly-written code.

This function does nothing.

Pops an animatable region from context . See gtk_style_context_push_animatable_region().

Parameters

context	a GtkStyleContext

Since: 3.0

---

gtk_style_context_push_animatable_region ()

```
void
gtk_style_context_push_animatable_region
                                (GtkStyleContext *context,
                                 gpointer region_id);
```

`gtk_style_context_push_animatable_region` has been deprecated since version 3.6 and should not be used in newly-written code.

This function does nothing.

Pushes an animatable region, so all further gtk_render_*() calls between this call and the following `gtk_style_context_pop_animatable_region()` will potentially show transition animations for this region if `gtk_style_context_notify_state_change()` is called for a given state, and the current theme/style defines transition animations for state changes.

The `region_id` used must be unique in `context` so the themes can uniquely identify rendered elements subject to a state transition.

Parameters

context	a GtkStyleContext
region_id	unique identifier for the animatable region

Since: 3.0

---

gtk_style_context_cancel_animations ()

```
void
gtk_style_context_cancel_animations (GtkStyleContext *context,
                                     gpointer region_id);
```

`gtk_style_context_cancel_animations` has been deprecated since version 3.6 and should not be used in newly-written code.

This function does nothing.

Stops all running animations for `region_id` and all animatable regions underneath.

A `NULL` `region_id` will stop all ongoing animations in `context` , when dealing with a GtkStyleContext obtained through `gtk_widget_get_style_context()`, this is normally done for you in all circumstances you would expect all widget to be stopped, so this should be only used in complex widgets with different animatable regions.

Parameters

context	a GtkStyleContext	
region_id	animatable region to stop, or `NULL`. See `gtk_style_context_push_animatable_region()`.	[allow-none]

Since: 3.0

---

gtk_style_context_scroll_animations ()

```
void
gtk_style_context_scroll_animations (GtkStyleContext *context,
                                     GdkWindow *window,
                                     gint dx,
                                     gint dy);
```

`gtk_style_context_scroll_animations` has been deprecated since version 3.6 and should not be used in newly-written code.

This function does nothing.

This function is analogous to `gdk_window_scroll()`, and should be called together with it so the invalidation areas for any ongoing animation are scrolled together with it.

Parameters

context	a GtkStyleContext
window	a GdkWindow used previously in `gtk_style_context_notify_state_change()`
dx	Amount to scroll in the X axis
dy	Amount to scroll in the Y axis

Since: 3.0

---

gtk_style_context_remove_provider ()

```
void
gtk_style_context_remove_provider (GtkStyleContext *context,
                                   GtkStyleProvider *provider);
```

Removes `provider` from the style providers list in `context` .

Parameters

context	a GtkStyleContext
provider	a GtkStyleProvider

Since: 3.0

---

gtk_style_context_remove_provider_for_screen ()

```
void
gtk_style_context_remove_provider_for_screen
                                (GdkScreen *screen,
                                 GtkStyleProvider *provider);
```

Removes `provider` from the global style providers list in `screen` .

Parameters

screen	a GdkScreen
provider	a GtkStyleProvider

Since: 3.0

---

gtk_style_context_reset_widgets ()

```
void
gtk_style_context_reset_widgets (GdkScreen *screen);
```

This function recomputes the styles for all widgets under a particular GdkScreen. This is useful when some global parameter has changed that affects the appearance of all widgets, because when a widget gets a new style, it will both redraw and recompute any cached information about its appearance. As an example, it is used when the color scheme changes in the related GtkSettings object.

Parameters

screen	a GdkScreen

Since: 3.0

---

gtk_style_context_set_background ()

```
void
gtk_style_context_set_background (GtkStyleContext *context,
                                  GdkWindow *window);
```

`gtk_style_context_set_background` has been deprecated since version 3.18 and should not be used in newly-written code.

Use `gtk_render_background()` instead. Note that clients still using this function are now responsible for calling this function again whenever `context` is invalidated.

Sets the background of `window` to the background pattern or color specified in `context` for its current state.

Parameters

context	a GtkStyleContext
window	a GdkWindow

Since: 3.0

---

gtk_style_context_restore ()

```
void
gtk_style_context_restore (GtkStyleContext *context);
```

Restores `context` state to a previous stage. See `gtk_style_context_save()`.

Parameters

context	a GtkStyleContext

Since: 3.0

---

gtk_style_context_save ()

```
void
gtk_style_context_save (GtkStyleContext *context);
```

Saves the `context` state, so temporary modifications done through `gtk_style_context_add_class()`, `gtk_style_context_remove_class()`, `gtk_style_context_set_state()`, etc. can quickly be reverted in one go through `gtk_style_context_restore()`.

The matching call to `gtk_style_context_restore()` must be done before GTK returns to the main loop.

Parameters

context	a GtkStyleContext

Since: 3.0

---

gtk_style_context_set_direction ()

```
void
gtk_style_context_set_direction (GtkStyleContext *context,
                                 GtkTextDirection direction);
```

`gtk_style_context_set_direction` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `gtk_style_context_set_state()` with GTK_STATE_FLAG_DIR_LTR and GTK_STATE_FLAG_DIR_RTL instead.

Sets the reading direction for rendering purposes.

If you are using a GtkStyleContext returned from `gtk_widget_get_style_context()`, you do not need to call this yourself.

Parameters

context	a GtkStyleContext
direction	the new direction.

Since: 3.0

---

gtk_style_context_set_junction_sides ()

```
void
gtk_style_context_set_junction_sides (GtkStyleContext *context,
                                      GtkJunctionSides sides);
```

Sets the sides where rendered elements (mostly through `gtk_render_frame()`) will visually connect with other visual elements.

This is merely a hint that may or may not be honored by themes.

Container widgets are expected to set junction hints as appropriate for their children, so it should not normally be necessary to call this function manually.

Parameters

context	a GtkStyleContext	
sides	sides where rendered elements are visually connected to other elements	

Since: 3.0

---

gtk_style_context_set_parent ()

```
void
gtk_style_context_set_parent (GtkStyleContext *context,
                              GtkStyleContext *parent);
```

Sets the parent style context for `context` . The parent style context is used to implement inheritance of properties.

If you are using a GtkStyleContext returned from `gtk_widget_get_style_context()`, the parent will be set for you.

Parameters

context	a GtkStyleContext	
parent	the new parent or `NULL`.	[allow-none]

Since: 3.4

---

gtk_style_context_set_path ()

```
void
gtk_style_context_set_path (GtkStyleContext *context,
                            GtkWidgetPath *path);
```

Sets the GtkWidgetPath used for style matching. As a consequence, the style will be regenerated to match the new given path.

If you are using a GtkStyleContext returned from `gtk_widget_get_style_context()`, you do not need to call this yourself.

Parameters

context	a GtkStyleContext
path	a GtkWidgetPath

Since: 3.0

---

gtk_style_context_add_class ()

```
void
gtk_style_context_add_class (GtkStyleContext *context,
                             const gchar *class_name);
```

Adds a style class to `context` , so posterior calls to `gtk_style_context_get()` or any of the gtk_render_*() functions will make use of this new class for styling.

In the CSS file format, a GtkEntry defining a “search” class, would be matched by:

```
1                                               entry.search { ... }
```

While any widget defining a “search” class would be matched by:

```
1                                               .search { ... }
```

Parameters

context	a GtkStyleContext
class_name	class name to use in styling

Since: 3.0

---

gtk_style_context_remove_class ()

```
void
gtk_style_context_remove_class (GtkStyleContext *context,
                                const gchar *class_name);
```

Removes `class_name` from `context` .

Parameters

context	a GtkStyleContext
class_name	class name to remove

Since: 3.0

---

gtk_style_context_has_class ()

```
gboolean
gtk_style_context_has_class (GtkStyleContext *context,
                             const gchar *class_name);
```

Returns `TRUE` if `context` currently has defined the given class name.

Parameters

context	a GtkStyleContext
class_name	a class name

Returns

`TRUE` if `context` has `class_name` defined

Since: 3.0

---

gtk_style_context_list_classes ()

```
GList *
gtk_style_context_list_classes (GtkStyleContext *context);
```

Returns the list of classes currently defined in `context` .

Parameters

context	a GtkStyleContext

Returns

a GList of strings with the currently defined classes. The contents of the list are owned by GTK+, but you must free the list itself with `g_list_free()` when you are done with it.

[transfer container][element-type utf8]

Since: 3.0

---

gtk_style_context_add_region ()

```
void
gtk_style_context_add_region (GtkStyleContext *context,
                              const gchar *region_name,
                              GtkRegionFlags flags);
```

`gtk_style_context_add_region` has been deprecated since version 3.14 and should not be used in newly-written code.

Adds a region to `context` , so posterior calls to `gtk_style_context_get()` or any of the gtk_render_*() functions will make use of this new region for styling.

In the CSS file format, a GtkTreeView defining a “row” region, would be matched by:

```
1 treeview row { ... }
```

Pseudo-classes are used for matching `flags` , so the two following rules:

```
1 treeview row:nth-child(even) { ... }
2 treeview row:nth-child(odd) { ... }
```

would apply to even and odd rows, respectively.

Region names must only contain lowercase letters and “-”, starting always with a lowercase letter.

Parameters

context	a GtkStyleContext
region_name	region name to use in styling
flags	flags that apply to the region

Since: 3.0

---

gtk_style_context_remove_region ()

```
void
gtk_style_context_remove_region (GtkStyleContext *context,
                                 const gchar *region_name);
```

`gtk_style_context_remove_region` has been deprecated since version 3.14 and should not be used in newly-written code.

Removes a region from `context` .

Parameters

context	a GtkStyleContext
region_name	region name to unset

Since: 3.0

---

gtk_style_context_has_region ()

```
gboolean
gtk_style_context_has_region (GtkStyleContext *context,
                              const gchar *region_name,
                              GtkRegionFlags *flags_return);
```

`gtk_style_context_has_region` has been deprecated since version 3.14 and should not be used in newly-written code.

Returns `TRUE` if `context` has the region defined. If `flags_return` is not `NULL`, it is set to the flags affecting the region.

Parameters

context	a GtkStyleContext	
region_name	a region name	
flags_return	return location for region flags.	[out][allow-none]

Returns

TRUE if region is defined

Since: 3.0

---

gtk_style_context_list_regions ()

```
GList *
gtk_style_context_list_regions (GtkStyleContext *context);
```

gtk_style_context_list_regions has been deprecated since version 3.14 and should not be used in newly-written code.

Returns the list of regions currently defined in context .

Parameters

context	a GtkStyleContext

Returns

a GList of strings with the currently defined regions. The contents of the list are owned by GTK+, but you must free the list itself with g_list_free() when you are done with it.

[transfer container][element-type utf8]

Since: 3.0

---

gtk_style_context_set_screen ()

```
void
gtk_style_context_set_screen (GtkStyleContext *context,
                              GdkScreen *screen);
```

Attaches context to the given screen.

The screen is used to add style information from “global” style providers, such as the screen’s GtkSettings instance.

If you are using a GtkStyleContext returned from gtk_widget_get_style_context(), you do not need to call this yourself.

Parameters

context	a GtkStyleContext
screen	a GdkScreen

Since: 3.0

---

gtk_style_context_set_frame_clock ()

```
void
gtk_style_context_set_frame_clock (GtkStyleContext *context,
                                   GdkFrameClock *frame_clock);
```

Attaches `context` to the given frame clock.

The frame clock is used for the timing of animations.

If you are using a GtkStyleContext returned from `gtk_widget_get_style_context()`, you do not need to call this yourself.

Parameters

context	a GdkFrameClock
frame_clock	a GdkFrameClock

Since: 3.8

---

gtk_style_context_set_state ()

```
void
gtk_style_context_set_state (GtkStyleContext *context,
                             GtkStateFlags flags);
```

Sets the state to be used for style matching.

Parameters

context	a GtkStyleContext
flags	state to represent

Since: 3.0

---

gtk_style_context_set_scale ()

```
void
gtk_style_context_set_scale (GtkStyleContext *context,
                             gint scale);
```

Sets the scale to use when getting image assets for the style.

Parameters

context	a GtkStyleContext
scale	scale

Since: 3.10

---

gtk_style_context_get_scale ()

```
gint
gtk_style_context_get_scale (GtkStyleContext *context);
```

Returns the scale used for assets.

Parameters

context	a GtkStyleContext

Returns

the scale

Since: 3.10

---

gtk_style_context_to_string ()

```
char *
gtk_style_context_to_string (GtkStyleContext *context,
                             GtkStyleContextPrintFlags flags);
```

Converts the style context into a string representation.

The string representation always includes information about the name, state, id, visibility and style classes of the CSS node that is backing `context` . Depending on the flags, more information may be included.

This function is intended for testing and debugging of the CSS implementation in GTK+. There are no guarantees about the format of the returned string, it may change.

Parameters

context	a GtkStyleContext
flags	Flags that determine what to print

Returns

a newly allocated string representing `context`

Since: 3.20

---

gtk_border_new ()

```
GtkBorder *
gtk_border_new (void);
```

Allocates a new GtkBorder and initializes its elements to zero.

Returns

a newly allocated GtkBorder. Free with `gtk_border_free()`.

[transfer full]

Since: 2.14

---

gtk_border_copy ()

```
GtkBorder *
gtk_border_copy (const GtkBorder *border_);
```

Copies a GtkBorder.

Parameters

border_	a GtkBorder

Returns

a copy of `border_`.

[transfer full]

---

gtk_border_free ()

```
void
gtk_border_free (GtkBorder *border_);
```

Frees a GtkBorder.

Parameters

border_	a GtkBorder

---

gtk_render_arrow ()

```
void
gtk_render_arrow (GtkStyleContext *context,
                  cairo_t *cr,
                  gdouble angle,
                  gdouble x,
```

```
                gdouble y,
                gdouble size);
```

Renders an arrow pointing to `angle` .

Typical arrow rendering at 0, 1⁄2 π;, π; and 3⁄2 π:

Parameters

context	a GtkStyleContext
cr	a cairo_t
angle	arrow angle from 0 to 2 * `G_PI`, being 0 the arrow pointing to the north
x	X origin of the render area
y	Y origin of the render area
size	square side for render area

Since: 3.0

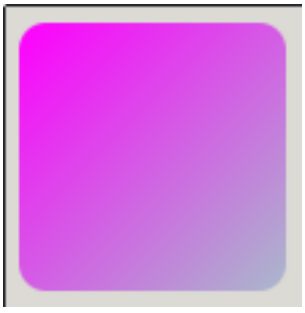
---

gtk_render_background ()

```
void
gtk_render_background (GtkStyleContext *context,
                       cairo_t *cr,
                       gdouble x,
                       gdouble y,
                       gdouble width,
                       gdouble height);
```

Renders the background of an element.

Typical background rendering, showing the effect of `background-image`, `border-width` and `border-radius`:



Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle

y	Y origin of the rectangle	
width	rectangle width	
height	rectangle height	

Since: 3.0.

---

gtk_render_background_get_clip ()

```
void
gtk_render_background_get_clip (GtkStyleContext *context,
                                gdouble x,
                                gdouble y,
                                gdouble width,
                                gdouble height,
                                GdkRectangle *out_clip);
```

Returns the area that will be affected (i.e. drawn to) when calling `gtk_render_background()` for the given `context` and rectangle.

Parameters

context	a GtkStyleContext	
x	X origin of the rectangle	
y	Y origin of the rectangle	
width	rectangle width	
height	rectangle height	
out_clip	return location for the clip.	[out]

Since: 3.20

---

gtk_render_check ()

```
void
gtk_render_check (GtkStyleContext *context,
                  cairo_t *cr,
                  gdouble x,
                  gdouble y,
                  gdouble width,
                  gdouble height);
```

Renders a checkmark (as in a GtkCheckButton).

The `GTK_STATE_FLAG_CHECKED` state determines whether the check is on or off, and `GTK_STATE_FLAG_INCONSISTENT` determines whether it should be marked as undefined.

Typical checkmark rendering:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_expander ()

```
void
gtk_render_expander (GtkStyleContext *context,
                     cairo_t *cr,
                     gdouble x,
                     gdouble y,
                     gdouble width,
                     gdouble height);
```

Renders an expander (as used in GtkTreeView and GtkExpander) in the area defined by `x` , `y` , `width` , `height` . The state `GTK_STATE_FLAG_CHECKED` determines whether the expander is collapsed or expanded.

Typical expander rendering:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_extension ()

```
void
gtk_render_extension (GtkStyleContext *context,
                      cairo_t *cr,
                      gdouble x,
                      gdouble y,
                      gdouble width,
                      gdouble height,
                      GtkPositionType gap_side);
```

Renders a extension (as in a GtkNotebook tab) in the rectangle defined by `x` , `y` , `width` , `height` . The side

where the extension connects to is defined by `gap_side` .

Typical extension rendering:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height
gap_side	side where the gap is

Since: 3.0

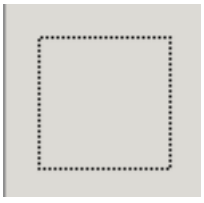
---

gtk_render_focus ()

```
void
gtk_render_focus (GtkStyleContext *context,
                  cairo_t *cr,
                  gdouble x,
                  gdouble y,
                  gdouble width,
                  gdouble height);
```

Renders a focus indicator on the rectangle determined by `x` , `y` , `width` , `height` .

Typical focus rendering:



Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

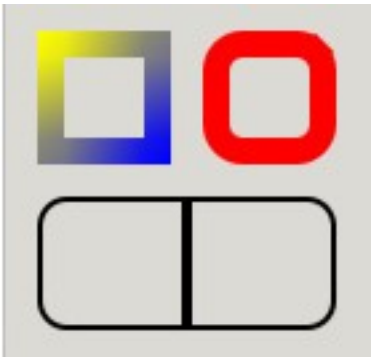
---

gtk_render_frame ()

```
void
gtk_render_frame (GtkStyleContext *context,
                  cairo_t *cr,
                  gdouble x,
                  gdouble y,
                  gdouble width,
                  gdouble height);
```

Renders a frame around the rectangle defined by `x`, `y`, `width`, `height`.

Examples of frame rendering, showing the effect of `border-image`, `border-color`, `border-width`, `border-radius` and junctions:



Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_frame_gap ()

```
void
gtk_render_frame_gap (GtkStyleContext *context,
                      cairo_t *cr,
                      gdouble x,
                      gdouble y,
                      gdouble width,
                      gdouble height,
                      GtkPositionType gap_side,
                      gdouble xy0_gap,
                      gdouble xy1_gap);
```

gtk_render_frame_gap has been deprecated since version 3.24 and should not be used in newly-written code.

Use gtk_render_frame() instead. Themes can create gaps by omitting borders via CSS.

Renders a frame around the rectangle defined by (x , y , width , height), leaving a gap on one side. xy0_gap and xy1_gap will mean X coordinates for GTK_POS_TOP and GTK_POS_BOTTOM gap sides, and Y coordinates for GTK_POS_LEFT and GTK_POS_RIGHT.

Typical rendering of a frame with a gap:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height
gap_side	side where the gap is
xy0_gap	initial coordinate (X or Y depending on gap_side) for the gap
xy1_gap	end coordinate (X or Y depending on gap_side) for the gap

Since: 3.0

---

gtk_render_handle ()

```
void
gtk_render_handle (GtkStyleContext *context,
                   cairo_t *cr,
                   gdouble x,
                   gdouble y,
                   gdouble width,
                   gdouble height);
```

Renders a handle (as in GtkHandleBox, GtkPaned and GtkWindow's resize grip), in the rectangle determined by x , y , width , height .

Handles rendered for the paned and grip classes:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_layout ()

```
void
gtk_render_layout (GtkStyleContext *context,
                   cairo_t *cr,
                   gdouble x,
                   gdouble y,
                   PangoLayout *layout);
```

Renders `layout` on the coordinates `x`, `y`

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin
y	Y origin
layout	the PangoLayout to render

Since: 3.0

---

gtk_render_line ()

```
void
gtk_render_line (GtkStyleContext *context,
                 cairo_t *cr,
                 gdouble x0,
                 gdouble y0,
                 gdouble x1,
                 gdouble y1);
```

Renders a line from (x0, y0) to (x1, y1).

Parameters

context	a GtkStyleContext
cr	a cairo_t

x0	X coordinate for the origin of the line
y0	Y coordinate for the origin of the line
x1	X coordinate for the end of the line
y1	Y coordinate for the end of the line

Since: 3.0

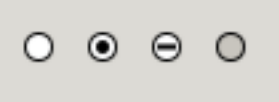
---

gtk_render_option ()

```
void
gtk_render_option (GtkStyleContext *context,
                   cairo_t *cr,
                   gdouble x,
                   gdouble y,
                   gdouble width,
                   gdouble height);
```

Renders an option mark (as in a GtkRadioButton), the GTK_STATE_FLAG_CHECKED state will determine whether the option is on or off, and GTK_STATE_FLAG_INCONSISTENT whether it should be marked as undefined.

Typical option mark rendering:



Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_slider ()

```
void
gtk_render_slider (GtkStyleContext *context,
                   cairo_t *cr,
                   gdouble x,
                   gdouble y,
                   gdouble width,
                   gdouble height,
                   GtkOrientation orientation);
```

Renders a slider (as in GtkScale) in the rectangle defined by `x`, `y`, `width`, `height`. `orientation` defines whether the slider is vertical or horizontal.

Typical slider rendering:

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height
orientation	orientation of the slider

Since: 3.0

---

gtk_render_activity ()

```
void
gtk_render_activity (GtkStyleContext *context,
                     cairo_t *cr,
                     gdouble x,
                     gdouble y,
                     gdouble width,
                     gdouble height);
```

Renders an activity indicator (such as in GtkSpinner). The state `GTK_STATE_FLAG_CHECKED` determines whether there is activity going on.

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin of the rectangle
y	Y origin of the rectangle
width	rectangle width
height	rectangle height

Since: 3.0

---

gtk_render_icon_pixbuf ()

```
GdkPixbuf *
gtk_render_icon_pixbuf (GtkStyleContext *context,
                        const GtkIconSource *source,
                        GtkIconSize size);
```

`gtk_render_icon_pixbuf` has been deprecated since version 3.10 and should not be used in newly-written code.

Use `gtk_icon_theme_load_icon()` instead.

Renders the icon specified by `source` at the given `size` , returning the result in a pixbuf.

Parameters

context	a GtkStyleContext	
source	the GtkIconSource specifying the icon to render	
size	the size (GtkIconSize) to render the icon at. A size of `(GtkIconSize) -1` means render at the size of the source and don’t scale.	[type int]

Returns

a newly-created GdkPixbuf containing the rendered icon.

[transfer full]

Since: 3.0

---

gtk_render_icon_surface ()

```
void
gtk_render_icon_surface (GtkStyleContext *context,
                         cairo_t *cr,
                         cairo_surface_t *surface,
                         gdouble x,
                         gdouble y);
```

Renders the icon in `surface` at the specified `x` and `y` coordinates.

Parameters

context	a GtkStyleContext
cr	a cairo_t
surface	a cairo_surface_t containing the icon to draw
x	X position for the `icon`
y	Y position for the `incon`

Since: 3.10

---

gtk_render_icon ()

```
void
gtk_render_icon (GtkStyleContext *context,
                 cairo_t *cr,
                 GdkPixbuf *pixbuf,
                 gdouble x,
                 gdouble y);
```

Renders the icon in `pixbuf` at the specified `x` and `y` coordinates.

This function will render the icon in `pixbuf` at exactly its size, regardless of scaling factors, which may not be appropriate when drawing on displays with high pixel densities.

You probably want to use `gtk_render_icon_surface()` instead, if you already have a Cairo surface.

Parameters

context	a GtkStyleContext
cr	a cairo_t
pixbuf	a GdkPixbuf containing the icon to draw
x	X position for the `pixbuf`
y	Y position for the `pixbuf`

Since: 3.2

---

gtk_render_insertion_cursor ()

```
void
gtk_render_insertion_cursor (GtkStyleContext *context,
                             cairo_t *cr,
                             gdouble x,
                             gdouble y,
                             PangoLayout *layout,
                             int index,
                             PangoDirection direction);
```

Draws a text caret on `cr` at the specified index of `layout` .

Parameters

context	a GtkStyleContext
cr	a cairo_t
x	X origin
y	Y origin
layout	the PangoLayout of the text
index	the index in the PangoLayout
direction	the PangoDirection of the text

Since: 3.4

Types and Values

GTK_STYLE_PROPERTY_BACKGROUND_COLOR

```
#define GTK_STYLE_PROPERTY_BACKGROUND_COLOR "background-color"
```

A property holding the background color of rendered elements as a GdkRGBA.

GTK_STYLE_PROPERTY_COLOR

```
#define GTK_STYLE_PROPERTY_COLOR "color"
```

A property holding the foreground color of rendered elements as a GdkRGBA.

GTK_STYLE_PROPERTY_FONT

```
#define GTK_STYLE_PROPERTY_FONT "font"
```

A property holding the font properties used when rendering text as a PangoFontDescription.

GTK_STYLE_PROPERTY_MARGIN

```
#define GTK_STYLE_PROPERTY_MARGIN "margin"
```

A property holding the rendered element's margin as a GtkBorder. The margin is defined as the spacing between the border of the element and its surrounding elements. It is external to GtkWidget's size allocations, and the most external spacing property of the padding/border/margin series.

GTK_STYLE_PROPERTY_PADDING

```
#define GTK_STYLE_PROPERTY_PADDING "padding"
```

A property holding the rendered element's padding as a GtkBorder. The padding is defined as the spacing between the inner part of the element border and its child. It's the innermost spacing property of the padding/border/margin series.

GTK_STYLE_PROPERTY_BORDER_WIDTH

```
#define GTK_STYLE_PROPERTY_BORDER_WIDTH "border-width"
```

A property holding the rendered element's border width in pixels as a GtkBorder. The border is the intermediary spacing property of the padding/border/margin series.

gtk_render_frame() uses this property to find out the frame line width, so GtkWidgets rendering frames may need to add up this padding when requesting size

---

GTK_STYLE_PROPERTY_BORDER_RADIUS

```
#define GTK_STYLE_PROPERTY_BORDER_RADIUS "border-radius"
```

A property holding the rendered element's border radius in pixels as a gint.

GTK_STYLE_PROPERTY_BORDER_STYLE

```
#define GTK_STYLE_PROPERTY_BORDER_STYLE "border-style"
```

A property holding the element's border style as a GtkBorderStyle.

GTK_STYLE_PROPERTY_BORDER_COLOR

```
#define GTK_STYLE_PROPERTY_BORDER_COLOR "border-color"
```

A property holding the element's border color as a GdkRGBA.

GTK_STYLE_PROPERTY_BACKGROUND_IMAGE

```
#define GTK_STYLE_PROPERTY_BACKGROUND_IMAGE "background-image"
```

A property holding the element's background as a cairo_pattern_t.

---

enum GtkBorderStyle

Describes how the border of a UI element should be rendered.

Members

GTK_BORDER_STYLE_NONE	No visible border
GTK_BORDER_STYLE_SOLID	A single line segment
GTK_BORDER_STYLE_INSET	Looks as if the content is sunken into the canvas
GTK_BORDER_STYLE_OUTSET	Looks as if the content is coming out of the canvas
GTK_BORDER_STYLE_HIDDEN	Same as `GTK_BORDER_STYLE_NONE`
GTK_BORDER_STYLE_DOTTED	A series of round dots
GTK_BORDER_STYLE_DASHED	A series of square-ended dashes
GTK_BORDER_STYLE_DOUBLE	Two parallel lines with some space between them
GTK_BORDER_STYLE_GROOVE	Looks as if it were carved in the canvas
GTK_BORDER_STYLE_RIDGE	Looks as if it were coming out of the canvas

---

GTK_STYLE_CLASS_ACCELERATOR

```
#define GTK_STYLE_CLASS_ACCELERATOR "accelerator"
```

A CSS class to match an accelerator.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_ARROW

```
#define GTK_STYLE_CLASS_ARROW "arrow"
```

A CSS class used when rendering an arrow element.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_BACKGROUND

```
#define GTK_STYLE_CLASS_BACKGROUND "background"
```

A CSS class to match the window background.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_BOTTOM

```
#define GTK_STYLE_CLASS_BOTTOM "bottom"
```

A CSS class to indicate an area at the bottom of a widget.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_BUTTON

```
#define GTK_STYLE_CLASS_BUTTON "button"
```

A CSS class to match buttons.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_CALENDAR

```
#define GTK_STYLE_CLASS_CALENDAR "calendar"
```

A CSS class to match calendars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_CELL

```
#define GTK_STYLE_CLASS_CELL "cell"
```

A CSS class to match content rendered in cell views.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_COMBOBOX_ENTRY

```
#define GTK_STYLE_CLASS_COMBOBOX_ENTRY "combobox-entry"
```

A CSS class to match combobox entries.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_CONTEXT_MENU

```
#define GTK_STYLE_CLASS_CONTEXT_MENU "context-menu"
```

A CSS class to match context menus.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_CHECK

```
#define GTK_STYLE_CLASS_CHECK "check"
```

A CSS class to match check boxes.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_CSD

```
#define GTK_STYLE_CLASS_CSD "csd"
```

A CSS class that gets added to windows which have client-side decorations.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_CURSOR_HANDLE

```
#define GTK_STYLE_CLASS_CURSOR_HANDLE "cursor-handle"
```

A CSS class used when rendering a drag handle for text selection.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_DEFAULT

```
#define GTK_STYLE_CLASS_DEFAULT "default"
```

A CSS class to match the default widget.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_DESTRUCTIVE_ACTION

```
#define GTK_STYLE_CLASS_DESTRUCTIVE_ACTION "destructive-action"
```

A CSS class used when an action (usually a button) is one that is expected to remove or destroy something visible to the user.

Refer to individual widget documentation for used style classes.

Since: 3.12

GTK_STYLE_CLASS_DIM_LABEL

```
#define GTK_STYLE_CLASS_DIM_LABEL "dim-label"
```

A CSS class to match dimmed labels.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_DND

```
#define GTK_STYLE_CLASS_DND "dnd"
```

A CSS class for a drag-and-drop indicator.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_DOCK

```
#define GTK_STYLE_CLASS_DOCK "dock"
```

A CSS class defining a dock area.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_ENTRY

```
#define GTK_STYLE_CLASS_ENTRY "entry"
```

A CSS class to match text entries.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_ERROR

```
#define GTK_STYLE_CLASS_ERROR "error"
```

A CSS class for an area displaying an error message, such as those in infobars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_EXPANDER

```
#define GTK_STYLE_CLASS_EXPANDER "expander"
```

A CSS class defining an expander, such as those in treeviews.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_FRAME

```
#define GTK_STYLE_CLASS_FRAME "frame"
```

A CSS class defining a frame delimiting content, such as GtkFrame or the scrolled window frame around the scrollable area.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_FLAT

```
#define GTK_STYLE_CLASS_FLAT "flat"
```

A CSS class that is added when widgets that usually have a frame or border (like buttons or entries) should appear without it.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_GRIP

```
#define GTK_STYLE_CLASS_GRIP "grip"
```

A CSS class defining a resize grip.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_HEADER

```
#define GTK_STYLE_CLASS_HEADER "header"
```

A CSS class to match a header element.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_HIGHLIGHT

```
#define GTK_STYLE_CLASS_HIGHLIGHT "highlight"
```

A CSS class defining a highlighted area, such as headings in assistants and calendars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_HORIZONTAL

```
#define GTK_STYLE_CLASS_HORIZONTAL "horizontal"
```

A CSS class for horizontally layered widgets.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_IMAGE

```
#define GTK_STYLE_CLASS_IMAGE "image"
```

A CSS class defining an image, such as the icon in an entry.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_INFO

```
#define GTK_STYLE_CLASS_INFO "info"
```

A CSS class for an area displaying an informational message, such as those in infobars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_INLINE_TOOLBAR

```
#define GTK_STYLE_CLASS_INLINE_TOOLBAR "inline-toolbar"
```

A CSS class to match inline toolbars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_INSERTION_CURSOR

```
#define GTK_STYLE_CLASS_INSERTION_CURSOR "insertion-cursor"
```

A CSS class used when rendering a drag handle for the insertion cursor position.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_LABEL

```
#define GTK_STYLE_CLASS_LABEL "label"
```

A CSS class to match labels.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_LEFT

```
#define GTK_STYLE_CLASS_LEFT "left"
```

A CSS class to indicate an area at the left of a widget.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_LEVEL_BAR

```
#define GTK_STYLE_CLASS_LEVEL_BAR "level-bar"
```

A CSS class used when rendering a level indicator, such as a battery charge level, or a password strength.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_LINKED

```
#define GTK_STYLE_CLASS_LINKED "linked"
```

A CSS class to match a linked area, such as a box containing buttons belonging to the same control.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_LIST

```
#define GTK_STYLE_CLASS_LIST "list"
```

A CSS class to match lists.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_LIST_ROW

```
#define GTK_STYLE_CLASS_LIST_ROW "list-row"
```

A CSS class to match list rows.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_MARK

```
#define GTK_STYLE_CLASS_MARK "mark"
```

A CSS class defining marks in a widget, such as in scales.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_MENU

```
#define GTK_STYLE_CLASS_MENU "menu"
```

A CSS class to match menus.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_MENUBAR

```
#define GTK_STYLE_CLASS_MENUBAR "menubar"
```

A CSS class to menubars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_MENUITEM

```
#define GTK_STYLE_CLASS_MENUITEM "menuitem"
```

A CSS class to match menu items.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_MESSAGE_DIALOG

```
#define GTK_STYLE_CLASS_MESSAGE_DIALOG "message-dialog"
```

A CSS class that is added to message dialogs.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_MONOSPACE

```
#define GTK_STYLE_CLASS_MONOSPACE "monospace"
```

A CSS class that is added to text view that should use a monospace font.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_NEEDS_ATTENTION

```
#define GTK_STYLE_CLASS_NEEDS_ATTENTION "needs-attention"
```

A CSS class used when an element needs the user attention, for instance a button in a stack switcher corresponding to a hidden page that changed state.

Refer to individual widget documentation for used style classes.

Since: 3.12

GTK_STYLE_CLASS_NOTEBOOK

```
#define GTK_STYLE_CLASS_NOTEBOOK "notebook"
```

A CSS class defining a notebook.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_OSD

```
#define GTK_STYLE_CLASS_OSD "osd"
```

A CSS class used when rendering an OSD (On Screen Display) element, on top of another container.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_OVERSHOOT

```
#define GTK_STYLE_CLASS_OVERSHOOT "overshoot"
```

A CSS class that is added on the visual hints that happen when scrolling is attempted past the limits of a scrollable area.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_PANE_SEPARATOR

```
#define GTK_STYLE_CLASS_PANE_SEPARATOR "pane-separator"
```

A CSS class for a pane separator, such as those in GtkPaned.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_PAPER

```
#define GTK_STYLE_CLASS_PAPER "paper"
```

A CSS class that is added to areas that should look like paper.

This is used in print previews and themes are encouraged to style it as black text on white background.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_POPUP

```
#define GTK_STYLE_CLASS_POPUP "popup"
```

A CSS class that is added to the toplevel windows used for menus.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_POPOVER

```
#define GTK_STYLE_CLASS_POPOVER "popover"
```

A CSS class that matches popovers.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_PRIMARY_TOOLBAR

```
#define GTK_STYLE_CLASS_PRIMARY_TOOLBAR "primary-toolbar"
```

A CSS class to match primary toolbars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_PROGRESSBAR

```
#define GTK_STYLE_CLASS_PROGRESSBAR "progressbar"
```

A CSS class to use when rendering activity as a progressbar.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_PULSE

```
#define GTK_STYLE_CLASS_PULSE "pulse"
```

A CSS class to use when rendering a pulse in an indeterminate progress bar.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_QUESTION

```
#define GTK_STYLE_CLASS_QUESTION "question"
```

A CSS class for an area displaying a question to the user, such as those in infobars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_RADIO

```
#define GTK_STYLE_CLASS_RADIO "radio"
```

A CSS class to match radio buttons.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_RAISED

```
#define GTK_STYLE_CLASS_RAISED "raised"
```

A CSS class to match a raised control, such as a raised button on a toolbar.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_READ_ONLY

```
#define GTK_STYLE_CLASS_READ_ONLY "read-only"
```

A CSS class used to indicate a read-only state.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_RIGHT

```
#define GTK_STYLE_CLASS_RIGHT "right"
```

A CSS class to indicate an area at the right of a widget.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_RUBBERBAND

```
#define GTK_STYLE_CLASS_RUBBERBAND "rubberband"
```

A CSS class to match the rubberband selection rectangle.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SCALE

```
#define GTK_STYLE_CLASS_SCALE "scale"
```

A CSS class to match scale widgets.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SCALE_HAS_MARKS_ABOVE

```
#define GTK_STYLE_CLASS_SCALE_HAS_MARKS_ABOVE "scale-has-marks-above"
```

A CSS class to match scale widgets with marks attached, all the marks are above for horizontal GtkScale. left for vertical GtkScale.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SCALE_HAS_MARKS_BELOW

```
#define GTK_STYLE_CLASS_SCALE_HAS_MARKS_BELOW "scale-has-marks-below"
```

A CSS class to match scale widgets with marks attached, all the marks are below for horizontal GtkScale, right for vertical GtkScale.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SCROLLBAR

```
#define GTK_STYLE_CLASS_SCROLLBAR "scrollbar"
```

A CSS class to match scrollbars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SCROLLBARS_JUNCTION

```
#define GTK_STYLE_CLASS_SCROLLBARS_JUNCTION "scrollbars-junction"
```

A CSS class to match the junction area between an horizontal and vertical scrollbar, when they’re both shown.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SEPARATOR

```
#define GTK_STYLE_CLASS_SEPARATOR "separator"
```

A CSS class for a separator.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SIDEBAR

```
#define GTK_STYLE_CLASS_SIDEBAR "sidebar"
```

A CSS class defining a sidebar, such as the left side in a file chooser.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SLIDER

```
#define GTK_STYLE_CLASS_SLIDER "slider"
```

A CSS class to match sliders.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SPINBUTTON

```
#define GTK_STYLE_CLASS_SPINBUTTON "spinbutton"
```

A CSS class defining an spinbutton.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_SPINNER

```
#define GTK_STYLE_CLASS_SPINNER "spinner"
```

A CSS class to use when rendering activity as a “spinner”.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_STATUSBAR

```
#define GTK_STYLE_CLASS_STATUSBAR "statusbar"
```

A CSS class to match statusbars.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_SUBTITLE

```
#define GTK_STYLE_CLASS_SUBTITLE "subtitle"
```

A CSS class used for the subtitle label in a titlebar in a toplevel window.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_SUGGESTED_ACTION

```
#define GTK_STYLE_CLASS_SUGGESTED_ACTION "suggested-action"
```

A CSS class used when an action (usually a button) is the primary suggested action in a specific context.

Refer to individual widget documentation for used style classes.

Since: 3.12

GTK_STYLE_CLASS_TITLE

```
#define GTK_STYLE_CLASS_TITLE "title"
```

A CSS class used for the title label in a titlebar in a toplevel window.

Refer to individual widget documentation for used style classes.

Since: 3.14

GTK_STYLE_CLASS_TITLEBAR

```
#define GTK_STYLE_CLASS_TITLEBAR "titlebar"
```

A CSS class used when rendering a titlebar in a toplevel window.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_TOOLBAR

```
#define GTK_STYLE_CLASS_TOOLBAR "toolbar"
```

A CSS class to match toolbars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_TOOLTIP

```
#define GTK_STYLE_CLASS_TOOLTIP "tooltip"
```

A CSS class to match tooltip windows.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_TOUCH_SELECTION

```
#define GTK_STYLE_CLASS_TOUCH_SELECTION "touch-selection"
```

A CSS class for touch selection popups on entries and text views.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_TOP

```
#define GTK_STYLE_CLASS_TOP "top"
```

A CSS class to indicate an area at the top of a widget.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_TROUGH

```
#define GTK_STYLE_CLASS_TROUGH "trough"
```

A CSS class to match troughs, as in scrollbars and progressbars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_UNDERSHOOT

```
#define GTK_STYLE_CLASS_UNDERSHOOT "undershoot"
```

A CSS class that is added on the visual hints that happen where content is 'scrolled off' and can be made visible by scrolling.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_CLASS_VERTICAL

```
#define GTK_STYLE_CLASS_VERTICAL "vertical"
```

A CSS class for vertically layered widgets.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_VIEW

```
#define GTK_STYLE_CLASS_VIEW "view"
```

A CSS class defining a view, such as iconviews or treeviews.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_WARNING

```
#define GTK_STYLE_CLASS_WARNING "warning"
```

A CSS class for an area displaying a warning message, such as those in infobars.

Refer to individual widget documentation for used style classes.

GTK_STYLE_CLASS_WIDE

```
#define GTK_STYLE_CLASS_WIDE "wide"
```

A CSS class to indicate that a UI element should be 'wide'. Used by GtkPaned.

Refer to individual widget documentation for used style classes.

Since: 3.16

GTK_STYLE_REGION_COLUMN

```
#define GTK_STYLE_REGION_COLUMN "column"
```

`GTK_STYLE_REGION_COLUMN` has been deprecated since version 3.20 and should not be used in newly-written code.

Don't use regions.

A widget region name to define a treeview column.

GTK_STYLE_REGION_COLUMN_HEADER

```
#define GTK_STYLE_REGION_COLUMN_HEADER "column-header"
```

`GTK_STYLE_REGION_COLUMN_HEADER` has been deprecated since version 3.20 and should not be used in newly-written code.

Don't use regions.

A widget region name to define a treeview column header.

GTK_STYLE_REGION_ROW

```
#define GTK_STYLE_REGION_ROW "row"
```

`GTK_STYLE_REGION_ROW` has been deprecated since version 3.20 and should not be used in newly-written code.

Don't use regions.

A widget region name to define a treeview row.

GTK_STYLE_REGION_TAB

```
#define GTK_STYLE_REGION_TAB "tab"
```

`GTK_STYLE_REGION_TAB` has been deprecated since version 3.20 and should not be used in newly-written code.

Don't use regions.

A widget region name to define a notebook tab.

GtkStyleContext

```
typedef struct _GtkStyleContext GtkStyleContext;
```

---

enum GtkJunctionSides

Describes how a rendered element connects to adjacent elements.

Members

GTK_JUNCTION_NONE	No junctions.
GTK_JUNCTION_CORNER_TOPLEFT	Element connects on the top-left corner.
GTK_JUNCTION_CORNER_TOPRIGHT	Element connects on the top-right corner.
GTK_JUNCTION_CORNER_BOTTOMLEFT	Element connects on the bottom-left corner.
GTK_JUNCTION_CORNER_BOTTOMRIGHT	Element connects on the bottom-right corner.
GTK_JUNCTION_TOP	Element connects on the top side.
GTK_JUNCTION_BOTTOM	Element connects on the bottom side.
GTK_JUNCTION_LEFT	Element connects on the left side.
GTK_JUNCTION_RIGHT	Element connects on the right side.

---

enum GtkRegionFlags

Describes a region within a widget.

Members

GTK_REGION_EVEN	Region has an even number within a set.
GTK_REGION_ODD	Region has an odd number within a set.
GTK_REGION_FIRST	Region is the first one within a set.
GTK_REGION_LAST	Region is the last one within a set.
GTK_REGION_ONLY	Region is the only one within a set.
GTK_REGION_SORTED	Region is part of a sorted area.

---

enum GtkStyleContextPrintFlags

Flags that modify the behavior of `gtk_style_context_to_string()`. New values may be added to this enumeration.

Members

GTK_STYLE_CONTEXT_PRINT_NONE	

GTK_STYLE_CONTEXT_PRINT_RECURSE	Print the entire tree of CSS nodes starting at the style context's node
GTK_STYLE_CONTEXT_PRINT_SHOW_STYLE	Show the values of the CSS properties for each node

---

struct GtkBorder

```
struct GtkBorder {
  gint16 left;
  gint16 right;
  gint16 top;
  gint16 bottom;
};
```

A struct that specifies a border around a rectangular area that can be of different width on each side.

Members

gint16 `left`;	The width of the left border
gint16 `right`;	The width of the right border
gint16 `top`;	The width of the top border
gint16 `bottom`;	The width of the bottom border

Property Details

The `“direction”` property

`“direction”` `GtkTextDirection`

Text direction.

Flags: Read / Write

Default value: GTK_TEXT_DIR_LTR

The `“paint-clock”` property

`“paint-clock”` `GdkFrameClock *`

The associated GdkFrameClock.

Flags: Read / Write

The `“parent”` property

`“parent”` `GtkStyleContext *`

Sets or gets the style context’s parent. See `gtk_style_context_set_parent()` for details.

Flags: Read / Write

Since: 3.4

The “screen” property

```
  “screen”                   GdkScreen *
```

The associated GdkScreen.

Flags: Read / Write

Signal Details

The “changed” signal

```
void
user_function (GtkStyleContext *stylecontext,
               gpointer         user_data)
```

The ::changed signal is emitted when there is a change in the GtkStyleContext.

For a GtkStyleContext returned by `gtk_widget_get_style_context()`, the “style-updated” signal/vfunc might be more convenient to use.

This signal is useful when using the theming layer standalone.

Parameters

user_data	user data set when the signal handler was connected.

Flags: Run First

Since: 3.0

GtkCssProvider

GtkCssProvider — CSS-like styling for widgets

Functions

GtkCssProvider *	gtk_css_provider_get_default ()
GtkCssProvider *	gtk_css_provider_get_named ()
gboolean	gtk_css_provider_load_from_data ()
gboolean	gtk_css_provider_load_from_file ()
gboolean	gtk_css_provider_load_from_path ()
void	gtk_css_provider_load_from_resource ()
GtkCssProvider *	gtk_css_provider_new ()
char *	gtk_css_provider_to_string ()
guint	gtk_css_section_get_end_line ()
guint	gtk_css_section_get_end_position ()
GFile *	gtk_css_section_get_file ()
GtkCssSection *	gtk_css_section_get_parent ()
GtkCssSectionType	gtk_css_section_get_section_type ()
guint	gtk_css_section_get_start_line ()
guint	gtk_css_section_get_start_position ()
GtkCssSection *	gtk_css_section_ref ()
void	gtk_css_section_unref ()

Signals

void	parsing-error	Run Last

Types and Values

struct	GtkCssProvider
#define	GTK_CSS_PROVIDER_ERROR
enum	GtkCssProviderError
	GtkCssSection
enum	GtkCssSectionType

Object Hierarchy

```
GBoxed
└── GtkCssSection
GObject
└── GtkCssProvider
```

Implemented Interfaces

GtkCssProvider implements GtkStyleProvider and GtkStyleProviderPrivate.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkCssProvider is an object implementing the GtkStyleProvider interface. It is able to parse CSS-like input in order to style widgets.

An application can make GTK+ parse a specific CSS style sheet by calling `gtk_css_provider_load_from_file()` or `gtk_css_provider_load_from_resource()` and adding the provider with `gtk_style_context_add_provider()` or `gtk_style_context_add_provider_for_screen()`.

In addition, certain files will be read when GTK+ is initialized. First, the file `$XDG_CONFIG_HOME/gtk-3.0/gtk.css` is loaded if it exists. Then, GTK+ loads the first existing file among `XDG_DATA_HOME/themes/THEME/gtk-VERSION/gtk.css`, `$HOME/.themes/THEME/gtk-VERSION/gtk.css`, `$XDG_DATA_DIRS/themes/THEME/gtk-VERSION/gtk.css` and `DATADIR/share/themes/THEME/gtk-VERSION/gtk.css`, where `THEME` is the name of the current theme (see the "gtk-theme-name" setting), `DATADIR` is the prefix configured when GTK+ was compiled (unless overridden by the `GTK_DATA_PREFIX` environment variable), and `VERSION` is the GTK+ version number. If no file is found for the current version, GTK+ tries older versions all the way back to 3.0.

In the same way, GTK+ tries to load a gtk-keys.css file for the current key theme, as defined by "gtk-key-theme-name".

Functions

gtk_css_provider_get_default ()

```
GtkCssProvider *
gtk_css_provider_get_default (void);
```

`gtk_css_provider_get_default` has been deprecated since version 3.24 and should not be used in newly-written code.

Use `gtk_css_provider_new()` instead.

Returns the provider containing the style settings used as a fallback for all widgets.

Returns

The provider used for fallback styling. This memory is owned by GTK+, and you must not free it.

[transfer none]

---

gtk_css_provider_get_named ()

```
GtkCssProvider *
gtk_css_provider_get_named (const gchar *name,
                            const gchar *variant);
```

Loads a theme from the usual theme paths

Parameters

name	A theme name	
variant	variant to load, for example, "dark", or `NULL` for the default.	[allow-none]

Returns

a GtkCssProvider with the theme loaded. This memory is owned by GTK+, and you must not free it.

[transfer none]

---

gtk_css_provider_load_from_data ()

```
gboolean
gtk_css_provider_load_from_data (GtkCssProvider *css_provider,
                                 const gchar *data,
                                 gssize length,
                                 GError **error);
```

Loads `data` into `css_provider` , and by doing so clears any previously loaded information.

Parameters

css_provider	a GtkCssProvider	
data	CSS data loaded in memory.	[array length=length][element-type guint8]
length	the length of `data` in bytes, or -1 for NUL terminated strings. If `length` is not -1, the code will assume it is not NUL terminated and will potentially do a copy.	
error	return location for a GError, or `NULL`.	[out][allow-none]

Returns

`TRUE`. The return value is deprecated and `FALSE` will only be returned for backwards compatibility reasons if an `error` is not `NULL` and a loading error occurred. To track errors while loading CSS, connect to the "parsing-error" signal.

---

gtk_css_provider_load_from_file ()

```
gboolean
gtk_css_provider_load_from_file (GtkCssProvider *css_provider,
                                 GFile *file,
                                 GError **error);
```

Loads the data contained in `file` into `css_provider` , making it clear any previously loaded information.

Parameters

css_provider	a GtkCssProvider	
file	GFile pointing to a file to load	
error	return location for a GError, or `NULL`.	[out][allow-none]

Returns

`TRUE`. The return value is deprecated and `FALSE` will only be returned for backwards compatibility reasons if an `error` is not `NULL` and a loading error occurred. To track errors while loading CSS, connect to the "parsing-error" signal.

---

gtk_css_provider_load_from_path ()

```
gboolean
gtk_css_provider_load_from_path (GtkCssProvider *css_provider,
                                 const gchar *path,
                                 GError **error);
```

Loads the data contained in `path` into `css_provider` , making it clear any previously loaded information.

Parameters

css_provider	a GtkCssProvider	
path	the path of a filename to load, in the GLib filename encoding	
error	return location for a GError, or `NULL`.	[out][allow-none]

Returns

`TRUE`. The return value is deprecated and `FALSE` will only be returned for backwards compatibility reasons if an `error` is not `NULL` and a loading error occurred. To track errors while loading CSS, connect to the "parsing-error" signal.

---

gtk_css_provider_load_from_resource ()

```
void
gtk_css_provider_load_from_resource (GtkCssProvider *css_provider,
                                     const gchar *resource_path);
```

Loads the data contained in the resource at `resource_path` into the GtkCssProvider, clearing any previously loaded information.

To track errors while loading CSS, connect to the "parsing-error" signal.

Parameters

css_provider	a GtkCssProvider
resource_path	a GResource resource path

Since: 3.16

---

gtk_css_provider_new ()

```
GtkCssProvider *
gtk_css_provider_new (void);
```

Returns a newly created GtkCssProvider.

Returns

A new GtkCssProvider

---

gtk_css_provider_to_string ()

```
char *
gtk_css_provider_to_string (GtkCssProvider *provider);
```

Converts the `provider` into a string representation in CSS format.

Using `gtk_css_provider_load_from_data()` with the return value from this function on a new provider created with `gtk_css_provider_new()` will basically create a duplicate of this `provider` .

Parameters

provider	the provider to write to a string

Returns

a new string representing the `provider` .

Since: 3.2

---

gtk_css_section_get_end_line ()

```
guint
gtk_css_section_get_end_line (const GtkCssSection *section);
```

Returns the line in the CSS document where this section end. The line number is 0-indexed, so the first line of the document will return 0. This value may change in future invocations of this function if `section` is not yet parsed completely. This will for example happen in the GtkCssProvider::parsing-error signal. The end position and line may be identical to the start position and line for sections which failed to parse anything successfully.

Parameters

section	the section

Returns

the line number

Since: 3.2

---

gtk_css_section_get_end_position ()

```
guint
gtk_css_section_get_end_position (const GtkCssSection *section);
```

Returns the offset in bytes from the start of the current line returned via `gtk_css_section_get_end_line()`. This value may change in future invocations of this function if `section` is not yet parsed completely. This will for example happen in the GtkCssProvider::parsing-error signal. The end position and line may be identical to the start position and line for sections which failed to parse anything successfully.

Parameters

section	the section

Returns

the offset in bytes from the start of the line.

Since: 3.2

---

gtk_css_section_get_file ()

```
GFile *
gtk_css_section_get_file (const GtkCssSection *section);
```

Gets the file that `section` was parsed from. If no such file exists, for example because the CSS was loaded via `gtk_css_provider_load_from_data()`, then `NULL` is returned.

Parameters

section	the section

Returns

the GFile that `section` was parsed from or `NULL` if `section` was parsed from other data.

[transfer none]

Since: 3.2

---

gtk_css_section_get_parent ()

```
GtkCssSection *
gtk_css_section_get_parent (const GtkCssSection *section);
```

Gets the parent section for the given `section`. The parent section is the section that contains this `section`. A special case are sections of type GTK_CSS_SECTION_DOCUMENT. Their parent will either be `NULL` if they are the original CSS document that was loaded by `gtk_css_provider_load_from_file()` or a section of type GTK_CSS_SECTION_IMPORT if it was loaded with an import rule from a different file.

Parameters

section	the section

Returns

the parent section or `NULL` if none.

[nullable][transfer none]

Since: 3.2

---

gtk_css_section_get_section_type ()

```
GtkCssSectionType
gtk_css_section_get_section_type (const GtkCssSection *section);
```

Gets the type of information that `section` describes.

Parameters

section	the section

Returns

the type of `section`

Since: 3.2

---

gtk_css_section_get_start_line ()

```
guint
gtk_css_section_get_start_line (const GtkCssSection *section);
```

Returns the line in the CSS document where this section starts. The line number is 0-indexed, so the first line of the document will return 0.

Parameters

section	the section

Returns

the line number

Since: 3.2

---

gtk_css_section_get_start_position ()

```
guint
gtk_css_section_get_start_position (const GtkCssSection *section);
```

Returns the offset in bytes from the start of the current line returned via `gtk_css_section_get_start_line()`.

Parameters

section	the section

Returns

the offset in bytes from the start of the line.

Since: 3.2

---

gtk_css_section_ref ()

```
GtkCssSection *
gtk_css_section_ref (GtkCssSection *section);
```

Increments the reference count on `section` .

Parameters

section	a GtkCssSection

Returns

`section` itself.

Since: 3.2

---

gtk_css_section_unref ()

```
void
gtk_css_section_unref (GtkCssSection *section);
```

Decrements the reference count on `section` , freeing the structure if the reference count reaches 0.

Parameters

section	a GtkCssSection

Since: 3.2

Types and Values

struct GtkCssProvider

```
struct GtkCssProvider;
```

GTK_CSS_PROVIDER_ERROR

```
#define GTK_CSS_PROVIDER_ERROR (gtk_css_provider_error_quark ())
```

Domain for GtkCssProvider errors.

---

enum GtkCssProviderError

Error codes for `GTK_CSS_PROVIDER_ERROR`.

Members

GTK_CSS_PROVIDER_ERROR_FAILED	Failed.
GTK_CSS_PROVIDER_ERROR_SYNTAX	Syntax error.
GTK_CSS_PROVIDER_ERROR_IMPORT	Import error.
GTK_CSS_PROVIDER_ERROR_NAME	Name error.
GTK_CSS_PROVIDER_ERROR_DEPRECATED	Deprecation error.
GTK_CSS_PROVIDER_ERROR_UNKNOWN_VALUE	Unknown value.

GtkCssSection

```
typedef struct _GtkCssSection GtkCssSection;
```

Defines a part of a CSS document. Because sections are nested into one another, you can use `gtk_css_section_get_parent()` to get the containing region.

Since: 3.2

---

enum GtkCssSectionType

The different types of sections indicate parts of a CSS document as parsed by GTK's CSS parser. They are oriented towards the CSS Grammar, but may contain extensions.

More types might be added in the future as the parser incorporates more features.

Members

GTK_CSS_SECTION_DOCUMENT	The section describes a complete document. This section time is the only one where `gtk_css_section_get_parent()` might return `NULL`.
GTK_CSS_SECTION_IMPORT	The section defines an import rule.
GTK_CSS_SECTION_COLOR_DEFINITION	The section defines a color. This is a GTK extension to CSS.
GTK_CSS_SECTION_BINDING_SET	The section defines a binding set. This is a GTK extension to CSS.
GTK_CSS_SECTION_RULESET	The section defines a CSS ruleset.
GTK_CSS_SECTION_SELECTOR	The section defines a CSS selector.
GTK_CSS_SECTION_DECLARATION	The section defines the declaration of a CSS variable.
GTK_CSS_SECTION_VALUE	The section defines the value of a CSS declaration.
GTK_CSS_SECTION_KEYFRAMES	The section defines keyframes. See CSS Animations for details. Since 3.6

Since: 3.2

Signal Details

The `"parsing-error"` signal

```
void
user_function (GtkCssProvider *provider,
               GtkCssSection  *section,
               GError         *error,
               gpointer        user_data)
```

Signals that a parsing error occurred. the `path` , `line` and `position` describe the actual location of the error as accurately as possible.

Parsing errors are never fatal, so the parsing will resume after the error. Errors may however cause parts of the given data or even all of it to not be parsed at all. So it is a useful idea to check that the parsing succeeds by connecting to this signal.

Note that this signal may be emitted at any time as the css provider may opt to defer parsing parts or all of the input to a later time than when a loading function was called.

Parameters

provider	the provider that had a parsing error
section	section the error happened in
error	The parsing error
user_data	user data set when the signal handler was connected.

Flags: Run Last

See Also

GtkStyleContext, GtkStyleProvider

GtkStyleProvider

GtkStyleProvider — Interface to provide style information to GtkStyleContext

Functions

GtkIconFactory *	gtk_style_provider_get_icon_factory ()
GtkStyleProperties *	gtk_style_provider_get_style ()
gboolean	gtk_style_provider_get_style_property ()

Types and Values

struct	GtkStyleProviderIface
	GtkStyleProvider
#define	GTK_STYLE_PROVIDER_PRIORITY_FALLBACK
#define	GTK_STYLE_PROVIDER_PRIORITY_THEME
#define	GTK_STYLE_PROVIDER_PRIORITY_SETTINGS
#define	GTK_STYLE_PROVIDER_PRIORITY_APPLICATION
#define	GTK_STYLE_PROVIDER_PRIORITY_USER

Object Hierarchy

```
GInterface
└── GtkStyleProvider
```

Known Implementations

GtkStyleProvider is implemented by GtkCssProvider and GtkSettings.

Includes

```
#include <gtk/gtk.h>
```

Description

GtkStyleProvider is an interface used to provide style information to a GtkStyleContext. See `gtk_style_context_add_provider()` and `gtk_style_context_add_provider_for_screen()`.

Functions

gtk_style_provider_get_icon_factory ()

```
GtkIconFactory *
gtk_style_provider_get_icon_factory (GtkStyleProvider *provider,
                                     GtkWidgetPath *path);
```

`gtk_style_provider_get_icon_factory` has been deprecated since version 3.8 and should not be used in newly-written code.

Will always return `NULL` for all GTK-provided style providers.

Returns the GtkIconFactory defined to be in use for `path` , or `NULL` if none is defined.

Parameters

provider	a GtkStyleProvider
path	GtkWidgetPath to query

Returns

The icon factory to use for `path` , or `NULL`.

[nullable][transfer none]

Since: 3.0

---

gtk_style_provider_get_style ()

```
GtkStyleProperties *
gtk_style_provider_get_style (GtkStyleProvider *provider,
                              GtkWidgetPath *path);
```

`gtk_style_provider_get_style` has been deprecated since version 3.8 and should not be used in newly-written code.

Will always return `NULL` for all GTK-provided style providers as the interface cannot correctly work the way CSS is specified.

Returns the style settings affecting a widget defined by `path` , or `NULL` if `provider` doesn't contemplate styling `path` .

Parameters

provider	a GtkStyleProvider
path	GtkWidgetPath to query

Returns

a GtkStyleProperties containing the style settings affecting `path` .

[nullable][transfer full]

Since: 3.0

---

gtk_style_provider_get_style_property ()

```
gboolean
gtk_style_provider_get_style_property (GtkStyleProvider *provider,
                                       GtkWidgetPath *path,
                                       GtkStateFlags state,
                                       GParamSpec *pspec,
                                       GValue *value);
```

Looks up a widget style property as defined by `provider` for the widget represented by `path` .

Parameters

provider	a GtkStyleProvider	
path	GtkWidgetPath to query	
state	state to query the style property for	
pspec	The GParamSpec to query	
value	return location for the property value.	[out]

Returns

`TRUE` if the property was found and has a value, `FALSE` otherwise

Since: 3.0

Types and Values

struct GtkStyleProviderIface

```
struct GtkStyleProviderIface {
  GtkStyleProperties * (* get_style) (GtkStyleProvider *provider,
                                      GtkWidgetPath    *path);

  gboolean (* get_style_property) (GtkStyleProvider *provider,
                                   GtkWidgetPath    *path,
                                   GtkStateFlags     state,
                                   GParamSpec       *pspec,
                                   GValue           *value);

  GtkIconFactory * (* get_icon_factory) (GtkStyleProvider *provider,
                                GtkWidgetPath    *path);
};
```

Members

`get_style` ()	Gets a set of style information that applies to a widget path.
`get_style_property` ()	Gets the value of a widget style property that applies to a widget path.
`get_icon_factory` ()	Gets the icon factory that applies to a widget path.

GtkStyleProvider

```
typedef struct _GtkStyleProvider GtkStyleProvider;
```

GTK_STYLE_PROVIDER_PRIORITY_FALLBACK

```
#define GTK_STYLE_PROVIDER_PRIORITY_FALLBACK      1
```

The priority used for default style information that is used in the absence of themes.

Note that this is not very useful for providing default styling for custom style classes - themes are likely to override styling provided at this priority with catch-all `* {...}` rules.

GTK_STYLE_PROVIDER_PRIORITY_THEME

```
#define GTK_STYLE_PROVIDER_PRIORITY_THEME     200
```

The priority used for style information provided by themes.

GTK_STYLE_PROVIDER_PRIORITY_SETTINGS

```
#define GTK_STYLE_PROVIDER_PRIORITY_SETTINGS    400
```

The priority used for style information provided via GtkSettings.

This priority is higher than GTK_STYLE_PROVIDER_PRIORITY_THEME to let settings override themes.

GTK_STYLE_PROVIDER_PRIORITY_APPLICATION

```
#define GTK_STYLE_PROVIDER_PRIORITY_APPLICATION 600
```

A priority that can be used when adding a GtkStyleProvider for application-specific style information.

GTK_STYLE_PROVIDER_PRIORITY_USER

```
#define GTK_STYLE_PROVIDER_PRIORITY_USER        800
```

The priority used for the style information from `XDG_CONFIG_HOME/gtk-3.0/gtk.css`.

You should not use priorities higher than this, to give the user the last word.

See Also

GtkStyleContext, GtkCssProvider

GtkStyleProperties

GtkStyleProperties — Store for style property information

Functions

void	gtk_style_properties_clear ()
void	gtk_style_properties_get ()
gboolean	gtk_style_properties_get_property ()
void	gtk_style_properties_get_valist ()
GtkSymbolicColor *	gtk_style_properties_lookup_color ()
gboolean	gtk_style_properties_lookup_property ()
void	gtk_style_properties_map_color ()
void	gtk_style_properties_merge ()
GtkStyleProperties *	gtk_style_properties_new ()
gboolean	(*GtkStylePropertyParser) ()
void	gtk_style_properties_register_property ()
void	gtk_style_properties_set ()
void	gtk_style_properties_set_property ()
void	gtk_style_properties_set_valist ()
void	gtk_style_properties_unset_property ()

Types and Values

struct	GtkStyleProperties

Includes

```
#include <gtk/gtk.h>
```

Description

GtkStyleProperties provides the storage for style information that is used by GtkStyleContext and other GtkStyleProvider implementations.

Before style properties can be stored in GtkStyleProperties, they must be registered with `gtk_style_properties_register_property()`.

Unless you are writing a GtkStyleProvider implementation, you are unlikely to use this API directly, as `gtk_style_context_get()` and its variants are the preferred way to access styling information from widget implementations and theming engine implementations should use the APIs provided by GtkThemingEngine instead.

GtkStyleProperties has been deprecated in GTK 3.16. The CSS machinery does not use it anymore and all users of this object have been deprecated.

Functions

gtk_style_properties_clear ()

```
void
gtk_style_properties_clear (GtkStyleProperties *props);
```

`gtk_style_properties_clear` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Clears all style information from `props` .

Parameters

props	a GtkStyleProperties

---

gtk_style_properties_get ()

```
void
gtk_style_properties_get (GtkStyleProperties *props,
                          GtkStateFlags state,
                          ...);
```

`gtk_style_properties_get` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Retrieves several style property values from `props` for a given state.

Parameters

props	a GtkStyleProperties
state	state to retrieve the property values for
...	property name /return value pairs, followed by `NULL`

Since: 3.0

---

gtk_style_properties_get_property ()

```
gboolean
gtk_style_properties_get_property (GtkStyleProperties *props,
                                   const gchar *property,
                                   GtkStateFlags state,
                                   GValue *value);
```

`gtk_style_properties_get_property` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Gets a style property from `props` for the given state. When done with `value` , `g_value_unset()` needs to be called to free any allocated memory.

Parameters

props	a GtkStyleProperties	
property	style property name	
state	state to retrieve the property value for	
value	return location for the style property value.	[out][transfer full]

Returns

`TRUE` if the property exists in `props` , `FALSE` otherwise

Since: 3.0

---

gtk_style_properties_get_valist ()

```
void
gtk_style_properties_get_valist (GtkStyleProperties *props,
                                 GtkStateFlags state,
                                 va_list args);
```

`gtk_style_properties_get_valist` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Retrieves several style property values from `props` for a given state.

Parameters

props	a GtkStyleProperties
state	state to retrieve the property values for
args	va_list of property name/return location pairs, followed by `NULL`

Since: 3.0

---

gtk_style_properties_lookup_color ()

```
GtkSymbolicColor *
gtk_style_properties_lookup_color (GtkStyleProperties *props,
                                   const gchar *name);
```

`gtk_style_properties_lookup_color` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Returns the symbolic color that is mapped to `name` .

Parameters

props	a GtkStyleProperties
name	color name to lookup

Returns

The mapped color.

[transfer none]

Since: 3.0

---

gtk_style_properties_lookup_property ()

```
gboolean
gtk_style_properties_lookup_property (const gchar *property_name,
                                      GtkStylePropertyParser *parse_func,
                                      GParamSpec **pspec);
```

`gtk_style_properties_lookup_property` has been deprecated since version 3.8 and should not be used in newly-written code.

This code could only look up custom properties and those are deprecated.

Returns `TRUE` if a property has been registered, if `pspec` or `parse_func` are not `NULL`, the GParamSpec and parsing function will be respectively returned.

[skip]

Parameters

property_name	property name to look up	
parse_func	return location for the parse function.	[out]
pspec	return location for the GParamSpec.	[out][transfer none]

Returns

`TRUE` if the property is registered, `FALSE` otherwise

Since: 3.0

---

gtk_style_properties_map_color ()

```
void
gtk_style_properties_map_color (GtkStyleProperties *props,
                                const gchar *name,
                                GtkSymbolicColor *color);
```

`gtk_style_properties_map_color` has been deprecated since version 3.8 and should not be used in newly-written code.

GtkSymbolicColor is deprecated.

Maps `color` so it can be referenced by `name` . See `gtk_style_properties_lookup_color()`

Parameters

props	a GtkStyleProperties
name	color name
color	GtkSymbolicColor to map `name` to

Since: 3.0

---

gtk_style_properties_merge ()

```
void
gtk_style_properties_merge (GtkStyleProperties *props,
                            const GtkStyleProperties *props_to_merge,
                            gboolean replace);
```

`gtk_style_properties_merge` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Merges into `props` all the style information contained in `props_to_merge` . If `replace` is `TRUE`, the values will be overwritten, if it is `FALSE`, the older values will prevail.

Parameters

props	a GtkStyleProperties
props_to_merge	a second GtkStyleProperties
replace	whether to replace values or not

Since: 3.0

---

gtk_style_properties_new ()

```
GtkStyleProperties *
gtk_style_properties_new (void);
```

`gtk_style_properties_new` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Returns a newly created GtkStyleProperties

Returns

a new GtkStyleProperties

GtkStylePropertyParser ()

```
gboolean
(*GtkStylePropertyParser) (const gchar *string,
                           GValue *value,
                           GError **error);
```

---

gtk_style_properties_register_property ()

```
void
gtk_style_properties_register_property
                                (GtkStylePropertyParser parse_func,
                                 GParamSpec *pspec);
```

`gtk_style_properties_register_property` has been deprecated since version 3.8 and should not be used in newly-written code.

Code should use the default properties provided by CSS.

Registers a property so it can be used in the CSS file format. This function is the low-level equivalent of `gtk_theming_engine_register_property()`, if you are implementing a theming engine, you want to use that function instead.

[skip]

Parameters

parse_func	parsing function to use, or `NULL`.	[nullable]
pspec	the GParamSpec for the new property	

Since: 3.0

---

gtk_style_properties_set ()

```
void
gtk_style_properties_set (GtkStyleProperties *props,
                          GtkStateFlags state,
                          ...);
```

`gtk_style_properties_set` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Sets several style properties on `props` .

Parameters

props	a GtkStyleProperties
state	state to set the values for
...	property name/value pairs, followed by `NULL`

Since: 3.0

---

gtk_style_properties_set_property ()

```
void
gtk_style_properties_set_property (GtkStyleProperties *props,
                                   const gchar *property,
                                   GtkStateFlags state,
                                   const GValue *value);
```

`gtk_style_properties_set_property` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Sets a styling property in `props` .

Parameters

props	a GtkStyleProperties
property	styling property to set
state	state to set the value for
value	new value for the property

Since: 3.0

---

gtk_style_properties_set_valist ()

```
void
gtk_style_properties_set_valist (GtkStyleProperties *props,
                                 GtkStateFlags state,
                                 va_list args);
```

`gtk_style_properties_set_valist` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Sets several style properties on `props` .

Parameters

props	a GtkStyleProperties
state	state to set the values for
args	va_list of property name/value pairs, followed by `NULL`

Since: 3.0

---

gtk_style_properties_unset_property ()

```
void
gtk_style_properties_unset_property (GtkStyleProperties *props,
                                     const gchar *property,
                                     GtkStateFlags state);
```

`gtk_style_properties_unset_property` has been deprecated since version 3.16 and should not be used in newly-written code.

GtkStyleProperties are deprecated.

Unsets a style property in `props` .

Parameters

props	a GtkStyleProperties
property	property to unset
state	state to unset

Since: 3.0

Types and Values

struct GtkStyleProperties

```
struct GtkStyleProperties {
};
```

GtkWidgetPath

GtkWidgetPath — Widget path abstraction

Functions

gint	gtk_widget_path_append_type ()
gint	gtk_widget_path_append_with_siblings ()
gint	gtk_widget_path_append_for_widget ()
GtkWidgetPath *	gtk_widget_path_copy ()
GtkWidgetPath *	gtk_widget_path_ref ()
void	gtk_widget_path_unref ()
void	gtk_widget_path_free ()
GType	gtk_widget_path_get_object_type ()
gboolean	gtk_widget_path_has_parent ()
gboolean	gtk_widget_path_is_type ()
void	gtk_widget_path_iter_add_class ()
void	gtk_widget_path_iter_add_region ()
void	gtk_widget_path_iter_clear_classes ()
void	gtk_widget_path_iter_clear_regions ()
const gchar *	gtk_widget_path_iter_get_name ()
const char *	gtk_widget_path_iter_get_object_name ()
GType	gtk_widget_path_iter_get_object_type ()
const GtkWidgetPath *	gtk_widget_path_iter_get_siblings ()
guint	gtk_widget_path_iter_get_sibling_index ()
GtkStateFlags	gtk_widget_path_iter_get_state ()
gboolean	gtk_widget_path_iter_has_class ()
gboolean	gtk_widget_path_iter_has_name ()
gboolean	gtk_widget_path_iter_has_qclass ()
gboolean	gtk_widget_path_iter_has_qname ()
gboolean	gtk_widget_path_iter_has_qregion ()
gboolean	gtk_widget_path_iter_has_region ()
GSList *	gtk_widget_path_iter_list_classes ()
GSList *	gtk_widget_path_iter_list_regions ()
void	gtk_widget_path_iter_remove_class ()
void	gtk_widget_path_iter_remove_region ()
void	gtk_widget_path_iter_set_name ()
void	gtk_widget_path_iter_set_object_name ()
void	gtk_widget_path_iter_set_object_type ()
void	gtk_widget_path_iter_set_state ()
gint	gtk_widget_path_length ()
GtkWidgetPath *	gtk_widget_path_new ()
void	gtk_widget_path_prepend_type ()
char *	gtk_widget_path_to_string ()

Types and Values

GtkWidgetPath

Includes

```
#include <gtk/gtk.h>
```

Description

GtkWidgetPath is a boxed type that represents a widget hierarchy from the topmost widget, typically a toplevel, to any child. This widget path abstraction is used in GtkStyleContext on behalf of the real widget in order to query style information.

If you are using GTK+ widgets, you probably will not need to use this API directly, as there is `gtk_widget_get_path()`, and the style context returned by `gtk_widget_get_style_context()` will be automatically updated on widget hierarchy changes.

The widget path generation is generally simple:

Defining a button within a window

```
1 {
2   GtkWidgetPath *path;
3
4   path = gtk_widget_path_new ();
5   gtk_widget_path_append_type (path, GTK_TYPE_WINDOW);
6   gtk_widget_path_append_type (path, GTK_TYPE_BUTTON);
7 }
```

Although more complex information, such as widget names, or different classes (property that may be used by other widget types) and intermediate regions may be included:

Defining the first tab widget in a notebook

```
1 {
2   GtkWidgetPath *path;
3   guint pos;
4
5   path = gtk_widget_path_new ();
6
7   pos = gtk_widget_path_append_type (path, GTK_TYPE_NOTEBOOK);
8   gtk_widget_path_iter_add_region (path, pos, "tab", GTK_REGION_EVEN |
9 GTK_REGION_FIRST);
10
11  pos = gtk_widget_path_append_type (path, GTK_TYPE_LABEL);
12  gtk_widget_path_iter_set_name (path, pos, "first tab label");
  }
```

All this information will be used to match the style information that applies to the described widget.

Functions

gtk_widget_path_append_type ()

```
gint
gtk_widget_path_append_type (GtkWidgetPath *path,
                             GType type);
```

Appends a widget type to the widget hierarchy represented by `path` .

Parameters

path	a GtkWidgetPath
type	widget type to append

Returns

the position where the element was inserted

Since: 3.0

---

gtk_widget_path_append_with_siblings ()

```
gint
gtk_widget_path_append_with_siblings (GtkWidgetPath *path,
                                      GtkWidgetPath *siblings,
                                      guint sibling_index);
```

Appends a widget type with all its siblings to the widget hierarchy represented by `path` . Using this function instead of `gtk_widget_path_append_type()` will allow the CSS theming to use sibling matches in selectors and apply :nth-`child()` pseudo classes. In turn, it requires a lot more care in widget implementations as widgets need to make sure to call `gtk_widget_reset_style()` on all involved widgets when the `siblings` path changes.

Parameters

path	the widget path to append to
siblings	a widget path describing a list of siblings. This path may not contain any siblings itself and it must not be modified afterwards.
sibling_index	index into `siblings` for where the added element is positioned.

Returns

the position where the element was inserted.

Since: 3.2

---

gtk_widget_path_append_for_widget ()

```
gint
gtk_widget_path_append_for_widget (GtkWidgetPath *path,
                                   GtkWidget *widget);
```

Appends the data from `widget` to the widget hierarchy represented by `path` . This function is a shortcut for adding information from `widget` to the given `path` . This includes setting the name or adding the style classes from `widget` .

Parameters

path	a widget path
widget	the widget to append to the widget path

Returns

the position where the data was inserted

Since: 3.2

---

gtk_widget_path_copy ()

```
GtkWidgetPath *
gtk_widget_path_copy (const GtkWidgetPath *path);
```

Returns a copy of `path`

Parameters

path	a GtkWidgetPath

Returns

a copy of `path` .

[transfer full]

Since: 3.0

---

gtk_widget_path_ref ()

```
GtkWidgetPath *
gtk_widget_path_ref (GtkWidgetPath *path);
```

Increments the reference count on `path` .

Parameters

path	a GtkWidgetPath

Returns

`path` itself.

Since: 3.2

---

gtk_widget_path_unref ()

```
void
gtk_widget_path_unref (GtkWidgetPath *path);
```

Decrements the reference count on `path` , freeing the structure if the reference count reaches 0.

Parameters

path	a GtkWidgetPath

Since: 3.2

---

gtk_widget_path_free ()

```
void
gtk_widget_path_free (GtkWidgetPath *path);
```

Decrements the reference count on `path` , freeing the structure if the reference count reaches 0.

Parameters

path	a GtkWidgetPath

Since: 3.0

---

gtk_widget_path_get_object_type ()

```
GType
gtk_widget_path_get_object_type (const GtkWidgetPath *path);
```

Returns the topmost object type, that is, the object type this path is representing.

Parameters

path	a GtkWidget

Returns

The object type

Since: 3.0

---

gtk_widget_path_has_parent ()

```
gboolean
gtk_widget_path_has_parent (const GtkWidgetPath *path,
                            GType type);
```

Returns `TRUE` if any of the parents of the widget represented in `path` is of type `type` , or any subtype of it.

Parameters

path	a GtkWidgetPath
type	widget type to check in parents

Returns

`TRUE` if any parent is of type `type`

Since: 3.0

---

gtk_widget_path_is_type ()

```
gboolean
gtk_widget_path_is_type (const GtkWidgetPath *path,
                         GType type);
```

Returns TRUE if the widget type represented by this path is type , or a subtype of it.

Parameters

path	a GtkWidgetPath
type	widget type to match

Returns

TRUE if the widget represented by path is of type type

Since: 3.0

---

gtk_widget_path_iter_add_class ()

```
void
gtk_widget_path_iter_add_class (GtkWidgetPath *path,
                                gint pos,
                                const gchar *name);
```

Adds the class name to the widget at position pos in the hierarchy defined in path . See gtk_style_context_add_class().

Parameters

path	a GtkWidget
pos	position to modify, -1 for the path head
name	a class name

Since: 3.0

---

gtk_widget_path_iter_add_region ()

```
void
gtk_widget_path_iter_add_region (GtkWidgetPath *path,
                                 gint pos,
                                 const gchar *name,
                                 GtkRegionFlags flags);
```

gtk_widget_path_iter_add_region has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

Adds the region name to the widget at position pos in the hierarchy defined in path . See gtk_style_context_add_region().

Region names must only contain lowercase letters and “-”, starting always with a lowercase letter.

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
name	region name
flags	flags affecting the region

Since: 3.0

---

gtk_widget_path_iter_clear_classes ()

```
void
gtk_widget_path_iter_clear_classes (GtkWidgetPath *path,
                                    gint pos);
```

Removes all classes from the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidget
pos	position to modify, -1 for the path head

Since: 3.0

---

gtk_widget_path_iter_clear_regions ()

```
void
gtk_widget_path_iter_clear_regions (GtkWidgetPath *path,
                                    gint pos);
```

`gtk_widget_path_iter_clear_regions` has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

Removes all regions from the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head

Since: 3.0

---

gtk_widget_path_iter_get_name ()

```
const gchar *
gtk_widget_path_iter_get_name (const GtkWidgetPath *path,
                               gint pos);
```

Returns the name corresponding to the widget found at the position `pos` in the widget hierarchy defined by `path`

Parameters

path	a GtkWidgetPath
pos	position to get the widget name for, -1 for the path head

Returns

The widget name, or `NULL` if none was set.

[nullable]

---

gtk_widget_path_iter_get_object_name ()

```
const char *
gtk_widget_path_iter_get_object_name (const GtkWidgetPath *path,
                                      gint pos);
```

Returns the object name that is at position `pos` in the widget hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to get the object name for, -1 for the path head

Returns

the name or `NULL`.

[nullable]

Since: 3.20

---

gtk_widget_path_iter_get_object_type ()

```
GType
gtk_widget_path_iter_get_object_type (const GtkWidgetPath *path,
                                      gint pos);
```

Returns the object GType that is at position `pos` in the widget hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to get the object type for, -1 for the path head

Returns

a widget type

Since: 3.0

---

gtk_widget_path_iter_get_siblings ()

```
const GtkWidgetPath *
gtk_widget_path_iter_get_siblings (const GtkWidgetPath *path,
                                   gint pos);
```

Returns the list of siblings for the element at `pos` . If the element was not added with siblings, `NULL` is returned.

Parameters

path	a GtkWidgetPath
pos	position to get the siblings for, -1 for the path head

Returns

`NULL` or the list of siblings for the element at `pos` .

---

gtk_widget_path_iter_get_sibling_index ()

```
guint
gtk_widget_path_iter_get_sibling_index
                                (const GtkWidgetPath *path,
                                 gint pos);
```

Returns the index into the list of siblings for the element at `pos` as returned by `gtk_widget_path_iter_get_siblings()`. If that function would return `NULL` because the element at `pos` has no siblings, this function will return 0.

Parameters

path	a GtkWidgetPath
pos	position to get the sibling index for, -1 for the path head

Returns

0 or the index into the list of siblings for the element at `pos` .

gtk_widget_path_iter_get_state ()

```
GtkStateFlags
gtk_widget_path_iter_get_state (const GtkWidgetPath *path,
                                gint pos);
```

Returns the state flags corresponding to the widget found at the position `pos` in the widget hierarchy defined by `path`

Parameters

path	a GtkWidgetPath
pos	position to get the state for, -1 for the path head

Returns

The state flags

Since: 3.14

---

gtk_widget_path_iter_has_class ()

```
gboolean
gtk_widget_path_iter_has_class (const GtkWidgetPath *path,
                                gint pos,
                                const gchar *name);
```

Returns `TRUE` if the widget at position `pos` has the class `name` defined, `FALSE` otherwise.

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head
name	class name

Returns

`TRUE` if the class `name` is defined for the widget at `pos`

Since: 3.0

---

gtk_widget_path_iter_has_name ()

```
gboolean
```

```
gtk_widget_path_iter_has_name (const GtkWidgetPath *path,
                               gint pos,
                               const gchar *name);
```

Returns TRUE if the widget at position pos has the name name , FALSE otherwise.

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head
name	a widget name

Returns

TRUE if the widget at pos has this name

Since: 3.0

---

gtk_widget_path_iter_has_qclass ()

```
gboolean
gtk_widget_path_iter_has_qclass (const GtkWidgetPath *path,
                                 gint pos,
                                 GQuark qname);
```

See gtk_widget_path_iter_has_class(). This is a version that operates with GQuarks.

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head
qname	class name as a GQuark

Returns

TRUE if the widget at pos has the class defined.

Since: 3.0

---

gtk_widget_path_iter_has_qname ()

```
gboolean
gtk_widget_path_iter_has_qname (const GtkWidgetPath *path,
                                gint pos,
                                GQuark qname);
```

See gtk_widget_path_iter_has_name(). This is a version that operates on GQuarks.

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head
qname	widget name as a GQuark

Returns

TRUE if the widget at pos has this name

Since: 3.0

---

gtk_widget_path_iter_has_qregion ()

```
gboolean
gtk_widget_path_iter_has_qregion (const GtkWidgetPath *path,
                                  gint pos,
                                  GQuark qname,
                                  GtkRegionFlags *flags);
```

gtk_widget_path_iter_has_qregion has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

See gtk_widget_path_iter_has_region(). This is a version that operates with GQuarks.

Parameters

path	a GtkWidgetPath	
pos	position to query, -1 for the path head	
qname	region name as a GQuark	
flags	return location for the region flags.	[out]

Returns

TRUE if the widget at pos has the region defined.

Since: 3.0

---

gtk_widget_path_iter_has_region ()

```
gboolean
gtk_widget_path_iter_has_region (const GtkWidgetPath *path,
                                 gint pos,
                                 const gchar *name,
                                 GtkRegionFlags *flags);
```

gtk_widget_path_iter_has_region has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

Returns `TRUE` if the widget at position `pos` has the class `name` defined, `FALSE` otherwise.

Parameters

path	a GtkWidgetPath	
pos	position to query, -1 for the path head	
name	region name	
flags	return location for the region flags.	[out]

Returns

`TRUE` if the class `name` is defined for the widget at `pos`

Since: 3.0

---

gtk_widget_path_iter_list_classes ()

```
GSList *
gtk_widget_path_iter_list_classes (const GtkWidgetPath *path,
                                   gint pos);
```

Returns a list with all the class names defined for the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head

Returns

The list of classes, This is a list of strings, the GSList contents are owned by GTK+, but you should use `g_slist_free()` to free the list itself.

[transfer container][element-type utf8]

Since: 3.0

---

gtk_widget_path_iter_list_regions ()

```
GSList *
gtk_widget_path_iter_list_regions (const GtkWidgetPath *path,
                                   gint pos);
```

`gtk_widget_path_iter_list_regions` has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

Returns a list with all the region names defined for the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to query, -1 for the path head

Returns

The list of regions, This is a list of strings, the GSList contents are owned by GTK+, but you should use `g_slist_free()` to free the list itself.

[transfer container][element-type utf8]

Since: 3.0

---

gtk_widget_path_iter_remove_class ()

```
void
gtk_widget_path_iter_remove_class (GtkWidgetPath *path,
                                   gint pos,
                                   const gchar *name);
```

Removes the class `name` from the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
name	class name

Since: 3.0

---

gtk_widget_path_iter_remove_region ()

```
void
gtk_widget_path_iter_remove_region (GtkWidgetPath *path,
                                    gint pos,
                                    const gchar *name);
```

`gtk_widget_path_iter_remove_region` has been deprecated since version 3.14 and should not be used in newly-written code.

The use of regions is deprecated.

Removes the region `name` from the widget at position `pos` in the hierarchy defined in `path` .

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
name	region name

Since: 3.0

---

gtk_widget_path_iter_set_name ()

```
void
gtk_widget_path_iter_set_name (GtkWidgetPath *path,
                               gint pos,
                               const gchar *name);
```

Sets the widget name for the widget found at position `pos` in the widget hierarchy defined by `path` .

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
name	widget name

Since: 3.0

---

gtk_widget_path_iter_set_object_name ()

```
void
gtk_widget_path_iter_set_object_name (GtkWidgetPath *path,
                                      gint pos,
                                      const char *name);
```

Sets the object name for a given position in the widget hierarchy defined by `path` .

When set, the object name overrides the object type when matching CSS.

Parameters

path	a GtkWidgetPath	
pos	position to modify, -1 for the path head	
name	object name to set or `NULL` to unset.	[allow-none]

Since: 3.20

---

gtk_widget_path_iter_set_object_type ()

```
void
gtk_widget_path_iter_set_object_type (GtkWidgetPath *path,
                                      gint pos,
                                      GType type);
```

Sets the object type for a given position in the widget hierarchy defined by `path` .

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
type	object type to set

Since: 3.0

---

gtk_widget_path_iter_set_state ()

```
void
gtk_widget_path_iter_set_state (GtkWidgetPath *path,
                                gint pos,
                                GtkStateFlags state);
```

Sets the widget name for the widget found at position `pos` in the widget hierarchy defined by `path` .

If you want to update just a single state flag, you need to do this manually, as this function updates all state flags.

Setting a flag

```
1gtk_widget_path_iter_set_state (path, pos, gtk_widget_path_iter_get_state (path, pos) |
 flag);
```

Unsetting a flag

```
1gtk_widget_path_iter_set_state (path, pos, gtk_widget_path_iter_get_state (path, pos) &
 ~flag);
```

Parameters

path	a GtkWidgetPath
pos	position to modify, -1 for the path head
state	state flags

Since: 3.14

---

gtk_widget_path_length ()

```
gint
gtk_widget_path_length (const GtkWidgetPath *path);
```

Returns the number of GtkWidget GTypes between the represented widget and its topmost container.

Parameters

path	a GtkWidgetPath

Returns

the number of elements in the path

Since: 3.0

---

gtk_widget_path_new ()

```
GtkWidgetPath *
gtk_widget_path_new (void);
```

Returns an empty widget path.

Returns

A newly created, empty, GtkWidgetPath.

[transfer full]

Since: 3.0

---

gtk_widget_path_prepend_type ()

```
void
gtk_widget_path_prepend_type (GtkWidgetPath *path,
                              GType type);
```

Prepends a widget type to the widget hierachy represented by `path` .

Parameters

path	a GtkWidgetPath
type	widget type to prepend

Since: 3.0

---

gtk_widget_path_to_string ()

```
char *
gtk_widget_path_to_string (const GtkWidgetPath *path);
```

Dumps the widget path into a string representation. It tries to match the CSS style as closely as possible (Note that there might be paths that cannot be represented in CSS).

The main use of this code is for debugging purposes, so that you can `g_print()` the path or dump it in a gdb session.

Parameters

path	the path

Returns

A new string describing `path` .

Since: 3.2

Types and Values

GtkWidgetPath

```
typedef struct _GtkWidgetPath GtkWidgetPath;
```

See Also

GtkStyleContext

GtkIconTheme

GtkIconTheme — Looking up icons by name

Functions

GtkIconTheme *	gtk_icon_theme_new ()
GtkIconTheme *	gtk_icon_theme_get_default ()
GtkIconTheme *	gtk_icon_theme_get_for_screen ()
void	gtk_icon_theme_set_screen ()
void	gtk_icon_theme_set_search_path ()
void	gtk_icon_theme_get_search_path ()
void	gtk_icon_theme_append_search_path ()
void	gtk_icon_theme_prepend_search_path ()
void	gtk_icon_theme_add_resource_path ()
void	gtk_icon_theme_set_custom_theme ()
gboolean	gtk_icon_theme_has_icon ()
GtkIconInfo *	gtk_icon_theme_lookup_icon ()
GtkIconInfo *	gtk_icon_theme_lookup_icon_for_scale ()
GtkIconInfo *	gtk_icon_theme_choose_icon ()
GtkIconInfo *	gtk_icon_theme_choose_icon_for_scale ()
GtkIconInfo *	gtk_icon_theme_lookup_by_gicon ()
GtkIconInfo *	gtk_icon_theme_lookup_by_gicon_for_scale ()
GdkPixbuf *	gtk_icon_theme_load_icon ()
GdkPixbuf *	gtk_icon_theme_load_icon_for_scale ()
cairo_surface_t *	gtk_icon_theme_load_surface ()
GList *	gtk_icon_theme_list_contexts ()
GList *	gtk_icon_theme_list_icons ()
gint *	gtk_icon_theme_get_icon_sizes ()
char *	gtk_icon_theme_get_example_icon_name ()
gboolean	gtk_icon_theme_rescan_if_needed ()
void	gtk_icon_theme_add_builtin_icon ()
GtkIconInfo *	gtk_icon_info_copy ()
void	gtk_icon_info_free ()
GtkIconInfo *	gtk_icon_info_new_for_pixbuf ()
gint	gtk_icon_info_get_base_size ()
gint	gtk_icon_info_get_base_scale ()
const gchar *	gtk_icon_info_get_filename ()

GdkPixbuf *	gtk_icon_info_get_builtin_pixbuf ()
GdkPixbuf *	gtk_icon_info_load_icon ()
cairo_surface_t *	gtk_icon_info_load_surface ()
void	gtk_icon_info_load_icon_async ()
GdkPixbuf *	gtk_icon_info_load_icon_finish ()
GdkPixbuf *	gtk_icon_info_load_symbolic ()
void	gtk_icon_info_load_symbolic_async ()
GdkPixbuf *	gtk_icon_info_load_symbolic_finish ()
GdkPixbuf *	gtk_icon_info_load_symbolic_for_style ()
GdkPixbuf *	gtk_icon_info_load_symbolic_for_context ()
void	gtk_icon_info_load_symbolic_for_context_async ()
GdkPixbuf *	gtk_icon_info_load_symbolic_for_context_finish ()
void	gtk_icon_info_set_raw_coordinates ()
gboolean	gtk_icon_info_get_embedded_rect ()
gboolean	gtk_icon_info_get_attach_points ()
const gchar *	gtk_icon_info_get_display_name ()
gboolean	gtk_icon_info_is_symbolic ()

Signals

void	changed	Run Last

Types and Values

	GtkIconInfo
struct	GtkIconTheme
struct	GtkIconThemeClass
enum	GtkIconLookupFlags
#define	GTK_ICON_THEME_ERROR
enum	GtkIconThemeError

Object Hierarchy

```
GObject
└── GtkIconTheme
```

Includes

```
#include <gtk/gtk.h>
```

Description

GtkIconTheme provides a facility for looking up icons by name and size. The main reason for using a name rather than simply providing a filename is to allow different icons to be used depending on what “icon theme” is selected by the user. The operation of icon themes on Linux and Unix follows the Icon Theme Specification There is a fallback icon theme, named `hicolor`, where applications should install their icons, but additional icon themes can be installed as operating system vendors and users choose.

Named icons are similar to the deprecated Stock Items, and the distinction between the two may be a bit confusing. A few things to keep in mind:

- Stock images usually are used in conjunction with Stock Items, such as GTK_STOCK_OK or GTK_STOCK_OPEN. Named icons are easier to set up and therefore are more useful for new icons that an application wants to add, such as application icons or window icons.
- Stock images can only be loaded at the symbolic sizes defined by the GtkIconSize enumeration, or by custom sizes defined by gtk_icon_size_register(), while named icons are more flexible and any pixel size can be specified.
- Because stock images are closely tied to stock items, and thus to actions in the user interface, stock images may come in multiple variants for different widget states or writing directions.

A good rule of thumb is that if there is a stock image for what you want to use, use it, otherwise use a named icon. It turns out that internally stock images are generally defined in terms of one or more named icons. (An example of the more than one case is icons that depend on writing direction; GTK_STOCK_GO_FORWARD uses the two themed icons “gtk-stock-go-forward-ltr” and “gtk-stock-go-forward-rtl”.)

In many cases, named themes are used indirectly, via GtkImage or stock items, rather than directly, but looking up icons directly is also simple. The GtkIconTheme object acts as a database of all the icons in the current theme. You can create new GtkIconTheme objects, but it’s much more efficient to use the standard icon theme for the GdkScreen so that the icon information is shared with other people looking up icons.

```
1  GError *error = NULL;
2  GtkIconTheme *icon_theme;
3  GdkPixbuf *pixbuf;
4
5  icon_theme = gtk_icon_theme_get_default ();
6  pixbuf = gtk_icon_theme_load_icon (icon_theme,
7                                     "my-icon-name", // icon name
8                                     48, // icon size
9                                     0,  // flags
10                                    &error);
11 if (!pixbuf)
12   {
13     g_warning ("Couldn’t load icon: %s", error->message);
14     g_error_free (error);
15   }
16 else
17   {
18     // Use the pixbuf
19     g_object_unref (pixbuf);
20   }
```

Functions

gtk_icon_theme_new ()

```
GtkIconTheme *
gtk_icon_theme_new (void);
```

Creates a new icon theme object. Icon theme objects are used to lookup up an icon by name in a particular icon theme. Usually, you’ll want to use gtk_icon_theme_get_default() or gtk_icon_theme_get_for_screen() rather than creating a new icon theme object for scratch.

Returns

the newly created GtkIconTheme object.

Since: 2.4

---

gtk_icon_theme_get_default ()

```
GtkIconTheme *
gtk_icon_theme_get_default (void);
```

Gets the icon theme for the default screen. See `gtk_icon_theme_get_for_screen()`.

Returns

A unique GtkIconTheme associated with the default screen. This icon theme is associated with the screen and can be used as long as the screen is open. Do not ref or unref it.

[transfer none]

Since: 2.4

---

gtk_icon_theme_get_for_screen ()

```
GtkIconTheme *
gtk_icon_theme_get_for_screen (GdkScreen *screen);
```

Gets the icon theme object associated with `screen` ; if this function has not previously been called for the given screen, a new icon theme object will be created and associated with the screen. Icon theme objects are fairly expensive to create, so using this function is usually a better choice than calling than `gtk_icon_theme_new()` and setting the screen yourself; by using this function a single icon theme object will be shared between users.

Parameters

screen	a GdkScreen

Returns

A unique GtkIconTheme associated with the given screen. This icon theme is associated with the screen and can be used as long as the screen is open. Do not ref or unref it.

[transfer none]

Since: 2.4

---

gtk_icon_theme_set_screen ()

```
void
gtk_icon_theme_set_screen (GtkIconTheme *icon_theme,
                           GdkScreen *screen);
```

Sets the screen for an icon theme; the screen is used to track the user's currently configured icon theme, which might be different for different screens.

Parameters

icon_theme	a GtkIconTheme
screen	a GdkScreen

Since: 2.4

---

gtk_icon_theme_set_search_path ()

```
void
gtk_icon_theme_set_search_path (GtkIconTheme *icon_theme,
                                const gchar *path[],
                                gint n_elements);
```

Sets the search path for the icon theme object. When looking for an icon theme, GTK+ will search for a subdirectory of one or more of the directories in `path` with the same name as the icon theme containing an index.theme file. (Themes from multiple of the path elements are combined to allow themes to be extended by adding icons in the user's home directory.)

In addition if an icon found isn't found either in the current icon theme or the default icon theme, and an image file with the right name is found directly in one of the elements of `path` , then that image will be used for the icon name. (This is legacy feature, and new icons should be put into the fallback icon theme, which is called hicolor, rather than directly on the icon path.)

Parameters

icon_theme	a GtkIconTheme	
path	array of directories that are searched for icon themes.	[array length=n_elements][element-type filename]
n_elements	number of elements in `path` .	

Since: 2.4

---

gtk_icon_theme_get_search_path ()

```
void
gtk_icon_theme_get_search_path (GtkIconTheme *icon_theme,
                                gchar **path[],
                                gint *n_elements);
```

Gets the current search path. See `gtk_icon_theme_set_search_path()`.

Parameters

icon_theme	a GtkIconTheme	
path	location to store a list of icon theme path directories or `NULL`. The stored value should be freed with `g_strfreev()`.	[allow-none][array length=n_elements][element-type filename][out]
n_elements	location to store number of elements in `path` , or `NULL`	

Since: 2.4

---

gtk_icon_theme_append_search_path ()

```
void
gtk_icon_theme_append_search_path (GtkIconTheme *icon_theme,
                                   const gchar *path);
```

Appends a directory to the search path. See `gtk_icon_theme_set_search_path()`.

Parameters

icon_theme	a GtkIconTheme	
path	directory name to append to the icon path.	[type filename]

Since: 2.4

---

gtk_icon_theme_prepend_search_path ()

```
void
gtk_icon_theme_prepend_search_path (GtkIconTheme *icon_theme,
                                    const gchar *path);
```

Prepends a directory to the search path. See `gtk_icon_theme_set_search_path()`.

Parameters

icon_theme	a GtkIconTheme	
path	directory name to prepend to the icon path.	[type filename]

Since: 2.4

---

gtk_icon_theme_add_resource_path ()

```
void
gtk_icon_theme_add_resource_path (GtkIconTheme *icon_theme,
                                  const gchar *path);
```

Adds a resource path that will be looked at when looking for icons, similar to search paths.

This function should be used to make application-specific icons available as part of the icon theme.

The resources are considered as part of the hicolor icon theme and must be located in subdirectories that are

defined in the hicolor icon theme, such as `@path/16x16/actions/run.png`. Icons that are directly placed in the resource path instead of a subdirectory are also considered as ultimate fallback.

Parameters

icon_theme	a GtkIconTheme
path	a resource path

Since: 3.14

---

gtk_icon_theme_set_custom_theme ()

```
void
gtk_icon_theme_set_custom_theme (GtkIconTheme *icon_theme,
                                 const gchar *theme_name);
```

Sets the name of the icon theme that the GtkIconTheme object uses overriding system configuration. This function cannot be called on the icon theme objects returned from `gtk_icon_theme_get_default()` and `gtk_icon_theme_get_for_screen()`.

Parameters

icon_theme	a GtkIconTheme	
theme_name	name of icon theme to use instead of configured theme, or `NULL` to unset a previously set custom theme.	[allow-none]

Since: 2.4

---

gtk_icon_theme_has_icon ()

```
gboolean
gtk_icon_theme_has_icon (GtkIconTheme *icon_theme,
                         const gchar *icon_name);
```

Checks whether an icon theme includes an icon for a particular name.

Parameters

icon_theme	a GtkIconTheme
icon_name	the name of an icon

Returns

`TRUE` if `icon_theme` includes an icon for `icon_name` .

Since: 2.4

---

gtk_icon_theme_lookup_icon ()

```
GtkIconInfo *
gtk_icon_theme_lookup_icon (GtkIconTheme *icon_theme,
                            const gchar *icon_name,
                            gint size,
                            GtkIconLookupFlags flags);
```

Looks up a named icon and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`. (`gtk_icon_theme_load_icon()` combines these two steps if all you need is the pixbuf.)

When rendering on displays with high pixel densities you should not use a `size` multiplied by the scaling factor returned by functions like `gdk_window_get_scale_factor()`. Instead, you should use `gtk_icon_theme_lookup_icon_for_scale()`, as the assets loaded for a given scaling factor may be different.

Parameters

icon_theme	a GtkIconTheme
icon_name	the name of the icon to lookup
size	desired icon size
flags	flags modifying the behavior of the icon lookup

Returns

a GtkIconInfo object containing information about the icon, or `NULL` if the icon wasn't found.

[nullable][transfer full]

Since: 2.4

---

gtk_icon_theme_lookup_icon_for_scale ()

```
GtkIconInfo *
gtk_icon_theme_lookup_icon_for_scale (GtkIconTheme *icon_theme,
                                      const gchar *icon_name,
                                      gint size,
                                      gint scale,
                                      GtkIconLookupFlags flags);
```

Looks up a named icon for a particular window scale and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`. (`gtk_icon_theme_load_icon()` combines these two steps if all you need is the pixbuf.)

Parameters

icon_theme	a GtkIconTheme
icon_name	the name of the icon to lookup
size	desired icon size

scale	the desired scale
flags	flags modifying the behavior of the icon lookup

Returns

a GtkIconInfo object containing information about the icon, or `NULL` if the icon wasn't found.

[nullable][transfer full]

Since: 3.10

---

gtk_icon_theme_choose_icon ()

```
GtkIconInfo *
gtk_icon_theme_choose_icon (GtkIconTheme *icon_theme,
                            const gchar *icon_names[],
                            gint size,
                            GtkIconLookupFlags flags);
```

Looks up a named icon and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`. (`gtk_icon_theme_load_icon()` combines these two steps if all you need is the pixbuf.)

If `icon_names` contains more than one name, this function tries them all in the given order before falling back to inherited icon themes.

Parameters

icon_theme	a GtkIconTheme	
icon_names	`NULL`-terminated array of icon names to lookup.	[array zero-terminated=1]
size	desired icon size	
flags	flags modifying the behavior of the icon lookup	

Returns

a GtkIconInfo object containing information about the icon, or `NULL` if the icon wasn't found.

[nullable][transfer full]

Since: 2.12

---

gtk_icon_theme_choose_icon_for_scale ()

```
GtkIconInfo *
gtk_icon_theme_choose_icon_for_scale (GtkIconTheme *icon_theme,
                                      const gchar *icon_names[],
                                      gint size,
                                      gint scale,
                                      GtkIconLookupFlags flags);
```

Looks up a named icon for a particular window scale and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`. (`gtk_icon_theme_load_icon()` combines these two steps if all you need is the pixbuf.)

If `icon_names` contains more than one name, this function tries them all in the given order before falling back to inherited icon themes.

Parameters

icon_theme	a GtkIconTheme	
icon_names	`NULL`-terminated array of icon names to lookup.	[array zero-terminated=1]
size	desired icon size	
scale	desired scale	
flags	flags modifying the behavior of the icon lookup	

Returns

a GtkIconInfo object containing information about the icon, or `NULL` if the icon wasn't found.

[nullable][transfer full]

Since: 3.10

---

gtk_icon_theme_lookup_by_gicon ()

```
GtkIconInfo *
gtk_icon_theme_lookup_by_gicon (GtkIconTheme *icon_theme,
                                GIcon *icon,
                                gint size,
                                GtkIconLookupFlags flags);
```

Looks up an icon and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`.

When rendering on displays with high pixel densities you should not use a `size` multiplied by the scaling factor returned by functions like `gdk_window_get_scale_factor()`. Instead, you should use `gtk_icon_theme_lookup_by_gicon_for_scale()`, as the assets loaded for a given scaling factor may be different.

Parameters

icon_theme	a GtkIconTheme
icon	the GIcon to look up
size	desired icon size
flags	flags modifying the behavior of the icon lookup

Returns

a GtkIconInfo containing information about the icon, or `NULL` if the icon wasn't found. Unref with `g_object_unref()`.

[nullable][transfer full]

Since: 2.14

---

gtk_icon_theme_lookup_by_gicon_for_scale ()

```
GtkIconInfo *
gtk_icon_theme_lookup_by_gicon_for_scale
                                (GtkIconTheme *icon_theme,
                                 GIcon *icon,
                                 gint size,
                                 gint scale,
                                 GtkIconLookupFlags flags);
```

Looks up an icon and returns a GtkIconInfo containing information such as the filename of the icon. The icon can then be rendered into a pixbuf using `gtk_icon_info_load_icon()`.

Parameters

icon_theme	a GtkIconTheme
icon	the GIcon to look up
size	desired icon size
scale	the desired scale
flags	flags modifying the behavior of the icon lookup

Returns

a GtkIconInfo containing information about the icon, or `NULL` if the icon wasn't found. Unref with `g_object_unref()`.

[nullable][transfer full]

Since: 3.10

---

gtk_icon_theme_load_icon ()

```
GdkPixbuf *
gtk_icon_theme_load_icon (GtkIconTheme *icon_theme,
                          const gchar *icon_name,
                          gint size,
                          GtkIconLookupFlags flags,
                          GError **error);
```

Looks up an icon in an icon theme, scales it to the given size and renders it into a pixbuf. This is a convenience function; if more details about the icon are needed, use `gtk_icon_theme_lookup_icon()` followed by

gtk_icon_info_load_icon().

Note that you probably want to listen for icon theme changes and update the icon. This is usually done by connecting to the GtkWidget::style-set signal. If for some reason you do not want to update the icon when the icon theme changes, you should consider using gdk_pixbuf_copy() to make a private copy of the pixbuf returned by this function. Otherwise GTK+ may need to keep the old icon theme loaded, which would be a waste of memory.

Parameters

icon_theme	a GtkIconTheme	
icon_name	the name of the icon to lookup	
size	the desired icon size. The resulting icon may not be exactly this size; see gtk_icon_info_load_icon().	
flags	flags modifying the behavior of the icon lookup	
error	Location to store error information on failure, or NULL.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use g_object_unref() to release your reference to the icon. NULL if the icon isn't found.

[nullable][transfer full]

Since: 2.4

---

gtk_icon_theme_load_icon_for_scale ()

```
GdkPixbuf *
gtk_icon_theme_load_icon_for_scale (GtkIconTheme *icon_theme,
                                    const gchar *icon_name,
                                    gint size,
                                    gint scale,
                                    GtkIconLookupFlags flags,
                                    GError **error);
```

Looks up an icon in an icon theme for a particular window scale, scales it to the given size and renders it into a pixbuf. This is a convenience function; if more details about the icon are needed, use gtk_icon_theme_lookup_icon() followed by gtk_icon_info_load_icon().

Note that you probably want to listen for icon theme changes and update the icon. This is usually done by connecting to the GtkWidget::style-set signal. If for some reason you do not want to update the icon when the icon theme changes, you should consider using gdk_pixbuf_copy() to make a private copy of the pixbuf returned by this function. Otherwise GTK+ may need to keep the old icon theme loaded, which would be a waste of memory.

Parameters

icon_theme	a GtkIconTheme

icon_name	the name of the icon to lookup	
size	the desired icon size. The resulting icon may not be exactly this size; see `gtk_icon_info_load_icon()`.	
scale	desired scale	
flags	flags modifying the behavior of the icon lookup	
error	Location to store error information on failure, or `NULL`.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use `g_object_unref()` to release your reference to the icon. `NULL` if the icon isn't found.

[nullable][transfer full]

Since: 3.10

---

gtk_icon_theme_load_surface ()

```
cairo_surface_t *
gtk_icon_theme_load_surface (GtkIconTheme *icon_theme,
                             const gchar *icon_name,
                             gint size,
                             gint scale,
                             GdkWindow *for_window,
                             GtkIconLookupFlags flags,
                             GError **error);
```

Looks up an icon in an icon theme for a particular window scale, scales it to the given size and renders it into a cairo surface. This is a convenience function; if more details about the icon are needed, use `gtk_icon_theme_lookup_icon()` followed by `gtk_icon_info_load_surface()`.

Note that you probably want to listen for icon theme changes and update the icon. This is usually done by connecting to the GtkWidget::style-set signal.

Parameters

icon_theme	a GtkIconTheme	
icon_name	the name of the icon to lookup	
size	the desired icon size. The resulting icon may not be exactly this size; see `gtk_icon_info_load_icon()`.	
scale	desired scale	
for_window	GdkWindow to optimize drawing for, or `NULL`.	[allow-none]
flags	flags modifying the behavior of the icon lookup	
error	Location to store error information on failure, or `NULL`.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use `cairo_surface_destroy()` to release your reference to the icon. `NULL` if the icon isn't found.

[nullable][transfer full]

Since: 3.10

---

gtk_icon_theme_list_contexts ()

```
GList *
gtk_icon_theme_list_contexts (GtkIconTheme *icon_theme);
```

Gets the list of contexts available within the current hierarchy of icon themes. See `gtk_icon_theme_list_icons()` for details about contexts.

Parameters

icon_theme	a GtkIconTheme

Returns

a GList list holding the names of all the contexts in the theme. You must first free each element in the list with `g_free()`, then free the list itself with `g_list_free()`.

[element-type utf8][transfer full]

Since: 2.12

---

gtk_icon_theme_list_icons ()

```
GList *
gtk_icon_theme_list_icons (GtkIconTheme *icon_theme,
                           const gchar *context);
```

Lists the icons in the current icon theme. Only a subset of the icons can be listed by providing a context string. The set of values for the context string is system dependent, but will typically include such values as “Applications” and “MimeTypes”. Contexts are explained in the Icon Theme Specification. The standard contexts are listed in the Icon Naming Specification. Also see `gtk_icon_theme_list_contexts()`.

Parameters

icon_theme	a GtkIconTheme	
context	a string identifying a particular type of icon, or `NULL` to list all icons.	[allow-none]

Returns

a GList list holding the names of all the icons in the theme. You must first free each element in the list with `g_free()`, then free the list itself with `g_list_free()`.

[element-type utf8][transfer full]

Since: 2.4

gtk_icon_theme_get_icon_sizes ()

```
gint *
gtk_icon_theme_get_icon_sizes (GtkIconTheme *icon_theme,
                               const gchar *icon_name);
```

Returns an array of integers describing the sizes at which the icon is available without scaling. A size of -1 means that the icon is available in a scalable format. The array is zero-terminated.

Parameters

icon_theme	a GtkIconTheme
icon_name	the name of an icon

Returns

An newly allocated array describing the sizes at which the icon is available. The array should be freed with `g_free()` when it is no longer needed.

[array zero-terminated=1][transfer full]

Since: 2.6

---

gtk_icon_theme_get_example_icon_name ()

```
char *
gtk_icon_theme_get_example_icon_name (GtkIconTheme *icon_theme);
```

Gets the name of an icon that is representative of the current theme (for instance, to use when presenting a list of themes to the user.)

Parameters

icon_theme	a GtkIconTheme

Returns

the name of an example icon or `NULL`. Free with `g_free()`.

[nullable]

Since: 2.4

---

gtk_icon_theme_rescan_if_needed ()

```
gboolean
gtk_icon_theme_rescan_if_needed (GtkIconTheme *icon_theme);
```

Checks to see if the icon theme has changed; if it has, any currently cached information is discarded and will be reloaded next time `icon_theme` is accessed.

Parameters

icon_theme	a GtkIconTheme

Returns

`TRUE` if the icon theme has changed and needed to be reloaded.

Since: 2.4

---

gtk_icon_theme_add_builtin_icon ()

```
void
gtk_icon_theme_add_builtin_icon (const gchar *icon_name,
                                 gint size,
                                 GdkPixbuf *pixbuf);
```

`gtk_icon_theme_add_builtin_icon` has been deprecated since version 3.14 and should not be used in newly-written code.

Use `gtk_icon_theme_add_resource_path()` to add application-specific icons to the icon theme.

Registers a built-in icon for icon theme lookups. The idea of built-in icons is to allow an application or library that uses themed icons to function requiring files to be present in the file system. For instance, the default images for all of GTK+'s stock icons are registered as built-icons.

In general, if you use `gtk_icon_theme_add_builtin_icon()` you should also install the icon in the icon theme, so that the icon is generally available.

This function will generally be used with pixbufs loaded via `gdk_pixbuf_new_from_inline()`.

Parameters

icon_name	the name of the icon to register
size	the size in pixels at which to register the icon (different images can be registered for the same icon name at different sizes.)
pixbuf	GdkPixbuf that contains the image to use for `icon_name`

Since: 2.4

---

gtk_icon_info_copy ()

```
GtkIconInfo *
gtk_icon_info_copy (GtkIconInfo *icon_info);
```

`gtk_icon_info_copy` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `g_object_ref()`

Make a copy of a GtkIconInfo.

[skip]

Parameters

icon_info	a GtkIconInfo

Returns

the new GtkIconInfo.

[transfer full]

Since: 2.4

---

gtk_icon_info_free ()

```
void
gtk_icon_info_free (GtkIconInfo *icon_info);
```

`gtk_icon_info_free` has been deprecated since version 3.8 and should not be used in newly-written code.

Use `g_object_unref()`

Free a GtkIconInfo and associated information

[skip]

Parameters

icon_info	a GtkIconInfo

Since: 2.4

---

gtk_icon_info_new_for_pixbuf ()

```
GtkIconInfo *
gtk_icon_info_new_for_pixbuf (GtkIconTheme *icon_theme,
                              GdkPixbuf *pixbuf);
```

Creates a GtkIconInfo for a GdkPixbuf.

Parameters

icon_theme	a GtkIconTheme
pixbuf	the pixbuf to wrap in a GtkIconInfo

Returns

a GtkIconInfo.

[transfer full]

Since: 2.14

---

gtk_icon_info_get_base_size ()

```
gint
gtk_icon_info_get_base_size (GtkIconInfo *icon_info);
```

Gets the base size for the icon. The base size is a size for the icon that was specified by the icon theme creator. This may be different than the actual size of image; an example of this is small emblem icons that can be attached to a larger icon. These icons will be given the same base size as the larger icons to which they are attached.

Note that for scaled icons the base size does not include the base scale.

Parameters

icon_info	a GtkIconInfo

Returns

the base size, or 0, if no base size is known for the icon.

Since: 2.4

---

gtk_icon_info_get_base_scale ()

```
gint
gtk_icon_info_get_base_scale (GtkIconInfo *icon_info);
```

Gets the base scale for the icon. The base scale is a scale for the icon that was specified by the icon theme creator. For instance an icon drawn for a high-dpi screen with window scale 2 for a base size of 32 will be 64 pixels tall and have a base scale of 2.

Parameters

icon_info	a GtkIconInfo

Returns

the base scale

Since: 3.10

---

gtk_icon_info_get_filename ()

```
const gchar *
gtk_icon_info_get_filename (GtkIconInfo *icon_info);
```

Gets the filename for the icon. If the `GTK_ICON_LOOKUP_USE_BUILTIN` flag was passed to `gtk_icon_theme_lookup_icon()`, there may be no filename if a builtin icon is returned; in this case, you should use `gtk_icon_info_get_builtin_pixbuf()`.

Parameters

icon_info	a GtkIconInfo

Returns

the filename for the icon, or `NULL` if `gtk_icon_info_get_builtin_pixbuf()` should be used instead. The return value is owned by GTK+ and should not be modified or freed.

[nullable][type filename]

Since: 2.4

---

gtk_icon_info_get_builtin_pixbuf ()

```
GdkPixbuf *
gtk_icon_info_get_builtin_pixbuf (GtkIconInfo *icon_info);
```

`gtk_icon_info_get_builtin_pixbuf` has been deprecated since version 3.14 and should not be used in newly-written code.

This function is deprecated, use `gtk_icon_theme_add_resource_path()` instead of builtin icons.

Gets the built-in image for this icon, if any. To allow GTK+ to use built in icon images, you must pass the `GTK_ICON_LOOKUP_USE_BUILTIN` to `gtk_icon_theme_lookup_icon()`.

Parameters

icon_info	a GtkIconInfo

Returns

the built-in image pixbuf, or `NULL`. No extra reference is added to the returned pixbuf, so if you want to keep it

around, you must use g_object_ref(). The returned image must not be modified.

[nullable][transfer none]

Since: 2.4

---

gtk_icon_info_load_icon ()

```
GdkPixbuf *
gtk_icon_info_load_icon (GtkIconInfo *icon_info,
                         GError **error);
```

Renders an icon previously looked up in an icon theme using gtk_icon_theme_lookup_icon(); the size will be based on the size passed to gtk_icon_theme_lookup_icon(). Note that the resulting pixbuf may not be exactly this size; an icon theme may have icons that differ slightly from their nominal sizes, and in addition GTK+ will avoid scaling icons that it considers sufficiently close to the requested size or for which the source image would have to be scaled up too far. (This maintains sharpness.). This behaviour can be changed by passing the GTK_ICON_LOOKUP_FORCE_SIZE flag when obtaining the GtkIconInfo. If this flag has been specified, the pixbuf returned by this function will be scaled to the exact size.

Parameters

icon_info	a GtkIconInfo from gtk_icon_theme_lookup_icon()	
error	location to store error information on failure, or NULL.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use g_object_unref() to release your reference to the icon.

[transfer full]

Since: 2.4

---

gtk_icon_info_load_surface ()

```
cairo_surface_t *
gtk_icon_info_load_surface (GtkIconInfo *icon_info,
                            GdkWindow *for_window,
                            GError **error);
```

Renders an icon previously looked up in an icon theme using gtk_icon_theme_lookup_icon(); the size will be based on the size passed to gtk_icon_theme_lookup_icon(). Note that the resulting surface may not be exactly this size; an icon theme may have icons that differ slightly from their nominal sizes, and in addition GTK+ will avoid scaling icons that it considers sufficiently close to the requested size or for which the source image would have to be scaled up too far. (This maintains sharpness.). This behaviour can be changed by passing the GTK_ICON_LOOKUP_FORCE_SIZE flag when obtaining the GtkIconInfo. If this flag has been specified, the pixbuf returned by this function will be scaled to the exact size.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
for_window	GdkWindow to optimize drawing for, or `NULL`.	[allow-none]
error	location for error information on failure, or `NULL`.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use `cairo_surface_destroy()` to release your reference to the icon.

[transfer full]

Since: 3.10

---

gtk_icon_info_load_icon_async ()

```
void
gtk_icon_info_load_icon_async (GtkIconInfo *icon_info,
                               GCancellable *cancellable,
                               GAsyncReadyCallback callback,
                               gpointer user_data);
```

Asynchronously load, render and scale an icon previously looked up from the icon theme using `gtk_icon_theme_lookup_icon()`.

For more details, see `gtk_icon_info_load_icon()` which is the synchronous version of this call.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
cancellable	optional GCancellable object, `NULL` to ignore.	[allow-none]
callback	a GAsyncReadyCallback to call when the request is satisfied.	[scope async]
user_data	the data to pass to callback function.	[closure]

Since: 3.8

---

gtk_icon_info_load_icon_finish ()

```
GdkPixbuf *
gtk_icon_info_load_icon_finish (GtkIconInfo *icon_info,
                                GAsyncResult *res,
                                GError **error);
```

Finishes an async icon load, see `gtk_icon_info_load_icon_async()`.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
res	a GAsyncResult	
error	location to store error information on failure, or `NULL`.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use `g_object_unref()` to release your reference to the icon.

[transfer full]

Since: 3.8

---

gtk_icon_info_load_symbolic ()

```
GdkPixbuf *
gtk_icon_info_load_symbolic (GtkIconInfo *icon_info,
                             const GdkRGBA *fg,
                             const GdkRGBA *success_color,
                             const GdkRGBA *warning_color,
                             const GdkRGBA *error_color,
                             gboolean *was_symbolic,
                             GError **error);
```

Loads an icon, modifying it to match the system colours for the foreground, success, warning and error colors provided. If the icon is not a symbolic one, the function will return the result from `gtk_icon_info_load_icon()`.

This allows loading symbolic icons that will match the system theme.

Unless you are implementing a widget, you will want to use `g_themed_icon_new_with_default_fallbacks()` to load the icon.

As implementation details, the icon loaded needs to be of SVG type, contain the “symbolic” term as the last component of the icon name, and use the “fg”, “success”, “warning” and “error” CSS styles in the SVG file itself.

See the Symbolic Icons Specification for more information about symbolic icons.

Parameters

icon_info	a GtkIconInfo	
fg	a GdkRGBA representing the foreground color of the icon	
success_color	a GdkRGBA representing the warning color of the icon or `NULL` to use the default color.	[allow-none]
warning_color	a GdkRGBA representing the warning color of the icon or `NULL` to use the default color.	[allow-none]
error_color	a GdkRGBA representing the error color of the icon or `NULL` to use the default color (allow-none).	[allow-none]

was_symbolic	a gboolean, returns whether the loaded icon was a symbolic one and whether the `fg` color was applied to it.	[out][allow-none]
error	location to store error information on failure, or `NULL`.	[allow-none]

Returns

a GdkPixbuf representing the loaded icon.

[transfer full]

Since: 3.0

---

gtk_icon_info_load_symbolic_async ()

```
void
gtk_icon_info_load_symbolic_async (GtkIconInfo *icon_info,
                                   const GdkRGBA *fg,
                                   const GdkRGBA *success_color,
                                   const GdkRGBA *warning_color,
                                   const GdkRGBA *error_color,
                                   GCancellable *cancellable,
                                   GAsyncReadyCallback callback,
                                   gpointer user_data);
```

Asynchronously load, render and scale a symbolic icon previously looked up from the icon theme using `gtk_icon_theme_lookup_icon()`.

For more details, see `gtk_icon_info_load_symbolic()` which is the synchronous version of this call.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
fg	a GdkRGBA representing the foreground color of the icon	
success_color	a GdkRGBA representing the warning color of the icon or `NULL` to use the default color.	[allow-none]
warning_color	a GdkRGBA representing the warning color of the icon or `NULL` to use the default color.	[allow-none]
error_color	a GdkRGBA representing the error color of the icon or `NULL` to use the default color (allow-none).	[allow-none]
cancellable	optional GCancellable object, `NULL` to ignore.	[allow-none]
callback	a GAsyncReadyCallback to call when the request is satisfied.	[scope async]
user_data	the data to pass to callback function.	[closure]

Since: 3.8

---

gtk_icon_info_load_symbolic_finish ()

```
GdkPixbuf *
gtk_icon_info_load_symbolic_finish (GtkIconInfo *icon_info,
```

```
                                    GAsyncResult *res,
                                    gboolean *was_symbolic,
                                    GError **error);
```

Finishes an async icon load, see gtk_icon_info_load_symbolic_async().

Parameters

icon_info	a GtkIconInfo from gtk_icon_theme_lookup_icon()	
res	a GAsyncResult	
was_symbolic	a gboolean, returns whether the loaded icon was a symbolic one and whether the fg color was applied to it.	[out][allow-none]
error	location to store error information on failure, or NULL.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use g_object_unref() to release your reference to the icon.

[transfer full]

Since: 3.8

---

gtk_icon_info_load_symbolic_for_style ()

```
GdkPixbuf *
gtk_icon_info_load_symbolic_for_style (GtkIconInfo *icon_info,
                                       GtkStyle *style,
                                       GtkStateType state,
                                       gboolean *was_symbolic,
                                       GError **error);
```

gtk_icon_info_load_symbolic_for_style has been deprecated since version 3.0 and should not be used in newly-written code.

Use gtk_icon_info_load_symbolic_for_context() instead

Loads an icon, modifying it to match the system colours for the foreground, success, warning and error colors provided. If the icon is not a symbolic one, the function will return the result from gtk_icon_info_load_icon().

This allows loading symbolic icons that will match the system theme.

See gtk_icon_info_load_symbolic() for more details.

Parameters

icon_info	a GtkIconInfo	
style	a GtkStyle to take the colors from	
state	the widget state to use for colors	
was_symbolic	a gboolean, returns whether the loaded icon was a symbolic one and whether	[out][allow-none]

	the `fg` color was applied to it.	
error	location to store error information on failure, or `NULL`.	[allow-none]

Returns

a GdkPixbuf representing the loaded icon.

[transfer full]

Since: 3.0

---

gtk_icon_info_load_symbolic_for_context ()

```
GdkPixbuf *
gtk_icon_info_load_symbolic_for_context
                                (GtkIconInfo *icon_info,
                                 GtkStyleContext *context,
                                 gboolean *was_symbolic,
                                 GError **error);
```

Loads an icon, modifying it to match the system colors for the foreground, success, warning and error colors provided. If the icon is not a symbolic one, the function will return the result from `gtk_icon_info_load_icon()`. This function uses the regular foreground color and the symbolic colors with the names “success_color”, “warning_color” and “error_color” from the context.

This allows loading symbolic icons that will match the system theme.

See `gtk_icon_info_load_symbolic()` for more details.

Parameters

icon_info	a GtkIconInfo	
context	a GtkStyleContext	
was_symbolic	a gboolean, returns whether the loaded icon was a symbolic one and whether the `fg` color was applied to it.	[out][allow-none]
error	location to store error information on failure, or `NULL`.	[allow-none]

Returns

a GdkPixbuf representing the loaded icon.

[transfer full]

Since: 3.0

---

gtk_icon_info_load_symbolic_for_context_async ()

```
void
```

```
gtk_icon_info_load_symbolic_for_context_async
                                (GtkIconInfo *icon_info,
                                 GtkStyleContext *context,
                                 GCancellable *cancellable,
                                 GAsyncReadyCallback callback,
                                 gpointer user_data);
```

Asynchronously load, render and scale a symbolic icon previously looked up from the icon theme using `gtk_icon_theme_lookup_icon()`.

For more details, see `gtk_icon_info_load_symbolic_for_context()` which is the synchronous version of this call.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
context	a GtkStyleContext	
cancellable	optional GCancellable object, `NULL` to ignore.	[allow-none]
callback	a GAsyncReadyCallback to call when the request is satisfied.	[scope async]
user_data	the data to pass to callback function.	[closure]

Since: 3.8

---

gtk_icon_info_load_symbolic_for_context_finish ()

```
GdkPixbuf *
gtk_icon_info_load_symbolic_for_context_finish
                                (GtkIconInfo *icon_info,
                                 GAsyncResult *res,
                                 gboolean *was_symbolic,
                                 GError **error);
```

Finishes an async icon load, see `gtk_icon_info_load_symbolic_for_context_async()`.

Parameters

icon_info	a GtkIconInfo from `gtk_icon_theme_lookup_icon()`	
res	a GAsyncResult	
was_symbolic	a gboolean, returns whether the loaded icon was a symbolic one and whether the `fg` color was applied to it.	[out][allow-none]
error	location to store error information on failure, or `NULL`.	[allow-none]

Returns

the rendered icon; this may be a newly created icon or a new reference to an internal icon, so you must not modify the icon. Use `g_object_unref()` to release your reference to the icon.

[transfer full]

Since: 3.8

---

gtk_icon_info_set_raw_coordinates ()

```
void
gtk_icon_info_set_raw_coordinates (GtkIconInfo *icon_info,
                                   gboolean raw_coordinates);
```

`gtk_icon_info_set_raw_coordinates` has been deprecated since version 3.14 and should not be used in newly-written code.

Embedded rectangles and attachment points are deprecated

Sets whether the coordinates returned by `gtk_icon_info_get_embedded_rect()` and `gtk_icon_info_get_attach_points()` should be returned in their original form as specified in the icon theme, instead of scaled appropriately for the pixbuf returned by `gtk_icon_info_load_icon()`.

Raw coordinates are somewhat strange; they are specified to be with respect to the unscaled pixmap for PNG and XPM icons, but for SVG icons, they are in a 1000x1000 coordinate space that is scaled to the final size of the icon. You can determine if the icon is an SVG icon by using `gtk_icon_info_get_filename()`, and seeing if it is non-`NULL` and ends in ".svg".

This function is provided primarily to allow compatibility wrappers for older API's, and is not expected to be useful for applications.

Parameters

icon_info	a GtkIconInfo
raw_coordinates	whether the coordinates of embedded rectangles and attached points should be returned in their original (unscaled) form.

Since: 2.4

---

gtk_icon_info_get_embedded_rect ()

```
gboolean
gtk_icon_info_get_embedded_rect (GtkIconInfo *icon_info,
                                 GdkRectangle *rectangle);
```

`gtk_icon_info_get_embedded_rect` has been deprecated since version 3.14 and should not be used in newly-written code.

Embedded rectangles are deprecated

This function is deprecated and always returns `FALSE`.

Parameters

icon_info	a GtkIconInfo	
rectangle	GdkRectangle in which to store embedded rectangle coordinates; coordinates are only stored when this function returns `TRUE`.	[out]

Returns

FALSE

Since: 2.4

---

gtk_icon_info_get_attach_points ()

```
gboolean
gtk_icon_info_get_attach_points (GtkIconInfo *icon_info,
                                 GdkPoint **points,
                                 gint *n_points);
```

`gtk_icon_info_get_attach_points` has been deprecated since version 3.14 and should not be used in newly-written code.

Attachment points are deprecated

This function is deprecated and always returns FALSE.

Parameters

icon_info	a GtkIconInfo	
points	location to store pointer to an array of points, or NULL free the array of points with g_free().	[allow-none][array length=n_points] [out]
n_points	location to store the number of points in points , or NULL.	[allow-none]

Returns

FALSE

Since: 2.4

---

gtk_icon_info_get_display_name ()

```
const gchar *
gtk_icon_info_get_display_name (GtkIconInfo *icon_info);
```

`gtk_icon_info_get_display_name` has been deprecated since version 3.14 and should not be used in newly-written code.

Display names are deprecated

This function is deprecated and always returns NULL.

Parameters

icon_info	a GtkIconInfo

Returns

`NULL`

Since: 2.4

---

gtk_icon_info_is_symbolic ()

```
gboolean
gtk_icon_info_is_symbolic (GtkIconInfo *icon_info);
```

Checks if the icon is symbolic or not. This currently uses only the file name and not the file contents for determining this. This behaviour may change in the future.

Parameters

icon_info	a GtkIconInfo

Returns

`TRUE` if the icon is symbolic, `FALSE` otherwise

Since: 3.12

Types and Values

GtkIconInfo

```
typedef struct _GtkIconInfo GtkIconInfo;
```

Contains information found when looking up an icon in an icon theme.

---

struct GtkIconTheme

```
struct GtkIconTheme;
```

Acts as a database of information about an icon theme. Normally, you retrieve the icon theme for a particular screen using `gtk_icon_theme_get_for_screen()` and it will contain information about current icon theme for that screen, but you can also create a new GtkIconTheme object and set the icon theme name explicitly using `gtk_icon_theme_set_custom_theme()`.

---

struct GtkIconThemeClass

```
struct GtkIconThemeClass {
  GObjectClass parent_class;
```

```
  void (* changed)  (GtkIconTheme *icon_theme);
};
```

Members

changed ()	Signal emitted when the current icon theme is switched or GTK+ detects that a change has occurred in the contents of the current icon theme.

---

enum GtkIconLookupFlags

Used to specify options for `gtk_icon_theme_lookup_icon()`

Members

GTK_ICON_LOOKUP_NO_SVG	Never get SVG icons, even if gdk-pixbuf supports them. Cannot be used together with `GTK_ICON_LOOKUP_FORCE_SVG`.
GTK_ICON_LOOKUP_FORCE_SVG	Get SVG icons, even if gdk-pixbuf doesn't support them. Cannot be used together with `GTK_ICON_LOOKUP_NO_SVG`.
GTK_ICON_LOOKUP_USE_BUILTIN	When passed to `gtk_icon_theme_lookup_icon()` includes builtin icons as well as files. For a builtin icon, `gtk_icon_info_get_filename()` is `NULL` and you need to call `gtk_icon_info_get_builtin_pixbuf()`.
GTK_ICON_LOOKUP_GENERIC_FALLBACK	Try to shorten icon name at '-' characters before looking at inherited themes. This flag is only supported in functions that take a single icon name. For more general fallback, see `gtk_icon_theme_choose_icon()`. Since 2.12.
GTK_ICON_LOOKUP_FORCE_SIZE	Always get the icon scaled to the requested size. Since 2.14.
GTK_ICON_LOOKUP_FORCE_REGULAR	Try to always load regular icons, even when symbolic icon names are given. Since 3.14.
GTK_ICON_LOOKUP_FORCE_SYMBOLIC	Try to always load symbolic icons, even when regular icon names are given. Since 3.14.
GTK_ICON_LOOKUP_DIR_LTR	Try to load a variant of the icon for left-to-right text direction. Since 3.14.
GTK_ICON_LOOKUP_DIR_RTL	Try to load a variant of the icon for right-to-left text direction. Since 3.14.

GTK_ICON_THEME_ERROR

```
#define GTK_ICON_THEME_ERROR gtk_icon_theme_error_quark ()
```

The GQuark used for GtkIconThemeError errors.

---

enum GtkIconThemeError

Error codes for GtkIconTheme operations.

Members

GTK_ICON_THEME_NOT_FOUND	The icon specified does not exist in the theme
GTK_ICON_THEME_FAILED	An unspecified error occurred.

Signal Details

The "changed" signal

```
void
user_function (GtkIconTheme *icon_theme,
               gpointer      user_data)
```

Emitted when the current icon theme is switched or GTK+ detects that a change has occurred in the contents of the current icon theme.

Parameters

icon_theme	the icon theme
user_data	user data set when the signal handler was connected.

Flags: Run Last
