# Short Contents

1. Overview ......................................................... 1  
2. ClutterAction .................................................... 2  
3. ClutterActorMeta ................................................. 3  
4. ClutterActor ...................................................... 5  
5. ClutterAlignConstraint .......................................... 63  
6. ClutterAlpha ..................................................... 65  
7. ClutterAnimatable ............................................... 68  
8. Implicit Animations ............................................... 69  
9. ClutterAnimator .................................................. 76  
10. ClutterBackend .................................................. 82  
11. ClutterBinLayout ............................................... 84  
12. ClutterBindConstraint .......................................... 87  
13. Key Bindings ................................................... 90  
14. ClutterBlurEffect ............................................... 94  
15. ClutterBoxLayout ............................................... 95  
16. ClutterBrightnessContrastEffect ............................. 100  
17. ClutterCairoTexture ............................................ 101  
18. ClutterCanvas .................................................. 103  
19. ClutterChildMeta .............................................. 104  
20. ClutterClickAction ............................................. 105  
21. ClutterClone ................................................... 108  
22. Colors ......................................................... 109  
23. ClutterColorizeEffect .......................................... 114  
24. ClutterConstraint ............................................... 115  
25. ClutterContainer ............................................... 117  
26. ClutterContent ................................................. 119  
27. ClutterDeformEffect ........................................... 120  
28. ClutterDesaturateEffect ....................................... 122  
29. ClutterDeviceManager ......................................... 123  
30. ClutterDragAction ............................................. 125  
31. ClutterDropAction ............................................. 127  
32. ClutterEffect .................................................. 128  
33. Events ........................................................ 132  
34. Features ....................................................... 141  
35. ClutterFixedLayout ............................................ 142
Undocumented ........................................ 265
Type Index .............................................. 270
Function Index .......................................... 271
1 Overview

(gnome clutter) wraps the Clutter graphical canvas toolkit for Guile. It is a part of Guile-GNOME.

See the documentation for (gobject) for more information on Guile-GNOME.
2 ClutterAction

Abstract class for event-related logic

2.1 Overview

<clutter-action> is an abstract base class for event-related actions that modify the user interaction of a <clutter-actor>, just like <clutter-constraint> is an abstract class for modifiers of an actor’s position or size.

Implementations of <clutter-action> are associated to an actor and can provide behavioral changes when dealing with user input - for instance drag and drop capabilities, or scrolling, or panning - by using the various event-related signals provided by <clutter-actor> itself.

<clutter-action> is available since Clutter 1.4

2.2 Usage
Chapter 3: ClutterActorMeta

3 ClutterActorMeta

Base class of actor modifiers

3.1 Overview

<clutter-actor-meta> is an abstract class providing a common API for modifiers of <clutter-actor> behaviour, appearance or layout.

A <clutter-actor-meta> can only be owned by a single <clutter-actor> at any time.

Every sub-class of <clutter-actor-meta> should check if the <"enabled"> property is set to '#t' before applying any kind of modification.

<clutter-actor-meta> is available since Clutter 1.4

3.2 Usage

clutter-actor-meta-set-name (self <clutter-actor-meta>) [Function]

(set-name mchars)

set-name [Method]

Sets the name of meta

The name can be used to identify the <clutter-actor-meta> instance

meta a <clutter-actor-meta>

name the name of meta

Since 1.4

clutter-actor-meta-get-name (self <clutter-actor-meta>) [Function]

⇒ (ret mchars)

get-name [Method]

Retrieves the name set using clutter-actor-meta-set-name

meta a <clutter-actor-meta>

ret the name of the <clutter-actor-meta> instance, or '#f' if none was set.

The returned string is owned by the <clutter-actor-meta> instance and it should not be modified or freed.

Since 1.4

clutter-actor-meta-set-enabled (self <clutter-actor-meta>) [Function]

(is_enabled bool)

set-enabled [Method]

Sets whether meta should be enabled or not

meta a <clutter-actor-meta>

is-enabled whether meta is enabled

Since 1.4
clutter-actor-meta-get-enabled (self <clutter-actor-meta>)  [Function]
⇒ (ret bool)
get-enabled
    Retrieves whether meta is enabled
    meta a <clutter-actor-meta>
    ret ‘#t’ if the <clutter-actor-meta> instance is enabled
    Since 1.4

clutter-actor-meta-get-actor (self <clutter-actor-meta>)  [Function]
⇒ (ret <clutter-actor>)
get-actor
    Retrieves a pointer to the <clutter-actor> that owns meta
    meta a <clutter-actor-meta>
    ret a pointer to a <clutter-actor> or ‘#f’.
    Since 1.4
Chapter 4: ClutterActor

The basic element of the scene graph

4.1 Overview

The ClutterActor class is the basic element of the scene graph in Clutter, and it encapsulates the position, size, and transformations of a node in the graph.

4.2 Actor transformations

Each actor can be transformed using methods like `clutter-actor-set-scale` or `clutter-actor-set-rotation`. The order in which the transformations are applied is decided by Clutter and it is the following:

1. translation by the origin of the "allocation";
2. translation by the actor's "depth";
3. scaling by the "scale-x" and "scale-y" factors;
4. rotation around the "rotation-z-angle" and "rotation-z-center";
5. rotation around the "rotation-y-angle" and "rotation-y-center";
6. rotation around the "rotation-x-angle" and "rotation-x-center";
7. negative translation by the "anchor-x" and "anchor-y" point.

4.3 Modifying an actor’s geometry

Each actor has a bounding box, called "allocation" which is either set by its parent or explicitly through the `clutter-actor-set-position` and `clutter-actor-set-size` methods. Each actor also has an implicit preferred size.

An actors preferred size can be defined by any subclass by overriding the `clutter-actor-class.get-preferred-width` and the `clutter-actor-class.get-preferred-height` virtual functions, or it can be explicitly set by using `clutter-actor-set-width` and `clutter-actor-set-height`.

An actors position can be set explicitly by using `clutter-actor-set-x` and `clutter-actor-set-y`; the coordinates are relative to the origin of the actors parent.
4.4 Managing actor children

Each actor can have multiple children, by calling `clutter-actor-add-child` to add a new child actor, and `clutter-actor-remove-child` to remove an existing child. `<clutter-actor>` will hold a reference on each child actor, which will be released when the child is removed from its parent, or destroyed using `clutter-actor-destroy`.

```c
ClutterActor *actor = clutter_actor_new ();

/* set the bounding box of the actor */
clutter_actor_set_position (actor, 0, 0);
clutter_actor_set_size (actor, 480, 640);

/* set the background color of the actor */
clutter_actor_set_background_color (actor, CLUTTER_COLOR_Orange);

/* set the bounding box of the child, relative to the parent */
ClutterActor *child = clutter_actor_new ();
clutter_actor_set_position (child, 20, 20);
clutter_actor_set_size (child, 80, 240);

/* set the background color of the child */
clutter_actor_set_background_color (child, CLUTTER_COLOR_Blue);

/* add the child to the actor */
clutter_actor_add_child (actor, child);
```

Children can be inserted at a given index, or above and below another child actor. The order of insertion determines the order of the children when iterating over them. Iterating over children is performed by using `clutter-actor-get-first-child`, `clutter-actor-get-previous-sibling`, `clutter-actor-get-next-sibling`, and `clutter-actor-get-last-child`. It is also possible to retrieve a list of children by using `clutter-actor-get-children`, as well as retrieving a specific child at a given index by using `clutter-actor-get-child-at-index`.

If you need to track additions of children to a `<clutter-actor>`, use the `<"actor-added">` signal; similarly, to track removals of children from a ClutterActor, use the `<"actor-removed">` signal.

4.5 Painting an actor

There are three ways to paint an actor:

- set a delegate `<clutter-content>` as the value for the `<"content">` property of the actor;
subclass `<clutter-actor>` and override the `clutter-actor-class.paint-node` virtual function;

subclass `<clutter-actor>` and override the `clutter-actor-class.paint` virtual function.

A `<clutter-content>` is a delegate object that takes over the painting operation of one, or more actors. The `<clutter-content>` painting will be performed on top of the `<"background-color">` of the actor, and before calling the `clutter-actor-class.paint-node` virtual function.

```c
ClutterActor *actor = clutter_actor_new();

// set the bounding box
clutter_actor_set_position(actor, 50, 50);
clutter_actor_set_size(actor, 100, 100);

// set the content; the image_content variable is set elsewhere
clutter_actor_set_content(actor, image_content);
```

The `clutter-actor-class.paint-node` virtual function is invoked whenever an actor needs to be painted. The implementation of the virtual function must only paint the contents of the actor itself, and not the contents of its children, if the actor has any.

The `<clutter-paint-node>` passed to the virtual function is the local root of the render tree; any node added to it will be rendered at the correct position, as defined by the actor's `<"allocation">`.

```c
static void
my_actor_paint_node (ClutterActor *actor,
                     ClutterPaintNode *root)
{
    ClutterPaintNode *node;
    ClutterActorBox box;

    // where the content of the actor should be painted
    clutter_actor_get_allocation_box (actor, &box);

    // the cogl_texture variable is set elsewhere
    node = clutter_texture_node_new (cogl_texture, CLUTTER_COLOR_White,
                                     CLUTTER_SCALING_FILTER_TRILINEAR,
                                     CLUTTER_SCALING_FILTER_LINEAR);

    // paint the content of the node using the allocation
    clutter_paint_node_add_rectangle (node, &box);

    // add the node, and transfer ownership
    clutter_paint_node_add_child (root, node);
```
The `clutter-actor-class.paint` virtual function is invoked when the `<"paint">` signal is emitted, and after the other signal handlers have been invoked. Overriding the paint virtual function gives total control to the paint sequence of the actor itself, including the children of the actor, if any.

It is strongly discouraged to override the `clutter-actor-class.paint` virtual function, as well as connecting to the `<"paint">` signal. These hooks into the paint sequence are considered legacy, and will be removed when the Clutter API changes.

### 4.6 Handling events on an actor

A `<clutter-actor>` can receive and handle input device events, for instance pointer events and key events, as long as its `<"reactive">` property is set to `'#t'`.

Once an actor has been determined to be the source of an event, Clutter will traverse the scene graph from the top-level actor towards the event source, emitting the `<"captured-event">` signal on each ancestor until it reaches the source; this phase is also called the *capture phase*. If the event propagation was not stopped, the graph is walked backwards, from the source actor to the top-level, and the `<"event">` signal, along with other event signals if needed, is emitted; this phase is also called the *bubble phase*. At any point of the signal emission, signal handlers can stop the propagation through the scene graph by returning `'CLUTTER_EVENT_STOP'`; otherwise, they can continue the propagation by returning `'CLUTTER_EVENT_PROPAGATE'`.

### 4.7 Animation

Animation is a core concept of modern user interfaces; Clutter provides a complete and powerful animation framework that automatically tweens the actor’s state without requiring direct, frame by frame manipulation from your application code.

The implicit animation model of Clutter assumes that all the changes in an actor state should be gradual and asynchronous; Clutter will automatically transition an actor’s property change between the current state and the desired one without manual intervention.

By default, in the 1.0 API series, the transition happens with a duration of zero milliseconds, and the implicit animation is an opt in feature to retain backwards compatibility. In order to enable implicit animations, it is necessary to change the easing state of an actor by using `clutter-actor-save-easing-state`:

```c
/* assume that the actor is currently positioned at (100, 100) */
clutter_actor_save_easing_state (actor);
clutter_actor_set_position (actor, 500, 500);
clutter_actor_restore_easing_state (actor);
```

The example above will trigger an implicit animation of the actor between its current position to a new position.
It is possible to animate multiple properties of an actor at the same time, and you can animate multiple actors at the same time as well, for instance:

```c
/* animate the actor's opacity and depth */
clutter_actor_save_easing_state (actor);
clutter_actor_set_opacity (actor, 0);
clutter_actor_set_depth (actor, -100);
clutter_actor_restore_easing_state (actor);

/* animate another actor's opacity */
clutter_actor_save_easing_state (another_actor);
clutter_actor_set_opacity (another_actor, 255);
clutter_actor_set_depth (another_actor, 100);
clutter_actor_restore_easing_state (another_actor);
```

Implicit animations use a default duration of 250 milliseconds, and a default easing mode of 'CLUTTER_EASE_OUT_CUBIC', unless you call `clutter-actor-set-easing-mode` and `clutter-actor-set-easing-duration` after changing the easing state of the actor.

It is important to note that if you modify the state on an animatable property while a transition is in flight, the transition's final value will be updated, as well as its duration and progress mode by using the current easing state; for instance, in the following example:

```c
clutter_actor_save_easing_state (actor);
clutter_actor_set_x (actor, 200);
clutter_actor_restore_easing_state (actor);

clutter_actor_save_easing_state (actor);
clutter_actor_set_x (actor, 100);
clutter_actor_restore_easing_state (actor);
```

the first call to `clutter-actor-set-x` will begin a transition of the "x" property to the value of 200; the second call to `clutter-actor-set-x` will change the transition's final value to 100.

It is possible to retrieve the `<clutter-transition>` used by the animatable properties by using `clutter-actor-get-transition` and using the property name as the transition name.

The explicit animation model supported by Clutter requires that you create a `<clutter-transition>` object, and set the initial and final values. The transition will not start unless you add it to the `<clutter-actor>`.

```c
ClutterTransition *transition;

transition = clutter_property_transition_new ("opacity");
clutter_timeline_set_duration (CLUTTER_TIMELINE (transition), 3000);
clutter_timeline_set_repeat_count (CLUTTER_TIMELINE (transition), 2);
```
clutter_timeline_set_auto_reverse (CLUTTER_TIMELINE (transition), TRUE);
clutter_transition_set_interval (transition, clutter_interval_new (G_TYPE_UINT, 255, 0));
clutter_actor_add_transition (actor, "animate-opacity", transition);

The example above will animate the "opacity" property of an actor between fully opaque and fully transparent, and back, over a span of 3 seconds. The animation does not begin until it is added to the actor.

The explicit animation API should also be used when using custom animatable properties for <clutter-action>, <clutter-constraint>, and <clutter-effect> instances associated to an actor; see the section on custom animatable properties below for an example.

Finally, explicit animations are useful for creating animations that run continuously, for instance:

```c
/* this animation will pulse the actor's opacity continuously */
ClutterTransition *transition;
ClutterInterval *interval;

transition = clutter_property_transition_new ("opacity");

/* we want to animate the opacity between 0 and 255 */
internal = clutter_interval_new (G_TYPE_UINT, 0, 255);
clutter_transition_set_interval (transition, interval);

/* over a one second duration, running an infinite amount of times */
clutter_timeline_set_duration (CLUTTER_TIMELINE (transition), 1000);
clutter_timeline_set_repeat_count (CLUTTER_TIMELINE (transition), -1);

/* we want to fade in and out, so we need to auto-reverse the transition */
clutter_timeline_set_auto_reverse (CLUTTER_TIMELINE (transition), TRUE);

/* and we want to use an easing function that eases both in and out */
clutter_timeline_set_progress_mode (CLUTTER_TIMELINE (transition),
                                  CLUTTER_EASE_IN_OUT_CUBIC);

/* add the transition to the desired actor; this will start the animation. */
clutter_actor_add_transition (actor, "opacityAnimation", transition);
```
4.8 Implementing an actor

Careful consideration should be given when deciding to implement a `<clutter-actor>` subclass. It is generally recommended to implement a sub-class of `<clutter-actor>` only for actors that should be used as leaf nodes of a scene graph.

If your actor should be painted in a custom way, you should override the `<"paint">` signal class handler. You can either opt to chain up to the parent class implementation or decide to fully override the default paint implementation; Clutter will set up the transformations and clip regions prior to emitting the `<"paint">` signal.

By overriding the `clutter-actor-class.get-preferred-width` and `clutter-actor-class.get-preferred-height` virtual functions it is possible to change or provide the preferred size of an actor; similarly, by overriding the `clutter-actor-class.allocate` virtual function it is possible to control the layout of the children of an actor. Make sure to always chain up to the parent implementation of the `clutter-actor-class.allocate` virtual function.

In general, it is strongly encouraged to use delegation and composition instead of direct subclassing.

4.9 ClutterActor custom properties for `<clutter-script>`

`<clutter-actor>` defines a custom "rotation" property which allows a short-hand description of the rotations to be applied to an actor.

The syntax of the "rotation" property is the following:

```
"rotation" : [
    { "<axis>" : [ <angle>, [ <center> ] ] }
]
```

where the `axis` is the name of an enumeration value of type `<clutter-rotate-axis>` and `angle` is a floating point value representing the rotation angle on the given axis, in degrees.

The `center` array is optional, and if present it must contain the center of rotation as described by two coordinates: Y and Z for "x-axis"; X and Z for "y-axis"; and X and Y for "z-axis".

`<clutter-actor>` will also parse every positional and dimensional property defined as a string through `clutter-units-from-string`; you should read the documentation for the `clutter-units` parser format for the valid units and syntax.

4.10 Custom animatable properties

`<clutter-actor>` allows accessing properties of `<clutter-action>`, `<clutter-effect>`, and `<clutter-constraint>` instances associated to an actor instance for animation purposes.

In order to access a specific `<clutter-action>` or a `<clutter-constraint>` property it is necessary to set the `<"name">` property on the given action or constraint.

The property can be accessed using the following syntax:
The initial @ is mandatory.

The section fragment can be one between "actions", "constraints" and "effects".

The meta-name fragment is the name of the action or constraint, as specified by the <"name"> property.

The property-name fragment is the name of the action or constraint property to be animated.

The example below animates a <clutter-bind-constraint> applied to an actor using clutter-actor-animate. The rect has a binding constraint for the origin actor, and in its initial state is overlapping the actor to which is bound to.

```c
constraint = clutter_bind_constraint_new (origin, CLUTTER_BIND_X, 0.0);
clutter_actor_meta_set_name (CLUTTER_ACTOR_META (constraint), "bind-x");
clutter_actor_add_constraint (rect, constraint);

constraint = clutter_bind_constraint_new (origin, CLUTTER_BIND_Y, 0.0);
clutter_actor_meta_set_name (CLUTTER_ACTOR_META (constraint), "bind-y");
clutter_actor_add_constraint (rect, constraint);

clutter_actor_set_reactive (origin, TRUE);

g_signal_connect (origin, "button-press-event",
    G_CALLBACK (on_button_press),
    rect);
```

On button press, the rectangle "slides" from behind the actor to which is bound to, using the <"offset"> property to achieve the effect:

```c
gboolean
on_button_press (ClutterActor *origin,
    ClutterEvent *event,
    ClutterActor *rect)
{
    ClutterTransition *transition;
    ClutterInterval *interval;

    /* the offset that we want to apply; this will make the actor slide in from behind the origin and rest at the right of the origin, plus a padding value.
    */
    float new_offset = clutter_actor_get_width (origin) + h_padding;

    /* the property we wish to animate; the @constraints" section
```
const char *prop = "@constraints.bind-x.offset";

/* create a new transition for the given property */
transition = clutter_property_transition_new (prop);

/* set the easing mode and duration */
clutter_timeline_set_progress_mode (CLUTTER_TIMELINE (transition), CLUTTER_EASE_OUT_CUBIC);
clutter_timeline_set_duration (CLUTTER_TIMELINE (transition), 500);

/* create the interval with the initial and final values */
interval = clutter_interval_new (G_TYPE_FLOAT, 0, new_offset);
clutter_transition_set_interval (transition, interval);

/* add the transition to the actor; this causes the animation to start. the name "offsetAnimation" can be used to retrieve the transition later. */
clutter_actor_add_transition (rect, "offsetAnimation", transition);

/* we handled the event */
return CLUTTER_EVENT_STOP;

4.11 Usage

clutter-actor-new ⇒ (ret <clutter-actor>) [Function]
Creates a new <clutter-actor>.
A newly created actor has a floating reference, which will be sunk when it is added to another actor.
ret the newly created <clutter-actor>.

Since 1.10

clutter-actor-set-flags (self <clutter-actor>) [Function]
(set-flags <clutter-actor-flags>)
Sets flags on self
This function will emit notifications for the changed properties
self a <clutter-actor>
flags the flags to set

Since 1.0

clutter-actor-unset-flags (self <clutter-actor>) [Function]
(\(flags\) <clutter-actor-flags>)

unset-flags [Method]
Unsets flags on self
This function will emit notifications for the changed properties

self a <clutter-actor>
flags the flags to unset

Since 1.0

clutter-actor-get-flags (self <clutter-actor>) [Function]
⇒ (ret <clutter-actor-flags>)

get-flags [Method]
Retrieves the flags set on self

self a <clutter-actor>
ret a bitwise or of <clutter-actor-flags> or 0

Since 1.0

clutter-actor-set-name (self <clutter-actor>) (name mchars) [Function]
set-name [Method]
Sets the given name to self. The name can be used to identify a <clutter-actor>.

self A <clutter-actor>
name Textual tag to apply to actor

clutter-actor-get-name (self <clutter-actor>) ⇒ (ret mchars) [Function]
get-name [Method]
Retrieves the name of self.

self A <clutter-actor>
ret the name of the actor, or ‘#f’. The returned string is owned by the actor
and should not be modified or freed.

clutter-actor-show (self <clutter-actor>) [Function]
show [Method]
Flags an actor to be displayed. An actor that isn’t shown will not be rendered on the
stage.
Actors are visible by default.
If this function is called on an actor without a parent, the <"show-on-set-parent”>
will be set to ‘#t’ as a side effect.

self A <clutter-actor>
Chapter 4: ClutterActor

**clutter-actor-hide** (*self <clutter-actor>*)

hide

Flags an actor to be hidden. A hidden actor will not be rendered on the stage.

Actors are visible by default.

If this function is called on an actor without a parent, the "show-on-set-parent" property will be set to '#f' as a side-effect.

* self A <clutter-actor>

**clutter-actor-realize** (*self <clutter-actor>*)

realize

Realization informs the actor that it is attached to a stage. It can use this to allocate resources if it wanted to delay allocation until it would be rendered. However it is perfectly acceptable for an actor to create resources before being realized because Clutter only ever has a single rendering context so that actor is free to be moved from one stage to another.

This function does nothing if the actor is already realized.

Because a realized actor must have realized parent actors, calling **clutter-actor-realize** will also realize all parents of the actor.

This function does not realize child actors, except in the special case that realizing the stage, when the stage is visible, will suddenly map (and thus realize) the children of the stage.

* self A <clutter-actor>

**clutter-actor-unrealize** (*self <clutter-actor>*)

unrealize

Unrealization informs the actor that it may be being destroyed or moved to another stage. The actor may want to destroy any underlying graphics resources at this point. However it is perfectly acceptable for it to retain the resources until the actor is destroyed because Clutter only ever uses a single rendering context and all of the graphics resources are valid on any stage.

Because mapped actors must be realized, actors may not be unrealized if they are mapped. This function hides the actor to be sure it isn’t mapped, an application-visible side effect that you may not be expecting.

This function should not be called by application code.

* self A <clutter-actor>

**clutter-actor-paint** (*self <clutter-actor>*)

paint

Renders the actor to display.

This function should not be called directly by applications. Call **clutter-actor-queue-redraw** to queue paints, instead.

This function is context-aware, and will either cause a regular paint or a pick paint.

This function will emit the "paint" signal or the "pick" signal, depending on the context.
This function does not paint the actor if the actor is set to 0, unless it is performing a pick paint.

```python
self A <clutter-actor>
```

### clutter-actor-continue-paint (self <clutter-actor>)

[Method]

Run the next stage of the paint sequence. This function should only be called within the implementation of the run virtual of a `<clutter-effect>`. It will cause the run method of the next effect to be applied, or it will paint the actual actor if the current effect is the last effect in the chain.

```python
self A <clutter-actor>
```

Since 1.8

### clutter-actor-queue-redraw (self <clutter-actor>)

[Method]

Queues up a redraw of an actor and any children. The redraw occurs once the main loop becomes idle (after the current batch of events has been processed, roughly).

Applications rarely need to call this, as redraws are handled automatically by modification functions.

This function will not do anything if `self` is not visible, or if the actor is inside an invisible part of the scenegraph.

Also be aware that painting is a NOP for actors with an opacity of 0.

When you are implementing a custom actor you must queue a redraw whenever some private state changes that will affect painting or picking of your actor.

```python
self A <clutter-actor>
```

### clutter-actor-queue-relayout (self <clutter-actor>)

[Method]

Indicates that the actor’s size request or other layout-affecting properties may have changed. This function is used inside `<clutter-actor>` subclass implementations, not by applications directly.

Queueing a new layout automatically queues a redraw as well.

```python
self A <clutter-actor>
```

Since 0.8

### clutter-actor-destroy (self <clutter-actor>)

[Method]

Destroys an actor. When an actor is destroyed, it will break any references it holds to other objects. If the actor is inside a container, the actor will be removed.

When you destroy a container, its children will be destroyed as well.

Note: you cannot destroy the `<clutter-stage>` returned by clutter-stage-get-default.

```python
self A <clutter-actor>
```
clutter-actor-event (self <clutter-actor>) [Function]
  (event <clutter-event>) (capture bool) ⇒ (ret bool)

This function is used to emit an event on the main stage. You should rarely need to use this function, except for synthetising events.

actor  a <clutter-actor>
event  a <clutter-event>
capture TRUE if event in in capture phase, FALSE otherwise.
ret  the return value from the signal emission: ‘#t’ if the actor handled the event, or ‘#f’ if the event was not handled

Since 0.6

clutter-actor-should-pick-paint (self <clutter-actor>) [Function]
  ⇒ (ret bool)

should-pick-paint [Method]
Should be called inside the implementation of the <"pick"> virtual function in order to check whether the actor should paint itself in pick mode or not.

This function should never be called directly by applications.

self  A <clutter-actor>
ret  ‘#t’ if the actor should paint its silhouette, ‘#f’ otherwise

clutter-actor-map (self <clutter-actor>) [Function]
map [Method]
Sets the ‘CLUTTER_ACTOR_MAPPED’ flag on the actor and possibly maps and realizes its children if they are visible. Does nothing if the actor is not visible.

Calling this function is strongly disencouraged: the default implementation of clutter-actor-class.map will map all the children of an actor when mapping its parent.

When overriding map, it is mandatory to chain up to the parent implementation.

self  A <clutter-actor>

Since 1.0

clutter-actor-unmap (self <clutter-actor>) [Function]
unmap [Method]
Unsets the ‘CLUTTER_ACTOR_MAPPED’ flag on the actor and possibly unmaps its children if they were mapped.

Calling this function is not encouraged: the default <clutter-actor> implementation of clutter-actor-class.unmap will also unmap any eventual children by default when their parent is unmapped.

When overriding clutter-actor-class.unmap, it is mandatory to chain up to the parent implementation.
It is important to note that the implementation of the `clutter-actor-class.unmap` virtual function may be called after the `clutter-actor-class.destroy` or the `g-object-class.dispose` implementation, but it is guaranteed to be called before the `g-object-class.finalize` implementation.

```python
self A <clutter-actor>

Since 1.0

clutter-actor-has-overlaps (self <clutter-actor>) ⇒ (ret bool) [Function]
has-overlaps [Method]
Asks the actor’s implementation whether it may contain overlapping primitives.
For example; Clutter may use this to determine whether the painting should be redirected to an offscreen buffer to correctly implement the opacity property.
Custom actors can override the default response by implementing the `<clutter-actor>has-overlaps` virtual function. See `clutter-actor-set-offscreen-redirect` for more information.
self A <clutter-actor>
ret ‘#t’ if the actor may have overlapping primitives, and ‘#f’ otherwise

Since 1.8

clutter-actor-allocate (self <clutter-actor>) [Function]
allocate [Method]
Called by the parent of an actor to assign the actor its size. Should never be called by applications (except when implementing a container or layout manager).
Actors can know from their allocation box whether they have moved with respect to their parent actor. The `flags` parameter describes additional information about the allocation, for instance whether the parent has moved with respect to the stage, for example because a grandparent’s origin has moved.
self A <clutter-actor>
box new allocation of the actor, in parent-relative coordinates
flags flags that control the allocation

Since 0.8

clutter-actor-allocate-align-fill (self <clutter-actor>) [Function]
allocation-align-fill [Method]
Allocates `self` by taking into consideration the available allocation area; an alignment factor on either axis; and whether the actor should fill the allocation on either axis.
The `box` should contain the available allocation width and height; if the `x1` and `y1` members of `<clutter-actor-box>` are not set to 0, the allocation will be offset by their value.
```
This function takes into consideration the geometry request specified by the "request-mode" property, and the text direction.
This function is useful for fluid layout managers, like <clutter-bin-layout> or <clutter-table-layout>

- self a <clutter-actor>
- box a <clutter-actor-box>, containing the available width and height
- x-align the horizontal alignment, between 0 and 1
- y-align the vertical alignment, between 0 and 1
- x-fill whether the actor should fill horizontally
- y-fill whether the actor should fill vertically
- flags allocation flags to be passed to clutter-actor-allocate

Since 1.4

```
clutter-actor-set-allocation (self <clutter-actor>)
  (box <clutter-actor-box>) (flags <clutter-allocation-flags>)

set-allocation
```

Stores the allocation of self as defined by box.

This function can only be called from within the implementation of the clutter-actor-class.allocate virtual function.

The allocation should have been adjusted to take into account constraints, alignment, and margin properties. If you are implementing a <clutter-actor> subclass that provides its own layout management policy for its children instead of using a <clutter-layout-manager> delegate, you should not call this function on the children of self; instead, you should call clutter-actor-allocate, which will adjust the allocation box for you.

This function should only be used by subclasses of <clutter-actor> that wish to store their allocation but cannot chain up to the parent’s implementation; the default implementation of the clutter-actor-class.allocate virtual function will call this function.

It is important to note that, while chaining up was the recommended behaviour for <clutter-actor> subclasses prior to the introduction of this function, it is recommended to call clutter-actor-set-allocation instead.

If the <clutter-actor> is using a <clutter-layout-manager> delegate object to handle the allocation of its children, this function will call the clutter-layout-manager-allocate function only if the ‘CLUTTER_DELEGATE_LAYOUT’ flag is set on flags, otherwise it is expected that the subclass will call clutter-layout-manager-allocate by itself. For instance, the following code:

```
static void
my_actor_allocate (ClutterActor *actor,
    const ClutterActorBox *allocation,
    ClutterAllocationFlags flags)
```
{  
    ClutterActorBox new_alloc;
    ClutterAllocationFlags new_flags;

    adjust_allocation (allocation, &new_alloc);

    new_flags = flags | CLUTTER_DELEGATE_LAYOUT;

    /* this will use the layout manager set on the actor */
    clutter_actor_set_allocation (actor, &new_alloc, new_flags);
}

is equivalent to this:

static void
my_actor_allocate (ClutterActor *actor,
                  const ClutterActorBox *allocation,
                  ClutterAllocationFlags flags)
{
    ClutterLayoutManager *layout;
    ClutterActorBox new_alloc;

    adjust_allocation (allocation, &new_alloc);

    clutter_actor_set_allocation (actor, &new_alloc, flags);

    layout = clutter_actor_get_layout_manager (actor);
    clutter_layout_manager_allocate (layout,
                                      CLUTTER_CONTAINER (actor),
                                      &new_alloc,
                                      flags);
}

self    a <clutter-actor>
box     a <clutter-actor-box>
flags   allocation flags

Since 1.10

clutter-actor-get-allocation-box (self <clutter-actor>)  [Function]
 (box <clutter-actor-box>)

get-allocation-box  [Method]

Gets the layout box an actor has been assigned. The allocation can only be assumed valid inside a paint method; anywhere else, it may be out-of-date.

An allocation does not incorporate the actor’s scale or anchor point; those transformations do not affect layout, only rendering.
Do not call any of the clutter_actor_get_allocation_*() family of functions inside the implementation of the get-preferred-width or get-preferred-height virtual functions.

self A <clutter-actor>

box the function fills this in with the actor’s allocation.

Since 0.8

clutter-actor-get-preferred-size (self <clutter-actor>) [Function]
⇒ (min_width_p float) (min_height_p float) (natural_width_p float)
(natural_height_p float)

get-preferred-size [Method]
Computes the preferred minimum and natural size of an actor, taking into account the actor’s geometry management (either height-for-width or width-for-height).

The width and height used to compute the preferred height and preferred width are the actor’s natural ones.

If you need to control the height for the preferred width, or the width for the preferred height, you should use clutter-actor-get-preferred-width and clutter-actor-get-preferred-height, and check the actor’s preferred geometry management using the "request-mode" property.

self a <clutter-actor>

min-width-p return location for the minimum width, or ‘#f’.

min-height-p return location for the minimum height, or ‘#f’.

natural-width-p return location for the natural width, or ‘#f’.

natural-height-p return location for the natural height, or ‘#f’.

Since 0.8

clutter-actor-get-preferred-width (self <clutter-actor>) [Function]
(for_height float) ⇒ (min_width_p float) (natural_width_p float)

get-preferred-width [Method]
Computes the requested minimum and natural widths for an actor, optionally depending on the specified height, or if they are already computed, returns the cached values.

An actor may not get its request - depending on the layout manager that’s in effect. A request should not incorporate the actor’s scale or anchor point; those transformations do not affect layout, only rendering.

self A <clutter-actor>

for-height available height when computing the preferred width, or a negative value to indicate that no height is defined
min-width-p
return location for minimum width, or ‘#f’.

natural-width-p
return location for the natural width, or ‘#f’.

Since 0.8

clutter-actor-get-preferred-height (self <clutter-actor>) [Function]
(for_width float) ⇒ (min_height_p float) (natural_height_p float)

get-preferred-height [Method]
Computes the requested minimum and natural heights for an actor, or if they are already computed, returns the cached values.

An actor may not get its request - depending on the layout manager that’s in effect.

A request should not incorporate the actor’s scale or anchor point; those transformations do not affect layout, only rendering.

self A <clutter-actor>
for-width available width to assume in computing desired height, or a negative value to indicate that no width is defined

min-height-p
return location for minimum height, or ‘#f’.

natural-height-p
return location for natural height, or ‘#f’.

Since 0.8

clutter-actor-set-request-mode (self <clutter-actor>) [Function]
(mode <clutter-request-mode>)

set-request-mode [Method]
Sets the geometry request mode of self.

The mode determines the order for invoking clutter-actor-get-preferred-width and clutter-actor-get-preferred-height

self a <clutter-actor>
mode the request mode

Since 1.2

clutter-actor-get-request-mode (self <clutter-actor>) [Function]
⇒ (ret <clutter-request-mode>)

get-request-mode [Method]
Retrieves the geometry request mode of self

self a <clutter-actor>
ret the request mode for the actor

Since 1.2
Chapter 4: ClutterActor

clutter-actor-has-allocation (self <clutter-actor>) ⇒ (ret bool)

has-allocation

Checks if the actor has an up-to-date allocation assigned to it. This means that
the actor should have an allocation: it’s visible and has a parent. It also means that
there is no outstanding relayout request in progress for the actor or its children (There
might be other outstanding layout requests in progress that will cause the actor to
get a new allocation when the stage is laid out, however).

If this function returns ‘#f’, then the actor will normally be allocated before it is next
drawn on the screen.

self a <clutter-actor>
ret ‘#t’ if the actor has an up-to-date allocation

Since 1.4

clutter-actor-set-x-align (self <clutter-actor>)
(x.align <clutter-actor-align>)

set-x-align

Sets the horizontal alignment policy of a <clutter-actor>, in case the actor received
extra horizontal space.

See also the "x-align" property.

self a <clutter-actor>
x-align the horizontal alignment policy

Since 1.10

clutter-actor-get-x-align (self <clutter-actor>) ⇒ (ret <clutter-actor-align>)

get-x-align

Retrieves the horizontal alignment policy set using clutter-actor-set-x-align.

self a <clutter-actor>
ret the horizontal alignment policy.

Since 1.10

clutter-actor-set-y-align (self <clutter-actor>)
(y.align <clutter-actor-align>)

set-y-align

Sets the vertical alignment policy of a <clutter-actor>, in case the actor received
extra vertical space.

See also the "y-align" property.

self a <clutter-actor>
y-align the vertical alignment policy

Since 1.10
**Chapter 4: ClutterActor**

**clutter-actor-get-y-align (self <clutter-actor>)**  
⇒ (ret <clutter-actor-align>)  

get-y-align  
Retrieves the vertical alignment policy set using `clutter-actor-set-y-align`.  

- **self** a `<clutter-actor>`  
- **ret** the vertical alignment policy.  

Since 1.10

**clutter-margin-new ⇒ (ret <clutter-margin>)**  

Creates a new `<clutter-margin>`.  

- **ret** a newly allocated `<clutter-margin>`. Use `clutter-margin-free` to free the resources associated with it when done.  

Since 1.10

**clutter-actor-set-margin (self <clutter-actor>)**  
(margin `<clutter-margin>`)  

set-margin  
Sets all the components of the margin of a `<clutter-actor>`.  

- **self** a `<clutter-actor>`  
- **margin** a `<clutter-margin>`  

Since 1.10

**clutter-actor-get-margin (self <clutter-actor>)**  
(margin `<clutter-margin>`)  

get-margin  
Retrieves all the components of the margin of a `<clutter-actor>`.  

- **self** a `<clutter-actor>`  
- **margin** return location for a `<clutter-margin>`.  

Since 1.10

**clutter-actor-set-margin-top (self <clutter-actor>)**  
(margin float)  

set-margin-top  
Sets the margin from the top of a `<clutter-actor>`.  

- **self** a `<clutter-actor>`  
- **margin** the top margin  

Since 1.10

**clutter-actor-get-margin-top (self <clutter-actor>)**  
⇒ (ret float)  

get-margin-top  
Retrieves the top margin of a `<clutter-actor>`.  

- **self** a `<clutter-actor>`  
- **ret** float
Chapter 4: ClutterActor

```python
self  a <clutter-actor>
ret   the top margin

Since 1.10

clutter-actor-set-margin-right (self <clutter-actor>)  [Function]
    (margin float)
set-margin-right  [Method]
    Sets the margin from the right of a <clutter-actor>.
    
self    a <clutter-actor>
margin  the right margin

Since 1.10

clutter-actor-get-margin-right (self <clutter-actor>)  [Function]
    ⇒ (ret float)
get-margin-right  [Method]
    Retrieves the right margin of a <clutter-actor>.
    
self    a <clutter-actor>
ret     the right margin

Since 1.10

clutter-actor-set-margin-bottom (self <clutter-actor>)  [Function]
    (margin float)
set-margin-bottom  [Method]
    Sets the margin from the bottom of a <clutter-actor>.
    
self    a <clutter-actor>
margin  the bottom margin

Since 1.10

clutter-actor-get-margin-bottom (self <clutter-actor>)  [Function]
    ⇒ (ret float)
get-margin-bottom  [Method]
    Retrieves the bottom margin of a <clutter-actor>.
    
self    a <clutter-actor>
ret     the bottom margin

Since 1.10

clutter-actor-set-margin-left (self <clutter-actor>)  [Function]
    (margin float)
set-margin-left  [Method]
    Sets the margin from the left of a <clutter-actor>.
    
self    a <clutter-actor>
margin  the left margin

Since 1.10
```
**clutter-actor-get-margin-left** *(self <clutter-actor>)*  
⇒ (ret float)

get-margin-left  
Retrieves the left margin of a `<clutter-actor>`.

self a `<clutter-actor>`  
ret the left margin

Since 1.10

**clutter-actor-set-layout-manager** *(self <clutter-actor>)*

(manager <clutter-layout-manager>)

set-layout-manager  
Sets the `<clutter-layout-manager>` delegate object that will be used to lay out the children of *self*.

The `<clutter-actor>` will take a reference on the passed *manager* which will be released either when the layout manager is removed, or when the actor is destroyed.

self a `<clutter-actor>`  
manager a `<clutter-layout-manager>`, or ‘#f’ to unset it.

Since 1.10

**clutter-actor-get-layout-manager** *(self <clutter-actor>)*

⇒ (ret <clutter-layout-manager>)

get-layout-manager  
Retrieves the `<clutter-layout-manager>` used by *self*.

self a `<clutter-actor>`  
ret a pointer to the `<clutter-layout-manager>`, or ‘#f’.

Since 1.10

**clutter-actor-set-background-color** *(self <clutter-actor>)*

(color <clutter-color>)

set-background-color  
Sets the background color of a `<clutter-actor>`.

The background color will be used to cover the whole allocation of the actor. The default background color of an actor is transparent.

To check whether an actor has a background color, you can use the `<background-color-set>` actor property.

The `<background-color>` property is animatable.

self a `<clutter-actor>`  
color a `<clutter-color>`, or ‘#f’ to unset a previously set color.

Since 1.10
clutter-actor-get-background-color (self <clutter-actor>) [Function]
(color <clutter-color>)

get-background-color [Method]
Retrieves the color set using clutter-actor-set-background-color.

  self    a <clutter-actor>
  color   return location for a <clutter-color>.

Since 1.10

clutter-actor-set-size (self <clutter-actor>) (width float) [Function]
(height float)

set-size [Method]
Sets the actor's size request in pixels. This overrides any "normal" size request the actor would have. For example a text actor might normally request the size of the text; this function would force a specific size instead.

If width and/or height are -1 the actor will use its "normal" size request instead of overriding it, i.e. you can "unset" the size with -1.

This function sets or unsets both the minimum and natural size.

  self    A <clutter-actor>
  width   New width of actor in pixels, or -1
  height  New height of actor in pixels, or -1

clutter-actor-get-size (self <clutter-actor>) ⇒ (width float) [Function]
(heigh float)

get-size [Method]
This function tries to "do what you mean" and return the size an actor will have. If the actor has a valid allocation, the allocation will be returned; otherwise, the actors natural size request will be returned.

If you care whether you get the request vs. the allocation, you should probably call a different function like clutter-actor-get-allocation-box or clutter-actor-get-preferred-width.

  self    A <clutter-actor>
  width   return location for the width, or '#f'.
  height  return location for the height, or '#f'.

Since 0.2

clutter-actor-set-position (self <clutter-actor>) (x float) [Function]
(y float)

set-position [Method]
Sets the actor's fixed position in pixels relative to any parent actor.

If a layout manager is in use, this position will override the layout manager and force a fixed position.

  self    A <clutter-actor>
Chapter 4: ClutterActor

\( x \)  
New left position of actor in pixels.

\( y \)  
New top position of actor in pixels.

**clutter-actor-get-position**

\[
\text{clutter-actor-get-position (self <clutter-actor>) ⇒ (x float) (y float)}
\]

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get-position</strong></td>
</tr>
</tbody>
</table>

This function tries to "do what you mean" and tell you where the actor is, prior to any transformations. Retrieves the fixed position of an actor in pixels, if one has been set; otherwise, if the allocation is valid, returns the actor’s allocated position; otherwise, returns 0,0.

The returned position is in pixels.

\[
\text{self a <clutter-actor>}
\]

\[
x \text{ return location for the X coordinate, or ‘#f’.}
\]

\[
y \text{ return location for the Y coordinate, or ‘#f’.}
\]

Since 0.6

**clutter-actor-set-width**

\[
\text{clutter-actor-set-width (self <clutter-actor>) (width float)}
\]

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>set-width</strong></td>
</tr>
</tbody>
</table>

Forces a width on an actor, causing the actor’s preferred width and height (if any) to be ignored.

If \( width \) is -1 the actor will use its preferred width request instead of overriding it, i.e. you can "unset" the width with -1.

This function sets both the minimum and natural size of the actor.

\[
\text{self A <clutter-actor>}
\]

\[
width \text{ Requested new width for the actor, in pixels, or -1}
\]

Since 0.2

**clutter-actor-get-width**

\[
\text{clutter-actor-get-width (self <clutter-actor>) ⇒ (ret float)}
\]

<table>
<thead>
<tr>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>get-width</strong></td>
</tr>
</tbody>
</table>

Retrieves the width of a <clutter-actor>.

If the actor has a valid allocation, this function will return the width of the allocated area given to the actor.

If the actor does not have a valid allocation, this function will return the actor’s natural width, that is the preferred width of the actor.

If you care whether you get the preferred width or the width that has been assigned to the actor, you should probably call a different function like clutter-actor-get-allocation-box to retrieve the allocated size or clutter-actor-get-preferred-width to retrieve the preferred width.

If an actor has a fixed width, for instance a width that has been assigned using clutter-actor-set-width, the width returned will be the same value.

\[
\text{self A <clutter-actor>}
\]

\[
ret \text{ the width of the actor, in pixels}
\]
clutter-actor-set-height \( (self \text{ <clutter-actor>}) \) \( (height \text{ float}) \) [Function]

set-height [Method]

Forces a height on an actor, causing the actor’s preferred width and height (if any) to be ignored.

If \( height \) is -1 the actor will use its preferred height instead of overriding it, i.e. you can "unset" the height with -1.

This function sets both the minimum and natural size of the actor.

\( self \quad \text{A <clutter-actor>} \)

\( height \quad \text{Requested new height for the actor, in pixels, or -1} \)

Since 0.2

clutter-actor-get-height \( (self \text{ <clutter-actor>}) \Rightarrow (ret \text{ float}) \) [Function]

get-height [Method]

Retrieves the height of a <clutter-actor>.

If the actor has a valid allocation, this function will return the height of the allocated area given to the actor.

If the actor does not have a valid allocation, this function will return the actor’s natural height, that is the preferred height of the actor.

If you care whether you get the preferred height or the height that has been assigned to the actor, you should probably call a different function like clutter-actor-get-allocation-box to retrieve the allocated size or clutter-actor-get-preferred-height to retrieve the preferred height.

If an actor has a fixed height, for instance a height that has been assigned using clutter-actor-set-height, the height returned will be the same value.

\( self \quad \text{A <clutter-actor>} \)

\( ret \quad \text{the height of the actor, in pixels} \)

clutter-actor-set-x \( (self \text{ <clutter-actor>}) \) \( (x \text{ float}) \) [Function]

set-x [Method]

Sets the actor’s X coordinate, relative to its parent, in pixels.

Overrides any layout manager and forces a fixed position for the actor.

The "x" property is animatable.

\( self \quad \text{a <clutter-actor>} \)

\( x \quad \text{the actor’s position on the X axis} \)

Since 0.6

clutter-actor-get-x \( (self \text{ <clutter-actor>}) \Rightarrow (ret \text{ float}) \) [Function]

get-x [Method]

Retrieves the X coordinate of a <clutter-actor>.

This function tries to "do what you mean", by returning the correct value depending on the actor’s state.
If the actor has a valid allocation, this function will return the X coordinate of the origin of the allocation box.

If the actor has any fixed coordinate set using `clutter-actor-set-x`, `clutter-actor-set-position` or `clutter-actor-set-geometry`, this function will return that coordinate.

If both the allocation and a fixed position are missing, this function will return 0.

```python
self  A <clutter-actor>
ret   the X coordinate, in pixels, ignoring any transformation (i.e. scaling, rotation)
```

`clutter-actor-set-y (self <clutter-actor>) (y float)`  
[Function]

`set-y`  
Sets the actor’s Y coordinate, relative to its parent, in pixels.  
Overrides any layout manager and forces a fixed position for the actor.  
The "y" property is animatable.

```python
self  a <clutter-actor>
y     the actor’s position on the Y axis
```

Since 0.6

`clutter-actor-get-y (self <clutter-actor>) ⇒ (ret float)`  
[Function]

`get-y`  
Retrieves the Y coordinate of a `<clutter-actor>`.  
This function tries to "do what you mean", by returning the correct value depending on the actor’s state.

If the actor has a valid allocation, this function will return the Y coordinate of the origin of the allocation box.

If the actor has any fixed coordinate set using `clutter-actor-set-y`, `clutter-actor-set-position` or `clutter-actor-set-geometry`, this function will return that coordinate.

If both the allocation and a fixed position are missing, this function will return 0.

```python
self  A <clutter-actor>
ret   the Y coordinate, in pixels, ignoring any transformation (i.e. scaling, rotation)
```

`clutter-actor-move-by (self <clutter-actor>) (dx float) (dy float)`  
[Function]

`move-by`  
Moves an actor by the specified distance relative to its current position in pixels.  
This function modifies the fixed position of an actor and thus removes it from any layout management.  
Another way to move an actor is with an anchor point, see `clutter-actor-set-anchor-point`.

```python
self  A <clutter-actor>
```
dx Distance to move Actor on X axis.

dy Distance to move Actor on Y axis.

Since 0.2

clutter-actor-set-depth (self <clutter-actor>) (depth float) {Function]
set-depth [Method]
Sets the Z coordinate of self to depth.
The unit used by depth is dependant on the perspective setup. See also clutter-stage-set-perspective.

self a <clutter-actor>
depth Z co-ord

clutter-actor-get-depth (self <clutter-actor>) ⇒ (ret float) {Function]
get-depth [Method]
Retrieves the depth of self.

self a <clutter-actor>
ret the depth of the actor

clutter-actor-set-scale (self <clutter-actor>) (scale_x double) (scale_y double) {Function]
set-scale [Method]
Scales an actor with the given factors. The scaling is relative to the scale center and the anchor point. The scale center is unchanged by this function and defaults to 0,0. The "scale-x" and "scale-y" properties are animatable.

self A <clutter-actor>
scale-x double factor to scale actor by horizontally.
scale-y double factor to scale actor by vertically.

Since 0.2

clutter-actor-set-scale-full (self <clutter-actor>) (scale_x double) (scale_y double) (center_x float) (center_y float) {Function]
set-scale-full [Method]
Scales an actor with the given factors around the given center point. The center point is specified in pixels relative to the anchor point (usually the top left corner of the actor).
The "scale-x" and "scale-y" properties are animatable.

self A <clutter-actor>
scale-x double factor to scale actor by horizontally.
scale-y double factor to scale actor by vertically.
center-x X coordinate of the center of the scale.
center-y Y coordinate of the center of the scale

Since 1.0
**clutter-actor-get-scale (self <clutter-actor>)**  
⇒ (scale_x double) (scale_y double)

Get-scale  
Retrieves an actor's scale factors.

* self  
  A <clutter-actor>

* scale-x  
  Location to store horizontal scale factor, or ‘#f’.

* scale-y  
  Location to store vertical scale factor, or ‘#f’.

Since 0.2

**clutter-actor-get-scale-center (self <clutter-actor>)**  
⇒ (center_x float) (center_y float)

Get-scale-center  
Retrieves the scale center coordinate in pixels relative to the top left corner of the actor. If the scale center was specified using a <clutter-gravity> this will calculate the pixel offset using the current size of the actor.

* self  
  A <clutter-actor>

* center-x  
  Location to store the X position of the scale center, or ‘#f’.

* center-y  
  Location to store the Y position of the scale center, or ‘#f’.

Since 1.0

**clutter-actor-get-scale-gravity (self <clutter-actor>)**  
⇒ (ret <clutter-gravity>)

Get-scale-gravity  
Retrieves the scale center as a compass direction. If the scale center was specified in pixels or units this will return ‘CLUTTER_GRAVITY_NONE’.

* self  
  A <clutter-actor>

* ret  
  the scale gravity

Since 1.0

**clutter-actor-is-scaled (self <clutter-actor>) ⇒ (ret bool)**  

Is-scaled  
Checks whether the actor is scaled in either dimension.

* self  
  a <clutter-actor>

* ret  
  ‘#t’ if the actor is scaled.

Since 0.6

**clutter-actor-set-rotation (self <clutter-actor>)**  
(axis <clutter-rotate-axis>) (angle double) (x float) (y float) (z float)

Set-rotation  
Sets the rotation angle of self around the given axis.

The rotation center coordinates used depend on the value of axis:
Chapter 4: ClutterActor

Since 0.8

clutter-actor-get-rotation (self <clutter-actor>) [Function]

(axis <clutter-rotate-axis>) ⇒ (ret double) (x float) (y float)

get-rotation [Method]

Retrieves the angle and center of rotation on the given axis, set using clutter-actor-set-rotation.

Since 0.8

clutter-actor-is-rotated (self <clutter-actor>) ⇒ (ret bool) [Function]

is-rotated [Method]

Checks whether any rotation is applied to the actor.

Since 0.6
ClutterActor

ClutterActor is a library for creating and managing actors, which are graphical elements in a Clutter scene. This chapter focuses on the functions and methods related to handling anchor points and transforming stage points.

### clutter-actor-set-anchor-point

**Signature:**

```plaintext
clutter-actor-set-anchor-point (self <clutter-actor>) (anchor_x float) (anchor_y float)
```

**Description:**

Sets an anchor point for `self`. The anchor point is a point in the coordinate space of an actor to which the actor position within its parent is relative; the default is (0, 0), i.e. the top-left corner of the actor.

- **Parameters:**
  - `self`: A `clutter-actor`
  - `anchor_x`: X coordinate of the anchor point
  - `anchor_y`: Y coordinate of the anchor point

**Since:** 0.6

### clutter-actor-get-anchor-point

**Signature:**

```plaintext
clutter-actor-get-anchor-point (self <clutter-actor>) ⇒ (anchor_x float) (anchor_y float)
```

**Description:**

Gets the current anchor point of the actor in pixels.

- **Parameters:**
  - `self`: A `clutter-actor`
  - `anchor_x`: Return location for the X coordinate of the anchor point.
  - `anchor_y`: Return location for the Y coordinate of the anchor point.

**Since:** 0.6

### clutter-actor-move-anchor-point

**Signature:**

```plaintext
clutter-actor-move-anchor-point (self <clutter-actor>) (anchor_x float) (anchor_y float)
```

**Description:**

Sets an anchor point for the actor, and adjusts the actor position so that the relative position of the actor toward its parent remains the same.

- **Parameters:**
  - `self`: A `clutter-actor`
  - `anchor_x`: X coordinate of the anchor point
  - `anchor_y`: Y coordinate of the anchor point

**Since:** 0.6

### clutter-actor-transform-stage-point

**Signature:**

```plaintext
clutter-actor-transform-stage-point (self <clutter-actor>) (x float) (y float) ⇒ (ret bool) (x_out float) (y_out float)
```

**Description:**

This function translates screen coordinates (x, y) to coordinates relative to the actor. For example, it can be used to translate screen events from global screen coordinates into actor-local coordinates.

The conversion can fail, notably if the transform stack results in the actor being projected on the screen as a mere line.

The conversion should not be expected to be pixel-perfect due to the nature of the operation. In general the error grows when the skewing of the actor rectangle on screen increases.

This function only works when the allocation is up-to-date, i.e. inside of `paint`.

This function is computationally intensive.
**Chapter 4: ClutterActor**

```
self  A <clutter-actor>
x     x screen coordinate of the point to unproject.
y     y screen coordinate of the point to unproject.
x-out return location for the unprojected x coordinate.
y-out return location for the unprojected y coordinate.
ret   ‘#t’ if conversion was successful.
```

Since 0.6

```clutter-actor-get-transformed-size (self <clutter-actor>) ⇒ (width float) (height float)
get-transformed-size
```

[Function]

Gets the absolute size of an actor in pixels, taking into account the scaling factors.

If the actor has a valid allocation, the allocated size will be used. If the actor has not
a valid allocation then the preferred size will be transformed and returned.

If you want the transformed allocation, see `clutter-actor-get-abs-allocation-vertices` instead.

---

When the actor (or one of its ancestors) is rotated around the X or Y axis, it no
longer appears as on the stage as a rectangle, but as a generic quadrangle; in that
case this function returns the size of the smallest rectangle that encapsulates the
entire quad. Please note that in this case no assumptions can be made about the
relative position of this envelope to the absolute position of the actor, as returned
by `clutter-actor-get-transformed-position`; if you need this information, you
need to use `clutter-actor-get-abs-allocation-vertices` to get the coords of the
actual quadrangle.

```
self  A <clutter-actor>
width return location for the width, or ‘#f’.
height return location for the height, or ‘#f’.
```

Since 0.8

```clutter-actor-get-paint-opacity (self <clutter-actor>) ⇒ (ret unsigned-int8)
get-paint-opacity
```

[Method]

Retrieves the absolute opacity of the actor, as it appears on the stage.

This function traverses the hierarchy chain and composites the opacity of the actor
with that of its parents.

This function is intended for subclasses to use in the paint virtual function, to paint
themselves with the correct opacity.

```
self  A <clutter-actor>
ret   The actor opacity value.
```

Since 0.8
clutter-actor-get-paint-visibility (self <clutter-actor>)  
⇒ (ret bool)  
get-paint-visibility  
Retrieves the ‘paint’ visibility of an actor recursively checking for non visible parents. 
This is by definition the same as ‘CLUTTER_ACTOR_IS_MAPPED’.

self  
A <clutter-actor>
ret  
‘#t’ if the actor is visible and will be painted.

Since 0.8.4

clutter-actor-get-paint-box (self <clutter-actor>)  
(box <clutter-actor-box>) ⇒ (ret bool)  
get-paint-box  
Retrieves the paint volume of the passed <clutter-actor>, and transforms it into a 2D bounding box in stage coordinates.
This function is useful to determine the on screen area occupied by the actor. The box is only an approximation and may often be considerably larger due to the optimizations used to calculate the box. The box is never smaller though, so it can reliably be used for culling.

There are times when a 2D paint box can’t be determined, e.g. because the actor isn’t yet parented under a stage or because the actor is unable to determine a paint volume.

self  
a <clutter-actor>
box  
return location for a <clutter-actor-box>.
ret  
‘#t’ if a 2D paint box could be determined, else ‘#f’.

Since 1.6

clutter-actor-set-content (self <clutter-actor>)  
(content <clutter-content>)  
set-content  
Sets the contents of a <clutter-actor>.

self  
a <clutter-actor>
content  
a <clutter-content>, or ‘#f’.

Since 1.10

clutter-actor-get-content (self <clutter-actor>)  
⇒ (ret <clutter-content>)  
get-content  
Retrieves the contents of self.

self  
a <clutter-actor>
ret  
a pointer to the <clutter-content> instance, or ‘#f’ if none was set.

Since 1.10
Chapter 4: ClutterActor

clutter-actor-set-content-gravity (self <clutter-actor>) (gravity <clutter-content-gravity>)

set-content-gravity
Sets the gravity of the <clutter-content> used by self.
See the description of the "content-gravity" property for more information.
The "content-gravity" property is animatable.

self a <clutter-actor>
gravity the <clutter-content-gravity>

Since 1.10

clutter-actor-get-content-gravity (self <clutter-actor>) ⇒ (ret <clutter-content-gravity>)

get-content-gravity
Retrieves the content gravity as set using clutter-actor-get-content-gravity.

self a <clutter-actor>
ret the content gravity

Since 1.10

clutter-actor-get-content-box (self <clutter-actor>) (box <clutter-actor-box>)

get-content-box
Retrieves the bounding box for the <clutter-content> of self.
The bounding box is relative to the actor’s allocation.
If no <clutter-content> is set for self, or if self has not been allocated yet, then the result is undefined.
The content box is guaranteed to be, at most, as big as the allocation of the <clutter-actor>.
If the <clutter-content> used by the actor has a preferred size, then it is possible to modify the content box by using the "content-gravity" property.

self a <clutter-actor>
box the return location for the bounding box for the <clutter-content>.

Since 1.10

clutter-actor-set-clip (self <clutter-actor>) (xoff float) (yoff float) (width float) (height float)

set-clip
Sets clip area for self. The clip area is always computed from the upper left corner of the actor, even if the anchor point is set otherwise.

self A <clutter-actor>
xoff X offset of the clip rectangle
yoff Y offset of the clip rectangle
width  Width of the clip rectangle
height Height of the clip rectangle

Since 0.6

clutter-actor-remove-clip (self <clutter-actor>)  [Function]
remove-clip  [Method]
Removes clip area from self.

self  A <clutter-actor>

clutter-actor-has-clip (self <clutter-actor>) ⇒ (ret bool)  [Function]
has-clip  [Method]
Determines whether the actor has a clip area set or not.

self  a <clutter-actor>
ret  ‘#t’ if the actor has a clip area set.

Since 0.1.1

clutter-actor-get-clip (self <clutter-actor>) ⇒ (xoff float)
(yoff float) (width float) (height float)  [Function]
get-clip  [Method]
Gets the clip area for self, if any is set

self  a <clutter-actor>
xoff  return location for the X offset of the clip rectangle, or ‘#f’.
yoff  return location for the Y offset of the clip rectangle, or ‘#f’.
width  return location for the width of the clip rectangle, or ‘#f’.
height  return location for the height of the clip rectangle, or ‘#f’.

Since 0.6

clutter-actor-set-opacity (self <clutter-actor>)  [Function]
(set-opacity unsigned-int8)  [Method]
Sets the actor's opacity, with zero being completely transparent and 255 (0xff) being fully opaque.
The "opacity" property is animatable.

self  A <clutter-actor>
opacity  New opacity value for the actor.

clutter-actor-get-opacity (self <clutter-actor>)  [Function]
⇒ (ret unsigned-int8)  [Method]
get-opacity
Retrieves the opacity value of an actor, as set by clutter-actor-set-opacity.
For retrieving the absolute opacity of the actor inside a paint virtual function, see clutter-actor-get-paint-opacity.
Chapter 4: ClutterActor

self a <clutter-actor>
ret the opacity of the actor

clutter-actor-is-in-clone-paint (self <clutter-actor>) [Function]
⇒ (ret bool)

is-in-clone-paint [Method]
Checks whether self is being currently painted by a <clutter-clone>
This function is useful only inside the ::paint virtual function implementations or
within handlers for the "paint" signal
This function should not be used by applications
self a <clutter-actor>
ret 't if the <clutter-actor> is currently being painted by a <clutter-clone>, and 'f otherwise

Since 1.0

clutter-actor-add-child (self <clutter-actor>) [Function]
(child <clutter-actor>)
add-child [Method]
Adds child to the children of self.
This function will acquire a reference on child that will only be released when calling
clutter-actor-remove-child.
This function will take into consideration the "depth" of child, and will keep the
list of children sorted.
This function will emit the "actor-added" signal on self.
self a <clutter-actor>
child a <clutter-actor>

Since 1.10

clutter-actor-insert-child-above (self <clutter-actor>) [Function]
(child <clutter-actor>) (sibling <clutter-actor>)
insert-child-above [Method]
Inserts child into the list of children of self, above another child of self or, if sibling
is 'f', above all the children of self.
This function will acquire a reference on child that will only be released when calling
clutter-actor-remove-child.
This function will not take into consideration the "depth" of child.
This function will emit the "actor-added" signal on self.
self a <clutter-actor>
child a <clutter-actor>
sibling a child of self, or 'f'.

Since 1.10
clutter-actor-insert-child-at-index (self <clutter-actor>) (child <clutter-actor>) (index int)

insert-child-at-index

Inserts child into the list of children of self, using the given index. If index is greater than the number of children in self, or is less than 0, then the new child is added at the end.

This function will acquire a reference on child that will only be released when calling clutter-actor-remove-child.

This function will not take into consideration the "depth" of child.

This function will emit the "actor-added" signal on self.

```
self  a <clutter-actor>
child a <clutter-actor>
index the index
```

Since 1.10

clutter-actor-insert-child-below (self <clutter-actor>) (child <clutter-actor>) (sibling <clutter-actor>)

insert-child-below

Inserts child into the list of children of self, below another child of self or, if sibling is ‘#f’, below all the children of self.

This function will acquire a reference on child that will only be released when calling clutter-actor-remove-child.

This function will not take into consideration the "depth" of child.

This function will emit the "actor-added" signal on self.

```
self  a <clutter-actor>
child a <clutter-actor>
sibling a child of self, or ‘#f’.
```

Since 1.10

clutter-actor-replace-child (self <clutter-actor>) (old_child <clutter-actor>) (new_child <clutter-actor>)

replace-child

Replaces old-child with new-child in the list of children of self.

```
sel  a <clutter-actor>
old-child the child of self to replace
new-child the <clutter-actor> to replace old-child
```

Since 1.10

clutter-actor-remove-child (self <clutter-actor>) (child <clutter-actor>)

remove-child

Removes child from the children of self.
This function will release the reference added by `clutter-actor-add-child`, so if you want to keep using `child` you will have to acquire a referenced on it before calling this function.

This function will emit the `<"actor-removed">` signal on `self`.

```plaintext
self a <clutter-actor>
child a <clutter-actor>
```

Since 1.10

**clutter-actor-remove-all-children (self <clutter-actor>)**  
[Function]

**remove-all-children**  
[Method]

Removes all children of `self`.

This function releases the reference added by inserting a child actor in the list of children of `self`.

If the reference count of a child drops to zero, the child will be destroyed. If you want to ensure the destruction of all the children of `self`, use `clutter-actor-destroy-all-children`.

```plaintext
self a <clutter-actor>
```

Since 1.10

**clutter-actor-destroy-all-children (self <clutter-actor>)**  
[Function]

**destroy-all-children**  
[Method]

Destroys all children of `self`.

This function releases the reference added by inserting a child actor in the list of children of `self`, and ensures that the `<"destroy">` signal is emitted on each child of the actor.

By default, `<clutter-actor>` will emit the `<"destroy">` signal when its reference count drops to 0; the default handler of the `<"destroy">` signal will destroy all the children of an actor. This function ensures that all children are destroyed, instead of just removed from `self`, unlike `clutter-actor-remove-all-children` which will merely release the reference and remove each child.

Unless you acquired an additional reference on each child of `self` prior to calling `clutter-actor-remove-all-children` and want to reuse the actors, you should use `clutter-actor-destroy-all-children` in order to make sure that children are destroyed and signal handlers are disconnected even in cases where circular references prevent this from automatically happening through reference counting alone.

```plaintext
self a <clutter-actor>
```

Since 1.10

**clutter-actor-get-first-child (self <clutter-actor>)**  
[Function]

⇒ (ret <clutter-actor>)

**get-first-child**  
[Method]

Retrieves the first child of `self`.

The returned pointer is only valid until the scene graph changes; it is not safe to modify the list of children of `self` while iterating it.
Chapter 4: ClutterActor

self
  a <clutter-actor>

ret
  a pointer to a <clutter-actor>, or ‘#f’.

Since 1.10

clutter-actor-get-next-sibling (self <clutter-actor>) ⇒ (ret <clutter-actor>)

get-next-sibling
Retrieves the sibling of self that comes after it in the list of children of self’s parent.
The returned pointer is only valid until the scene graph changes; it is not safe to modify the list of children of self while iterating it.

self
  a <clutter-actor>

ret
  a pointer to a <clutter-actor>, or ‘#f’.

Since 1.10

clutter-actor-get-previous-sibling (self <clutter-actor>) ⇒ (ret <clutter-actor>)

get-previous-sibling
Retrieves the sibling of self that comes before it in the list of children of self’s parent.
The returned pointer is only valid until the scene graph changes; it is not safe to modify the list of children of self while iterating it.

self
  a <clutter-actor>

ret
  a pointer to a <clutter-actor>, or ‘#f’.

Since 1.10

clutter-actor-get-last-child (self <clutter-actor>) ⇒ (ret <clutter-actor>)

get-last-child
Retrieves the last child of self.
The returned pointer is only valid until the scene graph changes; it is not safe to modify the list of children of self while iterating it.

self
  a <clutter-actor>

ret
  a pointer to a <clutter-actor>, or ‘#f’.

Since 1.10

clutter-actor-get-child-at-index (self <clutter-actor>) (index_int) ⇒ (ret <clutter-actor>)

get-child-at-index
Retrieves the actor at the given index inside the list of children of self.

self
  a <clutter-actor>

index
  the position in the list of children

ret
  a pointer to a <clutter-actor>, or ‘#f’.

Since 1.10
clutter-actor-get-children \((self <clutter-actor>)\) ⇒ \((ret \text{g-list-of})\)

get-children

Retrieves the list of children of \(self\).

\(self\) a \(<\text{clutter-actor}\>

\(ret\) A newly allocated \(<\text{g-list}\>\) of \(<\text{clutter-actor}\>\)s. Use \(\text{g-list-free}\) when done.

Since 1.10

clutter-actor-get-n-children \((self <clutter-actor>)\) ⇒ \((ret \text{int})\)

get-n-children

Retrieves the number of children of \(self\).

\(self\) a \(<\text{clutter-actor}\>

\(ret\) the number of children of an actor

Since 1.10

clutter-actor-get-parent \((self <clutter-actor>)\) ⇒ \((ret <\text{clutter-actor}>)\)

get-parent

Retrieves the parent of \(self\).

\(self\) A \(<\text{clutter-actor}\>

\(ret\) The \(<\text{clutter-actor}\>\) parent, or ‘#f’ if no parent is set.

clutter-actor-set-child-at-index \((self <clutter-actor>)\)

\((child <\text{clutter-actor}>)(index, \text{int})\)

set-child-at-index

Changes the index of \(child\) in the list of children of \(self\).

This function is logically equivalent to removing \(child\) and calling clutter-actor-insert-child-at-index, but it will not emit signals or change state on \(child\).

\(self\) a \(<\text{clutter-actor}\>

\(child\) a \(<\text{clutter-actor}\>\) child of \(self\)

\(index\) the new index for \(child\)

Since 1.10

clutter-actor-contains \((self <\text{clutter-actor}>)\)

\((\text{descendant} <\text{clutter-actor}>)⇒ (ret \text{bool})\)

contains

Determines if \(\text{descendant}\) is contained inside \(self\) (either as an immediate child, or as a deeper descendant). If \(self\) and \(\text{descendant}\) point to the same actor then it will also return ‘#t’.

\(self\) A \(<\text{clutter-actor}\>\)
descendant
A `<clutter-actor>` possibly contained in `self`

`ret` whether `descendent` is contained within `self`

Since 1.4

`clutter-actor-get-stage (self <clutter-actor>)` [Function]

`⇒ (ret <clutter-actor>)`

get-stage [Method]
Retrieves the `<clutter-stage>` where `actor` is contained.

`actor` a `<clutter-actor>`

`ret` the stage containing the actor, or `'#f'`.

Since 0.8

`clutter-actor-save-easing-state (self <clutter-actor>)` [Function]

save-easing-state [Method]
Saves the current easing state for animatable properties, and creates a new state with the default values for easing mode and duration.

`self` a `<clutter-actor>`

Since 1.10

`clutter-actor-restore-easing-state (self <clutter-actor>)` [Function]

restore-easing-state [Method]
Restores the easing state as it was prior to a call to `clutter-actor-save-easing-state`.

`self` a `<clutter-actor>`

Since 1.10

`clutter-actor-set-easing-duration (self <clutter-actor> (msecs unsigned-int))` [Function]

set-easing-duration [Method]
Sets the duration of the tweening for animatable properties of `self` for the current easing state.

`self` a `<clutter-actor>`

`msecs` the duration of the easing, or `'#f'`

Since 1.10

`clutter-actor-get-easing-duration (self <clutter-actor>)` [Function]

`⇒ (ret unsigned-int)`

get-easing-duration [Method]
Retrieves the duration of the tweening for animatable properties of `self` for the current easing state.

`self` a `<clutter-actor>`

`ret` the duration of the tweening, in milliseconds

Since 1.10
clutter-actor-set-easing-mode (self <clutter-actor>)(mode <clutter-animation-mode>)

set-easing-mode [Method]
Sets the easing mode for the tweening of animatable properties of self.

self a <clutter-actor>
mode an easing mode, excluding ‘CLUTTER_CUSTOM_MODE’

Since 1.10

clutter-actor-get-easing-mode (self <clutter-actor>)(⇒ (ret <clutter-animation-mode>))

get-easing-mode [Method]
Retrieves the easing mode for the tweening of animatable properties of self for the current easing state.

self a <clutter-actor>
ret an easing mode

Since 1.10

clutter-actor-set-easing-delay (self <clutter-actor>)(msecs unsigned-int)

set-easing-delay [Method]
Sets the delay that should be applied before tweening animatable properties.

self a <clutter-actor>
msecs the delay before the start of the tweening, in milliseconds

Since 1.10

clutter-actor-get-easing-delay (self <clutter-actor>)(⇒ (ret unsigned-int))

get-easing-delay [Method]
Retrieves the delay that should be applied when tweening animatable properties.

self a <clutter-actor>
ret a delay, in milliseconds

Since 1.10

clutter-actor-get-transition (self <clutter-actor>)(name mchars ⇒ (ret <clutter-transition>))

get-transition [Method]
Retrieves the <clutter-transition> of a <clutter-actor> by using the transition name.

Transitions created for animatable properties use the name of the property itself, for instance the code below:
clutter_actor_set.easing_duration (actor, 1000);
clutter_actor_set.rotation (actor, CLUTTER_Y_AXIS, 360.0, x, y, z);

transition = clutter_actor.get.transition (actor, "rotation-angle-y");
g_signal_connect (transition, "completed",
    G_CALLBACK (on_transition_complete),
    actor);

will call the on-transition-complete callback when the transition is complete.

Since 1.10

clutter-actor.add-transition (self <clutter-actor>)
(name mchars) (transition <clutter-transition>)

add-transition

Adds a transition to the <clutter-actor>'s list of animations.
The name string is a per-actor unique identifier of the transition: only one <clutter-transition> can be associated to the specified name.
The transition will be given the easing duration, mode, and delay associated to the actor’s current easing state; it is possible to modify these values after calling clutter-actor-add-transition.
The transition will be started once added.
This function will take a reference on the transition.
This function is usually called implicitly when modifying an animatable property.

Since 1.10

clutter-actor.remove-transition (self <clutter-actor>)
(name mchars)

remove-transition

Removes the transition stored inside a <clutter-actor> using name identifier.
If the transition is currently in progress, it will be stopped.
This function releases the reference acquired when the transition was added to the <clutter-actor>.

Since 1.10
**clutter-actor-set-reactive** (*self* `<clutter-actor>`)  
*set-reactive*  
Sets *actor* as reactive. Reactive actors will receive events.  
*actor* a `<clutter-actor>`  
*reactive* whether the actor should be reactive to events  
Since 0.6

**clutter-actor-get-reactive** (*self* `<clutter-actor>*) ⇒ (*ret* `bool`)  
*get-reactive*  
Checks whether *actor* is marked as reactive.  
*actor* a `<clutter-actor>`  
*ret* ‘#t’ if the actor is reactive  
Since 0.6

**clutter-actor-has-key-focus** (*self* `<clutter-actor>*) ⇒ (*ret* `bool`)  
*has-key-focus*  
Checks whether *self* is the `<clutter-actor>` that has key focus  
*self* a `<clutter-actor>`  
*ret* ‘#t’ if the actor has key focus, and ‘#f’ otherwise  
Since 1.4

**clutter-actor-grab-key-focus** (*self* `<clutter-actor>*)  
*grab-key-focus*  
Sets the key focus of the `<clutter-stage>` including *self* to this `<clutter-actor>`.  
*self* a `<clutter-actor>`  
Since 1.0

**clutter-actor-has-pointer** (*self* `<clutter-actor>*) ⇒ (*ret* `bool`)  
*has-pointer*  
Checks whether an actor contains the pointer of a `<clutter-input-device>`  
*self* a `<clutter-actor>`  
*ret* ‘#t’ if the actor contains the pointer, and ‘#f’ otherwise  
Since 1.2

**clutter-actor-get-pango-context** (*self* `<clutter-actor>*) ⇒ (*ret* `<pango-context>`)  
*get-pango-context*  
Retrieves the `<pango-context>` for *self*. The actor’s `<pango-context>` is already configured using the appropriate font map, resolution and font options.
Unlike `clutter-actor-create-pango-context`, this context is owned by the `<clutter-actor>` and it will be updated each time the options stored by the `<clutter-backend>` change.

You can use the returned `<pango-context>` to create a `<pango-layout>` and render text using `cogl-pango-render-layout` to reuse the glyphs cache also used by Clutter.

```
self a <clutter-actor>
ret the <pango-context> for a <clutter-actor>. The returned <pango-context> is owned by the actor and should not be unreferenced by the application code.
```

Since 1.0

```cpp
clutter-actor-create-pango-context (self <clutter-actor>) ⇒ (ret <pango-context>)
```

`create-pango-context` [Method]

Creates a `<pango-context>` for the given actor. The `<pango-context>` is already configured using the appropriate font map, resolution and font options.

See also `clutter-actor-get-pango-context`.

```
self a <clutter-actor>
ret the newly created <pango-context>. Use g-object-unref on the returned value to deallocate its resources.
```

Since 1.0

```cpp
clutter-actor-create-pango-layout (self <clutter-actor>) (text mchars) ⇒ (ret <pango-layout>)
```

`create-pango-layout` [Method]

Creates a new `<pango-layout>` from the same `<pango-context>` used by the `<clutter-actor>`. The `<pango-layout>` is already configured with the font map, resolution and font options, and the given `text`.

If you want to keep around a `<pango-layout>` created by this function you will have to connect to the "font-changed" and "resolution-changed" signals, and call `pango-layout-context-changed` in response to them.

```
self a <clutter-actor>
text (allow-none) the text to set on the <pango-layout>, or '#f'
ret the newly created <pango-layout>. Use g-object-unref when done.
```

Since 1.0

```cpp
clutter-actor-set-text-direction (self <clutter-actor>) (text_dir <clutter-text-direction>)
```

`set-text-direction` [Method]

Sets the `<clutter-text-direction>` for an actor

The passed text direction must not be 'CLUTTER_TEXT_DIRECTION_DEFAULT'

If `self` implements `<clutter-container>` then this function will recurse inside all the children of `self` (including the internal ones).
Composite actors not implementing `<clutter-container>`, or actors requiring special handling when the text direction changes, should connect to the `<"notify">` signal for the `<"text-direction">` property

```plaintext
self    a <clutter-actor>
text-dir the text direction for self
```

Since 1.2

```plaintext
clutter-actor-get-text-direction (self <clutter-actor>) ⇒ (ret <clutter-text-direction>)
```

*Function*

*Method*

Retrieves the value set using `clutter-actor-set-text-direction`

If no text direction has been previously set, the default text direction, as returned by `clutter-get-default-text-direction`, will be returned instead

```plaintext
self    a <clutter-actor>
ret    the <clutter-text-direction> for the actor
```

Since 1.2

```plaintext
clutter-actor-get-accessible (self <clutter-actor>) ⇒ (ret <atk-object>)
```

*Function*

*Method*

Returns the accessible object that describes the actor to an assistive technology.

If no class-specific `<atk-object>` implementation is available for the actor instance in question, it will inherit an `<atk-object>` implementation from the first ancestor class for which such an implementation is defined.

The documentation of the ATK library contains more information about accessible objects and their uses.

```plaintext
self    a <clutter-actor>
ret    the <atk-object> associated with actor.
```

```plaintext
clutter-actor-add-action (self <clutter-actor>) (action <clutter-action>)
```

*Function*

*Method*

Adds `action` to the list of actions applied to `self`

A `<clutter-action>` can only belong to one actor at a time

The `<clutter-actor>` will hold a reference on `action` until either `clutter-actor-remove-action` or `clutter-actor-clear-actions` is called

```plaintext
self    a <clutter-actor>
action  a <clutter-action>
```

Since 1.4
Chapter 4: ClutterActor

clutter-actor-add-action-with-name (self <clutter-actor>) [Function]
    (name mchars) (action <clutter-action>)
add-action-with-name [Method]
    A convenience function for setting the name of a <clutter-action> while adding it to the list of actions applied to self
    This function is the logical equivalent of:

    clutter_actor_meta_set_name (CLUTTER_ACTOR_META (action), name);
    clutter_actor_add_action (self, action);

    self a <clutter-actor>
    name the name to set on the action
    action a <clutter-action>
Since 1.4

clutter-actor-remove-action (self <clutter-actor>) [Function]
    (action <clutter-action>)
remove-action [Method]
    Removes action from the list of actions applied to self
    The reference held by self on the <clutter-action> will be released

    self a <clutter-actor>
    action a <clutter-action>
Since 1.4

clutter-actor-remove-action-by-name (self <clutter-actor>) [Function]
    (name mchars)
remove-action-by-name [Method]
    Removes the <clutter-action> with the given name from the list of actions applied to self

    self a <clutter-actor>
    name the name of the action to remove
Since 1.4

clutter-actor-has-actions (self <clutter-actor>) ⇒ (ret bool) [Function]
has-actions [Method]
    Returns whether the actor has any actions applied.

    self A <clutter-actor>
    ret ‘#t’ if the actor has any actions, ‘#f’ otherwise
Since 1.10

clutter-actor-get-actions (self <clutter-actor>) [Function]
    ⇒ (ret glist-of)
get-actions [Method]
    Retrieves the list of actions applied to self
Chapter 4: ClutterActor

self  a <clutter-actor>
ret  a copy of the list of <clutter-action>s. The contents of the list are owned by the <clutter-actor>. Use g-list-free to free the resources allocated by the returned <g-list>.

Since 1.4

clutter-actor-get-action (self <clutter-actor>) (name mchars)  [Function]
⇒ (ret <clutter-action>)
get-action  [Method]
Retrieves the <clutter-action> with the given name in the list of actions applied to self
self  a <clutter-actor>
name  the name of the action to retrieve
ret  a <clutter-action> for the given name, or '#f'. The returned <clutter-action> is owned by the actor and it should not be unreferenced directly.

Since 1.4

clutter-actor-clear-actions (self <clutter-actor>)  [Function]
clear-actions  [Method]
Clears the list of actions applied to self
self  a <clutter-actor>

Since 1.4

clutter-actor-add-constraint (self <clutter-actor>)  [Function]
(constraint <clutter-constraint>)
add-constraint  [Method]
Adds constraint to the list of <clutter-constraint>s applied to self
The <clutter-actor> will hold a reference on the constraint until either clutter-actor-remove-constraint or clutter-actor-clear-constraints is called.
sel  a <clutter-actor>
constraint  a <clutter-constraint>

Since 1.4

clutter-actor-remove-constraint (self <clutter-actor>)  [Function]
(constraint <clutter-constraint>)
remove-constraint  [Method]
Removes constraint from the list of constraints applied to self
The reference held by self on the <clutter-constraint> will be released
self  a <clutter-actor>
constraint  a <clutter-constraint>

Since 1.4
Chapter 4: ClutterActor

clutter-actor-has-constraints (self <clutter-actor>) ⇒ (ret bool)

has-constraints
Returns whether the actor has any constraints applied.

self A <clutter-actor>
ret ‘#t’ if the actor has any constraints, ‘#f’ otherwise

Since 1.10

clutter-actor-get-constraints (self <clutter-actor>) ⇒ (ret glist-of)
get-constraints
Retrieves the list of constraints applied to self

self a <clutter-actor>
ret a copy of the list of <clutter-constraint>s. The contents of the list are owned by the <clutter-actor>. Use g-list-free to free the resources allocated by the returned <g-list>.

Since 1.4

clutter-actor-get-constraint (self <clutter-actor>) (name mchars) ⇒ (ret <clutter-constraint>)
get-constraint
Retrieves the <clutter-constraint> with the given name in the list of constraints applied to self

self a <clutter-actor>
name the name of the constraint to retrieve
ret a <clutter-constraint> for the given name, or ‘#f’. The returned <clutter-constraint> is owned by the actor and it should not be un-referenced directly.

Since 1.4

clutter-actor-clear-constraints (self <clutter-actor>)
clear-constraints
Clears the list of constraints applied to self

self a <clutter-actor>

Since 1.4

clutter-actor-add-effect (self <clutter-actor>) (effect <clutter-effect>)
add-effect
Adds effect to the list of <clutter-effect>s applied to self
The <clutter-actor> will hold a reference on the effect until either clutter-actor-remove-effect or clutter-actor-clear-effects is called.
Chapter 4: ClutterActor

Since 1.4

`clutter-actor-add-effect-with-name (self <clutter-actor>)` [Function]

(name mchars) (effect <clutter-effect>)

`add-effect-with-name` [Method]

A convenience function for setting the name of a `<clutter-effect>` while adding it to the list of effects applied to `self`

This function is the logical equivalent of:

```c
clutter_actor_meta_set_name (CLUTTER_ACTOR_META (effect), name);
clutter_actor_add_effect (self, effect);
```

Since 1.4

`clutter-actor-remove-effect (self <clutter-actor>)` [Function]

(effect <clutter-effect>)

`remove-effect` [Method]

Removes `effect` from the list of effects applied to `self`

The reference held by `self` on the `<clutter-effect>` will be released

Since 1.4

`clutter-actor-remove-effect-by-name (self <clutter-actor>)` [Function]

(name mchars)

`remove-effect-by-name` [Method]

Removes the `<clutter-effect>` with the given name from the list of effects applied to `self`

Since 1.4

`clutter-actor-has-effects (self <clutter-actor>) ⇒ (ret bool)` [Function]

`has-effects` [Method]

Returns whether the actor has any effects applied.

Since 1.10
clutter-actor-get-effects (self <clutter-actor>)  
⇒ (ret glist-of)  
get-effects  
Retrieves the <clutter-effect>s applied on self, if any  
  self   a <clutter-actor>  
  ret    a list of <clutter-effect>s, or ‘#f’. The elements of the returned list  
         are owned by Clutter and they should not be freed. You should free the  
         returned list using g-list-free when done.  

Since 1.4  

clutter-actor-get-effect (self <clutter-actor>) (name mchars)  
⇒ (ret <clutter-effect>)  
get-effect  
Retrieves the <clutter-effect> with the given name in the list of effects applied to  
self  
  self   a <clutter-actor>  
  name   the name of the effect to retrieve  
  ret    a <clutter-effect> for the given name, or ‘#f’. The returned  
         <clutter-effect> is owned by the actor and it should not be  
         unreferenced directly.  

Since 1.4  

clutter-actor-clear-effects (self <clutter-actor>)  
clear-effects  
Clears the list of effects applied to self  
  self   a <clutter-actor>  

Since 1.4  

clutter-actor-box-new (x.1 float) (y.1 float) (x.2 float) (y.2 float)  
⇒ (ret <clutter-actor-box>)  
Allocates a new <clutter-actor-box> using the passed coordinates for the top left  
and bottom right points  
  x-1    X coordinate of the top left point  
  y-1    Y coordinate of the top left point  
  x-2    X coordinate of the bottom right point  
  y-2    Y coordinate of the bottom right point  
  ret    the newly allocated <clutter-actor-box>. Use clutter-actor-box-  
         free to free the resources  

Since 1.0
clutter-actor-box-init (self <clutter-actor-box>) (x_1 float) (y_1 float) (x_2 float) (y_2 float)
Initializes box with the given coordinates.

box a <clutter-actor-box>
x-1 X coordinate of the top left point
y-1 Y coordinate of the top left point
x-2 X coordinate of the bottom right point
y-2 Y coordinate of the bottom right point
Since 1.10

clutter-actor-box-init-rect (self <clutter-actor-box>) (x float) (y float) (width float) (height float)
Initializes box with the given origin and size.

box a <clutter-actor-box>
x X coordinate of the origin
y Y coordinate of the origin
width width of the box
height height of the box
Since 1.10

clutter-actor-box-equal (self <clutter-actor-box>) (box_b <clutter-actor-box>) ⇒ (ret bool)
Checks box-a and box-b for equality

box-a a <clutter-actor-box>
box-b a <clutter-actor-box>
ret ‘#t’ if the passed <clutter-actor-box> are equal
Since 1.0

clutter-actor-box-get-x (self <clutter-actor-box>) ⇒ (ret float)
Retrieves the X coordinate of the origin of box

box a <clutter-actor-box>
ret the X coordinate of the origin
Since 1.0

clutter-actor-box-get-y (self <clutter-actor-box>) ⇒ (ret float)
Retrieves the Y coordinate of the origin of box

box a <clutter-actor-box>
ret the Y coordinate of the origin
Since 1.0
clutter-actor-box-get-width (self <clutter-actor-box>)  [Function]
⇒ (ret float)
Retrieves the width of the box
box       a <clutter-actor-box>
ret       the width of the box
Since 1.0

clutter-actor-box-get-height (self <clutter-actor-box>)  [Function]
⇒ (ret float)
Retrieves the height of the box
box       a <clutter-actor-box>
ret       the height of the box
Since 1.0

clutter-actor-box-set-origin (self <clutter-actor-box>)  [Function]
(x float) (y float)
Changes the origin of box, maintaining the size of the <clutter-actor-box>.
box       a <clutter-actor-box>
x          the X coordinate of the new origin
y          the Y coordinate of the new origin
Since 1.6

clutter-actor-box-get-origin (self <clutter-actor-box>)  [Function]
⇒ (x float) (y float)
Retrieves the origin of box
box       a <clutter-actor-box>
x          return location for the X coordinate, or '#f'.
y          return location for the Y coordinate, or '#f'.
Since 1.0

clutter-actor-box-set-size (self <clutter-actor-box>)  [Function]
(width float) (height float)
Sets the size of box, maintaining the origin of the <clutter-actor-box>.
box       a <clutter-actor-box>
width     the new width
height    the new height
Since 1.6
**clutter-actor-box-get-size** (*self* <clutter-actor-box>)

⇒ (*width* float) (*height* float)

Retrieves the size of *box*

*box* a <clutter-actor-box>

*width* return location for the width, or ‘#f’.

*height* return location for the height, or ‘#f’.

Since 1.0

**clutter-actor-box-get-area** (*self* <clutter-actor-box>)

⇒ (*ret* float)

Retrieves the area of *box*

*box* a <clutter-actor-box>

*ret* the area of a <clutter-actor-box>, in pixels

Since 1.0

**clutter-actor-box-contains** (*self* <clutter-actor-box>)

(*x* float) (*y* float) ⇒ (*ret* bool)

Checks whether a point with *x*, *y* coordinates is contained within *box*

*box* a <clutter-actor-box>

*x* X coordinate of the point

*y* Y coordinate of the point

*ret* ‘#t’ if the point is contained by the <clutter-actor-box>

Since 1.0

**clutter-actor-box-clamp-to-pixel** (*self* <clutter-actor-box>)

Clamps the components of *box* to the nearest integer

*box* the <clutter-actor-box> to clamp.

Since 1.2

**clutter-actor-box-interpolate** (*self* <clutter-actor-box>)

(*final* <clutter-actor-box>) (*progress* double) ⇒ (*result* <clutter-actor-box>)

Interpolates between *initial* and *final*<clutter-actor-box>es using *progress*

*initial* the initial <clutter-actor-box>

*final* the final <clutter-actor-box>

*progress* the interpolation progress

*result* return location for the interpolation.

Since 1.2
**clutter-actor-box-union** *(self <clutter-actor-box>) [Function]*

(b <clutter-actor-box>) (result <clutter-actor-box>)

Unions the two boxes a and b and stores the result in result.

a (in) the first <clutter-actor-box>

b the second <clutter-actor-box>.

result the <clutter-actor-box> representing a union of a and b.

Since 1.4

**clutter-vertex-new** *(x float) (y float) (z float) [Function]*

⇒ (ret <clutter-vertex>)

Creates a new <clutter-vertex> for the point in 3D space identified by the 3 coordinates x, y, z

x X coordinate

y Y coordinate

z Z coordinate

ret the newly allocate <clutter-vertex>. Use clutter-vertex-free to free the resources

Since 1.0

**clutter-vertex-init** *(self <clutter-vertex>) (x float) (y float) (z float) [Function]*

Initializes vertex with the given coordinates.

vertex a <clutter-vertex>

x X coordinate

y Y coordinate

z Z coordinate

Since 1.10

**clutter-vertex-equal** *(self <clutter-vertex>) [Function]*

(vertex_b <clutter-vertex>) ⇒ (ret bool)

Compares vertex-a and vertex-b for equality

vertex-a a <clutter-vertex>

vertex-b a <clutter-vertex>

ret ‘#t’ if the passed <clutter-vertex> are equal

Since 1.0

**clutter-geometry-union** *(self <clutter-geometry>) [Function]*

(geometry_b <clutter-geometry>) (result <clutter-geometry>)

Find the union of two rectangles represented as <clutter-geometry>. 
geometry-a
   a <clutter-geometry>

geometry-b
   another <clutter-geometry>

result  location to store the result.

Since 1.4

clutter-geometry-intersects (self <clutter-geometry>)  [Function]
   (geometry1 <clutter-geometry>) ⇒ (ret bool)
Determines if geometry0 and geometry1 intersect returning ‘#t’ if they do else ‘#f’.

geometry0  The first geometry to test

geometry1  The second geometry to test

ret  ‘#t’ of geometry0 and geometry1 intersect else ‘#f’.

Since 1.4

clutter-paint-volume-set-origin (self <clutter-paint-volume>)  [Function]
   (origin <clutter-vertex>)
Sets the origin of the paint volume.

The origin is defined as the X, Y and Z coordinates of the top-left corner of an actor’s
paint volume, in actor coordinates.

The default is origin is assumed at: (0, 0, 0)

pv  a <clutter-paint-volume>

origin  a <clutter-vertex>

Since 1.6

clutter-paint-volume-get-origin (self <clutter-paint-volume>)  [Function]
   (vertex <clutter-vertex>)
Retrieves the origin of the <clutter-paint-volume>.

pv  a <clutter-paint-volume>

vertex  the return location for a <clutter-vertex>.

Since 1.6

clutter-paint-volume-set-width (self <clutter-paint-volume>)  [Function]
   (width float)
Sets the width of the paint volume. The width is measured along the x axis in the
actor coordinates that pv is associated with.

pv  a <clutter-paint-volume>

width  the width of the paint volume, in pixels

Since 1.6
**clutter-paint-volume-get-width** *(self <clutter-paint-volume>)*  
[Function]  
⇒ (ret float)  
Retrieves the width of the volume’s, axis aligned, bounding box.  
In other words; this takes into account what actor’s coordinate space \( pv \) belongs too and conceptually fits an axis aligned box around the volume. It returns the size of that bounding box as measured along the x-axis.  
If, for example, **clutter-actor-get-transformed-paint-volume** is used to transform a 2D child actor that is 100px wide, 100px high and 0px deep into container coordinates then the width might not simply be 100px if the child actor has a 3D rotation applied to it.  
Remember; after **clutter-actor-get-transformed-paint-volume** is used then a transformed child volume will be defined relative to the ancestor container actor and so a 2D child actor can have a 3D bounding volume.  

There are no accuracy guarantees for the reported width, except that it must always be \( \geq \) to the true width. This is because actors may report simple, loose fitting paint-volumes for efficiency.  

\( pv \)  
a <clutter-paint-volume>  
ret  
the width, in units of \( pv \)’s local coordinate system.  

Since 1.6

**clutter-paint-volume-set-height** *(self <clutter-paint-volume>)*  
(height float)  
Sets the height of the paint volume. The height is measured along the y axis in the actor coordinates that \( pv \) is associated with.  
\( pv \)  
a <clutter-paint-volume>  
height  
the height of the paint volume, in pixels  
Since 1.6

**clutter-paint-volume-get-height** *(self <clutter-paint-volume>)*  
[Function]  
⇒ (ret float)  
Retrieves the height of the volume’s, axis aligned, bounding box.  
In other words; this takes into account what actor’s coordinate space \( pv \) belongs too and conceptually fits an axis aligned box around the volume. It returns the size of that bounding box as measured along the y-axis.  
If, for example, **clutter-actor-get-transformed-paint-volume** is used to transform a 2D child actor that is 100px wide, 100px high and 0px deep into container coordinates then the height might not simply be 100px if the child actor has a 3D rotation applied to it.  
Remember; after **clutter-actor-get-transformed-paint-volume** is used then a transformed child volume will be defined relative to the ancestor container actor and so a 2D child actor can have a 3D bounding volume.
There are no accuracy guarantees for the reported height, except that it must always be \( \geq \) to the true height. This is because actors may report simple, loose fitting paint-volumes for efficiency.

\[
\begin{align*}
pv & \quad \text{a \texttt{<clutter-paint-volume>}} \\
ret & \quad \text{the height, in units of } pv\text{'s local coordinate system.}
\end{align*}
\]

Since 1.6

**clutter-paint-volume-set-depth** \((self \texttt{<clutter-paint-volume>})\)  \([\text{Function}]\)

\[
(depth \texttt{float})
\]

Sets the depth of the paint volume. The depth is measured along the z axis in the actor coordinates that \(pv\) is associated with.

\[
\begin{align*}
pv & \quad \text{a \texttt{<clutter-paint-volume>}} \\
depth & \quad \text{the depth of the paint volume, in pixels}
\end{align*}
\]

Since 1.6

**clutter-paint-volume-get-depth** \((self \texttt{<clutter-paint-volume>})\)  \([\text{Function}]\)

\[
\Rightarrow (ret \texttt{float})
\]

Retrieves the depth of the volume’s, axis aligned, bounding box.

In other words; this takes into account what actor’s coordinate space \(pv\) belongs too and conceptually fits an axis aligned box around the volume. It returns the size of that bounding box as measured along the z-axis.

If, for example, **clutter-actor-get-transformed-paint-volume** is used to transform a 2D child actor that is 100px wide, 100px high and 0px deep into container coordinates then the depth might not simply be 0px if the child actor has a 3D rotation applied to it.

Remember; after **clutter-actor-get-transformed-paint-volume** is used then the transformed volume will be defined relative to the container actor and in container coordinates a 2D child actor can have a 3D bounding volume.

There are no accuracy guarantees for the reported depth, except that it must always be \( \geq \) to the true depth. This is because actors may report simple, loose fitting paint-volumes for efficiency.

\[
\begin{align*}
pv & \quad \text{a \texttt{<clutter-paint-volume>}} \\
ret & \quad \text{the depth, in units of } pv\text{'s local coordinate system.}
\end{align*}
\]

Since 1.6

**clutter-paint-volume-union** \((self \texttt{<clutter-paint-volume>})\)  \([\text{Function}]\)

\[
(another\_pv \texttt{<clutter-paint-volume>})
\]

Updates the geometry of \(pv\) to encompass \(pv\) and \(another\_pv\).

There are no guarantees about how precisely the two volumes will be encompassed.
\textit{pv} \hspace{1em} The first \texttt{<clutter-paint-volume>} and destination for resulting union

\textit{another-pv} \hspace{1em} A second \texttt{<clutter-paint-volume>} to union with \textit{pv} \\

Since 1.6

\texttt{clutter-paint-volume-union-box (self <clutter-paint-volume>) (box <clutter-actor-box>)} \hspace{1em} [Function]  \\
Unions the 2D region represented by \textit{box} to a \texttt{<clutter-paint-volume>}.
This function is similar to \texttt{clutter-paint-volume-union}, but it is specific for 2D regions.

\textit{pv} \hspace{1em} a \texttt{<clutter-paint-volume>}

\textit{box} \hspace{1em} a \texttt{<clutter-actor-box>} to union to \textit{pv}  \\
Since 1.10
5 ClutterAlignConstraint

A constraint aligning the position of an actor

5.1 Overview

<clutter-align-constraint> is a <clutter-constraint> that aligns the position of the <clutter-actor> to which it is applied to the size of another <clutter-actor> using an alignment factor

<clutter-align-constraint> is available since Clutter 1.4

5.2 Usage

clutter-align-constraint-new (source <clutter-actor>) [Function]
( axis <clutter-align-axis>) (factor float)
⇒ (ret <clutter-constraint>)
Creates a new constraint, aligning a <clutter-actor>’s position with regards of the size of the actor to source, with the given alignment factor

source the <clutter-actor> to use as the source of the alignment, or ‘#f’.
axis the axis to be used to compute the alignment
factor the alignment factor, between 0.0 and 1.0
ret the newly created <clutter-align-constraint>

Since 1.4

clutter-align-constraint-set-source (self <clutter-align-constraint>) (source <clutter-actor>) [Function]
set-source [Method]
Sets the source of the alignment constraint

align a <clutter-align-constraint>
source a <clutter-actor>, or ‘#f’ to unset the source.

Since 1.4

clutter-align-constraint-get-source (self <clutter-align-constraint>) ⇒ (ret <clutter-actor>) [Function]
get-source [Method]
Retrieves the source of the alignment

align a <clutter-align-constraint>
ret the <clutter-actor> used as the source of the alignment.

Since 1.4
**clutter-align-constraint-set-factor** [Function]

\[
\text{(self <clutter-align-constraint>)(factor float)}
\]

**set-factor** [Method]

Sets the alignment factor of the constraint

The factor depends on the "align-axis" property and it is a value between 0.0 (meaning left, when "align-axis" is set to ‘CLUTTER_ALIGN_X_AXIS’); or meaning top, when "align-axis" is set to ‘CLUTTER_ALIGN_Y_AXIS’) and 1.0 (meaning right, when "align-axis" is set to ‘CLUTTER_ALIGN_X_AXIS’; or meaning bottom, when "align-axis" is set to ‘CLUTTER_ALIGN_Y_AXIS’). A value of 0.5 aligns in the middle in either cases

\[
\begin{align*}
\text{align} & \quad \text{a <clutter-align-constraint>} \\
\text{factor} & \quad \text{the alignment factor, between 0.0 and 1.0}
\end{align*}
\]

Since 1.4

**clutter-align-constraint-get-factor** [Function]

\[
\text{(self <clutter-align-constraint>) ⇒ (ret float)}
\]

**get-factor** [Method]

Retrieves the factor set using clutter-align-constraint-set-factor

\[
\begin{align*}
\text{align} & \quad \text{a <clutter-align-constraint>} \\
\text{ret} & \quad \text{the alignment factor}
\end{align*}
\]

Since 1.4
6 ClutterAlpha

A class for calculating a value as a function of time

6.1 Overview

<clutter-alpha> is a class for calculating an floating point value dependent only on the position of a <clutter-timeline>.

(code "<"property of " (code "") ", or the\n (code "function instead of " (code "") 


A <clutter-alpha> binds a <clutter-timeline> to a progress function which translates the time T into an adimensional factor alpha. The factor can then be used to drive a <clutter-behaviour>, which will translate the alpha value into something meaningful for a <clutter-actor>.

You should provide a <clutter-timeline> and bind it to the <clutter-alpha> instance using clutter-alpha-set-timeline. You should also set an "animation mode", either by using the <clutter-animation-mode> values that Clutter itself provides or by registering custom functions using clutter-alpha-register-func.

Instead of a <clutter-animation-mode> you may provide a function returning the alpha value depending on the progress of the timeline, using clutter-alpha-set-func or clutter-alpha-set-closure. The alpha function will be executed each time a new frame in the <clutter-timeline> is reached.

Since the alpha function is controlled by the timeline instance, you can pause, stop or resume the <clutter-alpha> from calling the alpha function by using the appropriate functions of the <clutter-timeline> object.

<clutter-alpha> is used to "drive" a <clutter-behaviour> instance, and it is internally used by the <clutter-animation> API.

(The missing figure, easing-modes)

6.2 ClutterAlpha custom properties for <clutter-script>

<clutter-alpha> defines a custom "function" property for <clutter-script> which allows to reference a custom alpha function available in the source code. Setting the "function" property is equivalent to calling clutter-alpha-set-func with the specified function name. No user data or <g-destroy-notify> is available to be passed.

The following JSON fragment defines a <clutter-alpha> using a <clutter-timeline> with id "sine-timeline" and an alpha function called my-sine-alpha. The defined <clutter-alpha> instance can be reused in multiple <clutter-behaviour> definitions or for <clutter-animation> definitions.

```json
{
    "id" : "sine-alpha",
    "timeline" : {
        "id" : "sine-timeline",
```
"duration" : 500,
"loop" : true
},
"function" : "my_sine_alpha"
}

For the way to define the <"mode"> property inside a ClutterScript fragment, see the corresponding section in <clutter-animation>.

### 6.3 Usage

**clutter-alpha-new ⇒ (ret <clutter-alpha>)**  
[Function]  
Creates a new <clutter-alpha> instance. You must set a function to compute the alpha value using clutter-alpha-set-func and bind a <clutter-timeline> object to the <clutter-alpha> instance using clutter-alpha-set-timeline.  
You should use the newly created <clutter-alpha> instance inside a <clutter-behaviour> object.

*ret* the newly created empty <clutter-alpha> instance.

Since 0.2

**clutter-alpha-set-timeline (self <clutter-alpha>)**  
(set-timeline <clutter-timeline>)  
[Function]  
Binds alpha to timeline.

*alpha* A <clutter-alpha>

*timeline* A <clutter-timeline>

Since 0.2

**clutter-alpha-get-timeline (self <clutter-alpha>)**  
⇒ (ret <clutter-timeline>)  
[Function]  
Gets the <clutter-timeline> bound to alpha.

*alpha* A <clutter-alpha>

*ret* a <clutter-timeline> instance.

Since 0.2

**clutter-alpha-set-mode (self <clutter-alpha>)**  
(set-mode <clutter-animation-mode>)  
[Function]  
Sets the progress function of alpha using the symbolic value of mode, as taken by the <clutter-animation-mode> enumeration or using the value returned by clutter-alpha-register-func.

*alpha* A <clutter-alpha>
mode a <clutter-animation-mode>

Since 1.0

clutter-alpha-get-mode (self <clutter-alpha>) ⇒ (ret <clutter-animation-mode>)

get-mode

Retrieves the <clutter-animation-mode> used by alpha.

alpha a <clutter-alpha>

ret the animation mode

Since 1.0

clutter-alpha-get-alpha (self <clutter-alpha>) ⇒ (ret double)

get-alpha

Query the current alpha value.

alpha A <clutter-alpha>

ret The current alpha value for the alpha

Since 0.2

clutter-alpha-set-closure (self <clutter-alpha>)

(closure <gclosure>)

set-closure

Sets the <gclosure> used to compute the alpha value at each frame of the <clutter-timeline> bound to alpha.

alpha A <clutter-alpha>

closure A <gclosure>

Since 0.8

clutter-alpha-register-closure (closure <gclosure>)

⇒ (ret unsigned-long)

<gclosure> variant of clutter-alpha-register-func.

Registers a global alpha function and returns its logical id to be used by clutter-alpha-set-mode or by <clutter-animation>.

The logical id is always greater than ‘CLUTTER_ANIMATION_LAST’.

Rename to: clutter_alpha_register_func

closure a <gclosure>

ret the logical id of the alpha function

Since 1.0
7 ClutterAnimatable

Interface for animatable classes

7.1 Overview

<clutter-animatable> is an interface that allows a <gobject> class to control how a <clutter-animation> will animate a property.

Each <clutter-animatable> should implement the animate-property virtual function of the interface to compute the animation state between two values of an interval depending on a progress factor, expressed as a floating point value.

If a <clutter-animatable> is animated by a <clutter-animation> instance, the <clutter-animation> will call clutter-animatable-animate-property passing the name of the currently animated property; the initial and final values of the animation interval; the progress factor. The <clutter-animatable> implementation should return the computed value for the animated property.

<clutter-animatable> is available since Clutter 1.0

7.2 Usage

clutter-animatable-find-property (self <clutter-animatable>)  
  (property_name mchars) ⇒ (ret <gparam>)

find-property
  Finds the <gparam> for property-name animatable a <clutter-animatable>

  property-name
    the name of the animatable property to find

  ret
    The <gparam> for the given property or ‘#f’.

Since 1.4

clutter-animatable-set-final-state
  (self <clutter-animatable>) (property_name mchars) (value <gvalue>)

set-final-state
  Sets the current state of property-name to value animatable a <clutter-animatable>

  property-name
    the name of the animatable property to set

  value
    the value of the animatable property to set

Since 1.4
8 Implicit Animations

Simple implicit animations

8.1 Overview
<clutter-animation> is an object providing simple, implicit animations for <gobject>s.

<clutter-animation> instances will bind one or more <gobject> properties belonging to a <gobject> to a <clutter-interval>, and will then use a <clutter-alpha> to interpolate the property between the initial and final values of the interval.

The duration of the animation is set using clutter-animation-set-duration. The easing mode of the animation is set using clutter-animation-set-mode.

If you want to control the animation you should retrieve the <clutter-timeline> using clutter-animation-get-timeline and then use <clutter-timeline> functions like clutter-timeline-start, clutter-timeline-pause or clutter-timeline-stop.

A <clutter-animation> will emit the "completed" signal when the <clutter-timeline> used by the animation is completed; unlike <clutter-timeline>, though, the "completed" will not be emitted if "loop" is set to '#t' - that is, a looping animation never completes.

If your animation depends on user control you can force its completion using clutter-animation-completed.

If the <gobject> instance bound to a <clutter-animation> implements the <clutter-animatable> interface it is possible for that instance to control the way the initial and final states are interpolated.

<clutter-animation>s are distinguished from <clutter-behaviour>s because the former can only control <gobject> properties of a single <gobject> instance, while the latter can control multiple properties using accessor functions inside the <clutter-behaviour>alpha-notify virtual function, and can control multiple <clutter-actor>s as well.

For convenience, it is possible to use the clutter-actor-animate function call which will take care of setting up and tearing down a <clutter-animation> instance and animate an actor between its current state and the specified final state.

8.2 Defining ClutterAnimationMode inside ClutterScript
When defining a <clutter-animation> inside a ClutterScript file or string the "mode" can be defined either using the <clutter-animation-mode> enumeration values through their "nick" (the short string used inside <g-enum-value>), their numeric id, or using the following strings:
Chapter 8: Implicit Animations

easeInCubic, easeOutCubic, easeInOutCubic
easeInQuart, easeOutQuart, easeInOutQuart
easeInQuint, easeOutQuint, easeInOutQuint
easeInSine, easeOutSine, easeInOutSine
easeInExpo, easeOutExpo, easeInOutExpo
easeInCirc, easeOutCirc, easeInOutCirc
easeInElastic, easeOutElastic, easeInOutElastic
easeInBack, easeOutBack, easeInOutBack
easeInBounce, easeOutBounce, easeInOutBounce

Corresponding to the quadratic easing modes
Corresponding to the cubic easing modes
Corresponding to the quartic easing modes
Corresponding to the quintic easing modes
Corresponding to the sine easing modes
Corresponding to the exponential easing modes
Corresponding to the circular easing modes
Corresponding to the overshooting elastic easing modes
Corresponding to the overshooting cubic easing modes
Corresponding to the bouncing easing modes

<clutter-animation> is available since Clutter 1.0

8.3 Usage

clutter-animation-new ⇒ (ret <clutter-animation>) \[Function\]
Creates a new <clutter-animation> instance. You should set the <gobject> to be animated using clutter-animation-set-object, set the duration with clutter-animation-set-duration and the easing mode using clutter-animation-set-mode.

Use clutter-animation-bind or clutter-animation-bind-interval to define the properties to be animated. The interval and the animated properties can be updated at runtime.

The clutter-actor-animate and relative family of functions provide an easy way to animate a <clutter-actor> and automatically manage the lifetime of a <clutter-animation> instance, so you should consider using those functions instead of manually creating an animation.

ret the newly created <clutter-animation>. Use g-object-unref to release the associated resources

Since 1.0

clutter-animation-set-object (self <clutter-animation>) \[Function\]
(object <gobject>)
set-object \[Method\]
Attaches animation to object. The <clutter-animation> will take a reference on object.
Chapter 8: Implicit Animations

animation a <clutter-animation>

object a <gobject>

Since 1.0

clutter-animation-get-object (self <clutter-animation>) [Function]
⇒ (ret <gobject>)

get-object [Method]
Retrieves the <gobject> attached to animation.

animation a <clutter-animation>
ret a <gobject>.

Since 1.0

clutter-animation-set-mode (self <clutter-animation>) [Function]
(mode unsigned-long)

set-mode [Method]
Sets the animation mode of animation. The animation mode is a logical id, either coming from the <clutter-animation-mode> enumeration or the return value of clutter-alpha-register-func.

This function will also set "alpha" if needed.

animation a <clutter-animation>
mode an animation mode logical id

Since 1.0

clutter-animation-get-mode (self <clutter-animation>) [Function]
⇒ (ret unsigned-long)

get-mode [Method]
Retrieves the animation mode of animation, as set by clutter-animation-set-mode.

animation a <clutter-animation>
ret the mode for the animation

Since 1.0

clutter-animation-set-duration (self <clutter-animation>) [Function]
(msecs unsigned-int)

set-duration [Method]
Sets the duration of animation in milliseconds.

This function will set "alpha" and "timeline" if needed.

animation a <clutter-animation>
msecs the duration in milliseconds

Since 1.0
clutter-animation-get-duration (self <clutter-animation>) ⇒ (ret unsigned-int)

get-duration
Retrieves the duration of animation, in milliseconds.

animation a <clutter-animation>
ret the duration of the animation

Since 1.0

clutter-animation-set-loop (self <clutter-animation>) (loop bool)

set-loop
Sets whether animation should loop over itself once finished.
A looping <clutter-animation> will not emit the "completed" signal when finished.
This function will set "alpha" and "timeline" if needed.

animation a <clutter-animation>
loop ‘#t’ if the animation should loop

Since 1.0

clutter-animation-get-loop (self <clutter-animation>) ⇒ (ret bool)

get-loop
Retrieves whether animation is looping.

animation a <clutter-animation>
ret ‘#t’ if the animation is looping

Since 1.0

clutter-animation-set-timeline (self <clutter-animation>) (timeline <clutter-timeline>)

set-timeline
Sets the <clutter-timeline> used by animation.
This function will take a reference on the passed timeline.

animation a <clutter-animation>
timeline a <clutter-timeline>, or ‘#f’ to unset the current <clutter-timeline>.

Since 1.0

clutter-animation-get-timeline (self <clutter-animation>) ⇒ (ret <clutter-timeline>)

get-timeline
Retrieves the <clutter-timeline> used by animation

animation a <clutter-animation>
ret the timeline used by the animation.

Since 1.0

**clutter-animation-completed** (*self <clutter-animation>*) [Function]
**completed** [Method]
Emits the ::completed signal on animation
When using this function with a <clutter-animation> created by the clutter-actor-animate family of functions, animation will be unreferenced and it will not be valid anymore, unless g-object-ref was called before calling this function or unless a reference was taken inside a handler for the "completed" signal

animation a <clutter-animation>

Since 1.0

**clutter-animation-bind** (*self <clutter-animation>*) [Function]
**(property_name mchars) (final <gvalue>)** ⇒ (*ret <clutter-animation>*) [Method]
Adds a single property with name property-name to the animation animation. For more information about animations, see clutter-actor-animate. This method returns the animation primarily to make chained calls convenient in language bindings.

animation a <clutter-animation>

property-name the property to control

final The final value of the property

ret The animation itself.

Since 1.0

**clutter-animation-bind-interval** (*self <clutter-animation>*) [Function]
**(property_name mchars) (interval <clutter-interval>)** ⇒ (*ret <clutter-animation>*) [Method]
Binds interval to the property-name of the <gobject> attached to animation. The <clutter-animation> will take ownership of the passed <clutter-interval>. For more information about animations, see clutter-actor-animate. If you need to update the interval instance use clutter-animation-update-interval instead.

animation a <clutter-animation>

property-name the property to control

interval a <clutter-interval>.

ret The animation itself.

Since 1.0
clutter-animation-update (self <clutter-animation>) \[\text{Function}\]
\[(\text{property-name} \ mchars) (\text{final} <\text{gvalue}>) \Rightarrow (\text{ret} <\text{clutter-animation}>)\]

**update** \[\text{Method}\]
Updates the \text{final} value of the interval for \text{property-name}

\textit{animation} a <clutter-animation>

\textit{property-name} name of the property

\textit{final} The final value of the property

\textit{ret} The animation itself.

Since 1.0

clutter-animation-update-interval (self <clutter-animation>) \[\text{Function}\]
\[(\text{property-name} \ mchars) (\text{interval} <\text{clutter-interval}>)\]

**update-interval** \[\text{Method}\]
Changes the \text{interval} for \text{property-name}. The <clutter-animation> will take ownership of the passed <clutter-interval>.

\textit{animation} a <clutter-animation>

\textit{property-name} name of the property

\textit{interval} a <clutter-interval>

Since 1.0

clutter-animation-has-property (self <clutter-animation>) \[\text{Function}\]
\[(\text{property-name} \ mchars) \Rightarrow (\text{ret} \text{bool})\]

**has-property** \[\text{Method}\]
Checks whether \text{animation} is controlling \text{property-name}.

\textit{animation} a <clutter-animation>

\textit{property-name} name of the property

\textit{ret} ‘#t’ if the property is animated by the <clutter-animation>, ‘#f’ otherwise

Since 1.0

clutter-animation-unbind-property (self <clutter-animation>) \[\text{Function}\]
\[(\text{property-name} \ mchars)\]

**unbind-property** \[\text{Method}\]
Removes \text{property-name} from the list of animated properties.

\textit{animation} a <clutter-animation>

\textit{property-name} name of the property

Since 1.0
clutter-animation-get-interval (self <clutter-animation>)  [Function]
  (property_name mchars) ⇒ (ret <clutter-interval>)
get-interval  [Method]
  Retrieves the <clutter-interval> associated to property-name inside animation.

animation  a <clutter-animation>

property-name
  name of the property

ret  a <clutter-interval> or '#f' if no property with the same name was found. The returned interval is owned by the <clutter-animation> and should not be unreferenced.

Since 1.0

clutter-actor-get-animation (self <clutter-actor>)  [Function]
  ⇒ (ret <clutter-animation>)
get-animation  [Method]
  Retrieves the <clutter-animation> used by actor, if clutter-actor-animate has been called on actor.

actor  a <clutter-actor>

ret  a <clutter-animation>, or '#f'.

Since 1.0

clutter-actor-detach-animation (self <clutter-actor>)  [Function]
detach-animation  [Method]
  Detaches the <clutter-animation> used by actor, if clutter-actor-animate has been called on actor.

Once the animation has been detached, it loses a reference. If it was the only reference then the <clutter-animation> becomes invalid.
The "completed" signal will not be emitted.

actor  a <clutter-actor>

Since 1.4
9 ClutterAnimator

Multi-actor tweener

9.1 Overview

<clutter-animator> is an object providing declarative animations for <gobject> properties belonging to one or more <gobject>s to <clutter-intervals>.

<clutter-animator> is used to build and describe complex animations in terms of "key frames". <clutter-animator> is meant to be used through the <clutter-script> definition format, but it comes with a convenience C API.

9.2 Key Frames

Every animation handled by a <clutter-animator> can be described in terms of "key frames". For each <gobject> property there can be multiple key frames, each one defined by the end value for the property to be computed starting from the current value to a specific point in time, using a given easing mode.

The point in time is defined using a value representing the progress in the normalized interval of \([0, 1]\). This maps the value returned by clutter-timeline-get-duration.

In the image above the duration of the animation is represented by the blue line. Each key frame is the white dot, along with its progress. The red line represents the computed function of time given the easing mode.

9.3 ClutterAnimator description for <clutter-script>

<clutter-animator> defines a custom "properties" property which allows describing the key frames for objects.

The "properties" property has the following syntax:

```
{
  "properties" : [
    {
      "object" : &lt;id of an object&gt;,  
      "name" : &lt;name of the property&gt;,  
      "ease-in" : &lt;boolean&gt;,  
      "interpolation" : &lt;#ClutterInterpolation value&gt;,  
      "keys" : [
        [ &lt;progress&gt;, &lt;easing mode&gt;, &lt;final value&gt; ]
      ]
    }
  ]
}
```

The following JSON fragment defines a <clutter-animator> with the duration of 1 second and operating on the x and y properties of a <clutter-actor> named "rect-01", with two frames for each property. The first frame will linearly move the actor from its
current position to the 100, 100 position in 20 percent of the duration of the animation; the second will using a cubic easing to move the actor to the 200, 200 coordinates.

```json
{
  "type" : "ClutterAnimator",
  "duration" : 1000,
  "properties" : [
    {
      "object" : "rect-01",
      "name" : "x",
      "ease-in" : true,
      "keys" : [
        [ 0.2, "linear", 100.0 ],
        [ 1.0, "easeOutCubic", 200.0 ]
      ]
    },
    {
      "object" : "rect-01",
      "name" : "y",
      "ease-in" : true,
      "keys" : [
        [ 0.2, "linear", 100.0 ],
        [ 1.0, "easeOutCubic", 200.0 ]
      ]
    }
  ]
}
```

<clutter-animator> is available since Clutter 1.2

### 9.4 Usage

`clutter-animator-new ⇒ (ret <clutter-animator>)`  
[Function]  
Creates a new <clutter-animator> instance

```diff
ret
```
a new <clutter-animator>.

Since 1.2

`clutter-animator-set-key (self <clutter-animator>) (object <gobject>) (property_name mchars) (mode unsigned-int) (progress double) (value <gvalue>) ⇒ (ret <clutter-animator>)`  
[Function]  
Sets a single key in the <clutter-animator> for the property-name of object at progress.

See also: clutter-animator-set

```diff
  animator
```
a <clutter-animator>
Chapter 9: ClutterAnimator

object  a <gobject>

property-name
the property to specify a key for

mode  the id of the alpha function to use

progress  the normalized range at which stage of the animation this value applies

value  the value property_name should have at progress.

ret  The animator instance.

Since 1.2

clutter-animator-remove-key (self <clutter-animator>)
(object <gobject>)
(property_name mchars)
(progress double)

remove-key
Removes all keys matching the conditions specified in the arguments.

animator  a <clutter-animator>

object  a <gobject> to search for, or ‘#f’ for all.

property-name
a specific property name to query for, or ‘#f’ for all.

progress  a specific progress to search for or a negative value for all

Since 1.2

clutter-animator-get-keys (self <clutter-animator>)
(object <gobject>)
(property_name mchars)
(progress double)
⇒
(ret glist-of)

get-keys
Returns a list of pointers to opaque structures with accessor functions that describe
the keys added to an animator.

animator  a <clutter-animator> instance

object  a <gobject> to search for, or ‘#f’ for all objects.

property-name
a specific property name to query for, or ‘#f’ for all properties.

progress  a specific progress to search for, or a negative value for all progresses

ret  a list of <clutter-animator-key>s; the contents of the list are owned
by the <clutter-animator>, but you should free the returned list when
done, using g-list-free.

Since 1.2

clutter-animator-start (self <clutter-animator>)
⇒  (ret <clutter-timeline>)

start
Start the ClutterAnimator, this is a thin wrapper that rewinds and starts the anima-
tors current timeline.
Chapter 9: ClutterAnimator

The clutter-animator that drives the animator. The returned timeline is owned by the <clutter-animator> and it should not be unreferenced.

Since 1.2

clutter-animator-compute-value (self <clutter-animator>)
  (object <gobject>) (property_name mchars) (progress double)
  (value <gvalue>) ⇒ (ret bool)

compute-value
  Compute the value for a managed property at a given progress.
  If the property is an ease-in property, the current value of the property on the object will be used as the starting point for computation.

  animator  a <clutter-animator>
  object    a <gobject>
  property-name
    the name of the property on object to check
  progress  a value between 0.0 and 1.0
  value     an initialized value to store the computed result
  ret       ‘#t’ if the computation yields has a value, otherwise (when an error occurs or the progress is before any of the keys) ‘#f’ is returned and the <gvalue> is left untouched

Since 1.2

clutter-animator-set-timeline (self <clutter-animator>)
  (timeline <clutter-timeline>)

set-timeline
  Sets an external timeline that will be used for driving the animation

  animator  a <clutter-animator>
  timeline  a <clutter-timeline>

Since 1.2

clutter-animator-get-timeline (self <clutter-animator>) ⇒ (ret <clutter-timeline>)

get-timeline
  Get the timeline hooked up for driving the <clutter-animator>

  animator  a <clutter-animator>
  ret       the <clutter-timeline> that drives the animator.

Since 1.2
clutter-animator-set-duration (self <clutter-animator>) [Function]
  (duration unsigned-int)
set-duration [Method]
  Runs the timeline of the <clutter-animator> with a duration in msecs as specified.
  animator a <clutter-animator>
duration milliseconds a run of the animator should last.
  Since 1.2

clutter-animator-get-duration (self <clutter-animator>) [Function]
  ⇒ (ret unsigned-int)
get-duration [Method]
  Retrieves the current duration of an animator
  animator a <clutter-animator>
  ret the duration of the animation, in milliseconds
  Since 1.2

clutter-animator-key-get-object (self <clutter-animator-key>) [Function]
  ⇒ (ret <gobject>)
Retrieves the object a key applies to.
  key a <clutter-animator-key>
  ret the object an animator_key exist for.
  Since 1.2

clutter-animator-key-get-mode (self <clutter-animator-key>) [Function]
  ⇒ (ret unsigned-long)
Retrieves the mode of a <clutter-animator> key, for the first key of a property for
an object this represents the whether the animation is open ended and or curved for
the remaining keys for the property it represents the easing mode.
  key a <clutter-animator-key>
  ret the mode of a <clutter-animator-key>
  Since 1.2

clutter-animator-key-get-progress
  (self <clutter-animator-key>) ⇒ (ret double)
Retrieves the progress of an clutter_animator_key
  key a <clutter-animator-key>
  ret the progress defined for a <clutter-animator> key.
  Since 1.2
clutter-animator-key-get-value (self <clutter-animator-key>)   [Function]
           (value <gvalue>) ⇒ (ret bool)
Retrieves a copy of the value for a <clutter-animator-key>.
The passed in <gvalue> needs to be already initialized for the value type of the key
or to a type that allow transformation from the value type of the key.
Use g-value-unset when done.

key        a <clutter-animator-key>
value      a <gvalue> initialized with the correct type for the animator key
ret        ‘#t’ if the passed <gvalue> was successfully set, and ‘#f’ otherwise

Since 1.2
Chapter 10: ClutterBackend

10 ClutterBackend

Backend abstraction

10.1 Overview

Clutter can be compiled against different backends. Each backend has to implement a set of functions, in order to be used by Clutter.

<clutter-backend> is the base class abstracting the various implementation; it provides a basic API to query the backend for generic information and settings.

<clutter-backend> is available since Clutter 0.4

10.2 Usage

clutter-get-default-backend ⇒ (ret <clutter-backend>) [Function]
Retrieves the default <clutter-backend> used by Clutter. The <clutter-backend> holds backend-specific configuration options.

ret the default backend. You should not ref or unref the returned object. Applications should rarely need to use this.

Since 0.4

clutter-backend-get-resolution (self <clutter-backend>) [Function]
 ⇒ (ret double)
get-resolution [Method]
Gets the resolution for font handling on the screen.
The resolution is a scale factor between points specified in a <pango-font-description> and cairo units. The default value is 96.0, meaning that a 10 point font will be 13 units high (10 * 96. / 72. = 13.3).
Clutter will set the resolution using the current backend when initializing; the resolution is also stored in the "font-dpi" property.

backend a <clutter-backend>
ret the current resolution, or -1 if no resolution has been set.

Since 0.4

clutter-backend-set-font-options (self <clutter-backend>) [Function]
 (options cairo-font-options-t)
set-font-options [Method]
Sets the new font options for backend. The <clutter-backend> will copy the <cairo-font-options-t>.
If options is '#f', the first following call to clutter-backend-get-font-options will return the default font options for backend.
This function is intended for actors creating a Pango layout using the PangoCairo API.

backend a <clutter-backend>
options  Cairo font options for the backend, or ‘#f’

Since 0.8

**clutter-check-windowing-backend** *(backend_type mchars)*  
* [Function]  
  
  Checks the run-time name of the Clutter windowing system backend, using the symbolic macros like ‘CLUTTER_WINDOWING_WIN32’ or ‘CLUTTER_WINDOWING_X11’.

  This function should be used in conjunction with the compile-time macros inside applications and libraries that are using the platform-specific windowing system API, to ensure that they are running on the correct windowing system; for instance:

  ```c
  ifdef CLUTTER_WINDOWING_X11
  if (clutter_check_windowing_backend (CLUTTER_WINDOWING_X11))
      /&x002A; it is safe to use the clutter_x11_* API &x002A;/
  else
  endif
  ifdef CLUTTER_WINDOWING_WIN32
  if (clutter_check_windowing_backend (CLUTTER_WINDOWING_WIN32))
      /&x002A; it is safe to use the clutter_win32_* API &x002A;/
  else
  endif
  g_error ("Unknown Clutter backend.");
  ```

* backend-type  
  the name of the backend to check

* ret  
  ‘#t’ if the current Clutter windowing system backend is the one checked, and ‘#f’ otherwise

Since 1.10
11 ClutterBinLayout

A simple layout manager

11.1 Overview

<clutter-bin-layout> is a layout manager which implements the following policy:
- the preferred size is the maximum preferred size between all the children of the container
  using the layout;
- each child is allocated in "layers", on on top of the other;
- for each layer there are horizontal and vertical alignment policies.

(The missing figure, bin-layout)

The image shows a <clutter-bin-layout> with three layers: a background <clutter-cairo-texture>, set to fill on both the X and Y axis; a <clutter-texture>, set to center on both the X and Y axis; and a <clutter-rectangle>, set to 'CLUTTER_BIN_ALIGNMENT_END' on both the X and Y axis.

The following code shows how to build a composite actor with a texture and a background, and add controls overlayed on top. The background is set to fill the whole allocation, whilst the texture is centered; there is a control in the top right corner and a label in the bottom, filling out the whole allocated width.

```c
ClutterLayoutManager *manager;
ClutterActor *box;

/* create the layout first */
layout = clutter_bin_layout_new (CLUTTER_BIN_ALIGNMENT_CENTER,
                              CLUTTER_BIN_ALIGNMENT_CENTER);
box = clutter_box_new (layout); /* then the container */

/* we can use the layout object to add actors */
clutter_bin_layout_add (CLUTTER_BIN_LAYOUT (layout), background,
                        CLUTTER_BIN_ALIGNMENT_FILL,
                        CLUTTER_BIN_ALIGNMENT_FILL);
clutter_bin_layout_add (CLUTTER_BIN_LAYOUT (layout), icon,
                        CLUTTER_BIN_ALIGNMENT_CENTER,
                        CLUTTER_BIN_ALIGNMENT_CENTER);

/* align to the bottom left */
clutter_bin_layout_add (CLUTTER_BIN_LAYOUT (layout), label,
                        CLUTTER_BIN_ALIGNMENT_START,
                        CLUTTER_BIN_ALIGNMENT_END);

/* align to the top right */
clutter_bin_layout_add (CLUTTER_BIN_LAYOUT (layout), button,
                        CLUTTER_BIN_ALIGNMENT_END,
                        CLUTTER_BIN_ALIGNMENT_START);
```
<clutter-bin-layout> is available since Clutter 1.2

### 11.2 Usage

**clutter-bin-layout-new**

```plaintext
clutter-bin-layout-new (x_align <clutter-bin-alignment>) (y_align <clutter-bin-alignment>) ⇒ (ret <clutter-layout-manager>)
```

Creates a new `<clutter-bin-layout>` layout manager

- **x-align**  the default alignment policy to be used on the horizontal axis
- **y-align**  the default alignment policy to be used on the vertical axis
- **ret**  the newly created layout manager

Since 1.2

**clutter-bin-layout-set-alignment**

```plaintext
clutter-bin-layout-set-alignment (self <clutter-bin-layout>) (child <clutter-actor>) (x_align <clutter-bin-alignment>) (y_align <clutter-bin-alignment>)
```

Sets the horizontal and vertical alignment policies to be applied to a *child* of *self*

- **self**  a `<clutter-bin-layout>`
- **child**  a child of container.
- **x-align**  the horizontal alignment policy to be used for the *child* inside container
- **y-align**  the vertical alignment policy to be used on the *child* inside container

Since 1.2

**clutter-bin-layout-get-alignment**

```plaintext
clutter-bin-layout-get-alignment (self <clutter-bin-layout>) (child <clutter-actor>) ⇒ (x_align <clutter-bin-alignment>) (y_align <clutter-bin-alignment>)
```

Retrieves the horizontal and vertical alignment policies for a child of *self*

- **self**  a `<clutter-bin-layout>`
- **child**  a child of container.
- **x-align**  return location for the horizontal alignment policy.
- **y-align**  return location for the vertical alignment policy.

Since 1.2
clutter-bin-layout-add (self <clutter-bin-layout>)
    (child <clutter-actor>) (x_align <clutter-bin-alignment>)
    (y_align <clutter-bin-alignment>)

[Function]

add

[Method]

Adds a <clutter-actor> to the container using self and sets the alignment policies for it.

This function is equivalent to clutter-container-add-actor and clutter-layout-manager-child-set-property but it does not require a pointer to the <clutter-container> associated to the <clutter-bin-layout>

self a <clutter-bin-layout>
child a <clutter-actor>
x-align horizontal alignment policy for child
y-align vertical alignment policy for child

Since 1.2


## 12 ClutterBindConstraint

A constraint binding the position or size of an actor

### 12.1 Overview

<clutter-bind-constraint> is a <clutter-constraint> that binds the position or the size of the <clutter-actor> to which it is applied to the the position or the size of another <clutter-actor>, or "source".

An offset can be applied to the constraint, to avoid overlapping. The offset can also be animated. For instance, the following code will set up three actors to be bound to the same origin:

```c
/* source */
rect[0] = clutter_rectangle_new_with_color (&red_color);
clutter_actor_set_position (rect[0], x_pos, y_pos);
clutter_actor_set_size (rect[0], 100, 100);

/* second rectangle */
rect[1] = clutter_rectangle_new_with_color (&green_color);
clutter_actor_set_size (rect[1], 100, 100);
clutter_actor_set_opacity (rect[1], 0);
constraint = clutter_bind_constraint_new (rect[0], CLUTTER_BIND_X, 0.0);
clutter_actor_add_constraint_with_name (rect[1], "green-x", constraint);
constraint = clutter_bind_constraint_new (rect[0], CLUTTER_BIND_Y, 0.0);
clutter_actor_add_constraint_with_name (rect[1], "green-y", constraint);

/* third rectangle */
rect[2] = clutter_rectangle_new_with_color (&blue_color);
clutter_actor_set_size (rect[2], 100, 100);
clutter_actor_set_opacity (rect[2], 0);
constraint = clutter_bind_constraint_new (rect[0], CLUTTER_BIND_X, 0.0);
clutter_actor_add_constraint_with_name (rect[2], "blue-x", constraint);
constraint = clutter_bind_constraint_new (rect[0], CLUTTER_BIND_Y, 0.0);
clutter_actor_add_constraint_with_name (rect[2], "blue-y", constraint);

The following code animates the second and third rectangles to "expand" them horizontally from underneath the first rectangle:

```c
clutter_actor_animate (rect[1], CLUTTER_EASE_OUT_CUBIC, 250,
"@constraints.green-x.offset", 100.0,
"opacity", 255,
NULL);
clutter_actor_animate (rect[2], CLUTTER_EASE_OUT_CUBIC, 250,
"@constraints.blue-x.offset", 200.0,
```
The example above creates eight rectangles and binds them to a rectangle positioned in the center of the stage; when the user presses the center rectangle, the \"offset\" property is animated through the \texttt{clutter-actor-animate} function to lay out the eight rectangles around the center one. Pressing one of the outer rectangles will animate the offset back to 0.

\texttt{<clutter-bind-constraint>} is available since Clutter 1.4

### 12.2 Usage

\texttt{clutter-bind-constraint-new} \texttt{(source <clutter-actor>)} \texttt{(coordinate <clutter-bind-coordinate>)} \texttt{(offset float)} \texttt{⇒ (ret <clutter-constraint>)}

Creates a new constraint, binding a \texttt{<clutter-actor>}’s position to the given \texttt{coordinate} of the position of \texttt{source}

- \texttt{source} the \texttt{<clutter-actor>} to use as the source of the binding, or ‘\#f’.
- \texttt{coordinate} the coordinate to bind
- \texttt{offset} the offset to apply to the binding, in pixels
- \texttt{ret} the newly created \texttt{<clutter-bind-constraint>}

Since 1.4

\texttt{clutter-bind-constraint-set-source} \texttt{(self <clutter-bind-constraint>)} \texttt{(source <clutter-actor>)}

Sets the source \texttt{<clutter-actor>} for the constraint

- \texttt{constraint} a \texttt{<clutter-bind-constraint>}
- \texttt{source} a \texttt{<clutter-actor>}, or ‘\#f’ to unset the source.

Since 1.4

\texttt{clutter-bind-constraint-get-source} \texttt{(self <clutter-bind-constraint>)} \texttt{⇒ (ret <clutter-actor>)}

Retrieves the \texttt{<clutter-actor>} set using \texttt{clutter-bind-constraint-set-source}

- \texttt{constraint} a \texttt{<clutter-bind-constraint>}
- \texttt{ret} a pointer to the source actor.

Since 1.4

\texttt{clutter-bind-constraint-set-offset} \texttt{(self <clutter-bind-constraint>)} \texttt{(offset float)}

Sets the offset to be applied to the constraint
constraint a <clutter-bind-constraint>

offset the offset to apply, in pixels

Since 1.4

clutter-bind-constraint-get-offset [Function]

(self <clutter-bind-constraint>) ⇒ (ret float)

get-offset [Method]

Retrieves the offset set using clutter-bind-constraint-set-offset

constraint a <clutter-bind-constraint>

ret the offset, in pixels

Since 1.4
13 Key Bindings

Pool for key bindings

13.1 Overview

<clutter-binding-pool> is a data structure holding a set of key bindings. Each key binding associates a key symbol (eventually with modifiers) to an action. A callback function is associated to each action.

For a given key symbol and modifier mask combination there can be only one action; for each action there can be only one callback. There can be multiple actions with the same name, and the same callback can be used to handle multiple key bindings.

Actors requiring key bindings should create a new <clutter-binding-pool> inside their class initialization function and then install actions like this:

```c
static void
foo_class_init (FooClass *klass)
{
    ClutterBindingPool *binding_pool;

    binding_pool = clutter_binding_pool_get_for_class (klass);

    clutter_binding_pool_install_action (binding_pool, "move-up",
                                         CLUTTER_Up, 0,
                                         G_CALLBACK (foo_action_move_up),
                                         NULL, NULL);
    clutter_binding_pool_install_action (binding_pool, "move-up",
                                         CLUTTER_KP_Up, 0,
                                         G_CALLBACK (foo_action_move_up),
                                         NULL, NULL);
}
```

The callback has a signature of:

```c
gboolean (* callback) (GObject *instance,
                        const gchar *action_name,
                        guint key_val,
                        ClutterModifierType modifiers,
                        gpointer user_data);
```

The actor should then override the "key-press-event" and use clutter-binding-pool-activate to match a <clutter-key-event> structure to one of the actions:

```c
ClutterBindingPool *pool;

// retrieve the binding pool for the type of the actor
pool = clutter_binding_pool_find (G_OBJECT_TYPE_NAME (actor));
```
/* activate any callback matching the key symbol and modifiers mask of the key event. The returned value can be directly used to signal that the actor has handled the event. */

return clutter_binding_pool_activate (pool, key_event->keyval, key_event->modifier_state, G_OBJECT (actor));

The `clutter-binding-pool-activate` function will return `
#f` if no action for the given key binding was found, if the action was blocked (using `clutter-binding-pool-block-action`) or if the key binding handler returned `
#f`.

`<clutter-binding-pool>` is available since Clutter 1.0

### 13.2 Usage

**clutter-binding-pool-new** (name mchars)  
⇒ (ret `<clutter-binding-pool>`)  

Creates a new `<clutter-binding-pool>` that can be used to store key bindings for an actor. The `name` must be a unique identifier for the binding pool, so that `clutter-binding-pool-find` will be able to return the correct binding pool.

- **name**: the name of the binding pool
- **ret**: the newly created binding pool with the given name. Use `g-object-unref` when done.

Since 1.0

**clutter-binding-pool-get-for-class** (klass `<g-object-class>`)  
⇒ (ret `<clutter-binding-pool>`)  

Retrieves the `<clutter-binding-pool>` for the given `<gobject>` class and, eventually, creates it. This function is a wrapper around `clutter-binding-pool-new` and uses the class type name as the unique name for the binding pool.

Calling this function multiple times will return the same `<clutter-binding-pool>`. A binding pool for a class can also be retrieved using `clutter-binding-pool-find` with the class type name:

```c
pool = clutter_binding_pool_find (G_OBJECT_TYPE_NAME (instance));
```

- **klass**: a `<g-object-class>` pointer
- **ret**: the binding pool for the given class. The returned `<clutter-binding-pool>` is owned by Clutter and should not be freed directly.

Since 1.0

**clutter-binding-pool-find** (name mchars)  
⇒ (ret `<clutter-binding-pool>`)  

Finds the `<clutter-binding-pool>` with `name`.
name the name of the binding pool to find
ret a pointer to the `<clutter-binding-pool>`, or ‘#f’.

Since 1.0

**clutter-binding-pool-find-action** [Function]

```
(self <clutter-binding-pool>) (key_val unsigned-int)
(modifiers <clutter-modifier-type>) ⇒ (ret mchars)
```

**find-action** [Method]
Retrieves the name of the action matching the given key symbol and modifiers bitmask.

pool a `<clutter-binding-pool>`
key_val a key symbol
modifiers a bitmask for the modifiers
ret the name of the action, if found, or ‘#f’. The returned string is owned by the binding pool and should never be modified or freed

Since 1.0

**clutter-binding-pool-remove-action** [Function]

```
(self <clutter-binding-pool>) (key_val unsigned-int)
(modifiers <clutter-modifier-type>)
```

**remove-action** [Method]
Removes the action matching the given key-val, modifiers pair, if any exists.

pool a `<clutter-binding-pool>`
key_val a key symbol
modifiers a bitmask for the modifiers

Since 1.0

**clutter-binding-pool-block-action** [Function]

```
(self <clutter-binding-pool>) (action_name mchars)
```

**block-action** [Method]
Blocks all the actions with name action-name inside pool.

pool a `<clutter-binding-pool>`
action_name an action name

Since 1.0

**clutter-binding-pool-unblock-action** [Function]

```
(self <clutter-binding-pool>) (action_name mchars)
```

**unblock-action** [Method]
Unblocks all the actions with name action-name inside pool.

Unblocking an action does not cause the callback bound to it to be invoked in case `clutter-binding-pool-activate` was called on an action previously blocked with `clutter-binding-pool-block-action`. 
Since 1.0

**activate**

Activates the callback associated to the action that is bound to the *key-val* and *modifiers* pair.

The callback has the following signature:

```c
void (* callback) (GObject *gobject,
                  const gchar *action_name,
                  guint key_val,
                  ClutterModifierType modifiers,
                  gpointer user_data);
```

Where the *gobject* instance is *gobject* and the user data is the one passed when installing the action with `clutter-binding-pool-install-action`.

If the action bound to the *key-val*, *modifiers* pair has been blocked using `clutter-binding-pool-block-action`, the callback will not be invoked, and this function will return `'#f'`.

### Since 1.0

**pool** a `<clutter-binding-pool>`

**action-name** an action name

Since 1.0

**clutter-binding-pool-activate** (*self* `<clutter-binding-pool>*) [Function]

```c
(key_val unsigned-int) (modifiers <clutter-modifier-type>)
(gobject <gobject>) ⇒ (ret bool)
```

**activate** [Method]

Activates the callback associated to the action that is bound to the *key-val* and *modifiers* pair.

The callback has the following signature:

```c
void (* callback) (GObject *gobject,
                  const gchar *action_name,
                  guint key_val,
                  ClutterModifierType modifiers,
                  gpointer user_data);
```

Where the *gobject* instance is *gobject* and the user data is the one passed when installing the action with `clutter-binding-pool-install-action`.

If the action bound to the *key-val*, *modifiers* pair has been blocked using `clutter-binding-pool-block-action`, the callback will not be invoked, and this function will return `'#f'`.

**pool** a `<clutter-binding-pool>`

**key-val** the key symbol

**modifiers** bitmask for the modifiers

**gobject** a `<gobject>`

**ret** `'#t'` if an action was found and was activated

Since 1.0
14 ClutterBlurEffect

A blur effect

14.1 Overview

<clutter-blur-effect> is a sub-class of <clutter-effect> that allows blurring a actor and its contents.

<clutter-blur-effect> is available since Clutter 1.4

14.2 Usage

clutter-blur-effect-new ⇒ (ret <clutter-effect>)  [Function]
Creates a new <clutter-blur-effect> to be used with clutter-actor-add-effect

ret the newly created <clutter-blur-effect> or ‘#f’

Since 1.4
15 ClutterBoxLayout

A layout manager arranging children on a single line

15.1 Overview

The <clutter-box-layout> is a <clutter-layout-manager> implementing the following layout policy:

- all children are arranged on a single line;
- the axis used is controlled by the <"vertical"> boolean property;
- the order of the packing is determined by the <"pack-start"> boolean property;
- each child will be allocated to its natural size or, if set to expand, the available size;
- if a child is set to fill on either (or both) axis, its allocation will match all the available size; the fill layout property only makes sense if the expand property is also set;
- if a child is set to expand but not to fill then it is possible to control the alignment using the X and Y alignment layout properties.
- if the <"homogeneous"> boolean property is set, then all widgets will get the same size, ignoring expand settings and the preferred sizes

(The missing figure, box-layout)

The image shows a <clutter-box-layout> with the <"vertical"> property set to ‘#f’.

It is possible to control the spacing between children of a <clutter-box-layout> by using clutter-box-layout-set-spacing.

In order to set the layout properties when packing an actor inside a <clutter-box-layout> you should use the clutter-box-layout-pack function.

<clutter-box-layout> is available since Clutter 1.2

15.2 Usage

clutter-box-layout-new ⇒ (ret <clutter-layout-manager>)  [Function]

Creates a new <clutter-box-layout> layout manager

ret the newly created <clutter-box-layout>

Since 1.2
clutter-box-layout-set-pack-start (self <clutter-box-layout>)  [Function]
  (pack_start bool)
set-pack-start  [Method]
  Sets whether children of layout should be layed out by appending them or by prepend-ing them
  layout a <clutter-box-layout>
  pack-start '#t' if the layout should pack children at the beginning of the layout
Since 1.2

clutter-box-layout-get-pack-start (self <clutter-box-layout>)  [Function]
  ⇒ (ret bool)
get-pack-start  [Method]
  Retrieves the value set using clutter-box-layout-set-pack-start
  layout a <clutter-box-layout>
  ret '#t' if the <clutter-box-layout> should pack children at the beginning of the layout, and '#f' otherwise
Since 1.2

clutter-box-layout-set-spacing (self <clutter-box-layout>)  [Function]
  (spacing unsigned-int)
set-spacing  [Method]
  Sets the spacing between children of layout
  layout a <clutter-box-layout>
  spacing the spacing between children of the layout, in pixels
Since 1.2

clutter-box-layout-get-spacing (self <clutter-box-layout>)  [Function]
  ⇒ (ret unsigned-int)
get-spacing  [Method]
  Retrieves the spacing set using clutter-box-layout-set-spacing
  layout a <clutter-box-layout>
  ret the spacing between children of the <clutter-box-layout>
Since 1.2

clutter-box-layout-set-vertical (self <clutter-box-layout>)  [Function]
  (vertical bool)
set-vertical  [Method]
  Sets whether layout should arrange its children vertically alongside the Y axis, instead of horizontally alongside the X axis
  layout a <clutter-box-layout>
  vertical '#t' if the layout should be vertical
Since 1.2
**Chapter 15: ClutterBoxLayout**

**clutter-box-layout-get-vertical** *(self <clutter-box-layout>)*  
* [Function]  
  \[
  \Rightarrow (ret bool)
  \]

**get-vertical**  
* [Method]  
  Retrieves the orientation of the layout as set using the clutter-box-layout-set-vertical function

- *layout* a <clutter-box-layout>
- *ret* ‘#t’ if the <clutter-box-layout> is arranging its children vertically, and ‘#f’ otherwise

Since 1.2

**clutter-box-layout-set-homogeneous**  
* (self <clutter-box-layout>) (homogeneous bool)  
* [Function]  

**set-homogeneous**  
* [Method]  
  Sets whether the size of layout children should be homogeneous

- *layout* a <clutter-box-layout>
- *homogeneous* ‘#t’ if the layout should be homogeneous

Since 1.4

**clutter-box-layout-get-homogeneous**  
* (self <clutter-box-layout>) ⇒ (ret bool)  
* [Function]  

**get-homogeneous**  
* [Method]  
  Retrieves if the children sizes are allocated homogeneously.

- *layout* a <clutter-box-layout>
- *ret* ‘#t’ if the <clutter-box-layout> is arranging its children homogeneously, and ‘#f’ otherwise

Since 1.4

**clutter-box-layout-pack** *(self <clutter-box-layout>)*  
* (actor <clutter-actor>) (expand bool) (x_fill bool) (y_fill bool)
  (x_align <clutter-box-alignment>)  
  (y_align <clutter-box-alignment>)  
* [Function]  

**pack**  
* [Method]  
  Packs actor inside the <clutter-container> associated to layout and sets the layout properties

- *layout* a <clutter-box-layout>
- *actor* a <clutter-actor>
- *expand* whether the actor should expand
- *x-fill* whether the actor should fill horizontally
- *y-fill* whether the actor should fill vertically
- *x-align* the horizontal alignment policy for actor
The vertical alignment policy for actor

Since 1.2

```scala
clutter-box-layout-set-alignment (self <clutter-box-layout>) (actor <clutter-actor>) (x_align <clutter-box-alignment>) (y_align <clutter-box-alignment>)
```

**set-alignment** [Method]
Sets the horizontal and vertical alignment policies for actor inside layout

- `layout`: a `<clutter-box-layout>`
- `actor`: a `<clutter-actor>` child of `layout`
- `x-align`: Horizontal alignment policy for actor
- `y-align`: Vertical alignment policy for actor

Since 1.2

```scala
clutter-box-layout-get-alignment (self <clutter-box-layout>) (actor <clutter-actor>) ⇒ (x_align <clutter-box-alignment>) (y_align <clutter-box-alignment>)
```

**get-alignment** [Method]
Retrieves the horizontal and vertical alignment policies for actor as set using `clutter-box-layout-pack` or `clutter-box-layout-set-alignment`

- `layout`: a `<clutter-box-layout>`
- `actor`: a `<clutter-actor>` child of `layout`
- `x-align`: return location for the horizontal alignment policy.
- `y-align`: return location for the vertical alignment policy.

Since 1.2

```scala
clutter-box-layout-set-expand (self <clutter-box-layout>) (actor <clutter-actor>) (expand bool)
```

**set-expand** [Method]
Sets whether actor should expand inside layout

- `layout`: a `<clutter-box-layout>`
- `actor`: a `<clutter-actor>` child of `layout`
- `expand`: whether actor should expand

Since 1.2

```scala
clutter-box-layout-get-expand (self <clutter-box-layout>) (actor <clutter-actor>) ⇒ (ret bool)
```

**get-expand** [Method]
Retrieves whether actor should expand inside layout

- `layout`: a `<clutter-box-layout>`
- `actor`: a `<clutter-actor>` child of `layout`
clutter-box-layout-set-fill (self <clutter-box-layout>) (actor <clutter-actor>) (x.fill bool) (y.fill bool)

set-fill
Sets the horizontal and vertical fill policies for actor inside layout

layout a <clutter-box-layout>
actor a <clutter-actor> child of layout
x-fill whether actor should fill horizontally the allocated space
y-fill whether actor should fill vertically the allocated space

clutter-box-layout-get-fill (self <clutter-box-layout>) (actor <clutter-actor>) ⇒ (x.fill bool) (y.fill bool)

get-fill
Retrieves the horizontal and vertical fill policies for actor as set using clutter-box-layout-pack or clutter-box-layout-set-fill

layout a <clutter-box-layout>
actor a <clutter-actor> child of layout
x-fill return location for the horizontal fill policy.
y-fill return location for the vertical fill policy.

clutter-box-layout-set-easing-mode (self <clutter-box-layout>) (mode unsigned-long)

set-easing-mode
Sets the easing mode to be used by layout when animating changes in layout properties

Use clutter-box-layout-set-use-animations to enable and disable the animations

layout a <clutter-box-layout>
mode an easing mode, either from <clutter-animation-mode> or a logical id from clutter-alpha-register-func

clutter-box-layout-get-easing-mode (self <clutter-box-layout>) ⇒ (ret unsigned-long)

get-easing-mode
Retrieves the easing mode set using clutter-box-layout-set-easing-mode

layout a <clutter-box-layout>
ret an easing mode

Since 1.2
16 ClutterBrightnessContrastEffect

Increase/decrease brightness and/or contrast of actor.

16.1 Overview
<clutter-brightness-contrast-effect> is a sub-class of <clutter-effect> that changes the overall brightness of a <clutter-actor>.

<clutter-brightness-contrast-effect> is available since Clutter 1.10

16.2 Usage
Chapter 17: ClutterCairoTexture

17 ClutterCairoTexture

Texture with Cairo integration

17.1 Overview

<clutter-cairo-texture> is a <clutter-texture> that displays the contents of a Cairo context. The <clutter-cairo-texture> actor will create a Cairo image surface which will then be uploaded to a GL texture when needed.

Since <clutter-cairo-texture> uses a Cairo image surface internally all the drawing operations will be performed in software and not using hardware acceleration. This can lead to performance degradation if the contents of the texture change frequently.

In order to use a <clutter-cairo-texture> you should connect to the <"draw"> signal; the signal is emitted each time the <clutter-cairo-texture> has been told to invalidate its contents, by using clutter-cairo-texture-invalid-rectangle or its sister function, clutter-cairo-texture-invalidate.

Each callback to the <"draw"> signal will receive a <cairo-t> context which can be used for drawing; the Cairo context is owned by the <clutter-cairo-texture> and should not be destroyed explicitly.

<clutter-cairo-texture> is available since Clutter 1.0.

17.2 Usage

clutter-cairo-texture-new (width unsigned-int) (height unsigned-int) ⇒ (ret <clutter-actor>)

 Creates a new <clutter-cairo-texture> actor, with a surface of width by height pixels.

 width the width of the surface
 height the height of the surface
 ret the newly created <clutter-cairo-texture> actor

Since 1.0

clutter-cairo-texture-invalidate (self <clutter-cairo-texture>)

 invalidate

 Invalidates the whole surface of a <clutter-cairo-texture>. This function will cause the <"draw"> signal to be emitted.

See also: clutter-cairo-texture-invalidate-rectangle

 self a <clutter-cairo-texture>

Since 1.8
clutter-cairo-texture-clear (self <clutter-cairo-texture>)  [Function]
clear  [Method]
    Clears self’s internal drawing surface, so that the next upload will replace the previous contents of the <clutter-cairo-texture> rather than adding to it.
    Calling this function from within a "draw" signal handler will clear the invalidated area.
self  a <clutter-cairo-texture>
Since 1.0

clutter-cairo-set-source-color (cr cairo-t)  [Function]
    (color <clutter-color>)
    Utility function for setting the source color of cr using a <clutter-color>. This function is the equivalent of:

        cairo_set_source_rgb (cr,
            color->red / 255.0,
            color->green / 255.0,
            color->blue / 255.0,
            color->alpha / 255.0);

    cr  a Cairo context
    color  a <clutter-color>
Since 1.0
18 ClutterCanvas

Content for 2D painting

18.1 Overview

The `<clutter-canvas>` class is a `<clutter-content>` implementation that allows drawing using the Cairo API on a 2D surface.

In order to draw on a `<clutter-canvas>`, you should connect a handler to the `<"draw">` signal; the signal will receive a `<cairo-t>` context that can be used to draw. `<clutter-canvas>` will emit the `<"draw">` signal when invalidated using `clutter-content-invalidate`.

`<clutter-canvas>` is available since Clutter 1.10.

18.2 Usage

`clutter-canvas-new ⇒ (ret <clutter-content>)` [Function]

Creates a new instance of `<clutter-canvas>`. You should call `clutter-canvas-set-size` to set the size of the canvas.

You should call `clutter-content-invalidate` every time you wish to draw the contents of the canvas.

`ret` The newly allocated instance of `<clutter-canvas>`. Use `g-object-unref` when done.

Since 1.10

`clutter-canvas-set-size (self <clutter-canvas>) (width int) (height int)` [Function]

`set-size` [Method]

Sets the size of the canvas.

This function will cause the canvas to be invalidated.

`canvas` a `<clutter-canvas>`

`width` the width of the canvas, in pixels

`height` the height of the canvas, in pixels

Since 1.10
19 ClutterChildMeta

Wrapper for actors inside a container

19.1 Overview

<clutter-child-meta> is a wrapper object created by <clutter-container> implementa-
tions in order to store child-specific data and properties.

A <clutter-child-meta> wraps a <clutter-actor> inside a <clutter-container>.

<clutter-child-meta> is available since Clutter 0.8

19.2 Usage

clutter-child-meta-get-container (self <clutter-child-meta>) ⇒ (ret <clutter-container>) [Function]
get-container [Method]

Retrieves the container using data

Retrieves the actor wrapped by data

Since 0.8

clutter-child-meta-get-actor (self <clutter-child-meta>) ⇒ (ret <clutter-actor>) [Function]
get-actor [Method]

Since 0.8
20 ClutterClickAction

Action for clickable actors

20.1 Overview

<clutter-click-action> is a sub-class of <clutter-action> that implements the logic for clickable actors, by using the low level events of <clutter-actor>, such as <"button-press-event"> and <"button-release-event">, to synthesize the high level <"clicked"> signal.

To use <clutter-click-action> you just need to apply it to a <clutter-actor> using clutter-actor-add-action and connect to the <"clicked"> signal:

```c
ClutterAction *action = clutter_click_action_new();
clutter_actor_add_action (actor, action);

g_signal_connect (action, "clicked", G_CALLBACK (on_clicked), NULL);
```

<clutter-click-action> also supports long press gestures: a long press is activated if the pointer remains pressed within a certain threshold (as defined by the <"long-press-threshold"> property) for a minimum amount of time (as the defined by the <"long-press-duration"> property). The <"long-press"> signal is emitted multiple times, using different <clutter-long-press-state> values; to handle long presses you should connect to the <"long-press"> signal and handle the different states:

```c
static gboolean
on_long_press (ClutterClickAction *action,
               ClutterActor *actor,
               ClutterLongPressState state)
{
    switch (state)
    {
    case CLUTTER_LONG_PRESS_QUERY:
        /* return TRUE if the actor should support long press
gestures, and FALSE otherwise; this state will be
emitted on button presses
        */
        return TRUE;
    case CLUTTER_LONG_PRESS_ACTIVATE:
        /* this state is emitted if the minimum duration has
been reached without the gesture being cancelled.
        */
        return TRUE;
    }
}
```
case CLUTTER_LONG_PRESS_CANCEL:
  /* this state is emitted if the long press was cancelled;
   * for instance, the pointer went outside the actor or the
   * allowed threshold, or the button was released before
   * the minimum duration was reached. the return value is
   * not used */
  return FALSE;
}

<clutter-click-action> is available since Clutter 1.4

20.2 Usage

<clutter-click-action-new> ⇒ (ret <clutter-action>) [Function]
Creates a new <clutter-click-action> instance

  ret the newly created <clutter-click-action>

Since 1.4

<clutter-click-action-get-button> (self <clutter-click-action>) ⇒ (ret unsigned-int) [Function]
get-button [Method]
Retrieves the button that was pressed.

  action a <clutter-click-action>
  ret the button value

Since 1.4

<clutter-click-action-get-state> (self <clutter-click-action>) ⇒ (ret <clutter-modifier-type>) [Function]
get-state [Method]
Retrieves the modifier state of the click action.

  action a <clutter-click-action>
  ret the modifier state parameter, or 0

Since 1.6

<clutter-click-action-get-coords> (self <clutter-click-action>) ⇒ (press_x float) (press_y float) [Function]
get-coords [Method]
Retrieves the screen coordinates of the button press.

  action a <clutter-click-action>
  press-x return location for the X coordinate, or ‘#f’.
  press-y return location for the Y coordinate, or ‘#f’.

Since 1.8
clutter-click-action-release (self <clutter-click-action>)  [Function]
release  [Method]
    Emulates a release of the pointer button, which ungrabs the pointer and unsets the "pressed" state.
    This function will also cancel the long press gesture if one was initiated.
    This function is useful to break a grab, for instance after a certain amount of time has passed.
    action  a <clutter-click-action>

Since 1.4
21 ClutterClone

An actor that displays a clone of a source actor

21.1 Overview

\textit{<clutter-clone>} is a \textit{<clutter-actor>} which draws with the paint function of another actor, scaled to fit its own allocation.

\textit{<clutter-clone>} can be used to efficiently clone any other actor.

This is different from \textit{clutter-texture-new-from-actor} which requires support for FBOs in the underlying GL implementation.

\textit{<clutter-clone>} is available since Clutter 1.0

21.2 Usage

\begin{verbatim}
clutter-clone-new (source <clutter-actor>) ⇒ (ret <clutter-actor>)
cretes a new <clutter-actor> which clones source/

source a <clutter-actor>, or ‘#f’
ret the newly created <clutter-clone>
Since 1.0

clutter-clone-set-source (self <clutter-clone>) (source <clutter-actor>)
set-source [Method]
Sets source as the source actor to be cloned by self.

self a <clutter-clone>
source a <clutter-actor>, or ‘#f’.
Since 1.0

clutter-clone-get-source (self <clutter-clone>) ⇒ (ret <clutter-actor>)
get-source [Method]
Retrieves the source <clutter-actor> being cloned by self.

self a <clutter-clone>
ret the actor source for the clone.
Since 1.0
\end{verbatim}
22 Colors

Color management and manipulation.

22.1 Overview

<clutter-color> is a simple type for representing colors in Clutter.

A <clutter-color> is expressed as a 4-tuple of values ranging from zero to 255, one for each color channel plus one for the alpha.

The alpha channel is fully opaque at 255 and fully transparent at 0.

22.2 Usage

clutter-color-new(red unsigned-int8) (green unsigned-int8) (blue unsigned-int8) (alpha unsigned-int8) ⇒ (ret <clutter-color>)

Creates a new <clutter-color> with the given values.

red  red component of the color, between 0 and 255
green green component of the color, between 0 and 255
blue  blue component of the color, between 0 and 255
alpha alpha component of the color, between 0 and 255
ret   the newly allocated color. Use clutter-color-free when done.

Since 0.8.4

clutter-color-get-static(color <clutter-static-color>) ⇒ (ret <clutter-color>)

Retrieves a static color for the given color name

Static colors are created by Clutter and are guaranteed to always be available and valid

color  the named global color
ret    a pointer to a static color; the returned pointer is owned by Clutter and it should never be modified or freed

Since 1.6

clutter-color-from-string(name mchars) ⇒ (ret scm)

Parses a string definition of a color, filling the "red", (structfield "alpha") channels of color.

The color is not allocated.

The format of str can be either one of:

•
•
•
a standard name (as taken from the X11 rgb.txt file)
an hexadecimal value in the form: ‘&#x0023;rgb’, ‘&#x0023;rrggbb’, ‘&#x0023;rgba’
or ‘&#x0023;rrggbbaa’
a RGB color in the form: ‘rgb(r, g, b)’
a RGB color in the form: ‘rgba(r, g, b, a)’
a HSL color in the form: ‘hsl(h, s, l)’
a HSL color in the form: ‘hsla(h, s, l, a)’
where ‘r’, ‘g’, ‘b’ and ‘a’ are (respectively) the red, green, blue color intensities and
the opacity. The ‘h’, ‘s’ and ‘l’ are (respectively) the hue, saturation and luminance
values.
In the rgb and rgba formats, the ‘r’, ‘g’, and ‘b’ values are either integers between
0 and 255, or percentage values in the range between 0% and 100%; the percentages
require the ‘%’ character. The ‘a’ value, if specified, can only be a floating point value
between 0.0 and 1.0.
In the hls and hlsa formats, the ‘h’ value (hue) it’s an angle between 0 and 360.0
degrees; the ‘l’ and ‘s’ values (luminance and saturation) are a floating point value
between 0.0 and 1.0. The ‘a’ value, if specified, can only be a floating point value
between 0.0 and 1.0.
Whitespace inside the definitions is ignored; no leading whitespace is allowed.
If the alpha component is not specified then it is assumed to be set to be fully opaque.

Since 1.0

clutter-color-to-string (self <clutter-color>) ⇒ (ret mchars) [Function]
Returns a textual specification of color in the hexadecimal form ‘&#x0023;rrggbbaa’,
where ‘r’, ‘g’, ‘b’ and ‘a’ are hexadecimal digits representing the red, green, blue and
alpha components respectively.

Since 0.2

clutter-color-from-hls (self <clutter-color>) (hue float) (luminance float) (saturation float) [Function]
Converts a color expressed in HLS (hue, luminance and saturation) values into a
<clutter-color>.

Since 0.2

clutter-color-from-hls (self <clutter-color>) (hue float) (luminance float) (saturation float) [Function]
Converts a color expressed in HLS (hue, luminance and saturation) values into a
<clutter-color>.

Since 0.2

clutter-color-from-hls (self <clutter-color>) (hue float) (luminance float) (saturation float) [Function]
Converts a color expressed in HLS (hue, luminance and saturation) values into a
<clutter-color>.

Since 0.2

clutter-color-from-hls (self <clutter-color>) (hue float) (luminance float) (saturation float) [Function]
Converts a color expressed in HLS (hue, luminance and saturation) values into a
<clutter-color>.
luminance  luminance value, in the 0 .. 1 range
saturation  saturation value, in the 0 .. 1 range

clutter-color-to-hls (self <clutter-color>) ⇒ (hue float)  [Function]
             (luminance float) (saturation float)
Converts color to the HLS format.
The hue value is in the 0 .. 360 range. The luminance and saturation values are in
the 0 .. 1 range.
color           a <clutter-color>
hue             return location for the hue value or ‘#f’.
luminance       return location for the luminance value or ‘#f’.
saturation      return location for the saturation value or ‘#f’.

clutter-color-from-pixel (self <clutter-color>)  [Function]
             (pixel unsigned-int32)
Converts pixel from the packed representation of a four 8 bit channel color to a
<clutter-color>.
color          return location for a <clutter-color>.
pixel           a 32 bit packed integer containing a color

clutter-color-to-pixel (self <clutter-color>)  [Function]
             ⇒ (ret unsigned-int32)
Converts color into a packed 32 bit integer, containing all the four 8 bit channels used
by <clutter-color>.
color           a <clutter-color>
ret              a packed color

clutter-color-add (self <clutter-color>) (b <clutter-color>)  [Function]
             (result <clutter-color>)
Adds a to b and saves the resulting color inside result.
The alpha channel of result is set as as the maximum value between the alpha channels
of a and b.
a           a <clutter-color>
b           a <clutter-color>
result   return location for the result.

clutter-color-subtract (self <clutter-color>)  [Function]
             (b <clutter-color>) (result <clutter-color>)
Subtracts b from a and saves the resulting color inside result.
This function assumes that the components of a are greater than the components of
b; the result is, otherwise, undefined.
The alpha channel of result is set as the minimum value between the alpha channels
of a and b.
Chapter 22: Colors

\begin{itemize}
\item \texttt{a} \quad \texttt{a \ <clutter-color>}
\item \texttt{b} \quad \texttt{a \ <clutter-color>}
\item \texttt{result} \quad \texttt{return location for the result.}
\end{itemize}

\begin{description}
\item \texttt{clutter-color-lighten} (\textit{self} <clutter-color>)
\item \texttt{(result} <clutter-color>)
\item \texttt{Lightens} \textit{color} \texttt{by a fixed amount, and saves the changed color in} \texttt{result.}
\item \texttt{color} \quad \texttt{a \ <clutter-color>}
\item \texttt{result} \quad \texttt{return location for the lighter color.}
\end{description}

\begin{description}
\item \texttt{clutter-color-darken} (\textit{self} <clutter-color>)
\item \texttt{(result} <clutter-color>)
\item \texttt{Darkens} \textit{color} \texttt{by a fixed amount, and saves the changed color in} \texttt{result.}
\item \texttt{color} \quad \texttt{a \ <clutter-color>}
\item \texttt{result} \quad \texttt{return location for the darker color.}
\end{description}

\begin{description}
\item \texttt{clutter-color-shade} (\textit{self} <clutter-color>) \texttt{(factor} double \texttt{)}
\item \texttt{(result} <clutter-color>)
\item \texttt{Shades} \textit{color} \texttt{by} \textit{factor} \texttt{and saves the modified color into} \texttt{result.}
\item \texttt{color} \quad \texttt{a \ <clutter-color>}
\item \texttt{factor} \quad \texttt{the shade factor to apply}
\item \texttt{result} \quad \texttt{return location for the shaded color.}
\end{description}

\begin{description}
\item \texttt{clutter-color-interpolate} (\textit{self} <clutter-color>)
\item \texttt{(final} <clutter-color>) \texttt{(progress} double \texttt{)}\texttt{(result} <clutter-color>)
\item \texttt{Interpolates} \texttt{between} \textit{initial} \texttt{and} \textit{final} \texttt{<clutter-color>s} \texttt{using} \texttt{progress}
\item \texttt{initial} \quad \texttt{the initial} <clutter-color>
\item \texttt{final} \quad \texttt{the final} <clutter-color>
\item \texttt{progress} \quad \texttt{the interpolation progress}
\item \texttt{result} \quad \texttt{return location for the interpolation.}
\end{description}

\texttt{Since 1.6}

\begin{description}
\item \texttt{clutter-value-set-color} (\textit{value} <gvalue>)
\item \texttt{(color} <clutter-color>)
\item \texttt{Sets} \textit{value} \texttt{to} \textit{color.}
\item \texttt{value} \quad \texttt{a \ <gvalue> initialized to} \texttt{<clutter-type-color>}
\item \texttt{color} \quad \texttt{the color to set}
\end{description}

\texttt{Since 0.8.4}
clutter-value-get-color (value <gvalue>)
⇒ (ret <clutter-color>)
Gets the <clutter-color> contained in value.

value    a <gvalue> initialized to <clutter-type-color>
ret      the color inside the passed <gvalue>.

Since 0.8.4
23 ClutterColorizeEffect

A colorization effect

23.1 Overview

<clutter-colorize-effect> is a sub-class of <clutter-effect> that colorizes an actor
with the given tint.

<clutter-colorize-effect> is available since Clutter 1.4

23.2 Usage

clutter-colorize-effect-new (self <clutter-color>) ⇒ (ret <clutter-effect>)

Creates a new <clutter-colorize-effect> to be used with clutter-actor-add-effect

tint the color to be used

ret the newly created <clutter-colorize-effect> or ‘#f’

Since 1.4

clutter-colorize-effect-set-tint (self <clutter-colorize-effect>) (tint <clutter-color>)

set-tint a <clutter-colorize-effect>

tint the color to be used

Since 1.4

clutter-colorize-effect-get-tint (self <clutter-colorize-effect>) (tint <clutter-color>)

get-tint a <clutter-colorize-effect>

tint return location for the color used.

Since 1.4
Chapter 24: ClutterConstraint

24 ClutterConstraint

Abstract class for constraints on position or size

24.1 Overview
<clutter-constraint> is a base abstract class for modifiers of a <clutter-actor> position or size.

A <clutter-constraint> sub-class should contain the logic for modifying the position or size of the <clutter-actor> to which it is applied, by updating the actor’s allocation. Each <clutter-constraint> can change the allocation of the actor to which they are applied by overriding the update-allocation virtual function.

24.2 Using Constraints
Constraints can be used with fixed layout managers, like <clutter-fixed-layout>, or with actors implicitly using a fixed layout manager, like <clutter-group> and <clutter-stage>.

Constraints provide a way to build user interfaces by using relations between <clutter-actor>s, without explicit fixed positioning and sizing, similarly to how fluid layout managers like <clutter-box-layout> and <clutter-table-layout> lay out their children.

Constraints are attached to a <clutter-actor>, and are available for inspection using clutter-actor-get-constraints.

Clutter provides different implementation of the <clutter-constraint> abstract class, for instance:

<clutter-bind-constraint>
this constraint binds the X, Y, width or height of an actor to the corresponding position or size of a source actor; it can also apply an offset.

<clutter-snap-constraint>
this constraint "snaps" together the edges of two <clutter-actor>s; if an actor uses two constraints on both its horizontal or vertical edges then it can also expand to fit the empty space.

The example below uses various <clutter-constraint>s to lay out three actors on a resizable stage. Only the central actor has an explicit size, and no actor has an explicit position.

1. The <clutter-rectangle> with <"name">layerA</"name> is explicitly sized to 100 pixels by 25 pixels, and it’s added to the <clutter-stage>;
2. two <clutter-align-constraint>s are used to anchor layerA to the center of the stage, by using 0.5 as the alignment <"factor"> on both the X and Y axis.
3. the <clutter-rectangle> with <"name">layerB</"name> is added to the <clutter-stage> with no explicit size;
4. the <"x"> and <"width"> of layerB are bound to the same properties of layerA using two <clutter-bind-constraint> objects, thus keeping layerB aligned to layerA;
5. the top edge of *layerB* is snapped together with the bottom edge of *layerA*; the bottom edge of *layerB* is also snapped together with the bottom edge of the `<clutter-stage>`; an offset is given to the two `<clutter-snap-constraint>`s to allow for some padding; since *layerB* is snapped between two different `<clutter-actor>`s, its height is stretched to match the gap;

6. the `<clutter-rectangle>` with `<"name">layerC` mirrors *layerB*, snapping the top edge of the `<clutter-stage>` to the top edge of *layerC* and the top edge of *layerA* to the bottom edge of *layerC*;

You can try resizing interactively the `<clutter-stage>` and verify that the three `<clutter-actor>`s maintain the same position and size relative to each other, and to the `<clutter-stage>`.

It’s important to note that Clutter does not avoid loops or competing constraints; if two or more `<clutter-constraint>`s are operating on the same positional or dimensional attributes of an actor, or if the constraints on two different actors depend on each other, then the behavior is undefined.

### 24.3 Implementing a ClutterConstraint

Creating a sub-class of `<clutter-constraint>` requires the implementation of the `update-allocation` virtual function.

The `update-allocation` virtual function is called during the allocation sequence of a `<clutter-actor>`, and allows any `<clutter-constraint>` attached to that actor to modify the allocation before it is passed to the `allocate` implementation.

The `<clutter-actor-box>` passed to the `update-allocation` implementation contains the original allocation of the `<clutter-actor>`, plus the eventual modifications applied by the other `<clutter-constraint>`s.

Constraints are queried in the same order as they were applied using `clutter-actor-add-constraint` or `clutter-actor-add-constraint-with-name`.

It is not necessary for a `<clutter-constraint>` sub-class to chain up to the parent’s implementation.

If a `<clutter-constraint>` is parametrized - i.e. if it contains properties that affect the way the constraint is implemented - it should call `clutter-actor-queue-relayout` on the actor to which it is attached to whenever any parameter is changed. The actor to which it is attached can be recovered at any point using `clutter-actor-meta-get-actor`.

 `<clutter-constraint>` is available since Clutter 1.4

### 24.4 Usage
25 ClutterContainer

An interface for container actors

25.1 Overview

<clutter-container> is an interface implemented by <clutter-actor>, and it provides some common API for notifying when a child actor is added or removed, as well as the infrastructure for accessing child properties through <clutter-child-meta>.

Until Clutter 1.10, the <clutter-container> interface was also the public API for implementing container actors; this part of the interface has been deprecated: <clutter-container> has a default implementation which defers to <clutter-actor> the child addition and removal, as well as the iteration. See the documentation of <clutter-container-_iface> for the list of virtual functions that should be overridden.

25.2 Usage

clutter-container-child-notify (self <clutter-container>) (child <clutter-actor>) (pspec <gparam>)

child-notify [Method]
Calls the clutter-container-iface.child-notify virtual function of <clutter-container>. The default implementation will emit the "child-notify" signal.

container a <clutter-container>
child a <clutter-actor>
pspec a <gparam>

Since 1.6

clutter-container-create-child-meta (self <clutter-container>) (actor <clutter-actor>)

create-child-meta [Method]
Creates the <clutter-child-meta> wrapping actor inside the container, if the "child-meta-type" class member is not set to ‘G_TYPE_INVALID’. This function is only useful when adding a <clutter-actor> to a <clutter-container> implementation outside of the <clutter-container>::add virtual function implementation.

Applications should not call this function.

container a <clutter-container>
actor a <clutter-actor>

Since 1.2

clutter-container-get-child-meta (self <clutter-container>) (actor <clutter-actor>) ⇒ (ret <clutter-child-meta>)

get-child-meta [Method]
Retrieves the <clutter-child-meta> which contains the data about the container specific state for actor.
container | a `<clutter-container>`
--- | ---
actor | a `<clutter-actor>` that is a child of container.
ret | the `<clutter-child-meta>` for the actor child of container or ‘#f’ if the specified actor does not exist or the container is not configured to provide `<clutter-child-meta>`s.

Since 0.8
26 ClutterContent

Delegate for painting the content of an actor

26.1 Overview

<clutter-content> is an interface to implement types responsible for painting the content of a <clutter-actor>.

Multiple actors can use the same <clutter-content> instance, in order to share the resources associated with painting the same content.

<clutter-content> is available since Clutter 1.10.

26.2 Usage

clutter-content-get-preferred-size (self <clutter-content>) ⇒ (ret bool) (width float) (height float)

get-preferred-size

Retrieves the natural size of the content, if any.

The natural size of a <clutter-content> is defined as the size the content would have regardless of the allocation of the actor that is painting it, for instance the size of an image data.

content a <clutter-content>

width return location for the natural width of the content.

height return location for the natural height of the content.

ret ‘#t’ if the content has a preferred size, and ‘#f’ otherwise

Since 1.10

clutter-content-invalidate (self <clutter-content>)

invalidate

Invalidates a <clutter-content>.

This function should be called by <clutter-content> implementations when they change the way a the content should be painted regardless of the actor state.

content a <clutter-content>

Since 1.10
27 ClutterDeformEffect

A base class for effects deforming the geometry of an actor

27.1 Overview
<clutter-deform-effect> is an abstract class providing all the plumbing for creating effects that result in the deformation of an actor’s geometry.

<clutter-deform-effect> uses offscreen buffers to render the contents of a <clutter-actor> and then the Cogl vertex buffers API to submit the geometry to the GPU.

27.2 Implementing ClutterDeformEffect
Sub-classes of <clutter-deform-effect> should override the deform-vertex virtual function; this function is called on every vertex that needs to be deformed by the effect. Each passed vertex is an in-out parameter that initially contains the position of the vertex and should be modified according to a specific deformation algorithm.

<clutter-deform-effect> is available since Clutter 1.4

27.3 Usage

clutter-deform-effect-set-n-tiles [Function]
(self <clutter-deform-effect>) (x_tiles unsigned-int)
(y_tiles unsigned-int)
set-n-tiles [Method]
Sets the number of horizontal and vertical tiles to be used when applying the effect
More tiles allow a finer grained deformation at the expenses of computation

effect a <clutter-deform-effect>
x-tiles number of horizontal tiles
y-tiles number of vertical tiles

Since 1.4

clutter-deform-effect-get-n-tiles [Function]
(self <clutter-deform-effect>) ⇒ (x_tiles unsigned-int)
(y_tiles unsigned-int)
get-n-tiles [Method]
Retrieves the number of horizontal and vertical tiles used to sub-divide the actor’s geometry during the effect

effect a <clutter-deform-effect>
x-tiles return location for the number of horizontal tiles, or ‘#f’.
y-tiles return location for the number of vertical tiles, or ‘#f’.

Since 1.4
clutter-deform-effect-invalidate  

(self <clutter-deform-effect>)

invalidate  

Invalidates the effect’s vertices and, if it is associated to an actor, it will queue a redraw

 Since 1.4
28 ClutterDesaturateEffect

A desaturation effect

28.1 Overview

<clutter-desaturate-effect> is a sub-class of <clutter-effect> that desaturates the color of an actor and its contents. The strenght of the desaturation effect is controllable and animatable through the "factor" property.

<clutter-desaturate-effect> is available since Clutter 1.4

28.2 Usage

clutter-desaturate-effect-new (factor double) [Function]
⇒ (ret <clutter-effect>)

Creates a new <clutter-desaturate-effect> to be used with clutter-actor-add-effect

factor the desaturation factor, between 0.0 and 1.0
ret the newly created <clutter-desaturate-effect> or ‘#f’

Since 1.4
29 ClutterDeviceManager

Maintains the list of input devices

29.1 Overview

<clutter-device-manager> is a singleton object, owned by Clutter, which maintains the list of <clutter-input-device>s.

Depending on the backend used by Clutter it is possible to use the <"device-added"> and <"device-removed"> to monitor addition and removal of devices.

<clutter-device-manager> is available since Clutter 1.2

29.2 Usage

clutter-device-manager-list-devices [Function]
  (self <clutter-device-manager>) ⇒ (ret gslist-of)
list-devices [Method]
  Lists all currently registered input devices
device-manager
  a <clutter-device-manager>
ret
  a newly allocated list of <clutter-input-device> objects. Use g-slist-free to deallocate it when done.

Since 1.2

clutter-device-manager-peek-devices [Function]
  (self <clutter-device-manager>) ⇒ (ret gslist-of)
peek-devices [Method]
  Lists all currently registered input devices
device-manager
  a <clutter-device-manager>
ret
  a pointer to the internal list of <clutter-input-device> objects. The returned list is owned by the <clutter-device-manager> and should never be modified or freed.

Since 1.2

clutter-device-manager-get-device [Function]
  (self <clutter-device-manager>) (device_id int) ⇒ (ret <clutter-input-device*>)
get-device [Method]
  Retrieves the <clutter-input-device> with the given device-id
device-manager
  a <clutter-device-manager>
device-id the integer id of a device
A `<clutter-input-device>` or `'#f'`. The returned device is owned by the `<clutter-device-manager>` and should never be modified or freed.

Since 1.2
30 ClutterDragAction

Action enabling dragging on actors

30.1 Overview

<clutter-drag-action> is a sub-class of <clutter-action> that implements all the necessary logic for dragging actors.

The simplest usage of <clutter-drag-action> consists in adding it to a <clutter-actor> and setting it as reactive; for instance, the following code:

```c
clutter_actor_add_action (actor, clutter_drag_action_new ());
clutter_actor_set_reactive (actor, TRUE);
```

will automatically result in the actor moving to follow the pointer whenever the pointer’s button is pressed over the actor and moved across the stage.

The <clutter-drag-action> will signal the begin and the end of a dragging through the <"drag-begin"> and <"drag-end"> signals, respectively. Each pointer motion during a drag will also result in the <"drag-motion"> signal to be emitted.

It is also possible to set another <clutter-actor> as the dragged actor by calling clutter-drag-action-set-drag-handle from within a handle of the <"drag-begin"> signal. The drag handle must be parented and exist between the emission of <"drag-begin"> and <"drag-end">.

The example program above allows dragging the rectangle around the stage using a <clutter-drag-action>. When pressing the "Shift") key the actor that is going to be dragged is a separate rectangle, and when the drag ends, the original rectangle will be animated to the final coordinates.

<clutter-drag-action> is available since Clutter 1.4

30.2 Usage

clutter-drag-action-new ⇒ (ret <clutter-action>) [Function]

Creates a new <clutter-drag-action> instance

ret the newly created <clutter-drag-action>

Since 1.4

clutter-drag-action-set-drag-handle

(self <clutter-drag-action>) (handle <clutter-actor>) [Function]

set-drag-handle

Sets the actor to be used as the drag handle.

action a <clutter-drag-action>

handle a <clutter-actor>, or ‘#f’ to unset.

Since 1.4
clutter-drag-action-get-drag-handle

\[(self \text{ <clutter-drag-action>}) \Rightarrow (ret \text{ <clutter-actor>})\]

get-drag-handle
Retrieves the drag handle set by clutter-drag-action-set-drag-handle

\[action\quad \text{a <clutter-drag-action>}\]
\[ret\quad \text{a <clutter-actor>}, \text{used as the drag handle, or ‘#f’ if none was set.}\]

Since 1.4

clutter-drag-action-set-drag-axis

\[(self \text{ <clutter-drag-action>}) (axis \text{ <clutter-drag-axis>})\]

set-drag-axis
Restricts the dragging action to a specific axis

\[action\quad \text{a <clutter-drag-action>}\]
\[axis\quad \text{the axis to constraint the dragging to}\]

Since 1.4

clutter-drag-action-get-drag-axis

\[(self \text{ <clutter-drag-action>}) \Rightarrow (ret \text{ <clutter-drag-axis>})\]

get-drag-axis
Retrieves the axis constraint set by clutter-drag-action-set-drag-axis

\[action\quad \text{a <clutter-drag-action>}\]
\[ret\quad \text{the axis constraint}\]

Since 1.4
31 ClutterDropAction

An action for drop targets

31.1 Overview

<clutter-drop-action> is a <clutter-action> that allows a <clutter-actor> implementation to control what happens when an actor dragged using a <clutter-drag-action> crosses the target area or when a dragged actor is released (or "dropped") on the target area.

A trivial use of <clutter-drop-action> consists in connecting to the "drop" signal and handling the drop from there, for instance:

```c
ClutterAction *action = clutter_drop_action ();

g_signal_connect (action, "drop", G_CALLBACK (on_drop), NULL);
clutter_actor_add_action (an_actor, action);
```

The "can-drop" can be used to control whether the "drop" signal is going to be emitted; returning '#f' from a handler connected to the "can-drop" signal will cause the "drop" signal to be skipped when the input device button is released.

It’s important to note that <clutter-drop-action> will only work with actors dragged using <clutter-drag-action>.

<clutter-drop-action> is available since Clutter 1.8

31.2 Usage

clutter-drop-action-new ⇒ (ret <clutter-action>) [Function]

Creates a new <clutter-drop-action>.

Use clutter-actor-add-action to add the action to a <clutter-actor>.

ret the newly created <clutter-drop-action>

Since 1.8
32 ClutterEffect

Base class for actor effects

32.1 Overview

The `<clutter-effect>` class provides a default type and API for creating effects for generic actors.

Effects are a `<clutter-actor-meta>` sub-class that modify the way an actor is painted in a way that is not part of the actor’s implementation.

Effects should be the preferred way to affect the paint sequence of an actor without sub-classing the actor itself and overriding the `<paint>` virtual function.

32.2 Implementing a ClutterEffect

Creating a sub-class of `<clutter-effect>` requires overriding the paint method. The implementation of the function should look something like this:

```c
void effect_paint (ClutterEffect *effect, ClutterEffectPaintFlags flags)
{
    /* Set up initialisation of the paint such as binding a
       CoglOffscreen or other operations */

    /* Chain to the next item in the paint sequence. This will either call
       paint on the next effect or just paint the actor if this is
       the last effect. */
    ClutterActor *actor =
        clutter_actor_meta_get_actor (CLUTTER_ACTOR_META (effect));
    clutter_actor_continue_paint (actor);

    /* perform any cleanup of state, such as popping the
       CoglOffscreen */
}
```

The effect can optionally avoid calling `clutter-actor-continue-paint` to skip any further stages of the paint sequence. This is useful for example if the effect contains a cached image of the actor. In that case it can optimise painting by avoiding the actor paint and instead painting the cached image. The ‘CLUTTER_EFFECT_PAINT_ACTOR_DIRTY’ flag is useful in this case. Clutter will set this flag when a redraw has been queued on the actor since it was last painted. The effect can use this information to decide if the cached image is still valid.

The paint virtual was added in Clutter 1.8. Prior to that there were two separate functions as follows.

- `pre-paint`, which is called before painting the `<clutter-actor>`.
- `post-paint`, which is called after painting the `<clutter-actor>`.
The pre-paint function was used to set up the `<clutter-effect>` right before the `<clutter-actor>`’s paint sequence. This function can fail, and return ‘#f’; in that case, no post-paint invocation will follow.

The post-paint function was called after the `<clutter-actor>`’s paint sequence.

With these two functions it is not possible to skip the rest of the paint sequence. The default implementation of the paint virtual calls pre-paint, clutter-actor-continue-paint and then post-paint so that existing actors that aren’t using the paint virtual will continue to work. New actors using the paint virtual do not need to implement pre or post paint.

The example below creates two rectangles: one will be painted "behind" the actor, while another will be painted "on top" of the actor. The set-actor implementation will create the two materials used for the two different rectangles; the paint function will paint the first material using cogl-rectangle, before continuing and then it will paint the second material after.

typedef struct {
    ClutterEffect parent_instance;

    CoglHandle rect_1;
    CoglHandle rect_2;
} MyEffect;

typedef struct _ClutterEffectClass MyEffectClass;

G_DEFINE_TYPE (MyEffect, my_effect, CLUTTER_TYPE_EFFECT);

static void
my_effect_set_actor (ClutterActorMeta *meta,
                     ClutterActor      *actor)
{
    MyEffect  *self = MY_EFFECT (meta);

    /* Clear the previous state */
    if (self->rect_1)
    {
        cogl_handle_unref (self->rect_1);
        self->rect_1 = NULL;
    }

    if (self->rect_2)
    {
        cogl_handle_unref (self->rect_2);
        self->rect_2 = NULL;
    }

    /* Maintain a pointer to the actor */
self->actor = actor;

/* If we've been detached by the actor then we should just bail out here */
if (self->actor == NULL)
    return;

/* Create a red material */
self->rect_1 = cogl_material_new();
cogl_material_set_color4f (self->rect_1, 1.0, 0.0, 0.0, 1.0);

/* Create a green material */
self->rect_2 = cogl_material_new();
cogl_material_set_color4f (self->rect_2, 0.0, 1.0, 0.0, 1.0);
}

static gboolean
my_effect_paint (ClutterEffect *effect)
{
    MyEffect *self = MY_EFFECT (effect);
    gfloat width, height;

    clutter_actor_get_size (self->actor, &width, &height);

    /* Paint the first rectangle in the upper left quadrant */
cogl_set_source (self->rect_1);
cogl_rectangle (0, 0, width / 2, height / 2);

    /* Continue to the rest of the paint sequence */
clutter_actor_continue_paint (self->actor);

    /* Paint the second rectangle in the lower right quadrant */
cogl_set_source (self->rect_2);
cogl_rectangle (width / 2, height / 2, width, height);
}

static void
my_effect_class_init (MyEffectClass *klass)
{
    ClutterActorMetaClas *meta_class = CLUTTER_ACTOR_META_CLASS (klass);

    meta_class->set_actor = my_effect_set_actor;

    klass->paint = my_effect_paint;
}
<clutter-effect> is available since Clutter 1.4

32.3 Usage

clutter-effect-queue-repaint (self <clutter-effect>)  [Function]
queue-repaint  [Method]

Queues a repaint of the effect. The effect can detect when the paint method is called as a result of this function because it will not have the ‘CLUTTER_EFFECT_PAINT_ACTOR_DIRTY’ flag set. In that case the effect is free to assume that the actor has not changed its appearance since the last time it was painted so it doesn’t need to call clutter-actor-continue-paint if it can draw a cached image. This is mostly intended for effects that are using a ‘CoglOffscreen’ to redirect the actor (such as ‘ClutterOffscreenEffect’). In that case the effect can save a bit of rendering time by painting the cached texture without causing the entire actor to be painted.

This function can be used by effects that have their own animatable parameters. For example, an effect which adds a varying degree of a red tint to an actor by redirecting it through a CoglOffscreen might have a property to specify the level of tint. When this value changes, the underlying actor doesn’t need to be redrawn so the effect can call clutter-effect-queue-repaint to make sure the effect is repainted.

Note however that modifying the position of the parent of an actor may change the appearance of the actor because its transformation matrix would change. In this case a redraw wouldn’t be queued on the actor itself so the ‘CLUTTER_EFFECT_PAINT_ACTOR_DIRTY’ would still not be set. The effect can detect this case by keeping track of the last modelview matrix that was used to render the actor and verifying that it remains the same in the next paint.

Any other effects that are layered on top of the passed in effect will still be passed the ‘CLUTTER_EFFECT_PAINT_ACTOR_DIRTY’ flag. If anything queues a redraw on the actor without specifying an effect or with an effect that is lower in the chain of effects than this one then that will override this call. In that case this effect will instead be called with the ‘CLUTTER_EFFECT_PAINT_ACTOR_DIRTY’ flag set.

effect  A <clutter-effect> which needs redrawing

Since 1.8
33 Events

User and window system events

33.1 Overview

Windowing events handled by Clutter.

The events usually come from the windowing backend, but can also be synthesized by Clutter itself or by the application code.

33.2 Usage

clutter-event-new (type <clutter-event-type>) ⇒ (ret <clutter-event>)

Creates a new <clutter-event> of the specified type.

*type* The type of event.

*ret* A newly allocated <clutter-event>.

clutter-event-type (self <clutter-event>) ⇒ (ret <clutter-event-type>)

Retrieves the type of the event.

*event* a <clutter-event>

*ret* a <clutter-event-type>

clutter-event-set-coords (self <clutter-event>) (x float) (y float)

Sets the coordinates of the event.

*event* a <clutter-event>

*x* the X coordinate of the event

*y* the Y coordinate of the event

Since 1.8

clutter-event-get-coords (self <clutter-event>) ⇒ (x float) (y float)

Retrieves the coordinates of *event* and puts them into *x* and *y*.

*event* a <clutter-event>

*x* return location for the X coordinate, or ‘#f’.

*y* return location for the Y coordinate, or ‘#f’.

Since 0.4
clutter-event-set-state (self <clutter-event>) (state <clutter-modifier-type>)

set-state
Sets the modifier state of the event.

event a <clutter-event>
state the modifier state to set

Since 1.8

clutter-event-get-state (self <clutter-event>) => (ret <clutter-modifier-type>)

get-state
Retrieves the modifier state of the event.

event a <clutter-event>
ret the modifier state parameter, or 0

Since 0.4

clutter-event-set-time (self <clutter-event>) (time unsigned-int32)

set-time
Sets the time of the event.

event a <clutter-event>
time the time of the event

Since 1.8

clutter-event-get-time (self <clutter-event>) => (ret unsigned-int32)

get-time
Retrieves the time of the event.

event a <clutter-event>
ret the time of the event, or ’CLUTTER_CURRENT_TIME’

Since 0.4

clutter-event-set-source (self <clutter-event>) (actor <clutter-actor>)

set-source
Sets the source <clutter-actor> of event.

event a <clutter-event>
actor a <clutter-actor>, or ‘#f’.

Since 1.8
**clutter-event-get-source** *(self <clutter-event>)*

⇒ *(ret <clutter-actor>)*

**get-source**

Retrieves the source `<clutter-actor>` the event originated from, or NULL if the event has no source.

*event* a `<clutter-event>`

*ret* a `<clutter-actor>`.

Since 0.6

**clutter-event-set-stage** *(self <clutter-event>)*

*(stage <clutter-stage>)*

**set-stage**

Sets the source `<clutter-stage>` of the event.

*event* a `<clutter-event>`

*stage* a `<clutter-stage>`, or ‘#f’.

Since 1.8

**clutter-event-get-stage** *(self <clutter-event>)*

⇒ *(ret <clutter-stage>)*

**get-stage**

Retrieves the source `<clutter-stage>` the event originated for, or ‘#f’ if the event has no stage.

*event* a `<clutter-event>`

*ret* a `<clutter-stage>`.

Since 0.8

**clutter-event-set-flags** *(self <clutter-event>)*

*(flags <clutter-event-flags>)*

**set-flags**

Sets the `<clutter-event-flags>` of *event*

*event* a `<clutter-event>`

*flags* a binary OR of `<clutter-event-flags>` values

Since 1.8

**clutter-event-get-flags** *(self <clutter-event>)*

⇒ *(ret <clutter-event-flags>)*

**get-flags**

Retrieves the `<clutter-event-flags>` of *event*

*event* a `<clutter-event>`

*ret* the event flags

Since 1.0
clutter-event-get-event-sequence (self <clutter-event>) ⇒ (ret <clutter-event-sequence*>)

get-event-sequence
Retrieves the <clutter-event-sequence> of event.

event a <clutter-event> of type 'CLUTTER_TOUCH_BEGIN', 'CLUTTER_TOUCH_UPDATE', 'CLUTTER_TOUCH_END', or 'CLUTTER_TOUCH_CANCEL'

ret the event sequence, or '#f'.

Since 1.10

clutter-event-get ⇒ (ret <clutter-event>)

Pops an event off the event queue. Applications should not need to call this.

ret A <clutter-event> or NULL if queue empty

Since 0.4

clutter-event-peek ⇒ (ret <clutter-event>)

Returns a pointer to the first event from the event queue but does not remove it.

ret A <clutter-event> or NULL if queue empty.

Since 0.4

clutter-event-put (self <clutter-event>)

put
Puts a copy of the event on the back of the event queue. The event will have the 'CLUTTER_EVENT_FLAG_SYNTHETIC' flag set. If the source is set event signals will be emitted for this source and capture/bubbling for its ancestors. If the source is not set it will be generated by picking or use the actor that currently has keyboard focus

event a <clutter-event>

Since 0.6

clutter-events-pending ⇒ (ret bool)

Checks if events are pending in the event queue.

ret TRUE if there are pending events, FALSE otherwise.

Since 0.4

clutter-event-set-button (self <clutter-event>)

(set-button unsigned-int32)

Sets the button number of event

event a <clutter-event> or type 'CLUTTER_BUTTONPRESS' or of type 'CLUTTER_BUTTON_RELEASE'

button the button number

Since 1.8
clutter-event-get-button (self <clutter-event>)  
⇒ (ret unsigned-int32)

get-button  
Retrieves the button number of event

  event a <clutter-event> of type ‘CLUTTER_BUTTON_PRESS’ or of type ‘CLUTTER_BUTTON_RELEASE’

  ret the button number

Since 1.0

clutter-event-get-click-count (self <clutter-event>)  
⇒ (ret unsigned-int)

get-click-count  
Retrieves the number of clicks of event

  event a <clutter-event> of type ‘CLUTTER_BUTTON_PRESS’ or of type ‘CLUTTER_BUTTON_RELEASE’

  ret the click count

Since 1.0

clutter-event-set-key-symbol (self <clutter-event>)  
(key_sym unsigned-int)

set-key-symbol  
Sets the key symbol of event.

  event a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or ‘CLUTTER_KEY_RELEASE’

  key-sym the key symbol representing the key

Since 1.8

clutter-event-get-key-symbol (self <clutter-event>)  
⇒ (ret unsigned-int)

get-key-symbol  
Retrieves the key symbol of event

  event a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or of type ‘CLUTTER_KEY_RELEASE’

  ret the key symbol representing the key

Since 1.0

clutter-event-set-key-code (self <clutter-event>)  
(key_code unsigned-int16)

set-key-code  
Sets the keycode of the event.

  event a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or ‘CLUTTER_KEY_RELEASE’
key-code  the keycode representing the key
Since 1.8

clutter-event-get-key-code (self <clutter-event>)  [Function]
⇒ (ret unsigned-int16)

get-key-code  [Method]
Retrieves the keycode of the key that caused event

event  a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or of type
‘CLUTTER_KEY_RELEASE’

ret  The keycode representing the key
Since 1.0

clutter-event-set-key-unicode (self <clutter-event>)  [Function]
(key_unicode unsigned-int32)

set-key-unicode  [Method]
Sets the Unicode value of event.

event  a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or
‘CLUTTER_KEY_RELEASE’

key-unicode  the Unicode value representing the key
Since 1.8

clutter-event-get-key-unicode (self <clutter-event>)  [Function]
⇒ (ret unsigned-int32)

get-key-unicode  [Method]
Retrieves the unicode value for the key that caused keyev.

event  a <clutter-event> of type ‘CLUTTER_KEY_PRESS’ or
‘CLUTTER_KEY_RELEASE’

ret  The unicode value representing the key
Since 1.8

clutter-keysym-to-unicode (keyval unsigned-int)  [Function]
⇒ (ret unsigned-int32)
Converts keyval from a Clutter key symbol to the corresponding ISO10646 (Unicode)
character.

keyval  a key symbol

ret  a Unicode character, or 0 if there is no corresponding character.

clutter-unicode-to-keysym (wc unsigned-int32)  [Function]
⇒ (ret unsigned-int)
Convert from a ISO10646 character to a key symbol.

wc  a ISO10646 encoded character

ret  the corresponding Clutter key symbol, if one exists. or, if there is no
corresponding symbol, wc | 0x01000000
Since 1.10
clutter-event-set-related (self <clutter-event>)  
    (actor <clutter-actor>)  
    float  
    Sets the related actor of a crossing event  
    event a <clutter-event> of type ‘CLUTTER_ENTER’ or ‘CLUTTER_LEAVE’  
    actor a <clutter-actor> or ‘#f’.  
    Since 1.8

clutter-event-get-related (self <clutter-event>)  
    ⇒ (ret <clutter-actor>)  
    float  
    Retrieves the related actor of a crossing event.  
    event a <clutter-event> of type ‘CLUTTER_ENTER’ or of type ‘CLUTTER_LEAVE’  
    ret the related <clutter-actor>, or ‘#f’.  
    Since 1.0

clutter-event-set-scroll-direction (self <clutter-event>)  
    (direction <clutter-scroll-direction>)  
    set-scroll-direction  
    float  
    Sets the direction of the scrolling of event  
    event a <clutter-event>  
    direction the scrolling direction  
    Since 1.8

clutter-event-get-scroll-delta (self <clutter-event>)  
    ⇒ (dx double) (dy double)  
    get-scroll-delta  
    float  
    Retrieves the precise scrolling information of event.  
    The event has to have a <clutter-scroll-event.direction> value of  
    ‘CLUTTER_SCROLL_SMOOTH’.  
    event a <clutter-event> of type ‘CLUTTER_SCROLL’  
    dx return location for the delta on the horizontal axis.  
    dy return location for the delta on the vertical axis.  
    Since 1.10

clutter-event-set-scroll-delta (self <clutter-event>)  
    (dx double) (dy double)  
    set-scroll-delta  
    float  
    Sets the precise scrolling information of event.  
    event a <clutter-event> of type ‘CLUTTER_SCROLL’  
    dx delta on the horizontal axis  
    dy delta on the vertical axis  
    Since 1.10
**clutter-event-set-device** *(self <clutter-event>)*  
*(device <clutter-input-device>)*  

**set-device**  
Sets the device for *event*.

*event* a <clutter-event>
*device* a <clutter-input-device>, or ‘#f’.

Since 1.6

**clutter-event-get-device** *(self <clutter-event>)*  
⇒ *(ret <clutter-input-device>)*  

**get-device**  
Retrieves the <clutter-input-device> for the event.

The <clutter-input-device> structure is completely opaque and should be cast to the platform-specific implementation.

*event* a <clutter-event>
*ret* the <clutter-input-device> or ‘#f’. The returned device is owned by the <clutter-event> and it should not be unreferenced.

Since 1.0

**clutter-event-set-source-device** *(self <clutter-event>)*  
*(device <clutter-input-device>)*  

**set-source-device**  
Sets the source <clutter-input-device> for *event*.

The <clutter-event> must have been created using clutter-event-new.

*event* a <clutter-event>
*device* a <clutter-input-device>.

Since 1.8

**clutter-event-get-source-device** *(self <clutter-event>)*  
⇒ *(ret <clutter-input-device>)*  

**get-source-device**  
Retrieves the hardware device that originated the event.

If you need the virtual device, use clutter-event-get-device. If no hardware device originated this event, this function will return the same device as clutter-event-get-device.

*event* a <clutter-event>
*ret* a pointer to a <clutter-input-device> or ‘#f’.

Since 1.6

**clutter-event-get-device-id** *(self <clutter-event>)*  
⇒ *(ret int)*  

**get-device-id**  
Retrieves the events device id if set.
**event**  
a clutter event

**ret**  
A unique identifier for the device or -1 if the event has no specific device set.

**clutter-event-get-device-type**  
\[(self <clutter-event>) \Rightarrow (ret <clutter-input-device-type>)\]  
[Function]  
Retrieves the type of the device for event

**get-device-type**  
[Method]  
Retrieves the type of the device for event

**event**  
a <clutter-event>

**ret**  
the <clutter-input-device-type> for the device, if any is set

Since 1.0

**clutter-get-current-event-time**  
\[(ret unsigned-int32)\]  
[Function]  
Retrieves the timestamp of the last event, if there is an event or if the event has a timestamp.

**ret**  
the event timestamp, or ‘CLUTTER_CURRENT_TIME’

Since 1.0

**clutter-get-current-event**  
\[(ret <clutter-event>)\]  
[Function]  
If an event is currently being processed, return that event. This function is intended to be used to access event state that might not be exposed by higher-level widgets. For example, to get the key modifier state from a Button ‘clicked’ event.

**ret**  
The current ClutterEvent, or ‘#f’ if none.

Since 1.2
34 Features

Run-time detection of Clutter features

34.1 Overview

Parts of Clutter depend on the underlying platform, including the capabilities of the backend used and the OpenGL features exposed through the Clutter and COGL API.

It is possible to ask whether Clutter has support for specific features at run-time.

See also cogl-get-features and <cogl-feature-flags>

34.2 Usage

\texttt{clutter-feature-available (feature <clutter-feature-flags>)} \quad [\text{Function}]
\begin{align*}
\Rightarrow \quad (ret \ bool) \\
\text{Checks whether feature is available. feature can be a logical OR of <clutter-feature-flags>.} \\
feature \quad \text{a <clutter-feature-flags>} \\
ret \quad \text{"#t" if a feature is available}
\end{align*}

Since 0.1.1

\texttt{clutter-feature-get-all \Rightarrow (ret <clutter-feature-flags>)} \quad [\text{Function}]
\begin{align*}
\text{Returns all the supported features.} \\
ret \quad \text{a logical OR of all the supported features.}
\end{align*}

Since 0.1.1
35 ClutterFixedLayout

A fixed layout manager

35.1 Overview

<clutter-fixed-layout> is a layout manager implementing the same layout policies as <clutter-group>.

<clutter-fixed-layout> is available since Clutter 1.2

35.2 Usage

clutter-fixed-layout-new ⇒ (ret <clutter-layout-manager>) [Function]

Creates a new <clutter-fixed-layout>

ret the newly created <clutter-fixed-layout>

Since 1.2
36 ClutterFlowLayout

A reflowing layout manager

36.1 Overview

<clutter-flow-layout> is a layout manager which implements the following policy:

- the preferred natural size depends on the value of the <"orientation"> property; the layout will try to maintain all its children on a single row or column;
- if either the width or the height allocated are smaller than the preferred ones, the layout will wrap; in this case, the preferred height or width, respectively, will take into account the amount of columns and rows;
- each line (either column or row) in reflowing will have the size of the biggest cell on that line; if the <"homogeneous"> property is set to '#f' the actor will be allocated within that area, and if set to '#t' instead the actor will be given exactly that area;
- the size of the columns or rows can be controlled for both minimum and maximum; the spacing can also be controlled in both columns and rows.

(The missing figure, flow-layout-image

The image shows a <clutter-flow-layout> with the <"orientation"> property set to 'CLUTTER_FLOW_HORIZONTAL'.

<clutter-flow-layout> is available since Clutter 1.2

36.2 Usage

clutter-flow-layout-new [Function]
(orientation <clutter-flow-orientation>)
⇒ (ret <clutter-layout-manager>)
Creates a new <clutter-flow-layout> with the given orientation

orientation the orientation of the flow layout
ret the newly created <clutter-flow-layout>

Since 1.2

clutter-flow-layout-set-homogeneous [Function]
(self <clutter-flow-layout>) (homogeneous bool)
set-homogeneous [Method]
Sets whether the layout should allocate the same space for each child

layout a <clutter-flow-layout>
homogeneous

whether the layout should be homogeneous or not

Since 1.2

**clutter-flow-layout-get-homogeneous**

(Function)

\[
\text{clutter-flow-layout-get-homogeneous} \quad (\text{self} \ <\ \text{clutter-flow-layout}) \Rightarrow (\text{ret} \ \text{bool})
\]

**get-homogeneous**

(Method)

Retrieves whether the layout is homogeneous

- **layout** a <clutter-flow-layout>
- **ret** ‘#t’ if the <clutter-flow-layout> is homogeneous

Since 1.2

**clutter-flow-layout-set-orientation**

(Function)

\[
\text{clutter-flow-layout-set-orientation} \quad (\text{self} \ <\ \text{clutter-flow-layout}) \quad (\text{orientation} \ <\ \text{clutter-flow-orientation})
\]

**set-orientation**

(Method)

Sets the orientation of the flow layout

The orientation controls the direction used to allocate the children: either horizontally or vertically. The orientation also controls the direction of the overflowing

- **layout** a <clutter-flow-layout>
- **orientation** the orientation of the layout

Since 1.2

**clutter-flow-layout-set-row-spacing**

(Function)

\[
\text{clutter-flow-layout-set-row-spacing} \quad (\text{self} \ <\ \text{clutter-flow-layout}) \quad (\text{spacing} \ \text{float})
\]

**set-row-spacing**

(Method)

Sets the spacing between rows, in pixels

- **layout** a <clutter-flow-layout>
- **spacing** the space between rows

Since 1.2

**clutter-flow-layout-get-row-spacing**

(Function)

\[
\text{clutter-flow-layout-get-row-spacing} \quad (\text{self} \ <\ \text{clutter-flow-layout}) \Rightarrow (\text{ret} \ \text{float})
\]

**get-row-spacing**

(Method)

Retrieves the spacing between rows

- **layout** a <clutter-flow-layout>
- **ret** the spacing between rows of the <clutter-flow-layout>, in pixels

Since 1.2
clutter-flow-layout-set-row-height  [Function]
  (self <clutter-flow-layout>) (min_height float) (max_height float)
set-row-height  [Method]
  Sets the minimum and maximum heights that a row can have
  layout   a <clutter-flow-layout>
  min-height  the minimum height of a row
  max-height  the maximum height of a row
Since 1.2

clutter-flow-layout-get-row-height  [Function]
  (self <clutter-flow-layout>) ⇒ (min_height float) (max_height float)
get-row-height  [Method]
  Retrieves the minimum and maximum row heights
  layout   a <clutter-flow-layout>
  min-height  return location for the minimum row height, or ‘#f’.
  max-height  return location for the maximum row height, or ‘#f’.
Since 1.2
37 ClutterGestureAction

Action for gesture gestures

37.1 Overview

<clutter-gesture-action> is a sub-class of <clutter-action> that implements the logic for recognizing gesture gestures. It listens for low level events such as <clutter-button-event> and <clutter-motion-event> on the stage to raise the <"gesture-begin">, <"gesture-progress">, and * <"gesture-end"> signals.

To use <clutter-gesture-action> you just need to apply it to a <clutter-actor> using clutter-actor-add-action and connect to the signals:

```c
ClutterAction *action = clutter_gesture_action_new();
clutter_actor_add_action (actor, action);

g_signal_connect (action, "gesture-begin", G_CALLBACK (on_gesture_begin), NULL);
```

37.2 Usage

clutter-gesture-action-new ⇒ (ret <clutter-action>)  [Function]
Creates a new <clutter-gesture-action> instance.

ret the newly created <clutter-gesture-action>

Since 1.8
38 ClutterImage

Image data content

38.1 Overview

<clutter-image> is a <clutter-content> implementation that displays image data.

<clutter-image> is available since Clutter 1.10.

38.2 Usage

clutter-image-new ⇒ (ret <clutter-content>)  [Function]

Creates a new <clutter-image> instance.

ret the newly created <clutter-image> instance. Use g-object-unref when done.

Since 1.10
Chapter 39: ClutterInputDevice

An input device managed by Clutter

39.1 Overview

<clutter-input-device> represents an input device known to Clutter.

The <clutter-input-device> class holds the state of the device, but its contents are usually defined by the Clutter backend in use.

39.2 Usage

clutter-input-device-get-device-id [Function]

\[
\text{clutter-input-device-get-device-id} \quad (\text{self <clutter-input-device*>}) \Rightarrow (\text{int ret})
\]

Retrieves the unique identifier of device

\[
\text{device} \quad \text{a <clutter-input-device>}
\]

\[
\text{ret} \quad \text{the identifier of the device}
\]

Since 1.0

clutter-input-device-get-has-cursor [Function]

\[
\text{clutter-input-device-get-has-cursor} \quad (\text{self <clutter-input-device*>}) \Rightarrow (\text{bool ret})
\]

Retrieves whether device has a pointer that follows the device motion.

\[
\text{device} \quad \text{a <clutter-input-device>}
\]

\[
\text{ret} \quad \text{‘#t’ if the device has a cursor}
\]

Since 1.6

clutter-input-device-set-enabled [Function]

\[
\text{clutter-input-device-set-enabled} \quad (\text{self <clutter-input-device*>}) (\text{enabled bool})
\]

Enables or disables a <clutter-input-device>.

Only devices with a "device-mode" property set to ‘CLUTTER_INPUT_MODE_SLAVE’ or ‘CLUTTER_INPUT_MODE_FLOATING’ can be disabled.

\[
\text{device} \quad \text{a <clutter-input-device>}
\]

\[
\text{enabled} \quad \text{‘#t’ to enable the device}
\]

Since 1.6

clutter-input-device-get-enabled [Function]

\[
\text{clutter-input-device-get-enabled} \quad (\text{self <clutter-input-device*>}) \Rightarrow (\text{bool ret})
\]

Retrieves whether device is enabled.

\[
\text{device} \quad \text{a <clutter-input-device>}
\]

\[
\text{ret} \quad \text{‘#t’ if the device is enabled}
\]

Since 1.6
clutter-input-device-get-n-keys

\[(\text{self} <\text{clutter-input-device}>) \Rightarrow (\text{ret} \text{ unsigned-int})\]

Retrieves the number of keys registered for \textit{device}.

\begin{itemize}
  \item \textit{device} a \text{clutter-input-device}
  \item \textit{ret} the number of registered keys
\end{itemize}

Since 1.6

clutter-input-device-set-key

\[(\text{self} <\text{clutter-input-device}>) \Rightarrow (\text{index} \text{ unsigned-int}) \Rightarrow (\text{keyval} \text{ unsigned-int}) \Rightarrow (\text{modifiers} <\text{clutter-modifier-type}>)\]

Sets the keyval and modifiers at the given \textit{index} for \textit{device}.

Clutter will use the keyval and modifiers set when filling out an event coming from the same input device.

\begin{itemize}
  \item \textit{device} a \text{clutter-input-device}
  \item \textit{index} the index of the key
  \item \textit{keyval} the keyval
  \item \textit{modifiers} a bitmask of modifiers
\end{itemize}

Since 1.6

clutter-input-device-get-key

\[(\text{self} <\text{clutter-input-device}>) \Rightarrow (\text{index} \text{ unsigned-int}) \Rightarrow (\text{ret} \text{ bool}) \Rightarrow (\text{keyval} \text{ unsigned-int}) \Rightarrow (\text{modifiers} <\text{clutter-modifier-type}>)\]

Retrieves the key set using \text{clutter-input-device-set-key}

\begin{itemize}
  \item \textit{device} a \text{clutter-input-device}
  \item \textit{index} the index of the key
  \item \textit{keyval} return location for the keyval at \textit{index}.
  \item \textit{modifiers} return location for the modifiers at \textit{index}.
  \item \textit{ret} ‘#t’ if a key was set at the given index
\end{itemize}

Since 1.6

clutter-input-device-get-n-axes

\[(\text{self} <\text{clutter-input-device}>) \Rightarrow (\text{ret} \text{ unsigned-int})\]

Retrieves the number of axes available on \textit{device}.

\begin{itemize}
  \item \textit{device} a \text{clutter-input-device}
  \item \textit{ret} the number of axes on the device
\end{itemize}

Since 1.6

clutter-input-device-get-axis

\[(\text{self} <\text{clutter-input-device}>) \Rightarrow (\text{index} \text{ unsigned-int}) \Rightarrow (\text{ret} <\text{clutter-input-axis}>)\]

Retrieves the type of axis on \textit{device} at the given index.
device  a <clutter-input-device>
index  the index of the axis
ret    the axis type

Since 1.6

clutter-input-device-get-axis-value  [Function]
  (self <clutter-input-device*>)(axis <clutter-input-axis>)
⇒ (ret bool)(axes double)(value double)

Extracts the value of the given axis of a <clutter-input-device> from an array of axis values.

An example of typical usage for this function is:

  ClutterInputDevice *device = clutter_event_get_device (event);
  gdoble *axes = clutter_event_get_axes (event, NULL);
  gdoble pressure_value = 0;

  clutter_input_device_get_axis_value (device, axes,
                                      CLUTTER_INPUT_AXIS_PRESSURE,
                                      &pressure_value);

device  a <clutter-input-device>
axes    an array of axes values, typically coming from clutter-event-get-axes.
axis    the axis to extract
value   return location for the axis value.
ret     ‘#t’ if the value was set, and ‘#f’ otherwise

Since 1.6

clutter-input-device-grab  [Function]
  (self <clutter-input-device*>)(actor <clutter-actor>)

Acquires a grab on actor for the given device.

Any event coming from device will be delivered to actor, bypassing the usual event delivery mechanism, until the grab is released by calling clutter-input-device-ungrab.

The grab is client-side: even if the windowing system used by the Clutter backend has the concept of "device grabs", Clutter will not use them.

Only <clutter-input-device> of types ‘CLUTTER_POINTER_DEVICE’ and ‘CLUTTER_KEYBOARD_DEVICE’ can hold a grab.

device  a <clutter-input-device>
actor   a <clutter-actor>

Since 1.10
clutter-input-device-ungrab (self <clutter-input-device*>)

Releases the grab on the device, if one is in place.

device a <clutter-input-device>

Since 1.10
40 Value intervals

An object holding an interval of two values

40.1 Overview

<clutter-interval> is a simple object that can hold two values defining an interval. <clutter-interval> can hold any value that can be enclosed inside a <gvalue>.

Once a <clutter-interval> for a specific <g-type> has been instantiated the "value-type" property cannot be changed anymore.

<clutter-interval> starts with a floating reference; this means that any object taking a reference on a <clutter-interval> instance should also take ownership of the interval by using g-object-ref-sink.

<clutter-interval> is used by <clutter-animation> to define the interval of values that an implicit animation should tween over.

<clutter-interval> can be subclassed to override the validation and value computation.

<clutter-interval> is available since Clutter 1.0

40.2 Usage

clutter-interval-new-with-values (gtype <gtype>)  [Function]
\[(\text{initial} <\text{gvalue}) (\text{final} <\text{gvalue}) \Rightarrow (\text{ret} <\text{clutter-interval})]\]
Creates a new <clutter-interval> of type gtype, between initial and final.

This function is useful for language bindings.

\begin{itemize}
  \item \textit{gtype} the type of the values in the interval
  \item \textit{initial} a <gvalue> holding the initial value of the interval
  \item \textit{final} a <gvalue> holding the final value of the interval
  \item \textit{ret} the newly created <clutter-interval>
\end{itemize}

Since 1.0

clutter-interval-clone (self <clutter-interval>)  [Function]
\[\Rightarrow (\text{ret} <\text{clutter-interval})\]
clone  [Method]

Creates a copy of interval.

\begin{itemize}
  \item \textit{interval} a <clutter-interval>
  \item \textit{ret} the newly created <clutter-interval>.
\end{itemize}

Since 1.0

clutter-interval-get-value-type (self <clutter-interval>)  [Function]
\[\Rightarrow (\text{ret} <\text{gtype})\]
get-value-type  [Method]

Retrieves the <g-type> of the values inside interval.
interval a <clutter-interval>
ret the type of the value, or G_TYPE_INVALID

Since 1.0

clutter-interval-set-initial-value (self <clutter-interval>) [Function]
  (value <gvalue>)
set-initial-value [Method]
  Sets the initial value of interval to value. The value is copied inside the <clutter-interval>.
  Rename to: clutter_interval_set_initial
interval a <clutter-interval>
value a <gvalue>

Since 1.0

clutter-interval-get-initial-value (self <clutter-interval>) [Function]
  ⇒ (ret <gvalue>)
get-initial-value [Method]
  Retrieves the initial value of interval and copies it into value.
  The passed <gvalue> must be initialized to the value held by the <clutter-interval>.
interval a <clutter-interval>
value a <gvalue>.

Since 1.0

clutter-interval-set-final-value (self <clutter-interval>) [Function]
  (value <gvalue>)
set-final-value [Method]
  Sets the final value of interval to value. The value is copied inside the <clutter-interval>.
  Rename to: clutter_interval_set_final
interval a <clutter-interval>
value a <gvalue>.

Since 1.0

clutter-interval-get-final-value (self <clutter-interval>) [Function]
  ⇒ (ret <gvalue>)
get-final-value [Method]
  Retrieves the final value of interval and copies it into value.
  The passed <gvalue> must be initialized to the value held by the <clutter-interval>.
interval a <clutter-interval>
value a <gvalue>.

Since 1.0
clutter-interval-validate (self <clutter-interval>)  [Function]
                       (pspec <gparam>) ⇒ (ret bool)

validate
Validates the initial and final values of interval against a <gparam>.

interval a <clutter-interval>
pspec a <gparam>
ret ‘#t’ if the <clutter-interval> is valid, ‘#f’ otherwise

Since 1.0

clutter-interval-compute (self <clutter-interval>)  [Function]
                       (factor double) ⇒ (ret <gvalue>)

compute
Computes the value between the interval boundaries given the progress factor
Unlike clutter-interval-compute-value, this function will return a const pointer to the computed value
You should use this function if you immediately pass the computed value to another function that makes a copy of it, like g-object-set-property

interval a <clutter-interval>
factor the progress factor, between 0 and 1
ret a pointer to the computed value, or ‘#f’ if the computation was not successful.

Since 1.4
41 ClutterLayoutManager

41.1 Overview

<clutter-layout-manager> is a base abstract class for layout managers. A layout manager implements the layouting policy for a composite or a container actor: it controls the preferred size of the actor to which it has been paired, and it controls the allocation of its children.

Any composite or container <clutter-actor> subclass can delegate the layouting of its children to a <clutter-layout-manager>. Clutter provides a generic container using <clutter-layout-manager> called <clutter-box>.

Clutter provides some simple <clutter-layout-manager> sub-classes, like <clutter-flow-layout> and <clutter-bin-layout>.

41.2 Using a Layout Manager inside an Actor

In order to use a <clutter-layout-manager> inside a <clutter-actor> sub-class you should invoke clutter-layout-manager-get-preferred-width inside the (structname "ClutterActor") : get-preferred-width virtual function and clutter-layout-manager-get-preferred-height inside the function implementations. You should also call clutter-layout-manager-allocate inside the implementation of the

In order to receive notifications for changes in the layout manager policies you should also connect to the "layout-changed" signal and queue a relayout on your actor. The following code should be enough if the actor does not need to perform specific operations whenever a layout manager changes:

```c
    g_signal_connect_swapped (layout_manager,
"layout-changed",
    G_CALLBACK (clutter_actor_queue_relayout),
    actor);
```

41.3 Implementing a ClutterLayoutManager

The implementation of a layout manager does not differ from the implementation of the size requisition and allocation bits of <clutter-actor>, so you should read the relative documentation for subclassing ClutterActor.

The layout manager implementation can hold a back pointer to the <clutter-container> by implementing the set-container virtual function. The layout manager should not hold a real reference (i.e. call g-object-ref) on the container actor, to avoid reference cycles.

If a layout manager has properties affecting the layout policies then it should emit the "layout-changed" signal on itself by using the clutter-layout-manager-layout-changed function whenever one of these properties changes.
41.4 Animating a ClutterLayoutManager

A layout manager is used to let a <clutter-container> take complete ownership over the layout (that is: the position and sizing) of its children; this means that using the Clutter animation API, like clutter-actor-animate, to animate the position and sizing of a child of a layout manager it is not going to work properly, as the animation will automatically override any setting done by the layout manager itself.

It is possible for a <clutter-layout-manager> sub-class to animate its children layout by using the base class animation support. The <clutter-layout-manager> animation support consists of three virtual functions: begin-animation, get-animation-progress and end-animation.

get-animation-progress
end-animation

This virtual function is invoked when the layout manager should begin an animation. The implementation should set up the state for the animation and create the ancillary objects for animating the layout. The default implementation creates a <clutter-timeline> for the given duration and a <clutter-alpha> binding the timeline to the given easing mode. This function returns a <clutter-alpha> which should be used to control the animation from the caller perspective.

This virtual function should be invoked when animating a layout manager. It returns the progress of the animation, using the same semantics as the "alpha" value.

This virtual function is invoked when the animation of a layout manager ends, and it is meant to be used for bookkeeping the objects created in the begin-animation function. The default implementation will call it implicitly when the timeline is complete.

The simplest way to animate a layout is to create a <clutter-timeline> inside the begin-animation virtual function, along with a <clutter-alpha>, and for each "new-frame" signal emission call clutter-layout-manager-layout-changed, which will cause a relayout. The "completed" signal emission should cause clutter-layout-manager-end-animation to be called. The default implementation provided internally by <clutter-layout-manager> does exactly this, so most sub-classes should either not override any animation-related virtual function or simply override begin-animation and end-animation to set up ad hoc state, and then chain up to the parent’s implementation.

The code below shows how a <clutter-layout-manager> sub-class should provide animating the allocation of its children from within the allocate virtual function implementation. The animation is computed between the last stable allocation performed before the animation started and the desired final allocation.

The <clutter-layout-manager> sub-class and it is updated by overriding the begin-animation and end-animation virtual functions and chaining up to the base class implementation.

The last stable allocation is stored within a <clutter-layout-meta> sub-class used by the implementation.

```c
static void
my_layout_manager_allocate (ClutterLayoutManager *manager,
                           ClutterContainer       *container,
```

const ClutterActorBox *allocation,
ClutterAllocationFlags flags)
{
    MyLayoutManager *self = MY_LAYOUT_MANAGER (manager);
    ClutterActor *child;

    for (child = clutter_actor_get_first_child (CLUTTER_ACTOR (container));
        child != NULL;
        child = clutter_actor_get_next_sibling (child))
    {
        ClutterLayoutMeta *meta;
        MyLayoutMeta *my_meta;

        /* retrieve the layout meta-object */
        meta = clutter_layout_manager_get_child_meta (manager,
                                                       container,
                                                       child);
        my_meta = MY_LAYOUT_META (meta);

        /* compute the desired allocation for the child */
        compute_allocation (self, my_meta, child,
                            allocation, flags,
                            &child_box);

        /* this is the additional code that deals with the animation */
        if (!self->is_animating)
        {
            my_meta->last_alloc = clutter_actor_box_copy (&child_box);
        }
        else
        {
            ClutterActorBox end = { 0, };
            gdouble p;

            /* get the progress of the animation */
            p = clutter_layout_manager_get_animation_progress (manager);

            if (my_meta->last_alloc != NULL)
            {
                /* copy the desired allocation as the final state */
                end = child_box;

                /* then interpolate the initial and final state */
            }
        }
    }
}
and put the result inside the box we will use to allocate the child:

```c
clutter_actor_box_interpolate (my_meta->last_alloc, &end, p, &child_box);
```

else
{
  /* if there is no stable allocation then the child was added while animating; one possible course of action is to just bail out and fall through to the allocation to position the child directly at its final state */
  my_meta->last_alloc = clutter_actor_box_copy (&child_box);
}

allocate the child:

```c
clutter_actor_allocate (child, &child_box, flags);
```

Sub-classes of `<clutter-layout-manager>` that support animations of the layout changes should call `clutter-layout-manager-begin-animation` whenever a layout property changes value, e.g.:

```c
if (self->orientation != new_orientation)
{
  ClutterLayoutManager *manager;

  self->orientation = new_orientation;

  manager = CLUTTER_LAYOUT_MANAGER (self);
  clutter_layout_manager_layout_changed (manager);
  clutter_layout_manager_begin_animation (manager, 500, CLUTTER_LINEAR);
  g_object_notify (G_OBJECT (self), "orientation");
}
```

The code above will animate a change in the layout property of a layout manager.
41.5 Layout Properties

If a layout manager has layout properties, that is properties that should exist only as the result of the presence of a specific (layout manager, container actor, child actor) combination, and it wishes to store those properties inside a `<clutter-layout-meta>`, then it should override the `::get-child-meta-type` virtual function to return the `<g-type>` of the `<clutter-layout-meta>` sub-class used to store the layout properties; optionally, the `<clutter-layout-manager>` sub-class might also override the (structname "ClutterLayoutManager") `::create-child-meta` virtual function to control how the `<clutter-layout-meta>` instance is created, otherwise the default implementation will be equivalent to:

```
ClutterLayoutManagerClass *klass;
GType meta_type;

klass = CLUTTER_LAYOUT_MANAGER_GET_CLASS (manager);
meta_type = klass->get_child_meta_type (manager);

return g_object_new (meta_type,
    "manager", manager,
    "container", container,
    "actor", actor,
    NULL);
```

Where (varname "container") is the `<clutter-container>` using the `<clutter-layout-manager>` and `<clutter-actor>` child of the `<clutter-container>`.

41.6 Using ClutterLayoutManager with ClutterScript

`<clutter-layout-manager>` instance can be created in the same way as other objects in `<clutter-script>`; properties can be set using the common syntax.

Layout properties can be set on children of a container with a `<clutter-layout-manager>` using the `layout::` modifier on the property name, for instance:

```
{
    "type" : "ClutterBox",
    "layout-manager" : { "type" : "ClutterTableLayout" },
    "children" : [
        {
            "type" : "ClutterTexture",
            "filename" : "image-00.png",
            "layout::row" : 0,
            "layout::column" : 0,
            "layout::x-align" : "left",
            "layout::y-align" : "center",
            "layout::x-expand" : true,
            "layout::y-expand" : true
        }
    ]
}
```
41.7 Usage

clutter-layout-manager-allocate  
(Function)

(clutter-layout-manager-allocate (self) (container) (allocation) (flags))

Allocate the children of container given an area.

See also clutter-actor-allocate

manager  a clutter-layout-manager

container the clutter-container using manager

allocation the clutter-actor-box containing the allocated area of container

flags  the allocation flags

Since 1.2
ClutterLayoutMeta

Wrapper for actors inside a layout manager

42.1 Overview

<clutter-layout-meta> is a wrapper object created by <clutter-layout-manager> implementations in order to store child-specific data and properties.

A <clutter-layout-meta> wraps a <clutter-actor> inside a <clutter-container> using a <clutter-layout-manager>.

<clutter-layout-meta> is available since Clutter 1.2

42.2 Usage

clutter-layout-meta-get-manager (self <clutter-layout-meta>) [Function]
⇒ (ret <clutter-layout-manager>)
get-manager [Method]
Retrieves the actor wrapped by data

data a <clutter-layout-meta>
ret a <clutter-layout-manager>.

Since 1.2
43 ClutterListModel

List model implementation

43.1 Overview

<clutter-list-model> is a <clutter-model> implementation provided by Clutter. 
<clutter-list-model> uses a <g-sequence> for storing the values for each row, so it’s 
optimized for insertion and look up in sorted lists.

<clutter-list-model> is available since Clutter 0.6

43.2 Usage
44 General

Various 'global' clutter functions.

44.1 Overview

Functions to retrieve various global Clutter resources and other utility functions for main-loops, events and threads

44.2 Threading Model

Clutter is thread-aware: all operations performed by Clutter are assumed to be under the big Clutter lock, which is created when the threading is initialized through `clutter-init`.

The code below shows how to correctly initialize Clutter in a multi-threaded environment. These operations are mandatory for applications that wish to use threads with Clutter.

```c
int main (int argc, char *argv[])
{
    /* initialize Clutter */
    clutter_init (&argc, &argv);

    /* program code */

    /* acquire the main lock */
    clutter_threads_enter ();

    /* start the main loop */
    clutter_main ();

    /* release the main lock */
    clutter_threads_leave ();

    /* clean up */
    return 0;
}
```

This threading model has the caveat that it is only safe to call Clutter's API when the lock has been acquired; which happens between pairs of `clutter-threads-enter` and `clutter-threads-leave` calls.

The only safe and portable way to use the Clutter API in a multi-threaded environment is to never access the API from a thread that did not call `clutter-init` and `clutter-main`.

The common pattern for using threads with Clutter is to use worker threads to perform blocking operations and then install idle or timeout sources with the result when the thread finished.
Clutter provides thread-aware variants of g-idle-add and g-timeout-add that acquire the Clutter lock before invoking the provided callback: clutter-threads-add-idle and clutter-threads-add-timeout.

The example below shows how to use a worker thread to perform a blocking operation, and perform UI updates using the main loop.

### 44.3 Usage

**clutter-main**

Starts the Clutter mainloop.

**clutter-main-quit**

Terminates the Clutter mainloop.

**clutter-main-level ⇒ (ret int)**

Retrieves the depth of the Clutter mainloop.

ret The level of the mainloop.

**clutter-get-default-frame-rate ⇒ (ret unsigned-int)**

Retrieves the default frame rate. See clutter-set-default-frame-rate.

ret the default frame rate

Since 0.6

**clutter-get-font-map ⇒ (ret <pango-font-map>)**

Retrieves the <pango-font-map> instance used by Clutter. You can use the global font map object with the COGL Pango API.

ret the <pango-font-map> instance. The returned value is owned by Clutter and it should never be unreferenced.

Since 1.0

**clutter-get-default-text-direction ⇒ (ret <clutter-text-direction>)**

Retrieves the default direction for the text. The text direction is determined by the locale and/or by the "CLUTTER_TEXT_DIRECTION") environment variable.

The default text direction can be overridden on a per-actor basis by using clutter-actor-set-text-direction.

ret the default text direction

Since 1.2

**clutter-get-accessibility-enabled ⇒ (ret bool)**

Returns whether Clutter has accessibility support enabled. As least, a value of TRUE means that there are a proper AtkUtil implementation available

ret ‘#t’ if Clutter has accessibility support enabled

Since 1.4
**clutter-get-keyboard-grab** ⇒ `(ret <clutter-actor>)`  
Queries the current keyboard grab of clutter.

*ret* the actor currently holding the keyboard grab, or NULL if there is no grab.

Since 0.6

**clutter-get-pointer-grab** ⇒ `(ret <clutter-actor>)`  
Queries the current pointer grab of clutter.

*ret* the actor currently holding the pointer grab, or NULL if there is no grab.

Since 0.6

**clutter-grab-keyboard** (*actor* <clutter-actor>)  
Grabs keyboard events, after the grab is done keyboard events ("key-press-event" and "key-release-event") are delivered to this actor directly. The source set in the event will be the actor that would have received the event if the keyboard grab was not in effect.

Like pointer grabs, keyboard grabs should only be used as a last resource.

See also **clutter-stage-set-key-focus** and **clutter-actor-grab-key-focus** to perform a "soft" key grab and assign key focus to a specific actor.

*actor* a <clutter-actor>

Since 0.6

**clutter-grab-pointer** (*actor* <clutter-actor>)  
Grabs pointer events, after the grab is done all pointer related events (press, motion, release, enter, leave and scroll) are delivered to this actor directly without passing through both capture and bubble phases of the event delivery chain. The source set in the event will be the actor that would have received the event if the pointer grab was not in effect.

Grabs completely override the entire event delivery chain done by Clutter. Pointer grabs should only be used as a last resource; using the "captured-event" signal should always be the preferred way to intercept event delivery to reactive actors.

This function should rarely be used.

If a grab is required, you are strongly encouraged to use a specific input device by calling **clutter-input-device-grab**.

*actor* a <clutter-actor>

Since 0.6

**clutter-ungrab-keyboard**  
Removes an existing grab of the keyboard.

Since 0.6

**clutter-ungrab-pointer**  
Removes an existing grab of the pointer.

Since 0.6
clutter-do-event (event <clutter-event>)

Processes an event.

The event must be a valid <clutter-event> and have a <clutter-stage> associated to it.

This function is only useful when embedding Clutter inside another toolkit, and it should never be called by applications.

event a <clutter-event>.

Since 0.4
45 ClutterMedia

An interface for controlling playback of media data

45.1 Overview

<clutter-media> is an interface for controlling playback of media sources.

Clutter core does not provide an implementation of this interface, but other integration libraries like Clutter-GStreamer implement it to offer a uniform API for applications.

<clutter-media> is available since Clutter 0.2

45.2 Usage

clutter-media-set-uri (self <clutter-media>) (uri mchars) [Function]
set-uri
Sets the URI of media to uri.

media a <clutter-media>
uri the URI of the media stream

Since 0.2

clutter-media-get-uri (self <clutter-media>) ⇒ (ret mchars) [Function]
get-uri
Retrieves the URI from media.

media a <clutter-media>
ret the URI of the media stream. Use g-free to free the returned string

Since 0.2

clutter-media-set-playing (self <clutter-media>) (playing bool) [Function]
set-playing
Starts or stops playing of media. The implementation might be asynchronous, so the way to know whether the actual playing state of the media is to use the "notify" signal on the "playing" property and then retrieve the current state with clutter-media-get-playing. ClutterGstVideoTexture in clutter-gst is an example of such an asynchronous implementation.

media a <clutter-media>
playing ‘#t’ to start playing

Since 0.2

clutter-media-get-playing (self <clutter-media>) ⇒ (ret bool) [Function]
get-playing
Retrieves the playing status of media.

media A <clutter-media> object
ret ‘#t’ if playing, ‘#f’ if stopped.

Since 0.2
clutter-media-set-progress (self <clutter-media>)  
(progress double)  
**set-progress**  
[Method]  
Sets the playback progress of media. The progress is a normalized value between 0.0 (begin) and 1.0 (end).

media  
a <clutter-media>  
progress  
the progress of the playback, between 0.0 and 1.0

Since 1.0

clutter-media-get-progress (self <clutter-media>)  
⇒ (ret double)  
**get-progress**  
[Method]  
Retrieves the playback progress of media.

media  
a <clutter-media>  
ret  
the playback progress, between 0.0 and 1.0

Since 1.0

clutter-media-set-subtitle-uri (self <clutter-media>)  
(uri mchars)  
**set-subtitle-uri**  
[Method]  
Sets the location of a subtitle file to display while playing media.

media  
a <clutter-media>  
uri  
the URI of a subtitle file

Since 1.2

clutter-media-get-subtitle-uri (self <clutter-media>)  
⇒ (ret mchars)  
**get-subtitle-uri**  
[Method]  
Retrieves the URI of the subtitle file in use.

media  
a <clutter-media>  
ret  
the URI of the subtitle file. Use `g-free` to free the returned string

Since 1.2

clutter-media-set-audio-volume (self <clutter-media>)  
(volume double)  
**set-audio-volume**  
[Method]  
Sets the playback volume of media to volume.

media  
a <clutter-media>  
volume  
the volume as a double between 0.0 and 1.0

Since 1.0
clutter-media-get-audio-volume (self <clutter-media>) ⇒ (ret double)

get-audio-volume
Retrieves the playback volume of media.

  media  a <clutter-media>
  ret    The playback volume between 0.0 and 1.0

Since 1.0

clutter-media-get-can-seek (self <clutter-media>) ⇒ (ret bool)

get-can-seek
Retrieves whether media is seekable or not.

  media  a <clutter-media>
  ret    ‘#t’ if media can seek, ‘#f’ otherwise.

Since 0.2

clutter-media-get-buffer-fill (self <clutter-media>) ⇒ (ret double)

get-buffer-fill
Retrieves the amount of the stream that is buffered.

  media  a <clutter-media>
  ret    the fill level, between 0.0 and 1.0

Since 1.0

clutter-media-get-duration (self <clutter-media>) ⇒ (ret double)

get-duration
Retrieves the duration of the media stream that media represents.

  media  a <clutter-media>
  ret    the duration of the media stream, in seconds

Since 0.2

clutter-media-set-filename (self <clutter-media>) (filename mchars)

set-filename
Sets the source of media using a file path.

  media  a <clutter-media>
  filename A filename

Since 0.2
46 ClutterModelIter

Iterates through a model

46.1 Overview

<clutter-model-iter> is an object used for iterating through all the rows of a <clutter-model>. It allows setting and getting values on the row which is currently pointing at.

A <clutter-model-iter> represents a position between two elements of the sequence. For example, the iterator returned by clutter-model-get-first-iter represents the gap immediately before the first row of the <clutter-model>, and the iterator returned by clutter-model-get-last-iter represents the gap immediately after the last row.

A <clutter-model-iter> can only be created by a <clutter-model> implementation and it is valid as long as the model does not change.

<clutter-model-iter> is available since Clutter 0.6

46.2 Usage
47 ClutterModel

A generic model implementation

47.1 Overview

<clutter-model> is a generic list model API which can be used to implement the model-view-controller architectural pattern in Clutter.

The <clutter-model> class is a list model which can accept most GObject types as a column type.

Creating a simple clutter model:

```c
enum {
    COLUMN_INT,
    COLUMN_STRING,
    N_COLUMNS
};

ClutterModel *model;

for (i = 0; i < 10; i++)
{
    gchar *string = g_strdup_printf("String %d", i);
    clutter_model_append(model, COLUMN_INT, i,
                           COLUMN_STRING, string,
                           -1);
    g_free(string);
}
```

Iterating through the model consists of retrieving a new <clutter-model-iter> pointing to the starting row, and calling clutter-model-iter-next or clutter-model-iter-prev to move forward or backwards, respectively.

A valid <clutter-model-iter> represents the position between two rows in the model. For example, the "first" iterator represents the gap immediately before the first row, and
the "last" iterator represents the gap immediately after the last row. In an empty sequence, the first and last iterators are the same.

Iterating a `<clutter-model>`:

```c
enum {
    COLUMN_INT,
    COLUMN_STRING,

    N_COLUMNS
};

{
    ClutterModel *model;
    ClutterModelIter *iter = NULL;

    /* Fill the model */
    model = populate_model ();

    /* Get the first iter */
    iter = clutter_model_get_first_iter (model);
    while (!clutter_model_iter_is_last (iter))
    {
        print_row (iter);

        iter = clutter_model_iter_next (iter);
    }

    /* Make sure to unref the iter */
    g_object_unref (iter);
}
```

`<clutter-model>` is an abstract class. Clutter provides a list model implementation called `<clutter-list-model>` which has been optimised for insertion and look up in sorted lists.

### 47.2 ClutterModel custom properties for `<clutter-script>`

`<clutter-model>` defines a custom property "columns" for `<clutter-script>` which allows defining the column names and types. It also defines a custom "rows" property which allows filling the `<clutter-model>` with some data.

The definition below will create a `<clutter-list-model>` with three columns: the first one with name "Name" and containing strings; the second one with name "Score" and containing integers; the third one with name "Icon" and containing `<clutter-texture>`s. The model is filled with three rows. A row can be defined either with an array that holds all columns of a row, or an object that holds "column-name" : "column-value" pairs.
Chapter 47: ClutterModel

```json
{
  "type": "ClutterListModel",
  "id": "teams-model",
  "columns": [
    [ "Name", "gchararray" ],
    [ "Score", "gint" ],
    [ "Icon", "ClutterTexture" ]
  ],
  "rows": [
    [ "Team 1", 42, { "type": "ClutterTexture", "filename": "team1.png" } ],
    [ "Team 2", 23, "team2-icon-script-id" ],
    { "Name": "Team 3", "Icon": "team3-icon-script-id" }
  ]
}
```

<clutter-model> is available since Clutter 0.6

### 47.3 Usage
48 ClutterOffscreenEffect

Base class for effects using offscreen buffers

48.1 Overview
<clutter-offscreen-effect> is an abstract class that can be used by <clutter-effect> sub-classes requiring access to an offscreen buffer.

Some effects, like the fragment shader based effects, can only use GL textures, and in order to apply those effects to any kind of actor they require that all drawing operations are applied to an offscreen framebuffer that gets redirected to a texture.

<clutter-offscreen-effect> provides all the heavy-lifting for creating the offscreen framebuffer, the redirection and the final paint of the texture on the desired stage.

48.2 Implementing a ClutterOffscreenEffect
Creating a sub-class of <clutter-offscreen-effect> requires, in case of overriding the <clutter-effect> virtual functions, to chain up to the <clutter-offscreen-effect>'s implementation.

On top of the <clutter-effect>'s virtual functions, <clutter-offscreen-effect> also provides a paint-target function, which encapsulates the effective painting of the texture that contains the result of the offscreen redirection.

The size of the target material is defined to be as big as the transformed size of the <clutter-actor> using the offscreen effect. Sub-classes of <clutter-offscreen-effect> can change the texture creation code to provide bigger textures by overriding the create-texture virtual function; no chain up to the <clutter-offscreen-effect> implementation is required in this case.

<clutter-offscreen-effect> is available since Clutter 1.4

48.3 Usage
49 ClutterPageTurnEffect

A page turning effect

49.1 Overview

A simple page turning effect

<clutter-page-turn-effect> is available since Clutter 1.4

49.2 Usage

clutter-page-turn-effect-new (period double) (angle double) (radius float) ⇒ (ret <clutter-effect>)

[Function]

Creates a new <clutter-page-turn-effect> instance with the given parameters

period the period of the page curl, between 0.0 and 1.0
angle the angle of the page curl, between 0.0 and 360.0
radius the radius of the page curl, in pixels
ret the newly created <clutter-page-turn-effect>

Since 1.4

clutter-page-turn-effect-set-period (self <clutter-page-turn-effect>) (period double)

[Method]

Sets the period of the page curling, between 0.0 (no curling) and 1.0 (fully curled)

effect a <clutter-page-turn-effect>
period the period of the page curl, between 0.0 and 1.0

Since 1.4

clutter-page-turn-effect-get-period (self <clutter-page-turn-effect>) ⇒ (ret double)

[Method]

Retrieves the value set using clutter-page-turn-effect-get-period

effect a <clutter-page-turn-effect>
ret the period of the page curling

Since 1.4

clutter-page-turn-effect-set-angle (self <clutter-page-turn-effect>) (angle double)

[Method]

Sets the angle of the page curling, in degrees

effect <clutter-page-turn-effect>
angle the angle of the page curl, in degrees

Since 1.4
clutter-page-turn-effect-get-angle

(Function)
(s elf <clutter-page-turn-effect>) ⇒ (ret double)

g et-angle

(Method)
Retrieves the value set using clutter-page-turn-effect-get-angle
effect a <clutter-page-turn-effect>:
ret the angle of the page curling

Since 1.4

c lutter-page-turn-effect-set-radius

(Function)
(s elf <clutter-page-turn-effect>) (radius float)

set-radius

(Method)
Sets the radius of the page curling
effect a <clutter-page-turn-effect>:
radius the radius of the page curling, in pixels

Since 1.4

c lutter-page-turn-effect-get-radius

(Function)
(s elf <clutter-page-turn-effect>) ⇒ (ret float)

get-radius

(Method)
Retrieves the value set using clutter-page-turn-effect-set-radius
effect a <clutter-page-turn-effect>
ret the radius of the page curling

Since 1.4
50 ClutterPaintNode

Paint objects

50.1 Overview

<clutter-paint-node> is an element in the render graph.

The render graph contains all the elements that need to be painted by Clutter when submitting a frame to the graphics system.

The render graph is distinct from the scene graph: the scene graph is composed by actors, which can be visible or invisible; the scene graph elements also respond to events. The render graph, instead, is only composed by nodes that will be painted.

Each <clutter-actor> can submit multiple <clutter-paint-node>s to the render graph.

50.2 Usage

clutter-paint-node-set-name (self <clutter-paint-node>) [Function]

(set-name mchars)

set-name [Method]

Sets a user-readable name for node.
The name will be used for debugging purposes.
The node will copy the passed string.

node a <clutter-paint-node>
name a string annotating the node

Since 1.10

clutter-paint-node-add-child (self <clutter-paint-node>) [Function]

(child <clutter-paint-node>)

add-child [Method]

Adds child to the list of children of node.
This function will acquire a reference on child.

node a <clutter-paint-node>
child the child <clutter-paint-node> to add

Since 1.10

clutter-paint-node-add-rectangle (self <clutter-paint-node>) [Function]

(rect <clutter-actor-box>)

add-rectangle [Method]

Adds a rectangle region to the node, as described by the passed rect.

node a <clutter-paint-node>
rect a <clutter-actor-box>

Since 1.10
51 Paint Nodes

ClutterPaintNode implementations

51.1 Overview

Clutter provides a set of predefined <clutter-paint-node> implementations that cover all the state changes available.

51.2 Usage

clutter-color-node-new (self <clutter-color>) ⇒ (ret <clutter-paint-node>)

Creates a new <clutter-paint-node> that will paint a solid color fill using color.

color the color to paint, or ‘#f’.

ret the newly created <clutter-paint-node>. Use clutter-paint-node-unref when done.

Since 1.10

clutter-text-node-new (layout <pango-layout>) ⇒ (ret <clutter-paint-node>)

Creates a new <clutter-paint-node> that will paint a <pango-layout> with the given color.

This function takes a reference on the passed layout, so it is safe to call g-object-unref after it returns.

layout a <pango-layout>, or ‘#f’.

color the color used to paint the layout, or ‘#f’.

ret the newly created <clutter-paint-node>. Use clutter-paint-node-unref when done.

Since 1.10

clutter-clip-node-new ⇒ (ret <clutter-paint-node>)

Creates a new <clutter-paint-node> that will clip its child nodes to the 2D regions added to it.

ret the newly created <clutter-paint-node>. Use clutter-paint-node-unref when done.

Since 1.10
52 ClutterPathConstraint

A constraint that follows a path

52.1 Overview

<clutter-path-constraint> is a simple constraint that modifies the allocation of the <clutter-actor> to which it has been applied using a <clutter-path>.

By setting the "offset" property it is possible to control how far along the path the <clutter-actor> should be.

ClutterPathConstraint is available since Clutter 1.6.

52.2 Usage

clutter-path-constraint-new (self <clutter-path>) (offset float) ⇒ (ret <clutter-constraint>)

constraint-new

Creates a new <clutter-path-constraint> with the given path and offset

path a <clutter-path>, or '#f'.
offset the offset along the <clutter-path>
ret the newly created <clutter-path-constraint>.

Since 1.6

clutter-path-constraint-set-path (self <clutter-path-constraint>) (path <clutter-path>)

set-path

Sets the path to be followed by the <clutter-path-constraint>.

The constraint will take ownership of the <clutter-path> passed to this function.

constraint a <clutter-path-constraint>
path a <clutter-path>.

Since 1.6

clutter-path-constraint-get-path (self <clutter-path-constraint>) ⇒ (ret <clutter-path>)

get-path

Retrieves a pointer to the <clutter-path> used by constraint.

constraint a <clutter-path-constraint>
ret the <clutter-path> used by the <clutter-path-constraint>, or '#f'.

The returned <clutter-path> is owned by the constraint and it should not be unreferenced.

Since 1.6
clutter-path-constraint-set-offset
\[ (self <\text{clutter-path-constraint}>) \text{ (offset float)} \]\n
set-offset
Sets the offset along the \text{<clutter-path>} used by \text{constraint}.

\text{constraint} \quad \text{a <clutter-path-constraint>}
\text{offset} \quad \text{the offset along the path}

Since 1.6

clutter-path-constraint-get-offset
\[ (self <\text{clutter-path-constraint}>) \Rightarrow (ret float) \]\n
get-offset
Retrieves the offset along the \text{<clutter-path>} used by \text{constraint}.

\text{constraint} \quad \text{a <clutter-path-constraint>}
\text{ret} \quad \text{the offset}

Since 1.6
53 ClutterPath

An object describing a path with straight lines and bezier curves.

53.1 Overview

A `<clutter-path>` contains a description of a path consisting of straight lines and bezier curves. This can be used in a `<clutter-behaviour-path>` to animate an actor moving along the path.

The path consists of a series of nodes. Each node is one of the following four types:

- `CLUTTER_PATH_LINE_TO`
- `CLUTTER_PATH_CURVE_TO`
- `CLUTTER_PATH_CLOSE`

Changes the position of the path to the given pair of coordinates. This is usually used as the first node of a path to mark the start position. If it is used in the middle of a path then the path will be disjoint and the actor will appear to jump to the new position when animated.

Creates a straight line from the previous point to the given point.

Creates a bezier curve. The end of the last node is used as the first control point and the three subsequent coordinates given in the node as used as the other three.

Creates a straight line from the last node to the last `CLUTTER_PATH_CLOSE` node. This can be used to close a path so that it will appear as a loop when animated.

The first three types have the corresponding relative versions `CLUTTER_PATH_REL_LINE_TO`, `CLUTTER_PATH_REL_CURVE_TO`. These are exactly the same except the coordinates are given relative to the previous node instead of as direct screen positions.

You can build a path using the node adding functions such as `clutter-path-add-line-to`. Alternatively the path can be described in a string using a subset of the SVG path syntax. See `clutter-path-add-string` for details.

`<clutter-path>` is available since Clutter 1.0

53.2 Usage

```plaintext
clutter-path-new \( \Rightarrow (\text{ret} \ <\text{clutter-path}>)) \) [Function]
```

Creates a new `<clutter-path>` instance with no nodes.

The object has a floating reference so if you add it to a `<clutter-behaviour-path>` then you do not need to unref it.

`ret` the newly created `<clutter-path>`

Since 1.0

```plaintext
clutter-path-new-with-description \( (\text{desc} \ \text{mchars}) \Rightarrow (\text{ret} \ <\text{clutter-path}>)) \) [Function]
```

Creates a new `<clutter-path>` instance with the nodes described in desc. See `clutter-path-add-string` for details of the format of the string.
The object has a floating reference so if you add it to a `<clutter-behaviour-path>` then you do not need to unref it.

**desc**  
a string describing the path

**ret**  
the newly created `<clutter-path>`

Since 1.0

```python
clutter-path-add-move-to (self <clutter-path>) (x int) (y int)  
add-move-to
```

Adds a `CLUTTER_PATH_MOVE_TO` type node to the path. This is usually used as the first node in a path. It can also be used in the middle of the path to cause the actor to jump to the new coordinate.

**path**  
a `<clutter-path>`

**x**  
the x coordinate

**y**  
the y coordinate

Since 1.0

```python
clutter-path-add-rel-move-to (self <clutter-path>) (x int) (y int)  
add-rel-move-to
```

Same as `clutter-path-add-move-to` except the coordinates are relative to the previous node.

**path**  
a `<clutter-path>`

**x**  
the x coordinate

**y**  
the y coordinate

Since 1.0

```python
clutter-path-add-line-to (self <clutter-path>) (x int) (y int)  
add-line-to
```

 Adds a `CLUTTER_PATH_LINE_TO` type node to the path. This causes the actor to move to the new coordinates in a straight line.

**path**  
a `<clutter-path>`

**x**  
the x coordinate

**y**  
the y coordinate

Since 1.0

```python
clutter-path-add-rel-line-to (self <clutter-path>) (x int) (y int)  
add-rel-line-to
```

Same as `clutter-path-add-line-to` except the coordinates are relative to the previous node.

**path**  
a `<clutter-path>`
clutter-path-add-curve-to (self <clutter-path>) (x_1 int) (y_1 int) (x_2 int) (y_2 int) (x_3 int) (y_3 int)

clutter-path-add-rel-curve-to (self <clutter-path>) (x_1 int) (y_1 int) (x_2 int) (y_2 int) (x_3 int) (y_3 int)

clutter-path-add-close (self <clutter-path>)

x the x coordinate
y the y coordinate

Since 1.0

add-curve-to
Adds a 'CLUTTER_PATH_CURVE_TO' type node to the path. This causes the actor to follow a bezier from the last node to (x-3, y-3) using (x-1, y-1) and (x-2, y-2) as control points.

path a <clutter-path>

x-1 the x coordinate of the first control point
y-1 the y coordinate of the first control point
x-2 the x coordinate of the second control point
y-2 the y coordinate of the second control point
x-3 the x coordinate of the third control point
y-3 the y coordinate of the third control point

Since 1.0

add-rel-curve-to
Same as clutter-path-add-curve-to except the coordinates are relative to the previous node.

path a <clutter-path>

x-1 the x coordinate of the first control point
y-1 the y coordinate of the first control point
x-2 the x coordinate of the second control point
y-2 the y coordinate of the second control point
x-3 the x coordinate of the third control point
y-3 the y coordinate of the third control point

Since 1.0

add-close
Adds a 'CLUTTER_PATH_CLOSE' type node to the path. This creates a straight line from the last node to the last 'CLUTTER_PATH_MOVE_TO' type node.

path a <clutter-path>

Since 1.0
clutter-path-add-string \texttt{(self clutter-path) (str mchars)} \textbf{[Function]}
\[ \Rightarrow (ret \ boole) \]

\textbf{add-string} \textbf{[Method]}

Adds new nodes to the end of the path as described in \textit{str}. The format is a subset of the SVG path format. Each node is represented by a letter and is followed by zero, one or three pairs of coordinates. The coordinates can be separated by spaces or a comma. The types are:

\begin{itemize}
  \item \texttt{L}
  \item \texttt{C}
  \item \texttt{z}
  \item \texttt{Z}
\end{itemize}

- Adds a ‘CLUTTER\_PATH\_MOVE\_TO’ node. Takes one pair of coordinates.
- Adds a ‘CLUTTER\_PATH\_LINE\_TO’ node. Takes one pair of coordinates.
- Adds a ‘CLUTTER\_PATH\_CURVE\_TO’ node. Takes three pairs of coordinates.
- Adds a ‘CLUTTER\_PATH\_CLOSE’ node. No coordinates are needed.

The M, L and C commands can also be specified in lower case which means the coordinates are relative to the previous node.

For example, to move an actor in a 100 by 100 pixel square centered on the point 300,300 you could use the following path:

\begin{verbatim}
M 250,350 l 0 -100 L 350,250 l 0 100 z
\end{verbatim}

If the path description isn’t valid ‘\#f’ will be returned and no nodes will be added.

\begin{itemize}
  \item \textit{path} \quad a clutter-path
  \item \textit{str} \quad a string describing the new nodes
  \item \textit{ret} \quad ‘\#t’ is the path description was valid or ‘\#f’ otherwise.
\end{itemize}

Since 1.0

\textbf{clutter-path-add-node \texttt{(self clutter-path) (node clutter-path-node)}} \textbf{[Function]}

\textbf{add-node} \textbf{[Method]}

Adds \textit{node} to the end of the path.

- \textit{path} \quad a clutter-path
- \textit{node} \quad a clutter-path-node

Since 1.0

\textbf{clutter-path-add-cairo-path \texttt{(self clutter-path) (cpath cairo-path-t)}} \textbf{[Function]}

\textbf{add-cairo-path} \textbf{[Method]}

Add the nodes of the Cairo path to the end of \textit{path}.

- \textit{path} \quad a clutter-path
- \textit{cpath} \quad a Cairo path

Since 1.0
clutter-path-get-n-nodes (self <clutter-path>) ⇒ (ret unsigned-int)

get-n-nodes [Method]
Retrieves the number of nodes in the path.
path a <clutter-path>
ret the number of nodes.
Since 1.0

clutter-path-get-node (self <clutter-path>)
(index unsigned-int) (node <clutter-path-node>)

get-node [Method]
Retrieves the node of the path indexed by index.
path a <clutter-path>
index the node number to retrieve
node a location to store a copy of the node.
Since 1.0

clutter-path-get-nodes (self <clutter-path>) ⇒ (ret gslist-of)

get-nodes [Method]
Returns a <gs-list> of <clutter-path-node>s. The list should be freed with gslist-free. The nodes are owned by the path and should not be freed. Altering the path may cause the nodes in the list to become invalid so you should copy them if you want to keep the list.
path a <clutter-path>
ret a list of nodes in the path.
Since 1.0

clutter-path-insert-node (self <clutter-path>) (index int)
(node <clutter-path-node>)

insert-node [Method]
Inserts node into the path before the node at the given offset. If index is negative it will append the node to the end of the path.
path a <clutter-path>
index offset of where to insert the node
node the node to insert
Since 1.0

clutter-path-remove-node (self <clutter-path>)
(index unsigned-int)

remove-node [Method]
Removes the node at the given offset from the path.
path a <clutter-path>
**clutter-path-replace-node**

- **Function**: `clutter-path-replace-node (self <clutter-path>) (index unsigned-int) (node <clutter-path-node>)`
- **Method**: `replace-node`

Replaces the node at offset `index` with `node`.

- `path`: a `<clutter-path>`
- `index`: index to the existing node
- `node`: the replacement node

**clutter-path-get-description**

- **Function**: `clutter-path-get-description (self <clutter-path>)` ⇒ `(ret mchars)`
- **Method**: `get-description`

Returns a newly allocated string describing the path in the same format as used by `clutter-path-add-string`.

- `path`: a `<clutter-path>`
- `ret`: a string description of the path. Free with `g-free`.

**clutter-path-set-description**

- **Function**: `clutter-path-set-description (self <clutter-path>) (str mchars)` ⇒ `(ret bool)`
- **Method**: `set-description`

Replaces all of the nodes in the path with nodes described by `str`. See `clutter-path-add-string` for details of the format.

If the string is invalid then ‘#f’ is returned and the path is unaltered.

- `path`: a `<clutter-path>`
- `str`: a string describing the path
- `ret`: ‘#t’ is the path was valid, ‘#f’ otherwise.

**clutter-path-to-cairo-path**

- **Function**: `clutter-path-to-cairo-path (self <clutter-path>) (cr cairo-t)`
- **Method**: `to-cairo-path`

Add the nodes of the ClutterPath to the path in the Cairo context.

- `path`: a `<clutter-path>`
- `cr`: a Cairo context

Since 1.0
clutter-path-clear \((self <clutter-path>)\)

**clear**

Removes all nodes from the path.

\(path\) a \(<clutter-path>\)

Since 1.0

clutter-path-get-position \((self <clutter-path>)\)

\((progress \text{ double}) (position <clutter-knot>) \Rightarrow (ret \text{ unsigned-int})\)

**get-position**

The value in \(progress\) represents a position along the path where 0.0 is the beginning and 1.0 is the end of the path. An interpolated position is then stored in \(position\).

\(path\) a \(<clutter-path>\)

\(progress\) a position along the path as a fraction of its length

\(position\) location to store the position.

\(ret\) index of the node used to calculate the position.

Since 1.0

clutter-path-get-length \((self <clutter-path>)\)

\(\Rightarrow (ret \text{ unsigned-int})\)

**get-length**

Retrieves an approximation of the total length of the path.

\(path\) a \(<clutter-path>\)

\(ret\) the length of the path.

Since 1.0

clutter-path-node-equal \((self <clutter-path-node>)\)

\((node\_b <clutter-path-node>) \Rightarrow (ret \text{ bool})\)

Compares two nodes and checks if they are the same type with the same coordinates.

\(node\_a\) First node

\(node\_b\) Second node

\(ret\) ‘#t’ if the nodes are the same.

Since 1.0
54 ClutterPropertyTransition

Property transitions

54.1 Overview

<clutter-property-transition> is a specialized <clutter-transition> that can be used to tween a property of a <clutter-animatable> instance.

<clutter-property-transition> is available since Clutter 1.10

54.2 Usage

clutter-property-transition-new (property_name mchars) ⇒ (ret <clutter-transition>)
Creates a new <clutter-property-transition>.

property-name
   a property of animatable, or ‘#f’.

ret
   the newly created <clutter-property-transition>. Use g-object-unref when done.

Since 1.10
55 ClutterScript

Loads a scene from UI definition data

55.1 Overview

<clutter-script> is an object used for loading and building parts or a complete scenegraph from external definition data in forms of string buffers or files.

The UI definition format is JSON, the JavaScript Object Notation as described by RFC 4627. <clutter-script> can load a JSON data stream, parse it and build all the objects defined into it. Each object must have an "id" and a "type" properties defining the name to be used to retrieve it from <clutter-script> with clutter-script-get-object, and the class type to be instanciated. Every other attribute will be mapped to the class properties.

A <clutter-script> holds a reference on every object it creates from the definition data, except for the stage. Every non-actor object will be finalized when the <clutter-script> instance holding it will be finalized, so they need to be referenced using g-object-ref in order for them to survive.

A simple object might be defined as:

```json
{
  "id" : "red-button",
  "type" : "ClutterRectangle",
  "width" : 100,
  "height" : 100,
  "color" : "#ff0000ff"
}
```

This will produce a red <clutter-rectangle>, 100x100 pixels wide, and with a ClutterScript id of "red-button"; it can be retrieved by calling:

```c
ClutterActor *red_button;
red_button = CLUTTER_ACTOR (clutter_script_get_object (script, "red-button"));
```

and then manipulated with the Clutter API. For every object created using ClutterScript it is possible to check the id by calling clutter-get-script-id.

Packing can be represented using the "children" member, and passing an array of objects or ids of objects already defined (but not packed: the packing rules of Clutter still apply, and an actor cannot be packed in multiple containers without unparenting it in between).

Behaviours and timelines can also be defined inside a UI definition buffer:

```json
{
  "id" : "rotate-behaviour",
  "type" : "ClutterBehaviourRotate",
  "angle-start" : 0.0,
  "angle-end" : 360.0,
  "axis" : "z-axis",
```
"alpha" : {
  "timeline" : { "duration" : 4000, "loop" : true },
  "mode" : "easeInSine"
}
}

And then to apply a defined behaviour to an actor defined inside the definition of an actor, the "behaviour" member can be used:

{
  "id" : "my-rotating-actor",
  "type" : "ClutterTexture",
  ...
  "behaviours" : [ "rotate-behaviour" ]
}

A <clutter-alpha> belonging to a <clutter-behaviour> can only be defined implicitly like in the example above, or explicitly by setting the "alpha" property to point to a previously defined <clutter-alpha>, e.g.:

{
  "id" : "rotate-behaviour",
  "type" : "ClutterBehaviourRotate",
  "angle-start" : 0.0,
  "angle-end" : 360.0,
  "axis" : "z-axis",
  "alpha" : {
    "id" : "rotate-alpha",
    "type" : "ClutterAlpha",
    "timeline" : {
      "id" : "rotate-timeline",
      "type" : "ClutterTimeline",
      "duration" : 4000,
      "loop" : true
    },
    "function" : "custom_sine_alpha"
  }
}

Implicitly defined <clutter-alpha>s and <clutter-timeline>s can omit the well as the clutter-script-get-object (they can, however, be extracted using the <clutter-behaviour> and <clutter-alpha> API respectively).

Signal handlers can be defined inside a Clutter UI definition file and then autoconnected to their respective signals using the clutter-script-connect-signals function:

...  
  "signals" : [
    { "name" : "button-press-event", "handler" : "on_button_press" },
  ]
}
Signal handler definitions must have a "name" and a "handler" members; they can also have the "after" and "swapped" boolean members (for the signal connection flags ‘G_CONNECT_AFTER’ and ‘G_CONNECT_SWAPPED’ respectively) and the "object" string member for calling g-signal-connect-object instead of g-signal-connect.

Signals can also be directly attached to a specific state defined inside a <clutter-state> instance, for instance:

```
... "signals" : [ 
    { 
      "name" : "enter-event",
      "states" : "button-states",
      "target-state" : "hover"
    },
    { 
      "name" : "leave-event",
      "states" : "button-states",
      "target-state" : "base"
    },
    { 
      "name" : "button-press-event",
      "states" : "button-states",
      "target-state" : "active",
      "warp" : true
    }
  ],
...```
using a '#f' name. The "warp" key can be used to warp to a specific state instead of animating to it. State changes on signal emission will not affect the signal emission chain.

Clutter reserves the following names, so classes defining properties through the usual GObject registration process should avoid using these names to avoid collisions:

"id" := the unique name of a ClutterScript object
"type" := the class literal name, also used to infer the type function
"type_func" := the GType function name, for non-standard classes
"children" := an array of names or objects to add as children
"behaviours" := an array of names or objects to apply to an actor
"signals" := an array of signal definitions to connect to an object
"is-default" := a boolean flag used when defining the #ClutterStage; if set to "true" the default stage will be used instead of creating a new #ClutterStage instance

<clutter-script> is available since Clutter 0.6

55.2 Usage

clutter-script-new ⇒ (ret <clutter-script>) [Function]
Creates a new <clutter-script> instance. <clutter-script> can be used to load objects definitions for scenegraph elements, like actors, or behavioural elements, like behaviours and timelines. The definitions must be encoded using the JavaScript Object Notation (JSON) language.

ret the newly created <clutter-script> instance. Use g-object_unref when done.

Since 0.6

clutter-script-load-from-data (self <clutter-script>) [Function]
load-from-data (data mchars) (length ssize_t) ⇒ (ret unsigned-int)
Loads the definitions from data into script and merges with the currently loaded ones, if any.

script a <clutter-script>
data a buffer containing the definitions
length the length of the buffer, or -1 if data is a NUL-terminated buffer
error return location for a <g-error>, or '#f'
ret on error, zero is returned and error is set accordingly. On success, the merge id for the UI definitions is returned. You can use the merge id with clutter-script-unmerge-objects.

Since 0.6
clutter-script-load-from-file (self <clutter-script>) [Function]
    (filename mchars) ⇒ (ret unsigned-int)
load-from-file [Method]
    Loads the definitions from filename into script and merges with the currently loaded ones, if any.
    
    script a <clutter-script>
    filename the full path to the definition file
    error return location for a <g-error>, or ‘#f’
    ret on error, zero is returned and error is set accordingly. On success, the merge id for the UI definitions is returned. You can use the merge id with clutter-script-unmerge-objects.

Since 0.6

clutter-script-load-from-resource (self <clutter-script>) [Function]
    (resource-path mchars) ⇒ (ret unsigned-int)
load-from-resource [Method]
    Loads the definitions from a resource file into script and merges with the currently loaded ones, if any.
    
    script a <clutter-script>
    resource-path the resource path of the file to parse
    error return location for a <g-error>, or ‘#f’
    ret on error, zero is returned and error is set accordingly. On success, the merge id for the UI definitions is returned. You can use the merge id with clutter-script-unmerge-objects.

Since 1.10

clutter-script-lookup-filename (self <clutter-script>) [Function]
    (filename mchars) ⇒ (ret mchars)
lookup-filename [Method]
    Looks up filename inside the search paths of script. If filename is found, its full path will be returned.
    
    script a <clutter-script>
    filename the name of the file to lookup
    ret the full path of filename or ‘#f’ if no path was found.

Since 0.8

clutter-script-get-object (self <clutter-script>) [Function]
    (name mchars) ⇒ (ret <gobject>)
get-object [Method]
    Retrieves the object bound to name. This function does not increment the reference count of the returned object.
script  a <clutter-script>
name  the name of the object to retrieve
ret  the named object, or ‘#f’ if no object with the given name was available.

Since 0.6

**clutter-script-unmerge-objects** *(self <clutter-script>)*  [Function]

**(merge_id unsigned-int)**  [Method]

unmerge-objects

Unmerges the objects identified by merge-id.

script  a <clutter-script>
merge-id  merge id returned when loading a UI definition

Since 0.6

**clutter-script-ensure-objects** *(self <clutter-script>)*  [Function]

**[Method]**

ensure-objects

Ensure that every object defined inside script is correctly constructed. You should rarely need to use this function.

script  a <clutter-script>

Since 0.6

**clutter-script-list-objects** *(self <clutter-script>)*  [Function]

⇒  *(ret glist-of)*  [Method]

list-objects

Retrieves all the objects created by script.

Note: this function does not increment the reference count of the objects it returns.

script  a <clutter-script>
ret  a list of <gobject>s, or ‘#f’. The objects are owned by the <clutter-script> instance. Use g-list-free on the returned list when done.

Since 0.8.2

**clutter-script-add-states** *(self <clutter-script>)*  [Function]

**(name mchars) (state <clutter-state>)**  [Method]

add-states

Associates a <clutter-state> to the <clutter-script> instance using the given name.

The <clutter-script> instance will use state to resolve target states when connecting signal handlers.

The <clutter-script> instance will take a reference on the <clutter-state> passed to this function.

script  a <clutter-script>
name  a name for the state, or ‘#f’ to set the default <clutter-state>.
state  a <clutter-state>

Since 1.8
clutter-script-get-states (self <clutter-script>)  
(name mchars) ⇒ (ret <clutter-state>)  

get-states  
Retrieves the <clutter-state> for the given state-name.  
If name is '#f', this function will return the default <clutter-state> instance.

script  a <clutter-script>  
name  the name of the <clutter-state>, or '#f'.  
ret  a pointer to the <clutter-state> for the given name. The <clutter-state> is owned by the <clutter-script> instance and it should not be unreferenced.

Since 1.8

clutter-script-get-type-from-name (self <clutter-script>)  
(type_name mchars) ⇒ (ret <gtype>)  

get-type-from-name  
Looks up a type by name, using the virtual function that <clutter-script> has for that purpose. This function should rarely be used.

script  a <clutter-script>  
type-name  name of the type to look up  
ret  the type for the requested type name, or 'G_TYPE_INVALID' if corresponding type was found.

Since 0.6

clutter-get-script-id (gobject <gobject>) ⇒ (ret mchars)  
Retrieves the Clutter script id, if any.

gobject  a <gobject>  
ret  the script id, or '#f' if object was not defined inside a UI definition file. The returned string is owned by the object and should never be modified or freed.

Since 0.6
56 ClutterScriptable

Override the UI definition parsing

56.1 Overview

The `<clutter-scriptable-iface>` interface exposes the UI definition parsing process to external classes. By implementing this interface, a class can override the UI definition parsing and transform complex data types into GObject properties, or allow custom properties.

`<clutter-scriptable>` is available since Clutter 0.6

56.2 Usage
57 ClutterSettings

Settings configuration

57.1 Overview

Clutter depends on some settings to perform operations like detecting multiple button press events, or font options to render text.

Usually, Clutter will strive to use the platform’s settings in order to be as much integrated as possible. It is, however, possible to change these settings on a per-application basis, by using the `<clutter-settings>` singleton object and setting its properties. It is also possible, for toolkit developers, to retrieve the settings from the `<clutter-settings>` properties when implementing new UI elements, for instance the default font name.

`<clutter-settings>` is available since Clutter 1.4

57.2 Usage

```clutter
clutter-settings-get-default ⇒ (ret <clutter-settings>) [Function]
```

Retrieves the singleton instance of `<clutter-settings>`

`ret` the instance of `<clutter-settings>`. The returned object is owned by Clutter and it should not be unreferenced directly.

Since 1.4
ClutterShaderEffect

Base class for shader effects

58.1 Overview
<clutter-shader-effect> is a class that implements all the plumbing for creating <clutter-effect>s using GLSL shaders.

<clutter-shader-effect> creates an offscreen buffer and then applies the GLSL shader (after checking whether the compilation and linking were successful) to the buffer before painting it on screen.

58.2 Implementing a ClutterShaderEffect

Creating a sub-class of <clutter-shader-effect> requires the overriding of the paint-target virtual function from the <clutter-offscreen-effect> class as well as the get-static-shader-source virtual from the <clutter-shader-effect> class.

The get-static-shader-source function should return a copy of the shader source to use. This function is only called once per subclass of <clutter-shader-effect> regardless of how many instances of the effect are created. The source for the shader is typically stored in a static const string which is returned from this function via g_strdup.

The paint-target should set the shader’s uniforms if any. This is done by calling clutter-shader-effect-set-uniform-value or clutter-shader-effect-set-uniform. The sub-class should then chain up to the <clutter-shader-effect> implementation.

The example below shows a typical implementation of the get-static-shader-source and paint-target phases of a <clutter-shader-effect> sub-class.

```c
static gchar *
my_effect_get_static_shader_source (ClutterShaderEffect *effect)
{
    return g_strdup (shader_source);
}

static gboolean
my_effect_paint_target (ClutterOffscreenEffect *effect)
{
    MyEffect *self = MY_EFFECT (effect);
    ClutterShaderEffect *shader = CLUTTER_SHADER_EFFECT (effect);
    ClutterEffectClass *parent_class;
    gfloat component_r, component_g, component_b;

    /\n    \#x002A; the "tex" uniform is declared in the shader as:
    /\n    \#x002A;
    /\n    \#x002A;
    /\n    \#x002A; uniform int tex;
    /\n    \#x002A;
    /\n    \#x002A; and it is passed a constant value of 0
```
clutter_shader_effect_set_uniform (shader, "tex", G_TYPE_INT, 1, 0);

/* the "component" uniform is declared in the shader as:
 component_r = self->color.red / 255.0f;
 component_g = self->color.green / 255.0f;
 component_b = self->color.blue / 255.0f;
 clutter_shader_effect_set_uniform (shader, "component",
 G_TYPE_FLOAT, 3,
 component_r, component_g, component_b);

 parent_class = CLUTTER_OFFSCREEN_EFFECT_CLASS (my_effect_parent_class);
 return parent_class->paint_target (effect); */

<clutter-shader-effect> is available since Clutter 1.4

58.3 Usage

clutter-shader-effect-new (shader_type <clutter-shader-type>) [Function]
 ⇒ (ret <clutter-effect>)
Creates a new <clutter-shader-effect>, to be applied to an actor using clutter-actor-add-effect.
The effect will be empty until clutter-shader-effect-set-shader-source is called.

shader-type
 the type of the shader, either ‘CLUTTER_FRAGMENT_SHADER’, or
 ‘CLUTTER_VERTEX_SHADER’

ret
 the newly created <clutter-shader-effect>. Use g-object-unref when done.

Since 1.8
59 Shaders

Programmable pipeline abstraction

59.1 Overview

<clutter-shader> is an object providing an abstraction over the OpenGL programmable pipeline. By using <clutter-shader>s is possible to override the drawing pipeline by using small programs also known as "shaders".

<clutter-shader> is available since Clutter 0.6.

<clutter-shader> is deprecated since Clutter 1.8; use <clutter-shader-effect> in newly written code.

59.2 Usage
60 ClutterSnapConstraint

A constraint snapping two actors together

60.1 Overview

<clutter-snap-constraint> is a constraint that snaps the edges of two actors together, expanding the actor’s allocation if necessary.

An offset can be applied to the constraint, to provide spacing.

<clutter-snap-constraint> is available since Clutter 1.6

60.2 Usage

clutter-snap-constraint-new (source <clutter-actor>) [Function]
(from_edge <clutter-snap-edge>) (to_edge <clutter-snap-edge>)
(offset float) ⇒ (ret <clutter-constraint>)

Creates a new <clutter-snap-constraint> that will snap a <clutter-actor> to the edge of source, with the given offset.

source the <clutter-actor> to use as the source of the constraint, or ‘#f’.
from-edge the edge of the actor to use in the constraint
to-edge the edge of source to use in the constraint
offset the offset to apply to the constraint, in pixels
ret the newly created <clutter-snap-constraint>

Since 1.6

clutter-snap-constraint-set-source [Function]
(self <clutter-snap-constraint>) (source <clutter-actor>)
set-source [Method]

Sets the source <clutter-actor> for the constraint

constraint a <clutter-snap-constraint>
source a <clutter-actor>, or ‘#f’ to unset the source.

Since 1.6

clutter-snap-constraint-get-source [Function]
(self <clutter-snap-constraint>) ⇒ (ret <clutter-actor>)
get-source [Method]

Retrieves the <clutter-actor> set using clutter-snap-constraint-set-source

constraint a <clutter-snap-constraint>
ret a pointer to the source actor.

Since 1.6
clutter-snap-constraint-set-edges [Function]

(self clutter-snap-constraint> (from_edge clutter-snap-edge>)
(to_edge clutter-snap-edge>)

set-edges [Method]
Sets the edges to be used by the constraint.

The from-edge is the edge on the <clutter-actor> to which constraint has been added. The to-edge is the edge of the <clutter-actor> inside the <"source"> property.

constraint a clutter-snap-constraint>
from-edge the edge on the actor
to-edge the edge on the source

Since 1.6

clutter-snap-constraint-get-edges [Function]

(self clutter-snap-constraint>)
⇒ (from_edge clutter-snap-edge>) (to_edge clutter-snap-edge>)

get-edges [Method]
Retrieves the edges used by the constraint.

constraint a clutter-snap-constraint>
from-edge return location for the actor’s edge, or ‘#f’.
to-edge return location for the source’s edge, or ‘#f’.

Since 1.6

clutter-snap-constraint-set-offset [Function]

(self clutter-snap-constraint>) (offset float)

set-offset [Method]
Sets the offset to be applied to the constraint.

constraint a clutter-snap-constraint>
offset the offset to apply, in pixels

Since 1.6

clutter-snap-constraint-get-offset [Function]

(self clutter-snap-constraint>) ⇒ (ret float)

get-offset [Method]
Retrieves the offset set using clutter-snap-constraint-set-offset.

constraint a clutter-snap-constraint>
ret the offset, in pixels

Since 1.6
61 Stage Manager

Maintains the list of stages

61.1 Overview

<clutter-stage-manager> is a singleton object, owned by Clutter, which maintains the list of currently active stages.

Every newly-created <clutter-stage> will cause the emission of the "stage-added" signal; once a <clutter-stage> has been destroyed, the "stage-removed" signal will be emitted.

<clutter-stage-manager> is available since Clutter 0.8

61.2 Usage

clutter-stage-manager-get-default [Function]

⇒ (ret <clutter-stage-manager>)

Returns the default <clutter-stage-manager>.

ret the default stage manager instance. The returned object is owned by Clutter and you should not reference or unreference it.

Since 0.8

clutter-stage-manager-list-stages [Function]

(self <clutter-stage-manager>) ⇒ (ret glist-of)

list-stages [Method]

Lists all currently used stages.

stage-manager a <clutter-stage-manager>

ret a newly allocated list of <clutter-stage> objects. Use g-list-free to deallocate it when done.

Since 0.8

clutter-stage-manager-peek-stages [Function]

(self <clutter-stage-manager>) ⇒ (ret glist-of)

peek-stages [Method]

Lists all currently used stages.

stage-manager a <clutter-stage-manager>

ret a pointer to the internal list of <clutter-stage> objects. The returned list is owned by the <clutter-stage-manager> and should never be modified or freed.

Since 1.0
62 ClutterStage

Top level visual element to which actors are placed.

62.1 Overview

<clutter-stage> is a top level ‘window’ on which child actors are placed and manipulated.

Backends might provide support for multiple stages. The support for this feature can be checked at run-time using the clutter-feature-available function and the ‘CLUTTER_FEATURE_STAGE_MULTIPLE’ flag. If the backend used supports multiple stages, new <clutter-stage> instances can be created using clutter-stage-new. These stages must be managed by the developer using clutter-actor-destroy, which will take care of destroying all the actors contained inside them.

<clutter-stage> is a proxy actor, wrapping the backend-specific implementation of the windowing system. It is possible to subclass <clutter-stage>, as long as every overridden virtual function chains up to the parent class corresponding function.

62.2 Usage

clutter-stage-new ⇒ (ret <clutter-actor>)  [Function]

Creates a new, non-default stage. A non-default stage is a new top-level actor which can be used as another container. It works exactly like the default stage, but while clutter-stage-get-default will always return the same instance, you will have to keep a pointer to any <clutter-stage> returned by clutter-stage-new.

The ability to support multiple stages depends on the current backend. Use clutter-feature-available and ‘CLUTTER_FEATURE_STAGE_MULTIPLE’ to check at runtime whether a backend supports multiple stages.

ret a new stage, or ‘#f’ if the default backend does not support multiple stages. Use clutter-actor-destroy to programmatically close the returned stage.

Since 0.8

clutter-stage-set-fullscreen (self <clutter-stage>)  [Function]

(fullscreen bool)

set-fullscreen  [Method]

Asks to place the stage window in the fullscreen or unfullscreen states.

( Note that you shouldn’t assume the window is definitely full screen afterward, because other entities (e.g. the user or window manager) could unfullscreen it again, and not all window managers honor requests to fullscreen windows.

If you want to receive notification of the fullscreen state you should either use the "fullscreen" and "unfullscreen" signals, or use the notify signal for the "fullscreen-set" property

stage a <clutter-stage>

fullscreen ‘#t’ to to set the stage fullscreen

Since 1.0
Chapter 62: ClutterStage

clutter-stage-get-fullscreen (self <clutter-stage>) [Function]
⇒ (ret bool)

get-fullscreen [Method]
Retrieves whether the stage is full screen or not

stage a <clutter-stage>
ret ‘#t’ if the stage is full screen

Since 1.0

clutter-stage-show-cursor (self <clutter-stage>) [Function]
show-cursor [Method]
Shows the cursor on the stage window

stage a <clutter-stage>

clutter-stage-hide-cursor (self <clutter-stage>) [Function]
hide-cursor [Method]
Makes the cursor invisible on the stage window

stage a <clutter-stage>

Since 0.4

clutter-stage-get-actor-at-pos (self <clutter-stage>) [Function]
(pick_mode <clutter-pick-mode>) (x int) (y int)
⇒ (ret <clutter-actor>)

get-actor-at-pos [Method]
Checks the scene at the coordinates x and y and returns a pointer to the <clutter-actor> at those coordinates.

By using pick-mode it is possible to control which actors will be painted and thus available.

stage a <clutter-stage>
pick-mode how the scene graph should be painted
x X coordinate to check
y Y coordinate to check
ret the actor at the specified coordinates, if any.

clutter-stage-ensure-current (self <clutter-stage>) [Function]
ensure-current [Method]
This function essentially makes sure the right GL context is current for the passed stage. It is not intended to be used by applications.

stage the <clutter-stage>

Since 0.8
clutter-stage-ensure-viewport (self <clutter-stage>)

**ensure-viewport**

Ensures that the GL viewport is updated with the current stage window size.

This function will queue a redraw of stage.

This function should not be called by applications; it is used when embedding a <clutter-stage> into a toolkit with another windowing system, like GTK+.

```
stage  a <clutter-stage>
```

Since 1.0

clutter-stage-ensure-redraw (self <clutter-stage>)

**ensure-redraw**

Ensures that stage is redrawn.

This function should not be called by applications; it is used when embedding a <clutter-stage> into a toolkit with another windowing system, like GTK+.

```
stage  a <clutter-stage>
```

Since 1.0

clutter-stage-event (self <clutter-stage>)

```
event <clutter-event> ⇒ (ret bool)
```

**event**

This function is used to emit an event on the main stage.

You should rarely need to use this function, except for synthetised events.

```
stage  a <clutter-stage>
event  a <clutter-event>
ret    the return value from the signal emission
```

Since 0.4

clutter-stage-set-key-focus (self <clutter-stage>)

```
(actor <clutter-actor>)
```

**set-key-focus**

Sets the key focus on actor. An actor with key focus will receive all the key events.

If actor is ‘#f’, the stage will receive focus.

```
stage  the <clutter-stage>
actor  the actor to set key focus to, or ‘#f’.
```

Since 0.6

clutter-stage-get-key-focus (self <clutter-stage>)

```
⇒ (ret <clutter-actor>)
```

**get-key-focus**

Retrieves the actor that is currently under key focus.

```
stage  the <clutter-stage>
ret    the actor with key focus, or the stage.
```
clutter-stage-set-use-alpha (self <clutter-stage>) [Function]
( use_alpha bool )
set-use-alpha [Method]
Sets whether the stage should honour the "opacity" and the alpha channel of the "color"

stage a <clutter-stage>
use-alpha whether the stage should honour the opacity or the alpha channel of the stage color

Since 1.2

clutter-stage-get-use-alpha (self <clutter-stage>) [Function]
⇒ ( ret bool )
get-use-alpha [Method]
Retrieves the value set using clutter-stage-set-use-alpha

stage a <clutter-stage>
ret '#t' if the stage should honour the opacity and the alpha channel of the stage color

Since 1.2

clutter-stage-set-minimum-size (self <clutter-stage>) [Function]
( width unsigned-int ) ( height unsigned-int )
set-minimum-size [Method]
Sets the minimum size for a stage window, if the default backend uses <clutter-stage> inside a window
This is a convenience function, and it is equivalent to setting the "min-width" and "min-height" on stage
If the current size of stage is smaller than the minimum size, the stage will be resized to the new width and height
This function has no effect if stage is fullscreen

stage a <clutter-stage>
width width, in pixels
height height, in pixels

Since 1.2

clutter-stage-get-minimum-size (self <clutter-stage>) [Function]
⇒ ( width unsigned-int ) ( height unsigned-int )
get-minimum-size [Method]
Retrieves the minimum size for a stage window as set using clutter-stage-set-minimum-size.
The returned size may not correspond to the actual minimum size and it is specific to the <clutter-stage> implementation inside the Clutter backend

stage a <clutter-stage>
width  return location for the minimum width, in pixels, or ‘#f’.
height return location for the minimum height, in pixels, or ‘#f’.

Since 1.2

clutter-stage-set-no-clear-hint (self <clutter-stage>)  [Function]
(set-no_clear bool)
set-no-clear-hint  [Method]
Sets whether the stage should clear itself at the beginning of each paint cycle or not.
Clearing the <clutter-stage> can be a costly operation, especially if the stage is always covered - for instance, in a full-screen video player or in a game with a background texture.
This setting is a hint; Clutter might discard this hint depending on its internal state.
If parts of the stage are visible and you disable clearing you might end up with visual artifacts while painting the contents of the stage.

stage  a <clutter-stage>
no-clear  ‘#t’ if the stage should not clear itself on every repaint cycle

Since 1.4

clutter-stage-get-no-clear-hint (self <clutter-stage>)  [Function]
⇒ (ret bool)
get-no-clear-hint  [Method]
Retrieves the hint set with clutter-stage-set-no-clear-hint

stage  a <clutter-stage>
ret  ‘#t’ if the stage should not clear itself on every paint cycle, and ‘#f’ otherwise

Since 1.4

clutter-stage-set-accept-focus (self <clutter-stage>)  [Function]
(set-accept_focus bool)
set-accept-focus  [Method]
Sets whether the stage should accept the key focus when shown.
This function should be called before showing stage using clutter-actor-show.

stage  a <clutter-stage>
accept-focus  ‘#t’ to accept focus on show

Since 1.6

clutter-stage-get-accept-focus (self <clutter-stage>)  [Function]
⇒ (ret bool)
get-accept-focus  [Method]
Retrieves the value set with clutter-stage-set-accept-focus.

stage  a <clutter-stage>
"ret" if the <clutter-stage> should accept focus, and "#f" otherwise

Since 1.6

**clutter-stage-set-perspective (self <clutter-stage>)**

(set-perspective <clutter-perspective>)

**set-perspective**

Sets the stage perspective. Using this function is not recommended because it will disable Clutter’s attempts to generate an appropriate perspective based on the size of the stage.

*stage* A <clutter-stage>

*perspective* A <clutter-perspective>

**clutter-stage-get-perspective (self <clutter-stage>)**

⇒ (ret scm)

**get-perspective**

Retrieves the stage perspective.

*stage* A <clutter-stage>

*perspective* return location for a <clutter-perspective>.

**clutter-stage-set-title (self <clutter-stage>) (title mchars)**

**set-title**

Sets the stage title.

*stage* A <clutter-stage>

*title* A utf8 string for the stage windows title.

Since 0.4

**clutter-stage-get-title (self <clutter-stage>) ⇒ (ret mchars)**

**get-title**

Gets the stage title.

*stage* A <clutter-stage>

*ret* pointer to the title string for the stage. The returned string is owned by the actor and should not be modified or freed.

Since 0.4

**clutter-stage-set-user-resizable (self <clutter-stage>)**

(resizable bool)

**set-user-resizable**

Sets if the stage is resizable by user interaction (e.g. via window manager controls)

*stage* a <clutter-stage>

*resizable* whether the stage should be user resizable.

Since 0.4
clutter-stage-get-user-resizable \( (\text{self} \ <\text{clutter-stage}>) \) \quad \text{[Function]}
\[ \Rightarrow (\text{ret} \ \text{bool}) \]

get-user-resizable

Retrieves the value set with clutter-stage-set-user-resizable.

\[ \text{stage} \quad \text{a} \ <\text{clutter-stage}> \]
\[ \text{ret} \quad \text{‘#t’ if the stage is resizable by the user.} \]

Since 0.4
63 ClutterState

State machine with animated transitions

63.1 Overview

<clutter-state> is an object controlling the tweening of properties on multiple actors between a set of named states. <clutter-state-key>s define how the properties are animated. If the source.state.name for a key is NULL it is used for transition to the target state unless a specific key exists for transitioning from the current state to the requested state.

The following example defines a "base" and a "hover" state in a <clutter-state> instance.

```c
ClutterState *state = clutter_state_new ();
ClutterColor color = { 0, };

/* transition from any state to the "base" state */
clutter_color_from_string (&color, "rgb(255, 0, 0)");
clutter_state_set (state, NULL, "base",
    actor, "color", CLUTTER_LINEAR, &color,
    actor, "scale-x", CLUTTER_EASE_IN_BOUNCE, 1.0,
    actor, "scale-y", CLUTTER_EASE_IN_BOUNCE, 1.0,
    NULL);

/* transition from the "base" state to the "hover" state */
clutter_color_from_string (&color, "rgb(0, 0, 255)");
clutter_state_set (state, "base", "hover",
    actor, "color", CLUTTER_LINEAR, &color,
    actor, "scale-x", CLUTTER_EASE_OUT_BOUNCE, 1.7,
    actor, "scale-y", CLUTTER_EASE_OUT_BOUNCE, 1.7,
    NULL);

/* the default duration of any transition */
clutter_state_set_duration (state, NULL, NULL, 500);

/* set "base" as the initial state */
clutter_state_warp_to_state (state, "base");
```

The actor then uses the <clutter-state> to animate through the two states using callbacks for the "enter-event" and "leave-event" signals.

```c
static gboolean
on_enter (ClutterActor *actor,
    ClutterEvent *event,
    ClutterState *state)
```
63.2 ClutterState description for `<clutter-script>`

`<clutter-state>` defines a custom `transitions` property which allows describing the states. The `transitions` property has the following syntax:

```
{
    "transitions" : [
    {
        "source" : "<source-state>",
        "target" : "<target-state>",
        "duration" : <milliseconds>,
        "keys" : [
            ["<object-id>",
             "<property-name>",
             "<easing-mode>",
             "<final-value>",
             ],
            ["<object-id>",
             "<property-name>",
             "<easing-mode>",
             "<final-value>",
             <pre-delay>,
             <post-delay>
            ],
            ...
        ],
    },
    {...
    }
}
```
Each element of the transitions array follows the same rules as clutter-state-set-key.

The source and target values control the source and target state of the transition. The key and animator are mutually exclusive. The pre-delay and post-delay values are optional.

The example below is a translation into a <clutter-script> definition of the code in the example above.

```json
{
    "id" : "button-state",
    "type" : "ClutterState",
    "duration" : 500,
    "transitions" : [
        {
            "source" : "*",
            "target" : "base",
            "keys" : [
                [ "button", "color", "linear", "rgb(255, 0, 0)" ],
                [ "button", "scale-x", "easeInBounce", 1.0 ],
                [ "button", "scale-y", "easeInBounce", 1.0 ]
            ]
        },
        {
            "source" : "base",
            "target" : "hover",
            "keys" : [
                [ "button", "color", "linear", "rgb(0, 0, 255)" ],
                [ "button", "scale-x", "easeOutBounce", 1.7 ],
                [ "button", "scale-y", "easeOutBounce", 1.7 ]
            ]
        }
    ]
}
```

<clutter-state> is available since Clutter 1.4.
63.3 Usage

**clutter-state-new** ⇒ (ret <clutter-state>)  
Creates a new <clutter-state>

ret the newly create <clutter-state> instance

**clutter-state-set-state** (self <clutter-state>)  
(target-state-name mchars) ⇒ (ret <clutter-timeline>)  
**set-state**  
Change the current state of <clutter-state> to target-state-name.

The state will animate during its transition, see <clutter-state-warp-to-state> for animation-free state switching.

Setting a '#f' state will stop the current animation and unset the current state, but keys will be left intact.

state a <clutter-state>  
target-state-name the state to transition to  
ret the <clutter-timeline> that drives the state transition. The returned timeline is owned by the <clutter-state> and it should not be unreferenced.

Since 1.4

**clutter-state-get-state** (self <clutter-state>) ⇒ (ret mchars)  
**get-state**  
Queries the currently set target state.

During a transition this function will return the target of the transition.

This function is useful when called from handlers of the "completed" signal.

state a <clutter-state>  
ret a string containing the target state. The returned string is owned by the <clutter-state> and should not be modified or freed.

Since 1.4

**clutter-state-warp-to-state** (self <clutter-state>)  
(target-state-name mchars) ⇒ (ret <clutter-timeline>)  
**warp-to-state**  
Change to the specified target state immediately with no animation.

See clutter-state-set-state.

state a <clutter-state>  
target-state-name the state to transition to  
ret the <clutter-timeline> that drives the state transition. The returned timeline is owned by the <clutter-state> and it should not be unreferenced.
Since 1.4

```clutter-state-set-key (self <clutter-state>)
  (source_state_name mchars) (target_state_name mchars) (object <gobject>)
  (property_name mchars) (mode unsigned-int) (value <gvalue>)
  (pre_delay double) (post_delay double) ⇒ (ret <clutter-state>)```

`set-key` [Method]
Sets one specific end key for a state name, object, property-name combination.

- **state**: a `<clutter-state>` instance.
- **source-state-name**: the source transition to specify transition for, or ‘#f’ to specify the default fallback when a more specific source state doesn’t exist.
- **target-state-name**: the name of the transition to set a key value for.
- **object**: the `<gobject>` to set a key for
- **property-name**: the property to set a key for
- **mode**: the id of the alpha function to use
- **value**: the value for property-name of object in state-name
- **pre-delay**: relative time of the transition to be idle in the beginning of the transition
- **post-delay**: relative time of the transition to be idle in the end of the transition
- **ret**: the `<clutter-state>` instance, allowing chaining of multiple calls.

Since 1.4

```clutter-state-set-duration (self <clutter-state>)
  (source_state_name mchars) (target_state_name mchars)
  (duration unsigned-int)```

`set-duration` [Method]
Sets the duration of a transition.

If both state names are ‘#f’ the default duration for state is set.

If only **target-state-name** is specified, the passed **duration** becomes the default duration for transitions to the target state.

If both states names are specified, the passed **duration** only applies to the specified transition.

- **state**: a `<clutter-state>`
- **source-state-name**: the name of the source state, or ‘#f’.
- **target-state-name**: the name of the target state, or ‘#f’.
- **duration**: the duration of the transition, in milliseconds
**clutter-state-get-duration** *(self <clutter-state>)*

**(source-state-name mchars) (target-state-name mchars)**

⇒ *(ret unsigned-int)*

duration

Queries the duration used for transitions between a source and target state pair.

The semantics for the query are the same as the semantics used for setting the duration with **clutter-state-set-duration**.

state a <clutter-state>

source-state-name

the name of the source state to get the duration of, or ‘#f’.

target-state-name

the name of the source state to get the duration of, or ‘#f’.

ret

the duration, in milliseconds

Since 1.4

**clutter-state-get-states** *(self <clutter-state>)*

⇒ *(ret glist-of)*

get-states

Gets a list of all the state names managed by this <clutter-state>.

state a <clutter-state> instance.

ret a newly allocated <g-list> of state names. The contents of the returned <g-list> are owned by the <clutter-state> and should not be modified or freed. Use **g-list-free** to free the resources allocated by the returned list when done using it.

Since 1.4

**clutter-state-get-keys** *(self <clutter-state>)*

**(source-state-name mchars) (target-state-name mchars) (object <gobject>) (property-name mchars)**

⇒ *(ret glist-of)*

get-keys

Returns a list of pointers to opaque structures with accessor functions that describe the keys added to an animator.

state a <clutter-state> instance.

source-state-name

the source transition name to query, or ‘#f’ for all source states.

target-state-name

the target transition name to query, or ‘#f’ for all target states.

object

the specific object instance to list keys for, or ‘#f’ for all managed objects.

property-name

the property name to search for, or ‘#f’ for all properties.
a newly allocated <g-list> of <clutter-state-key>s. The contents of the returned list are owned by the <clutter-state> and should not be modified or freed. Use g-list-free to free the resources allocated by the returned list when done using it.

Since 1.4

**clutter-state-remove-key (self <clutter-state>)**

(s source-state-name mchars) (target-state-name mchars) (object <gobject>)

**remove-key**

Removes all keys matching the search criteria passed in arguments.

*state* a <clutter-state> instance.

*source-state-name* the source state name to query, or ‘#f’ for all source states.

*target-state-name* the target state name to query, or ‘#f’ for all target states.

*object* the specific object instance to list keys for, or ‘#f’ for all managed objects.

*property-name* the property name to search for, or ‘#f’ for all properties.

Since 1.4

**clutter-state-get-timeline (self <clutter-state>)**

⇒ (ret <clutter-timeline>)

**get-timeline**

Gets the timeline driving the <clutter-state>

*state* a <clutter-state>

*ret* the <clutter-timeline> that drives the state change animations. The returned timeline is owned by the <clutter-state> and it should not be unreferenced directly.

Since 1.4

**clutter-state-set-animator (self <clutter-state>)**

(source-state-name mchars) (target-state-name mchars) (animator <clutter-animator>)

**set-animator**

Specifies a <clutter-animator> to be used when transitioning between the two named states.

The animator allows specifying a transition between the state that is more elaborate than the basic transitions allowed by the tweening of properties defined in the <clutter-state> keys.

If animator is ‘#f’ it will unset an existing animator.

<clutter-state> will take a reference on the passed animator, if any
state  a `<clutter-state>` instance.

source-state-name  
the name of a source state

target-state-name  
the name of a target state

 animator  a `<clutter-animator>` instance, or ‘#f’ to unset an existing `<clutter-animator>`.

Since 1.4

**clutter-state-get-animator (self `<clutter-state>`)**  
[Function]  
\(\text{(source-state-name mchars)}\) (\text{target-state-name mchars}) \Rightarrow (\text{ret `<clutter-animator>`})

get-animator  [Method]  
Retrieves the `<clutter-animator>` that is being used for transitioning between the two states, if any has been set

state  a `<clutter-state>` instance.

source-state-name  
the name of a source state

target-state-name  
the name of a target state

ret  a `<clutter-animator>` instance, or ‘#f’.

Since 1.4

**clutter-state-key-get-object (self `<clutter-state-key>`)**  
[Function]  
\Rightarrow (\text{ret `<gobject>`})

Retrieves the object instance this `<clutter-state-key>` applies to.

state-key  a `<clutter-state-key>`

ret  the object this state key applies to.

Since 1.4

**clutter-state-key-get-property-name**  
[Function]  
\((\text{self `<clutter-state-key>`}) \Rightarrow (\text{ret mchars})\)

Retrieves the name of the property this `<clutter-state-key>` applies to

state-key  a `<clutter-state-key>`

ret  the name of the property. The returned string is owned by the `<clutter-state-key>` and should never be modified or freed

Since 1.4

**clutter-state-key-get-mode (self `<clutter-state-key>`)**  
[Function]  
\Rightarrow (\text{ret unsigned-long})

Retrieves the easing mode used for state-key.
Chapter 63: ClutterState

state-key a \texttt{<clutter-state-key>}

ret the mode of a \texttt{<clutter-state-key>}

Since 1.4

\texttt{clutter-state-key-get-value (self <clutter-state-key>) \quad [Function]}
\[ (\text{value} \ < \texttt{gvalue}> ) \Rightarrow (\text{ret} \ \texttt{bool}) \]
Retrieves a copy of the value for a \texttt{<clutter-state-key>}.

The \texttt{<gvalue>} needs to be already initialized for the value type of the property or to a type that allow transformation from the value type of the key.

Use \texttt{g-value-unset} when done.

\texttt{state-key} a \texttt{<clutter-state-key>}

\texttt{value} a \texttt{<gvalue>} initialized with the correct type for the \texttt{state-key}

\texttt{ret} ‘\#t’ if the value was successfully retrieved, and ‘\#f’ otherwise

Since 1.4

\texttt{clutter-state-key-get-property-type} \quad [Function]
\[ (\text{self} \ < \texttt{clutter-state-key}> ) \Rightarrow (\text{ret} \ < \texttt{gtype}> ) \]
Retrieves the \texttt{<g-type>} of the property a key applies to

You can use this type to initialize the \texttt{<gvalue>} to pass to \texttt{clutter-state-key-get-value}

\texttt{key} a \texttt{<clutter-state-key>}

\texttt{ret} the \texttt{<g-type>} of the property

Since 1.4

\texttt{clutter-state-key-get-pre-delay (self <clutter-state-key>) \quad [Function]}
\[ \Rightarrow (\text{ret} \ \texttt{double}) \]
Retrieves the pause before transitioning starts as a fraction of the total transition time.

\texttt{state-key} a \texttt{<clutter-state-key>}

\texttt{ret} the pre delay used before starting the transition.

Since 1.4

\texttt{clutter-state-key-get-post-delay (self <clutter-state-key>) \quad [Function]}
\[ \Rightarrow (\text{ret} \ \texttt{double}) \]
Retrieves the duration of the pause after transitioning is complete as a fraction of the total transition time.

\texttt{state-key} a \texttt{<clutter-state-key>}

\texttt{ret} the post delay, used after doing the transition.

Since 1.4
64 ClutterSwipeAction

Action for swipe gestures

64.1 Overview

<clutter-swipe-action> is a sub-class of <clutter-gesture-action> that implements the logic for recognizing swipe gestures.

64.2 Usage

clutter-swipe-action-new ⇒ (ret <clutter-action>)  [Function]

Creates a new <clutter-swipe-action> instance

ret the newly created <clutter-swipe-action>

Since 1.8
Chapter 65: ClutterTableLayout

65 ClutterTableLayout

A layout manager arranging children in rows and columns

65.1 Overview

The `<clutter-table-layout>` is a `<clutter-layout-manager>` implementing the following layout policy:

- children are arranged in a table
- each child specifies the specific row and column cell to appear;
- a child can also set a span, and this way, take more than one cell both horizontally and vertically;
- each child will be allocated to its natural size or, if set to expand, the available size;
- if a child is set to fill on either (or both) axis, its allocation will match all the available size; the fill layout property only makes sense if the expand property is also set;
- if a child is set to expand but not to fill then it is possible to control the alignment using the horizontal and vertical alignment layout properties.

It is possible to control the spacing between children of a `<clutter-table-layout>` by using `clutter-table-layout-set-row-spacing` and `clutter-table-layout-set-column-spacing`.

In order to set the layout properties when packing an actor inside a `<clutter-table-layout>` you should use the `clutter-table-layout-pack` function.

A `<clutter-table-layout>` can use animations to transition between different values of the layout management properties; the easing mode and duration used for the animations are controlled by the `<"easing-mode">` and `<"easing-duration">` properties and their accessor functions.

(The missing figure, table-layout-image

The image shows a `<clutter-table-layout>`.

`<clutter-table-layout>` is available since Clutter 1.4

65.2 Usage

`clutter-table-layout-new` ⇒ `(ret `<clutter-layout-manager>`)` [Function]

Creates a new `<clutter-table-layout>` layout manager

`ret` the newly created `<clutter-table-layout>`

Since 1.4
**clutter-table-layout-get-row-count**  
*Function*  
\[
\text{clutter-table-layout-get-row-count} \quad \text{(self <clutter-table-layout>) } \Rightarrow \text{(ret int)}
\]

**get-row-count**  
*Method*  
Retrieve the current number rows in the *layout*

- *layout*  
  A *clutter-table-layout>*
- *ret*  
  the number of rows

Since 1.4

**clutter-table-layout-pack**  
*Function*  
\[
\text{clutter-table-layout-pack} \quad \text{(self <clutter-table-layout>) } \Rightarrow \text{(actor <clutter-actor>) (column int) (row int)}
\]

**pack**  
*Method*  
Packs *actor* inside the *<clutter-container>* associated to *layout* at the given row and column.

- *layout*  
  a *clutter-table-layout>*
- *actor*  
  a *clutter-actor>*
- *column*  
  the column the *actor* should be put, or -1 to append
- *row*  
  the row the *actor* should be put, or -1 to append

Since 1.4

**clutter-table-layout-set-alignment**  
*Function*  
\[
\text{clutter-table-layout-set-alignment} \quad \text{(self <clutter-table-layout>) (actor <clutter-actor>) (x_align <clutter-table-alignment>) (y_align <clutter-table-alignment>)}
\]

**set-alignment**  
*Method*  
Sets the horizontal and vertical alignment policies for *actor* inside *layout*

- *layout*  
  a *clutter-table-layout>*
- *actor*  
  a *clutter-actor>* child of *layout*
- *x-align*  
  Horizontal alignment policy for *actor*
- *y-align*  
  Vertical alignment policy for *actor*

Since 1.4

**clutter-table-layout-get-alignment**  
*Function*  
\[
\text{clutter-table-layout-get-alignment} \quad \text{(self <clutter-table-layout>) (actor <clutter-actor>) } \Rightarrow \text{(x_align <clutter-table-alignment>) (y_align <clutter-table-alignment>)}
\]

**get-alignment**  
*Method*  
Retrieves the horizontal and vertical alignment policies for *actor* as set using clutter-table-layout-pack or clutter-table-layout-set-alignment.

- *layout*  
  a *clutter-table-layout>*
- *actor*  
  a *clutter-actor>* child of *layout*
- *x-align*  
  return location for the horizontal alignment policy.
Chapter 65: ClutterTableLayout

\[ y\text{-align} \] return location for the vertical alignment policy.

Since 1.4

\textbf{clutter-table-layout-set-expand (self <clutter-table-layout>) [Function]}

\textbf{set-expand}
Sets the horizontal and vertical expand policies for \textit{actor} inside \textit{layout}

\textit{layout} a <clutter-table-layout>
\textit{actor} a <clutter-actor> child of \textit{layout}
\textit{x-expand} whether \textit{actor} should allocate extra space horizontally
\textit{y-expand} whether \textit{actor} should allocate extra space vertically

Since 1.4

\textbf{clutter-table-layout-get-expand (self <clutter-table-layout>) [Function]}

\textbf{get-expand}
Retrieves the horizontal and vertical expand policies for \textit{actor} as set using clutter-table-layout-pack or clutter-table-layout-set-expand

\textit{layout} a <clutter-table-layout>
\textit{actor} a <clutter-actor> child of \textit{layout}
\textit{x-expand} return location for the horizontal expand policy.
\textit{y-expand} return location for the vertical expand policy.

Since 1.4

\textbf{clutter-table-layout-set-fill (self <clutter-table-layout>) [Function]}

\textbf{set-fill}
Sets the horizontal and vertical fill policies for \textit{actor} inside \textit{layout}

\textit{layout} a <clutter-table-layout>
\textit{actor} a <clutter-actor> child of \textit{layout}
\textit{x-fill} whether \textit{actor} should fill horizontally the allocated space
\textit{y-fill} whether \textit{actor} should fill vertically the allocated space

Since 1.4

\textbf{clutter-table-layout-get-fill (self <clutter-table-layout>) [Function]}

\textbf{get-fill}
Retrieves the horizontal and vertical fill policies for \textit{actor} as set using clutter-table-layout-pack or clutter-table-layout-set-fill

\textit{layout} a <clutter-table-layout>
actor a `<clutter-actor>` child of `layout`

`x-fill` return location for the horizontal fill policy.

`y-fill` return location for the vertical fill policy.

Since 1.4

`clutter-table-layout-get-span (self <clutter-table-layout>)` [Function]

`(actor <clutter-actor>) ⇒ (column-span int) (row-span int)`

`get-span` [Method]

Retrieves the row and column span for `actor` as set using `clutter-table-layout-pack` or `clutter-table-layout-set-span`.

`layout` a `<clutter-table-layout>`

`actor` a `<clutter-actor>` child of `layout`

`column-span` return location for the col span.

`row-span` return location for the row span.

Since 1.4

`clutter-table-layout-set-span (self <clutter-table-layout>)` [Function]

`(actor <clutter-actor>) (column-span int) (row-span int)`

`set-span` [Method]

Sets the row and column span for `actor` inside `layout`.

`layout` a `<clutter-table-layout>`

`actor` a `<clutter-actor>` child of `layout`

`column-span` Column span for `actor`

`row-span` Row span for `actor`

Since 1.4
Chapter 66: ClutterTextBuffer

66 ClutterTextBuffer

Text buffer for ClutterText

66.1 Overview

The <clutter-text-buffer> class contains the actual text displayed in a <clutter-text> widget.

A single <clutter-text-buffer> object can be shared by multiple <clutter-text> widgets which will then share the same text content, but not the cursor position, visibility attributes, icon etc.

A <clutter-text-buffer> may be derived from. Such a derived class might allow text to be stored in an alternate location, such as non-pageable memory, useful in the case of important passwords. Or a derived class could integrate with an application’s concept of undo/redo.

66.2 Usage

clutter-text-buffer-new ⇒ (ret <clutter-text-buffer>) [Function]

Create a new ClutterTextBuffer object.

ret A new ClutterTextBuffer object.

Since 1.10

clutter-text-buffer-new-with-text (text mchars) (text_len ssize_t) ⇒ (ret <clutter-text-buffer>) [Function]

Create a new ClutterTextBuffer object with some text.

text initial buffer text.

text-len initial buffer text length, or -1 for null-terminated.

ret A new ClutterTextBuffer object.

Since 1.10

clutter-text-buffer-set-text (self <clutter-text-buffer>) (chars mchars) (n_chars int) [Function]

set-text

Sets the text in the buffer.

This is roughly equivalent to calling clutter-text-buffer-delete-text and clutter-text-buffer-insert-text.

Note that n-chars is in characters, not in bytes.

buffer a <clutter-text-buffer>

chars the new text

n-chars the number of characters in text, or -1

Since 1.10
clutter-text-buffer-get-text (self <clutter-text-buffer>) ⇒ (ret mchars)

get-text
Retrieves the contents of the buffer.

The memory pointer returned by this call will not change unless this object emits a signal, or is finalized.

buffer a <clutter-text-buffer>
ret a pointer to the contents of the widget as a string. This string points to internally allocated storage in the buffer and must not be freed, modified or stored.

Since 1.10

clutter-text-buffer-get-bytes (self <clutter-text-buffer>) ⇒ (ret size_t)

get-bytes
Retrieves the length in bytes of the buffer. See clutter-text-buffer-get-length.

buffer a <clutter-text-buffer>
ret The byte length of the buffer.

Since 1.10

clutter-text-buffer-get-length (self <clutter-text-buffer>) ⇒ (ret unsigned-int)

get-length
Retrieves the length in characters of the buffer.

buffer a <clutter-text-buffer>
ret The number of characters in the buffer.

Since 1.10

clutter-text-buffer-set-max-length
(sself <clutter-text-buffer>) (max_length int)

set-max-length
Sets the maximum allowed length of the contents of the buffer. If the current contents are longer than the given length, then they will be truncated to fit.

buffer a <clutter-text-buffer>
max-length the maximum length of the entry buffer, or 0 for no maximum. (other than the maximum length of entries.) The value passed in will be clamped to the range [ 0, ‘CLUTTER_TEXT_BUFFER_MAX_SIZE’ ].

Since 1.10
**clutter-text-buffer-get-max-length**  
* (self <clutter-text-buffer>) ⇒ (ret int)

**get-max-length**  
Retrieves the maximum allowed length of the text in buffer. See clutter-text-buffer-set-max-length.

* buffer a <clutter-text-buffer>

* ret the maximum allowed number of characters in <clutter-text-buffer>, or 0 if there is no maximum.

Since 1.10

**clutter-text-buffer-insert-text (self <clutter-text-buffer>)**  
* (position unsigned-int) (chars mchars) (n_chars int)

**insert-text**  
Inserts n-chars characters of chars into the contents of the buffer, at position position. If n-chars is negative, then characters from chars will be inserted until a null-terminator is found. If position or n-chars are out of bounds, or the maximum buffer text length is exceeded, then they are coerced to sane values.

Note that the position and length are in characters, not in bytes.

* buffer a <clutter-text-buffer>

* position the position at which to insert text.

* chars the text to insert into the buffer.

* n-chars the length of the text in characters, or -1

* ret The number of characters actually inserted.

Since 1.10

**clutter-text-buffer-delete-text (self <clutter-text-buffer>)**  
* (position unsigned-int) (n_chars int) ⇒ (ret unsigned-int)

**delete-text**  
Deletes a sequence of characters from the buffer. n-chars characters are deleted starting at position. If n-chars is negative, then all characters until the end of the text are deleted.

If position or n-vars are out of bounds, then they are coerced to sane values.

Note that the positions are specified in characters, not bytes.

* buffer a <clutter-text-buffer>

* position position at which to delete text

* n-vars number of characters to delete

* ret The number of characters deleted.

Since 1.10
67 ClutterText

An actor for displaying and editing text

67.1 Overview

<clutter-text> is an actor that displays custom text using Pango as the text rendering engine.

<clutter-text> also allows inline editing of the text if the actor is set editable using clutter-text-set-editable.

Selection using keyboard or pointers can be enabled using clutter-text-set-selectable.

<clutter-text> is available since Clutter 1.0

67.2 Usage

clutter-text-new ⇒ (ret <clutter-actor>) [Function]

Creates a new <clutter-text> actor. This actor can be used to display and edit text.

ret the newly created <clutter-text> actor

Since 1.0

clutter-text-new-full (font-name mchars) (text mchars) (color <clutter-color>) ⇒ (ret <clutter-actor>) [Function]

Creates a new <clutter-text> actor, using font-name as the font description; text will be used to set the contents of the actor; and color will be used as the color to render text.

This function is equivalent to calling clutter-text-new, clutter-text-set-font-name, clutter-text-set-text and clutter-text-set-color.

font-name a string with a font description

text the contents of the actor

color the color to be used to render text

ret the newly created <clutter-text> actor

Since 1.0

clutter-text-new-with-text (font-name mchars) (text mchars) [Function]

⇒ (ret <clutter-actor>)

Creates a new <clutter-text> actor, using font-name as the font description; text will be used to set the contents of the actor.

This function is equivalent to calling clutter-text-new, clutter-text-set-font-name, and clutter-text-set-text.

font-name a string with a font description

text the contents of the actor
Chapter 67: ClutterText

\[ ret \quad \text{the newly created <clutter-text> actor} \]

Since 1.0

\textbf{clutter-text-new-with-buffer} (\textit{buffer <clutter-text-buffer>}) \quad \textbf{[Function]}

\[ \Rightarrow (ret <clutter-actor>) \]

Creates a new entry with the specified text buffer.

\textit{buffer} \quad \text{The buffer to use for the new <clutter-text>.

\textit{ret} \quad \text{a new <clutter-text>}

Since 1.10

\textbf{clutter-text-set-buffer} (\textit{self <clutter-text>}) \quad \textbf{set-buffer} \quad \textbf{[Method]}

Set the <clutter-text-buffer> object which holds the text for this widget.

\textit{self} \quad \text{a <clutter-text>

\textit{buffer} \quad \text{a <clutter-text-buffer>}

Since 1.10

\textbf{clutter-text-get-buffer} (\textit{self <clutter-text>}) \quad \textbf{get-buffer} \quad \textbf{[Method]}

Get the <clutter-text-buffer> object which holds the text for this widget.

\textit{self} \quad \text{a <clutter-text>

\textit{ret} \quad \text{A <gtk-entry-buffer> object.

Since 1.10

\textbf{clutter-text-set-text} (\textit{self <clutter-text>}) (\textit{text mchars}) \quad \textbf{set-text} \quad \textbf{[Method]}

Sets the contents of a <clutter-text> actor.

If the "\text{use-markup}" property was set to 't' it will be reset to 'f' as a side effect.

If you want to maintain the "\text{use-markup}" you should use the clutter-text-set-markup function instead

\textit{self} \quad \text{a <clutter-text>

\textit{text} \quad \text{the text to set. Passing 'f' is the same as passing "" (the empty string).

Since 1.0

\textbf{clutter-text-set-markup} (\textit{self <clutter-text>}) (\textit{markup mchars}) \quad \textbf{set-markup} \quad \textbf{[Method]}

Sets \textit{markup} as the contents of a <clutter-text>.

This is a convenience function for setting a string containing Pango markup, and it is logically equivalent to:
/* the order is important */
clutter_text_set_text (CLUTTER_TEXT (actor), markup);
clutter_text_set_use_markup (CLUTTER_TEXT (actor), TRUE);

self  a <clutter-text>

markup  a string containing Pango markup. Passing ‘#f’ is the same as passing "" (the empty string).

Since 1.0

clutter-text-get-text (self <clutter-text>) ⇒ (ret mchars)  [Function]
get-text  [Method]
Retrieves a pointer to the current contents of a <clutter-text> actor.
If you need a copy of the contents for manipulating, either use g-strdup on the returned string, or use:

    copy = clutter_text_get_chars (text, 0, -1);
Which will return a newly allocated string.
If the <clutter-text> actor is empty, this function will return an empty string, and not ‘#f’.

self  a <clutter-text>
ret  the contents of the actor. The returned string is owned by the <clutter-text> actor and should never be modified or freed.

Since 1.0

clutter-text-set-activatable (self <clutter-text>)  [Function]
(activatable bool)  [Method]
set-activatable  Sets whether a <clutter-text> actor should be activatable.
An activatable <clutter-text> actor will emit the "activate" signal whenever the 'Enter' (or 'Return') key is pressed; if it is not activatable, a new line will be appended to the current content.
An activatable <clutter-text> must also be set as editable using clutter-text-set-editable.

self  a <clutter-text>
activatable  whether the <clutter-text> actor should be activatable

Since 1.0

clutter-text-get-activatable (self <clutter-text>)  [Function]
⇒ (ret bool)  [Method]
get-activatable  Retrieves whether a <clutter-text> is activatable or not.
Chapter 67: ClutterText

self a <clutter-text>
ret ‘#t’ if the actor is activatable

Since 1.0

clutter-text-set-attributes (self <clutter-text>) [Function]
  (attrs <pango-attr-list>)
set-attributes [Method]
Sets the attributes list that are going to be applied to the <clutter-text> contents. The <clutter-text> actor will take a reference on the <pango-attr-list> passed to this function.

self a <clutter-text>
attrs a <pango-attr-list> or ‘#f’ to unset the attributes

Since 1.0

clutter-text-get-attributes (self <clutter-text>) [Function]
  ⇒ (ret <pango-attr-list>)
get-attributes [Method]
Gets the attribute list that was set on the <clutter-text> actor clutter-text-set-attributes, if any.

self a <clutter-text>
ret the attribute list, or ‘#f’ if none was set. The returned value is owned by the <clutter-text> and should not be unreferenced.

Since 1.0

clutter-text-set-color (self <clutter-text>) [Function]
  (color <clutter-color>)
set-color [Method]
Sets the color of the contents of a <clutter-text> actor.
The overall opacity of the <clutter-text> actor will be the result of the alpha value of color and the composited opacity of the actor itself on the scenegraph, as returned by clutter-actor-get-paint-opacity.

self a <clutter-text>
color a <clutter-color>

Since 1.0

clutter-text-get-color (self <clutter-text>) [Function]
  (color <clutter-color>)
get-color [Method]
Retrieves the text color as set by clutter-text-set-color.

self a <clutter-text>
color return location for a <clutter-color>.

Since 1.0
**clutter-text-set-ellipsize (self <clutter-text>)**

*set-ellipsize*

Sets the mode used to ellipsize (add an ellipsis: ". . .") to the text if there is not enough space to render the entire contents of a <clutter-text> actor.

```python
self a <clutter-text>
mode a <pango-ellipsize-mode>
```

Since 1.0

**clutter-text-get-ellipsize (self <clutter-text>)**

*get-ellipsize*

Returns the ellipsizing position of a <clutter-text> actor, as set by clutter-text-set-ellipsize.

```python
self a <clutter-text>
ret a <pango-ellipsize-mode>
```

Since 1.0

**clutter-text-set-font-name (self <clutter-text>)**

*set-font-name*

Sets the font used by a <clutter-text>. The font-name string must either be ‘#f’, which means that the font name from the default <clutter-backend> will be used; or be something that can be parsed by the `pango-font-description-from-string` function, like:

```python
clutter_text_set_font_name (text, "Sans 10pt");
clutter_text_set_font_name (text, "Serif 16px");
clutter_text_set_font_name (text, "Helvetica 10");
```

```python
self a <clutter-text>
font-name a font name, or ‘#f’ to set the default font name.
```

Since 1.0

**clutter-text-get-font-name (self <clutter-text>)**

*get-font-name*

Retrieves the font name as set by clutter-text-set-font-name.

```python
self a <clutter-text>
ret a string containing the font name. The returned string is owned by the <clutter-text> actor and should not be modified or freed
```

Since 1.0
clutter-text-set-font-description (self <clutter-text>) (font-desc <pango-font-description>)

set-font-description
Sets font-desc as the font description for a <clutter-text>
The <pango-font-description> is copied by the <clutter-text> actor so you can safely call pango-font-description-free on it after calling this function.

self a <clutter-text>
font-desc a <pango-font-description>

Since 1.2

clutter-text-set-password-char (self <clutter-text>) (wc unsigned-int32)

set-password-char
Sets the character to use in place of the actual text in a password text actor.
If wc is 0 the text will be displayed as it is entered in the <clutter-text> actor.

self a <clutter-text>
wc a Unicode character, or 0 to unset the password character

Since 1.0

clutter-text-get-password-char (self <clutter-text>) ⇒ (ret unsigned-int32)

get-password-char
Retrieves the character to use in place of the actual text as set by clutter-text-set-password-char.

self a <clutter-text>
ret a Unicode character or 0 if the password character is not set

Since 1.0

clutter-text-set-justify (self <clutter-text>) (justify bool)

set-justify
Sets whether the text of the <clutter-text> actor should be justified on both margins. This setting is ignored if Clutter is compiled against Pango < 1.18.

self a <clutter-text>
justify whether the text should be justified

Since 1.0

clutter-text-get-justify (self <clutter-text>) ⇒ (ret bool)

get-justify
Retrieves whether the <clutter-text> actor should justify its contents on both margins.

self a <clutter-text>
ret ’#t’ if the text should be justified

Since 0.6
clutter-text-get-layout (self <clutter-text>) ⇒ (ret <pango-layout>)

get-layout
Retrieves the current <pango-layout> used by a <clutter-text> actor.

self a <clutter-text>
ret a <pango-layout>. The returned object is owned by the <clutter-text> actor and should not be modified or freed.

Since 1.0

clutter-text-set-line-alignment (self <clutter-text>) (alignment <pango-alignment>)

set-line-alignment
Sets the way that the lines of a wrapped label are aligned with respect to each other. This does not affect the overall alignment of the label within its allocated or specified width.
To align a <clutter-text> actor you should add it to a container that supports alignment, or use the anchor point.

self a <clutter-text>
alignment A <pango-alignment>

Since 1.0

clutter-text-get-line-alignment (self <clutter-text>) ⇒ (ret <pango-alignment>)

get-line-alignment
Retrieves the alignment of a <clutter-text>, as set by clutter-text-set-line-alignment.

self a <clutter-text>
ret a <pango-alignment>

Since 1.0

clutter-text-set-line-wrap (self <clutter-text>) (line-wrap bool)

set-line-wrap
Sets whether the contents of a <clutter-text> actor should wrap, if they don’t fit the size assigned to the actor.

self a <clutter-text>
line-wrap whether the contents should wrap

Since 1.0

clutter-text-get-line-wrap (self <clutter-text>) ⇒ (ret bool)

get-line-wrap
Retrieves the value set using clutter-text-set-line-wrap.
ClutterText

Chapter 67: ClutterText

Since 1.0

clutter-text-set-line-wrap-mode (self <clutter-text>) [Function]

(set-line-wrap-mode)

If line wrapping is enabled (see clutter-text-set-line-wrap) this function controls how the line wrapping is performed. The default is ‘PANGO_WRAP_WORD’ which means wrap on word boundaries.

Since 1.0

clutter-text-get-line-wrap-mode (self <clutter-text>) [Function]

:get-line-wrap-mode

Retrieves the line wrap mode used by the <clutter-text> actor.
See clutter-text-set-line-wrap-mode.

Since 1.0

clutter-text-set-max-length (self <clutter-text>) (max int) [Function]

(set-max-length)

Sets the maximum allowed length of the contents of the actor. If the current contents are longer than the given length, then they will be truncated to fit.

Since 1.0

clutter-text-get-max-length (self <clutter-text>) ⇒ (ret int) [Function]

:get-max-length

Gets the maximum length of text that can be set into a text actor.
See clutter-text-set-max-length.

Since 1.0
**clutter-text-set-selectable** (self <clutter-text>) [Function]

(set-selectable bool)

set-selectable [Method]
Sets whether a <clutter-text> actor should be selectable.

A selectable <clutter-text> will allow selecting its contents using the pointer or the keyboard.

self a <clutter-text>

selectable whether the <clutter-text> actor should be selectable

Since 1.0

**clutter-text-get-selectable** (self <clutter-text>) ⇒ (ret bool) [Function]

get-selectable [Method]
Retrieves whether a <clutter-text> is selectable or not.

self a <clutter-text>

ret ‘#t’ if the actor is selectable

Since 1.0

**clutter-text-set-selection** (self <clutter-text>) [Function]

(start-pos ssize_t) (end-pos ssize_t)

set-selection [Method]
Selects the region of text between start-pos and end-pos.

This function changes the position of the cursor to match start-pos and the selection bound to match end-pos.

self a <clutter-text>

start-pos start of the selection, in characters

end-pos end of the selection, in characters

Since 1.0

**clutter-text-get-selection** (self <clutter-text>) [Function]

⇒ (ret mchars)

get-selection [Method]
Retrieves the currently selected text.

self a <clutter-text>

ret a newly allocated string containing the currently selected text, or ‘#f’. Use g-free to free the returned string.

Since 1.0

**clutter-text-set-selection-bound** (self <clutter-text>) [Function]

(selection_bound int)

set-selection-bound [Method]
Sets the other end of the selection, starting from the current cursor position.

If selection-bound is -1, the selection unset.
self a <clutter-text>

selection-bound
the position of the end of the selection, in characters

Since 1.0

clutter-text-get-selection-bound (self <clutter-text>) ⇒ (ret int)
get-selection-bound [Method]
Retrieves the other end of the selection of a <clutter-text> actor, in characters
from the current cursor position.

self a <clutter-text>
ret the position of the other end of the selection

Since 1.0

clutter-text-set-single-line-mode (self <clutter-text>) (single_line bool)
set-single-line-mode [Method]
Sets whether a <clutter-text> actor should be in single line mode or not. Only
editable <clutter-text>s can be in single line mode.
A text actor in single line mode will not wrap text and will clip the visible area to
the predefined size. The contents of the text actor will scroll to display the end of
the text if its length is bigger than the allocated width.
When setting the single line mode the "activatable" property is also set as a
side effect. Instead of entering a new line character, the text actor will emit the
"activate" signal.

self a <clutter-text>
single-line whether to enable single line mode

Since 1.0

clutter-text-get-single-line-mode (self <clutter-text>) ⇒ (ret bool)
get-single-line-mode [Method]
Retrieves whether the <clutter-text> actor is in single line mode.

self a <clutter-text>
ret ‘#t’ if the <clutter-text> actor is in single line mode

Since 1.0

clutter-text-set-use-markup (self <clutter-text>) (setting bool)
set-use-markup [Method]
Sets whether the contents of the <clutter-text> actor contains markup in Pango’s
text markup language.
Setting "use-markup" on an editable <clutter-text> will not have any effect
except hiding the markup.
See also "use-markup".
Chapter 67: ClutterText

self a <clutter-text>

setting ‘#t’ if the text should be parsed for markup.

Since 1.0

clutter-text-get-use-markup (self <clutter-text>) ⇒ (ret bool) [Function]
get-use-markup
Retrieves whether the contents of the <clutter-text> actor should be parsed for the Pango text markup.

self a <clutter-text>
ret ‘#t’ if the contents will be parsed for markup

Since 1.0

clutter-text-set-editable (self <clutter-text>) (editable bool) [Function]
set-editable
Sets whether the <clutter-text> actor should be editable.

An editable <clutter-text> with key focus set using clutter-actor-grab-key-focus or clutter-stage-set-key-focus will receive key events and will update its contents accordingly.

self a <clutter-text>
editable whether the <clutter-text> should be editable

Since 1.0

clutter-text-get-editable (self <clutter-text>) ⇒ (ret bool) [Function]
get-editable
Retrieves whether a <clutter-text> is editable or not.

self a <clutter-text>
ret ‘#t’ if the actor is editable

Since 1.0

clutter-text-insert-text (self <clutter-text>) (text mchars) (position ssize_t) [Function]
insert-text
Inserts text into a <clutter-actor> at the given position.

If position is a negative number, the text will be appended at the end of the current contents of the <clutter-text>.

The position is expressed in characters, not in bytes.

self a <clutter-text>
text the text to be inserted
position the position of the insertion, or -1

Since 1.0
**clutter-text-insert-unichar** (*self <clutter-text>*))  
(\textit{wc unsigned-int32})  
[Function]

**insert-unichar**  
[Method]

Inserts \textit{wc} at the current cursor position of a \textit{<clutter-text>} actor.

*\textit{self} a \textit{<clutter-text>}*  
\textit{wc} a Unicode character

Since 1.0

**clutter-text-delete-chars** (*self <clutter-text>*))  
(\textit{n_chars unsigned-int})  
[Function]

**delete-chars**  
[Method]

Deletes \textit{n_chars} inside a \textit{<clutter-text>} actor, starting from the current cursor position.

Somewhat awkwardly, the cursor position is decremented by the same number of characters you’ve deleted.

*\textit{self} a \textit{<clutter-text>}*  
\textit{n_chars} the number of characters to delete

Since 1.0

**clutter-text-delete-text** (*self <clutter-text>*))  
\textit{(start_pos ssize_t) (end_pos ssize_t)}  
[Function]

**delete-text**  
[Method]

Deletes the text inside a \textit{<clutter-text>} actor between \textit{start-pos} and \textit{end-pos}.

The starting and ending positions are expressed in characters, not in bytes.

*\textit{self} a \textit{<clutter-text>}*  
\textit{start-pos} starting position  
\textit{end-pos} ending position

Since 1.0

**clutter-text-delete-selection** (*self <clutter-text>*))  
\Rightarrow (\textit{ret bool})  
[Function]

**delete-selection**  
[Method]

Deletes the currently selected text

This function is only useful in subclasses of \textit{<clutter-text>}

*\textit{self} a \textit{<clutter-text>}*  
\textit{ret} ‘\#t’ if text was deleted or if the text actor is empty, and ‘\#f’ otherwise

Since 1.0

**clutter-text-get-chars** (*self <clutter-text>*)) \textit{(start_pos ssize_t) (end_pos ssize_t)}  
\Rightarrow (\textit{ret mchars})  
[Function]

**get-chars**  
[Method]

Retrieves the contents of the \textit{<clutter-text>} actor between \textit{start-pos} and \textit{end-pos}, but not including \textit{end-pos}.

The positions are specified in characters, not in bytes.
Chapter 67: ClutterText

self
a <clutter-text>

start-pos
start of text, in characters

disco-pos
end of text, in characters

ret
a newly allocated string with the contents of the text actor between the specified positions. Use g-free to free the resources when done

Since 1.0

clutter-text-set-cursor-color (self <clutter-text>)

(color <clutter-color>)

set-cursor-color
Sets the color of the cursor of a <clutter-text> actor.
If color is ‘#f’, the cursor color will be the same as the text color.

self
a <clutter-text>

color
the color of the cursor, or ‘#f’ to unset it

Since 1.0

clutter-text-get-cursor-color (self <clutter-text>)

(color <clutter-color>)

g-cursor-color
Retrieves the color of the cursor of a <clutter-text> actor.

self
a <clutter-text>

color
return location for a <clutter-color>.

Since 1.0

clutter-text-set-selection-color (self <clutter-text>)

(color <clutter-color>)

set-selection-color
Sets the color of the selection of a <clutter-text> actor.
If color is ‘#f’, the selection color will be the same as the cursor color, or if no cursor color is set either then it will be the same as the text color.

self
a <clutter-text>

color
the color of the selection, or ‘#f’ to unset it

Since 1.0

clutter-text-get-selection-color (self <clutter-text>)

(color <clutter-color>)

g-selection-color
Retrieves the color of the selection of a <clutter-text> actor.

self
a <clutter-text>

color
return location for a <clutter-color>.

Since 1.0
clutter-text-set-cursor-position (self <clutter-text>) [Function]
  (position int)
set-cursor-position [Method]
  Sets the cursor of a <clutter-text> actor at position.
The position is expressed in characters, not in bytes.
  self  a <clutter-text>
  position the new cursor position, in characters
Since 1.0

clutter-text-get-cursor-position (self <clutter-text>) [Function]
  ⇒ (ret int)
get-cursor-position [Method]
  Retrieves the cursor position.
  self  a <clutter-text>
  ret the cursor position, in characters
Since 1.0

clutter-text-set-cursor-visible (self <clutter-text>) [Function]
  (cursor_visible bool)
set-cursor-visible [Method]
  Sets whether the cursor of a <clutter-text> actor should be visible or not.
The color of the cursor will be the same as the text color unless clutter-text-set-
cursor-color has been called.
The size of the cursor can be set using clutter-text-set-cursor-size.
The position of the cursor can be changed programmatically using clutter-text-
set-cursor-position.
  self  a <clutter-text>
  cursor-visible whether the cursor should be visible
Since 1.0

clutter-text-get-cursor-visible (self <clutter-text>) [Function]
  ⇒ (ret bool)
get-cursor-visible [Method]
  Retrieves whether the cursor of a <clutter-text> actor is visible.
  self  a <clutter-text>
  ret ‘#t’ if the cursor is visible
Since 1.0

clutter-text-set-cursor-size (self <clutter-text>) (size int) [Function]
set-cursor-size [Method]
  Sets the size of the cursor of a <clutter-text>. The cursor will only be visible if the
  "cursor-visible" property is set to ‘#t’. 
self a <clutter-text>

size the size of the cursor, in pixels, or -1 to use the default value

Since 1.0

clutter-text-get-cursor-size (self <clutter-text>) [Function]
⇒ (ret unsigned-int)

get-cursor-size [Method]
Retrieves the size of the cursor of a <clutter-text> actor.

self a <clutter-text>
ret the size of the cursor, in pixels

Since 1.0

clutter-text-activate (self <clutter-text>) ⇒ (ret bool) [Function]
activate [Method]
Emits the "activate" signal, if self has been set as activatable using clutter-text-set-activatable.

This function can be used to emit the ::activate signal inside a <"captured-event"> or <"key-press-event"> signal handlers before the default signal handler for the <clutter-text> is invoked.

self a <clutter-text>
ret '#t' if the ::activate signal has been emitted, and '#f' otherwise

Since 1.0

clutter-text-coords-to-position (self <clutter-text>) [Function]
(x float) (y float) ⇒ (ret int)

coords-to-position [Method]
Retrieves the position of the character at the given coordinates.

Return: the position of the character

self a <clutter-text>
x the X coordinate, relative to the actor
y the Y coordinate, relative to the actor

Since 1.10

clutter-text-position-to-coords (self <clutter-text>) [Function]
(position int) ⇒ (ret bool) (x float) (y float) (line_height float)

position-to-coords [Method]
Retrieves the coordinates of the given position.

self a <clutter-text>
position position in characters
x return location for the X coordinate, or '#f'.
y return location for the Y coordinate, or '#f'.
Since 1.0

**clutter-text-set-preedit-string** (self <clutter-text>) [Function]

(preedit_str mchars) (preedit_attrs <pango-attr-list>)

(cursor_pos unsigned-int)

**set-preedit-string** [Method]

Sets, or unsets, the pre-edit string. This function is useful for input methods to display a string (with eventual specific Pango attributes) before it is entered inside the <clutter-text> buffer.

The preedit string and attributes are ignored if the <clutter-text> actor is not editable.

This function should not be used by applications

**self** a <clutter-text>

**preedit-str**

the pre-edit string, or ‘#f’ to unset it.

**preeditAttrs**

the pre-edit string attributes.

**cursor-pos** the cursor position for the pre-edit string

Since 1.2

**clutter-text-get-layout-offsets** (self <clutter-text>) [Function]

⇒ (x int) (y int)

**get-layout-offsets** [Method]

Obtains the coordinates where the <clutter-text> will draw the <pango-layout> representing the text.

**self** a <clutter-text>

**x** location to store X offset of layout, or ‘#f’.

**y** location to store Y offset of layout, or ‘#f’.

Since 1.8
68 ClutterTexture

An actor for displaying and manipulating images.

68.1 Overview

<clutter-texture> is a base class for displaying and manipulating pixel buffer type data.

The clutter-texture-set-from-rgb-data and clutter-texture-set-from-file functions are used to copy image data into texture memory and subsequently realize the texture.

Note: a ClutterTexture will scale its contents to fit the bounding box requested using clutter-actor-set-size. To display an area of a texture without scaling, you should set the clip area using clutter-actor-set-clip.

68.2 Usage

clutter-texture-new ⇒ (ret <clutter-actor>) [Function]
Creates a new empty <clutter-texture> object.

ret A newly created <clutter-texture> object.

clutter-texture-new-from-file (filename mchars) ⇒ (ret <clutter-actor>) [Function]
Creates a new ClutterTexture actor to display the image contained a file. If the image failed to load then NULL is returned and error is set.

filename The name of an image file to load.
error Return locatoin for an error.
ret A newly created <clutter-texture> object or NULL on error.

Since 0.8

clutter-texture-set-from-file (self <clutter-texture>) (filename mchars) ⇒ (ret bool) [Function]
set-from-file
Sets the <clutter-texture> image data from an image file. In case of failure, ‘#f’ is returned and error is set.

If "load-async" is set to ‘#t’, this function will return as soon as possible, and the actual image loading from disk will be performed asynchronously. "size-change" will be emitted when the size of the texture is available and "load-finished" will be emitted when the image has been loaded or if an error occurred.

texture A <clutter-texture>
filename The filename of the image in GLib file name encoding
error Return location for a <g-error>, or ‘#f’
ret ‘#t’ if the image was successfully loaded and set

Since 0.8
clutter-texture-get-base-size \( (\text{self} \ <\ \text{clutter-texture}) \) [Function]
\[ \Rightarrow \ (\text{width} \ \text{int}) \ (\text{height} \ \text{int}) \]

get-base-size [Method]
Gets the size in pixels of the untransformed underlying image

\begin{itemize}
  \item \textit{texture} \ a \ <\ \text{clutter-texture} \\
  \item \textit{width} \ \text{return location for the width, or ‘#f’}. \\
  \item \textit{height} \ \text{return location for the height, or ‘#f’}. \\
\end{itemize}

clutter-texture-get-max-tile-waste \( (\text{self} \ <\ \text{clutter-texture}) \) [Function]
\[ \Rightarrow \ (\text{ret} \ \text{int}) \]

get-max-tile-waste [Method]
Gets the maximum waste that will be used when creating a texture or -1 if slicing is disabled.

\begin{itemize}
  \item \textit{texture} \ A \ <\ \text{clutter-texture} \\
  \item \textit{ret} \ The maximum waste or -1 if the texture waste is unlimited. \\
\end{itemize}

Since 0.8

clutter-texture-set-filter-quality \( (\text{self} \ <\ \text{clutter-texture}) \) [Function]
\[ (\text{filter-quality} \ <\ \text{clutter-texture-quality}) \]

set-filter-quality [Method]
Sets the filter quality when scaling a texture. The quality is an enumeration currently the following values are supported: ‘CLUTTER_TEXTURE_QUALITY_LOW’ which is fast but only uses nearest neighbour interpolation, ‘CLUTTER_TEXTURE_QUALITY_MEDIUM’ which is computationally a bit more expensive (bilinear interpolation), and ‘CLUTTER_TEXTURE_QUALITY_HIGH’ which uses extra texture memory resources to improve scaled down rendering as well (by using mipmaps). The default value is ‘CLUTTER_TEXTURE_QUALITY_MEDIUM’.

\begin{itemize}
  \item \textit{texture} \ a \ <\ \text{clutter-texture} \\
  \item \textit{filter-quality} \ \text{new filter quality value} \\
\end{itemize}

Since 0.8

clutter-texture-get-sync-size \( (\text{self} \ <\ \text{clutter-texture}) \) [Function]
\[ \Rightarrow \ (\text{ret} \ \text{bool}) \]

get-sync-size [Method]
Retrieves the value set with clutter-texture-set-sync-size

\begin{itemize}
  \item \textit{texture} \ a \ <\ \text{clutter-texture} \\
  \item \textit{ret} \ ‘#t’ if the <clutter-texture> should have the same preferred size of the underlying image data \\
\end{itemize}

Since 1.0
clutter-texture-set-sync-size (self <clutter-texture>)  
(sync_size bool)  
set-sync-size  
Sets whether texture should have the same preferred size as the underlying image data.

texture a <clutter-texture>
sync-size ‘#t’ if the texture should have the same size of the underlying image data
Since 1.0

clutter-texture-get-repeat (self <clutter-texture>)  
⇒ (repeat_x bool) (repeat_y bool)
get-repeat  
Retrieves the horizontal and vertical repeat values set using clutter-texture-set-repeat

texture a <clutter-texture>
repeat-x return location for the horizontal repeat.
repeat-y return location for the vertical repeat.
Since 1.0

clutter-texture-set-repeat (self <clutter-texture>)  
(repeat_x bool) (repeat_y bool)
set-repeat  
Sets whether the texture should repeat horizontally or vertically when the actor size is bigger than the image size

texture a <clutter-texture>
repeat-x ‘#t’ if the texture should repeat horizontally
repeat-y ‘#t’ if the texture should repeat vertically
Since 1.0

clutter-texture-get-load-async (self <clutter-texture>)  
⇒ (ret bool)  
get-load-async  
Retrieves the value set using clutter-texture-set-load-async

texture a <clutter-texture>
ret ‘#t’ if the <clutter-texture> should load the data from disk asynchronously
Since 1.0

clutter-texture-set-load-async (self <clutter-texture>)  
(load_async bool)
set-load-async  
Sets whether texture should use a worker thread to load the data from disk asynchronously. Setting load-async to ‘#t’ will make clutter-texture-set-from-file return immediately.
See the "load-async" property documentation, and clutter-texture-set-load-data-async.

texture a <clutter-texture>

load-async

'\#t' if the texture should asynchronously load data from a filename

Since 1.0

clutter-texture-get-load-data-async (self <clutter-texture>) [Function]
⇒ (ret bool)

get-load-data-async [Method]
Retrieves the value set by clutter-texture-set-load-data-async

texture a <clutter-texture>
ret '\#t' if the <clutter-texture> should load the image data from a file asynchronously

Since 1.0

clutter-texture-set-load-data-async (self <clutter-texture>) [Function]
(load_async bool)

set-load-data-async [Method]
Sets whether texture should use a worker thread to load the data from disk asynchronously. Setting load-asnc to '\#t' will make clutter-texture-set-from-file block until the <clutter-texture> has determined the width and height of the image data.

See the "load-async" property documentation, and clutter-texture-set-load-data-async.

texture a <clutter-texture>
load-async

'\#t' if the texture should asynchronously load data from a filename

Since 1.0

clutter-texture-get-pick-with-alpha (self <clutter-texture>) [Function]
⇒ (ret bool)

get-pick-with-alpha [Method]
Retrieves the value set by clutter-texture-set-load-data-async

texture a <clutter-texture>
ret '\#t' if the <clutter-texture> should define its shape using the alpha channel when picking.

Since 1.4

clutter-texture-set-pick-with-alpha (self <clutter-texture>) [Function]
(pick_with_alpha bool)

set-pick-with-alpha [Method]
Sets whether texture should have it’s shape defined by the alpha channel when picking.
Be aware that this is a bit more costly than the default picking due to the texture lookup, extra test against the alpha value and the fact that it will also interrupt the batching of geometry done internally.

Also there is currently no control over the threshold used to determine what value of alpha is considered pickable, and so only fully opaque parts of the texture will react to picking.

```
texture  a <clutter-texture>
pick-with-alpha
  ‘#t’ if the alpha channel should affect the picking shape
```

Since 1.4
69 ClutterTimeline

A class for time-based events

69.1 Overview

<clutter-timeline> is a base class for managing time-based event that cause Clutter to redraw a stage, such as animations.

Each <clutter-timeline> instance has a duration: once a timeline has been started, using clutter-timeline-start, it will emit a signal that can be used to update the state of the actors.

It is important to note that <clutter-timeline> is not a generic API for calling closures after an interval; each Timeline is tied into the master clock used to drive the frame cycle. If you need to schedule a closure after an interval, see clutter-threads-add-timeout instead.

Users of <clutter-timeline> should connect to the <"new-frame"> signal, which is emitted each time a timeline is advanced during the maste clock iteration. The <"new-frame"> signal provides the time elapsed since the beginning of the timeline, in milliseconds. A normalized progress value can be obtained by calling clutter-timeline-get-progress. By using clutter-timeline-get-delta it is possible to obtain the wallclock time elapsed since the last emission of the <"new-frame"> signal.

Initial state can be set up by using the <"started"> signal, while final state can be set up by using the <"completed"> signal. The <clutter-timeline> guarantees the emission of at least a single <"new-frame"> signal, as well as the emission of the <"completed"> signal.

It is possible to connect to specific points in the timeline progress by adding markers using clutter-timeline-add-marker-at-time and connecting to the <"marker-reached"> signal.

Timelines can be made to loop once they reach the end of their duration, by using clutter-timeline-set-repeat-count; a looping timeline will still emit the <"completed"> signal once it reaches the end of its duration.

Timelines have a <"direction">: the default direction is 'CLUTTER_TIMELINE_FORWARD', and goes from 0 to the duration; it is possible to change the direction to 'CLUTTER_TIMELINE_BACKWARD', and have the timeline go from the duration to 0. The direction can be automatically reversed when reaching completion by using the <"auto-reverse"> property.

Timelines are used in the Clutter animation framework by classes like <clutter-animation>, <clutter-animator>, and <clutter-state>.

69.2 Defining Timelines in ClutterScript

A <clutter-timeline> can be described in <clutter-script> like any other object. Additionally, it is possible to define markers directly inside the JSON definition by using the markers JSON object member, such as:

```json
{
```
"type" : "ClutterTimeline",
"duration" : 1000,
"markers" : [
  { "name" : "quarter", "time" : 250 },
  { "name" : "half-time", "time" : 500 },
  { "name" : "three-quarters", "time" : 750 }
]
}

### 69.3 Usage

**clutter-timeline-new (msecs unsigned-int)**  
⇒ (ret <clutter-timeline>)  
[Function]

Creates a new <clutter-timeline> with a duration of `msecs`.

- **msecs**  
  Duration of the timeline in milliseconds

- **ret**  
  the newly created <clutter-timeline> instance. Use `g-object-unref` when done using it

Since 0.6

**clutter-timeline-set-duration (self <clutter-timeline>)**  
(msecs unsigned-int)  
[Function]

Sets the duration of the timeline, in milliseconds. The speed of the timeline depends on the ClutterTimeline:fps setting.

- **timeline**  
  a <clutter-timeline>

- **msecs**  
  duration of the timeline in milliseconds

Since 0.6

**clutter-timeline-get-duration (self <clutter-timeline>)**  
⇒ (ret unsigned-int)  
[Function]

Retrieves the duration of a <clutter-timeline> in milliseconds. See clutter-timeline-set-duration.

- **timeline**  
  a <clutter-timeline>

- **ret**  
  the duration of the timeline, in milliseconds.

Since 0.6

**clutter-timeline-set-repeat-count (self <clutter-timeline>)**  
(count int)  
[Function]

Sets the number of times the timeline should repeat.

- **count**  
  number of repeats

If `count` is 0, the timeline never repeats.

If `count` is -1, the timeline will always repeat until it’s stopped.
Chapter 69: ClutterTimeline

**timeline**  a `<clutter-timeline>`

**count**  the number of times the timeline should repeat

Since 1.10

**clutter-timeline-get-repeat-count (self `<clutter-timeline>`)**  [Function]

⇒ (ret int)

**get-repeat-count**  [Method]

Retrieves the number set using clutter-timeline-set-repeat-count.

**timeline**  a `<clutter-timeline>`

**ret**  the number of repeats

Since 1.10

**clutter-timeline-set-delay (self `<clutter-timeline>`)**  [Function]

(msecs unsigned-int)

**set-delay**  [Method]

Sets the delay, in milliseconds, before timeline should start.

**timeline**  a `<clutter-timeline>`

**msecs**  delay in milliseconds

Since 0.4

**clutter-timeline-get-delay (self `<clutter-timeline>`)**  [Function]

⇒ (ret unsigned-int)

**get-delay**  [Method]

Retrieves the delay set using clutter-timeline-set-delay.

**timeline**  a `<clutter-timeline>`

**ret**  the delay in milliseconds.

Since 0.4

**clutter-timeline-set-direction (self `<clutter-timeline>`)**  [Function]

(direction `<clutter-timeline-direction>`)  [Function]

**set-direction**  [Method]

Sets the direction of timeline, either ‘CLUTTER_TIMELINE_FORWARD’ or ‘CLUTTER_TIMELINE_BACKWARD’.

**timeline**  a `<clutter-timeline>`

**direction**  the direction of the timeline

Since 0.6

**clutter-timeline-get-direction (self `<clutter-timeline>`)**  [Function]

⇒ (ret `<clutter-timeline-direction>`)  [Method]

**get-direction**  [Method]

Retrieves the direction of the timeline set with clutter-timeline-set-direction.

**timeline**  a `<clutter-timeline>`
Since 0.6

**clutter-timeline-set-auto-reverse** *(self <clutter-timeline>)*  
*reverse bool*

**set-auto-reverse**  
Sets whether timeline should reverse the direction after the emission of the 
"completed" signal.

Setting the "auto-reverse" property to ‘#t’ is the equivalent of connecting a 
callback to the "completed" signal and changing the direction of the timeline from 
that callback; for instance, this code:

```c
static void
reverse_timeline (ClutterTimeline *timeline)
{
    ClutterTimelineDirection dir = clutter_timeline_get_direction (timeline);

    if (dir == CLUTTER_TIMELINE_FORWARD)
        dir = CLUTTER_TIMELINE_BACKWARD;
    else
        dir = CLUTTER_TIMELINE_FORWARD;

    clutter_timeline_set_direction (timeline, dir);
}
```

...  
```c
timeline = clutter_timeline_new (1000);
clutter_timeline_set_repeat_count (timeline, -1);
g_signal_connect (timeline, "completed",
    G_CALLBACK (reverse_timeline),
    NULL);
```
can be effectively replaced by:

```c
timeline = clutter_timeline_new (1000);
clutter_timeline_set_repeat_count (timeline, -1);
clutter_timeline_set_auto_reverse (timeline);
```

`timeline`  
a <clutter-timeline>

`reverse`  
‘#t’ if the timeline should reverse the direction

Since 1.6

**clutter-timeline-get-auto-reverse** *(self <clutter-timeline>)*  
⇒ *(ret bool)*

**get-auto-reverse**  
Retrieves the value set by clutter-timeline-set-auto-reverse.

`timeline`  
a <clutter-timeline>
Chapter 69: ClutterTimeline

`ret` 't' if the timeline should automatically reverse, and 'f' otherwise

Since 1.6

```
clutter-timeline-set-progress-mode (self <clutter-timeline>)  [Function]
  (mode <clutter-animation-mode>)  [Method]

Sets the progress function using a value from the <clutter-animation-mode> enumeration. The mode cannot be 'CLUTTER_CUSTOM_MODE' or bigger than 'CLUTTER_ANIMATION_LAST'.

timeline   a <clutter-timeline>
mode       the progress mode, as a <clutter-animation-mode>
```

Since 1.10

```
clutter-timeline-get-progress-mode (self <clutter-timeline>)  [Function]
  ⇒ (ret <clutter-animation-mode>)  [Method]

Retrieves the progress mode set using clutter-timeline-set-progress-mode or clutter-timeline-set-progress-func.

timeline   a <clutter-timeline>
ret         a <clutter-animation-mode>
```

Since 1.10

```
clutter-timeline-get-duration-hint (self <clutter-timeline>)  [Function]
  ⇒ (ret int64)  [Method]

Retrieves the full duration of the timeline, taking into account the current value of the "repeat-count" property.
If the "repeat-count" property is set to -1, this function will return 'G_MAXINT64'.
The returned value is to be considered a hint, and it's only valid as long as the timeline hasn't been changed.

timeline   a <clutter-timeline>
ret         the full duration of the <clutter-timeline>
```

Since 1.10

```
clutter-timeline-get-current-repeat (self <clutter-timeline>)  [Function]
  ⇒ (ret int)  [Method]

Retrieves the current repeat for a timeline.
Repeats start at 0.

timeline   a <clutter-timeline>
ret         the current repeat
```

Since 1.10
clutter-timeline-start (self <clutter-timeline>)

[Function]

[Method]

Starts the <clutter-timeline> playing.

timeline A <clutter-timeline>

clutter-timeline-pause (self <clutter-timeline>)

[Function]

[Method]

Pauses the <clutter-timeline> on current frame

timeline A <clutter-timeline>

clutter-timeline-stop (self <clutter-timeline>)

[Function]

[Method]

Stops the <clutter-timeline> and moves to frame 0

timeline A <clutter-timeline>

clutter-timeline-rewind (self <clutter-timeline>)

[Function]

[Method]

Rewinds <clutter-timeline> to the first frame if its direction is 'CLUTTER_TIMELINE_FORWARD' and the last frame if it is 'CLUTTER_TIMELINE_BACKWARD'.

timeline A <clutter-timeline>

clutter-timeline-skip (self <clutter-timeline>)

(msecs unsigned-int)

[Function]

[Method]

Advance timeline by the requested time in milliseconds

timeline A <clutter-timeline>

msecs Amount of time to skip

clutter-timeline-advance (self <clutter-timeline>)

(msecs unsigned-int)

[Function]

[Method]

Advance timeline to the requested point. The point is given as a time in milliseconds since the timeline started.

The timeline will not emit the "new-frame" signal for the given time. The first ::new-frame signal after the call to clutter-timeline-advance will be emit the skipped markers.

timeline A <clutter-timeline>

msecs Time to advance to

clutter-timeline-get-elapsed-time (self <clutter-timeline>)

⇒ (ret unsigned-int)

[Function]

[Method]

Request the current time position of the timeline.

timeline A <clutter-timeline>

ret current elapsed time in milliseconds.
clutter-timeline-get-delta (self <clutter-timeline>)  
⇒ (ret unsigned-int)

get-delta
Retrieves the amount of time elapsed since the last ClutterTimeline::new-frame signal.
This function is only useful inside handlers for the ::new-frame signal, and its beha-

vior is undefined if the timeline is not playing.

timeline   a <clutter-timeline>
ret        the amount of time in milliseconds elapsed since the last frame
Since 0.6

clutter-timeline-get-progress (self <clutter-timeline>)  
⇒ (ret double)

get-progress
The position of the timeline in a normalized [-1, 2] interval.
The return value of this function is determined by the progress mode set using
clutter-timeline-set-progress-mode, or by the progress function set using
clutter-timeline-set-progress-func.

timeline   a <clutter-timeline>
ret        the normalized current position in the timeline.
Since 0.6

clutter-timeline-is-playing (self <clutter-timeline>)  
⇒ (ret bool)
is-playing
Queries state of a <clutter-timeline>.

timeline   A <clutter-timeline>
ret        ‘#t’ if timeline is currently playing

clutter-timeline-add-marker-at-time (self <clutter-timeline>)  
(markers mchars) (msecs unsigned-int)
add-marker-at-time
Adds a named marker that will be hit when the timeline has been running for msecs
milliseconds. Markers are unique string identifiers for a given time. Once timeline
reaches msecs, it will emit a ::marker-reached signal for each marker attached to that
time.
A marker can be removed with clutter-timeline-remove-marker. The timeline
can be advanced to a marker using clutter-timeline-advance-to-marker.

timeline   a <clutter-timeline>
marker-name
the unique name for this marker
msecs     position of the marker in milliseconds
Since 0.8
clutter-timeline-has-marker (self <clutter-timeline>)
  (marker_name mchars) ⇒ (ret bool)

has-marker
  Checks whether timeline has a marker set with the given name.
  
  timeline a <clutter-timeline>
  marker-name the name of the marker
  ret ‘#t’ if the marker was found

Since 0.8

clutter-timeline-remove-marker (self <clutter-timeline>)
  (marker_name mchars)

remove-marker
  Removes marker-name, if found, from timeline.
  
  timeline a <clutter-timeline>
  marker-name the name of the marker to remove

Since 0.8

clutter-timeline-advance-to-marker (self <clutter-timeline>)
  (marker_name mchars)

advance-to-marker
  Advances timeline to the time of the given marker-name.
  Like clutter-timeline-advance, this function will not emit the <"new-frame"> for the time where marker-name is set, nor it will emit <"marker-reached"> for marker-name.
  
  timeline a <clutter-timeline>
  marker-name the name of the marker

Since 0.8
70 ClutterTransition

Transition between two values

70.1 Overview

<clutter-transition> is a subclass of <clutter-timeline> that computes the interpolation between two values, stored by a <clutter-interval>.

70.2 Usage

clutter-transition-set-interval (self <clutter-transition>) (interval <clutter-interval>)  [Function]
set-interval  [Method]
Sets the "interval" property using interval.
The transition will acquire a reference on the interval, sinking the floating flag on it if necessary.

transition  a <clutter-transition>
interval  a <clutter-interval>, or '#f'.

Since 1.10

clutter-transition-get-interval (self <clutter-transition>)  [Function]
⇒ (ret <clutter-interval>)
get-interval  [Method]
Retrieves the interval set using clutter-transition-set-interval

transition  a <clutter-transition>
ret  a <clutter-interval>, or '#f'; the returned interval is owned by the <clutter-transition> and it should not be freed directly.

Since 1.10

clutter-transition-set-animatable (self <clutter-transition>) (animatable <clutter-animatable>)  [Function]
set-animatable  [Method]
Sets the "animatable" property.
The transition will acquire a reference to the animatable instance, and will call the clutter-transition-class.attached virtual function.
If an existing <clutter-animatable> is attached to transition, the reference will be released, and the clutter-transition-class.detached virtual function will be called.

transition  a <clutter-transition>
animatable  a <clutter-animatable>, or '#f'.

Since 1.10
clutter-transition-get-animatable (self <clutter-transition>)  [Function]
⇒ (ret <clutter-animatable>)

get-animatable  [Method]
Retrieves the <clutter-animatable> set using clutter-transition-set-animatable.

  transition a <clutter-transition>

  ret a <clutter-animatable>, or ‘#f’; the returned animatable is owned by
  the <clutter-transition>, and it should not be freed directly.

Since 1.10
71 Unit conversion

A logical distance unit

71.1 Overview

<clutter-units> is a structure holding a logical distance value along with its type, expressed as a value of the <clutter-unit-type> enumeration. It is possible to use <clutter-units> to store a position or a size in units different than pixels, and convert them whenever needed (for instance inside the <clutter-actor>::allocate virtual function, or inside the <clutter-actor>::get-preferred-width and <clutter-actor>::get-preferred-height virtual functions.

In order to register a <clutter-units> property, the <clutter-param-spec-units><gparam> sub-class should be used:

```
GParamSpec *pspec;

pspec = clutter_param_spec_units("active-width",
   "Width",
   "Width of the active area, in millimeters",
   CLUTTER_UNIT_MM,
   0.0, 12.0,
   12.0,
   G_PARAM_READWRITE);

g_object_class_install_property (gobject_class, PROP_WIDTH, pspec);
```

A <gvalue> holding units can be manipulated using clutter-value-set-units and clutter-value-get-units. <gvalue>s containing a <clutter-units> value can also be transformed to <gvalue>s initialized with ‘G_TYPE_INT’, ‘G_TYPE_FLOAT’ and ‘G_TYPE_STRING’ through implicit conversion and using g-value-transform.

<clutter-units> is available since Clutter 1.0

71.2 Usage

clutter-units-from-cm (self <clutter-units>) (cm float) [Function]
Stores a value in centimeters inside units

units a <clutter-units>.

Since 1.2

clutter-units-from-em (em float) ⇒ (ret <clutter-units>) [Function]
Stores a value in em inside units, using the default font name as returned by clutter-backend-get-font-name

units a <clutter-units>.

em em

Since 1.0
clutter-units-from-em-for-font \((font:\text{name} \ m\text{chars}) (em \ \text{float})\) [Function]
\(\Rightarrow (ret <\text{clutter-units}>)\)
Stores a value in em inside units using font-name
\(\text{units}\): a <clutter-units>.
\(\text{font-name}\): the font name and size.
\(\text{em}\): em
Since 1.0

clutter-units-from-mm \((mm \ \text{float})\) \(\Rightarrow (ret <\text{clutter-units}>)\) [Function]
Stores a value in millimeters inside units
\(\text{units}\): a <clutter-units>.
\(\text{mm}\): millimeters
Since 1.0

clutter-units-from-pixels \((px \ \text{int})\) \(\Rightarrow (ret <\text{clutter-units}>)\) [Function]
Stores a value in pixels inside units
\(\text{units}\): a <clutter-units>.
\(\text{px}\): pixels
Since 1.0

clutter-units-from-pt \((pt \ \text{float})\) \(\Rightarrow (ret <\text{clutter-units}>)\) [Function]
Stores a value in typographic points inside units
\(\text{units}\): a <clutter-units>.
\(\text{pt}\): typographic points
Since 1.0

clutter-units-to-pixels \((self <\text{clutter-units}>)\) \(\Rightarrow (ret \ \text{float})\) [Function]
Converts a value in <clutter-units> to pixels
\(\text{units}\): units to convert
\(\text{ret}\): the value in pixels
Since 1.0

clutter-units-get-unit-type \((self <\text{clutter-units}>)\) [Function]
\(\Rightarrow (ret <\text{clutter-unit-type}>)\)
Retrieves the unit type of the value stored inside units
\(\text{units}\): a <clutter-units>
\(\text{ret}\): a unit type
Since 1.0
clutter-units-get-unit-value (self <clutter-units>) ⇒ (ret float)
Retrieves the value stored inside units
units a <clutter-units>
ret the value stored inside a <clutter-units>
Since 1.0

clutter-units-from-string (str mchars) ⇒ (ret <clutter-units>) [Function]
Parses a value and updates units with it
A <clutter-units> expressed in string should match:
units: wsp* unit-value wsp* unit-name? wsp*
unit-value: number
unit-name: 'px' | 'pt' | 'mm' | 'em' | 'cm'
number: digit+
  | digit* sep digit+
sep: '.' | ','
digit: '0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9'
wsp: (0x20 | 0x9 | 0xA | 0xB | 0xC | 0xD)+

For instance, these are valid strings:

  10 px
  5.1 em
  24 pt
  12.6 mm
  .3 cm

While these are not:

  42 cats
  omg!1!ponies

If no unit is specified, pixels are assumed.

units a <clutter-units>.
str the string to convert
ret ‘#t’ if the string was successfully parsed, and ‘#f’ otherwise
Since 1.0

clutter-units-to-string (self <clutter-units>) ⇒ (ret mchars) [Function]
Converts units into a string
See clutter-units-from-string for the units syntax and for examples of output

Fractional values are truncated to the second decimal position for em, mm and cm, and to the first decimal position for typographic points. Pixels are integers.
units a <clutter-units>
ret a newly allocated string containing the encoded <clutter-units> value. Use g-free to free the string

Since 1.0
72 Utilities

Utility functions

72.1 Overview

Various miscellaneous utility functions.

72.2 Usage
73 Versioning Macros

Versioning utility macros

73.1 Overview

Clutter offers a set of macros for checking the version of the library at compile time; it also provides a function to perform the same check at run time.

Clutter adds version information to both API deprecations and additions; by defining the macros ‘CLUTTER_VERSION_MIN_REQUIRED’ and ‘CLUTTER_VERSION_MAX_ALLOWED’, you can specify the range of Clutter versions whose API you want to use. Functions that were deprecated before, or introduced after, this range will trigger compiler warnings. For instance, if we define the following symbols:

```plaintext
CLUTTER_VERSION_MIN_REQUIRED = CLUTTER_VERSION_1_6
CLUTTER_VERSION_MAX_ALLOWED = CLUTTER_VERSION_1_8
```

and we have the following functions annotated in the Clutter headers:

```plaintext
void clutter_function_A (void) CLUTTER_DEPRECATED_IN_1_4;
void clutter_function_B (void) CLUTTER_DEPRECATED_IN_1_6;
void clutter_function_C (void) CLUTTER_AVAILABLE_IN_1_8;
void clutter_function_D (void) CLUTTER_AVAILABLE_IN_1_10;
```

then any application code using the functions above will get the output:

```plaintext
clutter_function_A: deprecation warning
clutter_function_B: no warning
clutter_function_C: no warning
clutter_function_D: symbol not available warning
```

It is possible to disable the compiler warnings by defining the macro ‘CLUTTER_DISABLE_DEPRECATED_WARNINGS’ before including the clutter.h header.

73.2 Usage

`clutter-check-version (major unsigned-int)
(minor unsigned-int) (micro unsigned-int) ⇒ (ret bool)`

Run-time version check, to check the version the Clutter library that an application is currently linked against

This is the run-time equivalent of the compile-time ‘CLUTTER_CHECK_VERSION’ preprocessor macro

- `major` major version, like 1 in 1.2.3
- `minor` minor version, like 2 in 1.2.3
- `micro` micro version, like 3 in 1.2.3
- `ret` ‘#t’ if the version of the Clutter library is greater than (major, minor, micro), and ‘#f’ otherwise

Since 1.2
74 Undocumented

The following symbols, if any, have not been properly documented.

74.1 (gnome clutter)

clutter-interval-get-final interval
clutter-interval-get-initial interval
clutter-interval-get-interval interval
clutter-interval-new type from to
clutter-interval-set-final interval val
clutter-interval-set-initial interval val
clutter-interval-set-interval interval initial final

74.2 (gnome gw clutter)

<clutter-animatable>
<clutter-event-sequence*>
<clutter-input-device*>
clutter-actor-add-constraint-with-name
clutter-actor-allocate-available-size
clutter-actor-allocate-preferred-size
clutter-actor-apply-relative-transform-to-point
clutter-actor-apply-transform-to-point
clutter-actor-get-allocation-geometry
clutter-actor-get-anchor-point-gravity
clutter-actor-get-clip-to-allocation
clutter-actor-get-content-scaling-filters
clutter-actor-get-default-paint-volume
clutter-actor-get-fixed-position-set
clutter-actor-get-offscreen-redirect
clutter-actor-get-paint-volume
clutter-actor-get-transformed-paint-volume
clutter-actor-get-transformed-position
clutter-actor-get-z-rotation-gravity
clutter-actor-move-anchor-point-from-gravity
<table>
<thead>
<tr>
<th>Function/Variable</th>
</tr>
</thead>
<tbody>
<tr>
<td>clutter-actor-remove-all-transitions</td>
</tr>
<tr>
<td>clutter-actor-remove-constraint-by-name</td>
</tr>
<tr>
<td>clutter-actor-set-anchor-point-from-gravity</td>
</tr>
<tr>
<td>clutter-actor-set-child-above-sibling</td>
</tr>
<tr>
<td>clutter-actor-set-child-below-sibling</td>
</tr>
<tr>
<td>clutter-actor-set-clip-to-allocation</td>
</tr>
<tr>
<td>clutter-actor-set-content-scaling-filters</td>
</tr>
<tr>
<td>clutter-actor-set-fixed-position-set</td>
</tr>
<tr>
<td>clutter-actor-set-offscreen-redirect</td>
</tr>
<tr>
<td>clutter-actor-set-scale-with-gravity</td>
</tr>
<tr>
<td>clutter-actor-set-z-rotation-from-gravity</td>
</tr>
<tr>
<td>clutter-align-constraint-get-align-axis</td>
</tr>
<tr>
<td>clutter-align-constraint-set-align-axis</td>
</tr>
<tr>
<td>clutter-animatable-get-initial-state</td>
</tr>
<tr>
<td>clutter-animatable-interpolate-value</td>
</tr>
<tr>
<td>clutter-animator-key-get-property-name</td>
</tr>
<tr>
<td>clutter-animator-key-get-property-type_</td>
</tr>
<tr>
<td>clutter-animator-property-get-ease-in</td>
</tr>
<tr>
<td>clutter-animator-property-get-interpolation</td>
</tr>
<tr>
<td>clutter-animator-property-set-ease-in</td>
</tr>
<tr>
<td>clutter-animator-property-set-interpolation</td>
</tr>
<tr>
<td>clutter-backend-get-font-options_</td>
</tr>
<tr>
<td>clutter-base-init</td>
</tr>
<tr>
<td>clutter-bind-constraint-get-coordinate</td>
</tr>
<tr>
<td>clutter-bind-constraint-set-coordinate</td>
</tr>
<tr>
<td>clutter-binding-pool-install-closure</td>
</tr>
<tr>
<td>clutter-binding-pool-override-closure</td>
</tr>
<tr>
<td>clutter-box-layout-get-easing-duration</td>
</tr>
<tr>
<td>clutter-box-layout-get-use-animations</td>
</tr>
<tr>
<td>clutter-box-layout-set-easing-duration</td>
</tr>
<tr>
<td>clutter-box-layout-set-use-animations</td>
</tr>
<tr>
<td>clutter-brightness-contrast-effect-get-brightness_</td>
</tr>
<tr>
<td>clutter-brightness-contrast-effect-get-contrast_</td>
</tr>
</tbody>
</table>
clutter-brightness-contrast-effect-new [Variable]
clutter-brightness-contrast-effect-set-brightness [Variable]
clutter-brightness-contrast-effect-set-brightness-full [Variable]
clutter-brightness-contrast-effect-set-contrast [Variable]
clutter-brightness-contrast-effect-set-contrast-full [Variable]
clutter-cairo-texture-get-auto-resize [Variable]
clutter-cairo-texture-get-surface-size [Function]
clutter-cairo-texture-set-auto-resize [Variable]
clutter-cairo-texture-set-surface-size [Variable]
clutter-container-child-get-property [Variable]
clutter-container-child-set-property [Variable]
clutter-container-destroy-child-meta [Variable]
clutter-container-find-child-by-name [Variable]
clutter-desaturate-effect-get-factor [Variable]
clutter-desaturate-effect-set-factor [Variable]
clutter-device-manager-get-core-device [Variable]
clutter-device-manager-get-default [Variable]
clutter-drag-action-get-drag-threshold [Function]
clutter-drag-action-get-motion-coords [Function]
clutter-drag-action-get-press-coords [Function]
clutter-drag-action-set-drag-threshold [Variable]
clutter-event-get-scroll-direction [Variable]
clutter-flow-layout-get-column-spacing [Variable]
clutter-flow-layout-get-column-width [Function]
clutter-flow-layout-get-orientation [Variable]
clutter-flow-layout-set-column-spacing [Variable]
clutter-flow-layout-set-column-width [Variable]
clutter-gesture-action-get-motion-coords [Function]
clutter-gesture-action-get-press-coords [Function]
clutter-gesture-action-get-release-coords [Function]
clutter-image-error-quark [Variable]
clutter-input-device-get-associated-device [Variable]
clutter-input-device-get-device-coords [Function]
clutter-input-device-get-device-mode  [Variable]
clutter-input-device-get-device-name  [Variable]
clutter-input-device-get-device-type  [Variable]
clutter-input-device-get-grabbed-actor  [Variable]
clutter-input-device-get-pointer-actor  [Variable]
clutter-input-device-get-pointer-stage  [Variable]
clutter-input-device-get-slave-devices  [Function]
clutter-input-device-keycode-to-evdev  [Function]
clutter-input-device-update-from-event  [Variable]
clutter-knot-equal  [Variable]
clutter-layout-manager-begin-animation  [Variable]
clutter-layout-manager-child-get-property  [Variable]
clutter-layout-manager-child-set-property  [Variable]
clutter-layout-manager-end-animation  [Variable]
clutter-layout-manager-find-child-property  [Variable]
clutter-layout-manager-get-animation-progress  [Variable]
clutter-layout-manager-get-child-meta  [Variable]
clutter-layout-manager-get-preferred-height  [Function]
clutter-layout-manager-get-preferred-width  [Function]
clutter-layout-manager-layout-changed  [Variable]
clutter-layout-manager-set-container  [Variable]
clutter-media-get-subtitle-font-name  [Variable]
clutter-media-set-subtitle-font-name  [Variable]
clutter-offscreen-effect-get-target-size  [Function]
clutter-offscreen-effect-paint-target  [Variable]
clutter-paint-node-add-texture-rectangle  [Variable]
clutter-paint-volume-set-from-allocation  [Variable]
clutter-property-transition-get-property-name  [Variable]
clutter-property-transition-set-property-name  [Variable]
clutter-script-error-quark  [Variable]
clutter-script-get-translation-domain  [Variable]
clutter-script-set-translation-domain  [Variable]
clutter-shader-effect-set-shader-source  [Variable]
clutter-shader-effect-set-uniform-value
clutter-shader-error-quark
clutter-stage-get-motion-events-enabled
clutter-stage-get-throttle-motion-events
clutter-stage-manager-get-default-stage
clutter-stage-set-motion-events-enabled
clutter-stage-set-throttle-motion-events
clutter-state-key-get-source-state-name
clutter-state-key-get-target-state-name
clutter-table-layout-get-column-count
clutter-table-layout-get-column-spacing
clutter-table-layout-get-easing-duration
clutter-table-layout-get-easing-mode
clutter-table-layout-get-row-spacing
clutter-table-layout-get-use-animations
clutter-table-layout-set-column-spacing
clutter-table-layout-set-easing-duration
clutter-table-layout-set-easing-mode
clutter-table-layout-set-row-spacing
clutter-table-layout-set-use-animations
clutter-text-buffer-emit-deleted-text
clutter-text-buffer-emit-inserted-text
clutter-text-get-font-description
clutter-text-get-selected-text-color
clutter-text-set-selected-text-color
clutter-texture-error-quark
clutter-texture-get-filter-quality
clutter-texture-get-keep-aspect-ratio
clutter-texture-set-keep-aspect-ratio
clutter-transition-get-remove-on-complete
clutter-transition-set-remove-on-complete
Type Index
Function Index

A
activate ........................................ 93, 242
add ............................................ 86
add-action ........................................ 49
add-action-with-name ........................................ 50
add-cairo-path ........................................ 184
add-child ........................................ 39, 177
add-close ........................................ 183
add-constraint ........................................ 51
add-curve-to ........................................ 183
add-effect ........................................ 52
add-effect-with-name ........................................ 53
add-line-to ........................................ 182
add-marker-at-time ........................................ 255
add-move-to ........................................ 182
add-node ........................................ 184
add-rectangle ........................................ 177
add-rel-curve-to ........................................ 183
add-rel-line-to ........................................ 182
add-rel-move-to ........................................ 182
add-states ........................................ 194
add-string ........................................ 184
add-transition ........................................ 46
advance ........................................ 254
advance-to-marker ........................................ 256
allocate ........................................ 18, 160
allocate-align-fill ........................................ 18
clutter-actor-box-get-area ...................... 57
clutter-actor-box-get-height .................. 56
clutter-actor-box-get-origin .................. 56
clutter-actor-box-get-size ..................... 57
clutter-actor-box-get-width ................... 56
clutter-actor-box-get-x ......................... 55
clutter-actor-box-get-y ......................... 55
clutter-actor-box-init ........................................ 55
clutter-actor-box-init-rect .................... 55
clutter-actor-box-interpolate .................. 57
clutter-actor-box-new ............................ 54
clutter-actor-box-set-origin .................. 56
clutter-actor-box-set-size .................... 56
clutter-actor-box-union ........................................ 58
clutter-actor-clear-actions .................... 51
clutter-actor-clear-constraints ............... 52
clutter-actor-clear-effects ................... 54
clutter-actor-clear-effects-contents ........ 43
clutter-actor-continue-paint .................. 16
clutter-actor-create-pango-context ............ 48
clutter-actor-create-pango-layout ............. 48
clutter-actor-destroy ........................................ 16
clutter-actor-destroy-all-children .......... 41
clutter-actor-drag ........................................ 75
clutter-actor-event ........................................ 17
clutter-actor-get-accessible ................... 49
clutter-actor-get-action .......................... 51
clutter-actor-get-actions ....................... 50
clutter-actor-get-allocation-box .............. 20
clutter-actor-get-anchor-point ................ 34
clutter-actor-get-animation ...................... 75
clutter-actor-get-background-color ............ 27
clutter-actor-get-child-at-index .............. 42
clutter-actor-get-children ...................... 43
clutter-actor-get-clip .......................... 38
clutter-actor-get-constraint ................... 52
clutter-actor-get-constraints .................. 52
clutter-actor-get-content ....................... 36
clutter-actor-get-content-box ................. 37
clutter-actor-get-content-gravity ............. 37
clutter-actor-get-content-scaling-filters .... 265
clutter-actor-get-depth .......................... 31
clutter-actor-get-easing-delay ................. 45
clutter-actor-get-easing-duration .............. 44
clutter-actor-get-easing-mode .................. 45
clutter-actor-get-effect .......................... 54
clutter-actor-get-effects .......................... 54
clutter-actor-get-first-child ................... 41
clutter-actor-get-flags .......................... 14
clutter-actor-get-height ......................... 29
clutter-actor-get-last-child .................... 42
clutter-actor-get-layout-manager .......... 26
clutter-actor-get-margin ......................... 24
clutter-actor-get-margin-bottom .............. 25
<table>
<thead>
<tr>
<th>Function Name</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>clutter-actor-get-margin-left</td>
<td>26</td>
</tr>
<tr>
<td>clutter-actor-get-margin-right</td>
<td>25</td>
</tr>
<tr>
<td>clutter-actor-get-margin-top</td>
<td>24</td>
</tr>
<tr>
<td>clutter-actor-get-n-children</td>
<td>43</td>
</tr>
<tr>
<td>clutter-actor-get-name</td>
<td>14</td>
</tr>
<tr>
<td>clutter-actor-get-next-sibling</td>
<td>42</td>
</tr>
<tr>
<td>clutter-actor-get-opacity</td>
<td>38</td>
</tr>
<tr>
<td>clutter-actor-get-paint-box</td>
<td>36</td>
</tr>
<tr>
<td>clutter-actor-get-paint-opacity</td>
<td>36</td>
</tr>
<tr>
<td>clutter-actor-get-pango-context</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-get-paint-visibility</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-get-x-align</td>
<td>27</td>
</tr>
<tr>
<td>clutter-actor-get-x-stage</td>
<td>44</td>
</tr>
<tr>
<td>clutter-actor-get-text-direction</td>
<td>49</td>
</tr>
<tr>
<td>clutter-actor-get-transformed-position</td>
<td>265</td>
</tr>
<tr>
<td>clutter-actor-get-transformed-size</td>
<td>35</td>
</tr>
<tr>
<td>clutter-actor-get-transition</td>
<td>45</td>
</tr>
<tr>
<td>clutter-actor-get-width</td>
<td>28</td>
</tr>
<tr>
<td>clutter-actor-get-x</td>
<td>29</td>
</tr>
<tr>
<td>clutter-actor-get-x-align</td>
<td>23</td>
</tr>
<tr>
<td>clutter-actor-get-y</td>
<td>30</td>
</tr>
<tr>
<td>clutter-actor-get-y-align</td>
<td>24</td>
</tr>
<tr>
<td>clutter-actor-grab-key-focus</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-has-actions</td>
<td>50</td>
</tr>
<tr>
<td>clutter-actor-has-allocation</td>
<td>23</td>
</tr>
<tr>
<td>clutter-actor-has-clip</td>
<td>38</td>
</tr>
<tr>
<td>clutter-actor-has-constraints</td>
<td>52</td>
</tr>
<tr>
<td>clutter-actor-has-effects</td>
<td>53</td>
</tr>
<tr>
<td>clutter-actor-has-key-focus</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-has-overlaps</td>
<td>18</td>
</tr>
<tr>
<td>clutter-actor-has-pointer</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-hide</td>
<td>15</td>
</tr>
<tr>
<td>clutter-actor-insert-child-above</td>
<td>39</td>
</tr>
<tr>
<td>clutter-actor-insert-child-at-index</td>
<td>40</td>
</tr>
<tr>
<td>clutter-actor-insert-child-below</td>
<td>40</td>
</tr>
<tr>
<td>clutter-actor-is-in-clone-paint</td>
<td>39</td>
</tr>
<tr>
<td>clutter-actor-is-rotated</td>
<td>33</td>
</tr>
<tr>
<td>clutter-actor-is-scaled</td>
<td>32</td>
</tr>
<tr>
<td>clutter-actor-map</td>
<td>17</td>
</tr>
<tr>
<td>clutter-actor-meta-get-actor</td>
<td>4</td>
</tr>
<tr>
<td>clutter-actor-meta-get-enabled</td>
<td>4</td>
</tr>
<tr>
<td>clutter-actor-meta-get-name</td>
<td>3</td>
</tr>
<tr>
<td>clutter-actor-meta-set-enabled</td>
<td>3</td>
</tr>
<tr>
<td>clutter-actor-meta-set-name</td>
<td>3</td>
</tr>
<tr>
<td>clutter-actor-move-anchor-point</td>
<td>34</td>
</tr>
<tr>
<td>clutter-actor-move-by</td>
<td>30</td>
</tr>
<tr>
<td>clutter-actor-new</td>
<td>13</td>
</tr>
<tr>
<td>clutter-actor-paint</td>
<td>15</td>
</tr>
<tr>
<td>clutter-actor-queue-redraw</td>
<td>16</td>
</tr>
<tr>
<td>clutter-actor-queue-relayout</td>
<td>16</td>
</tr>
<tr>
<td>clutter-actor-realize</td>
<td>15</td>
</tr>
<tr>
<td>clutter-actor-remove-action</td>
<td>50</td>
</tr>
<tr>
<td>clutter-actor-remove-action-by-name</td>
<td>50</td>
</tr>
<tr>
<td>clutter-actor-remove-all-children</td>
<td>41</td>
</tr>
<tr>
<td>clutter-actor-remove-child</td>
<td>40</td>
</tr>
<tr>
<td>clutter-actor-remove-clip</td>
<td>38</td>
</tr>
<tr>
<td>clutter-actor-remove-constraint</td>
<td>51</td>
</tr>
<tr>
<td>clutter-actor-remove-effect</td>
<td>53</td>
</tr>
<tr>
<td>clutter-actor-remove-effect-by-name</td>
<td>53</td>
</tr>
<tr>
<td>clutter-actor-remove-transition</td>
<td>46</td>
</tr>
<tr>
<td>clutter-actor-replace-child</td>
<td>40</td>
</tr>
<tr>
<td>clutter-actor-restore-easing-state</td>
<td>44</td>
</tr>
<tr>
<td>clutter-actor-save-easing-state</td>
<td>44</td>
</tr>
<tr>
<td>clutter-actor-set-allocation</td>
<td>19</td>
</tr>
<tr>
<td>clutter-actor-set-anchor-point</td>
<td>34</td>
</tr>
<tr>
<td>clutter-actor-set-background-color</td>
<td>26</td>
</tr>
<tr>
<td>clutter-actor-set-child-at-index</td>
<td>43</td>
</tr>
<tr>
<td>clutter-actor-set-clap</td>
<td>37</td>
</tr>
<tr>
<td>clutter-actor-set-content</td>
<td>36</td>
</tr>
<tr>
<td>clutter-actor-set-content-gravity</td>
<td>37</td>
</tr>
<tr>
<td>clutter-actor-set-depth</td>
<td>31</td>
</tr>
<tr>
<td>clutter-actor-set-easing-delay</td>
<td>45</td>
</tr>
<tr>
<td>clutter-actor-set-easing-duration</td>
<td>44</td>
</tr>
<tr>
<td>clutter-actor-set-easing-mode</td>
<td>45</td>
</tr>
<tr>
<td>clutter-actor-set-flags</td>
<td>13</td>
</tr>
<tr>
<td>clutter-actor-set-height</td>
<td>29</td>
</tr>
<tr>
<td>clutter-actor-set-layout-manager</td>
<td>26</td>
</tr>
<tr>
<td>clutter-actor-set-margin</td>
<td>24</td>
</tr>
<tr>
<td>clutter-actor-set-margin-bottom</td>
<td>25</td>
</tr>
<tr>
<td>clutter-actor-set-margin-left</td>
<td>25</td>
</tr>
<tr>
<td>clutter-actor-set-margin-right</td>
<td>25</td>
</tr>
<tr>
<td>clutter-actor-set-margin-top</td>
<td>24</td>
</tr>
<tr>
<td>clutter-actor-set-name</td>
<td>14</td>
</tr>
<tr>
<td>clutter-actor-set-opacity</td>
<td>38</td>
</tr>
<tr>
<td>clutter-actor-set-position</td>
<td>27</td>
</tr>
<tr>
<td>clutter-actor-set-reactive</td>
<td>47</td>
</tr>
<tr>
<td>clutter-actor-set-request-mode</td>
<td>22</td>
</tr>
<tr>
<td>clutter-actor-set-rotation</td>
<td>32</td>
</tr>
<tr>
<td>clutter-actor-set-scale</td>
<td>31</td>
</tr>
<tr>
<td>clutter-actor-set-scale-full</td>
<td>31</td>
</tr>
<tr>
<td>clutter-actor-set-size</td>
<td>27</td>
</tr>
<tr>
<td>clutter-actor-set-text-direction</td>
<td>48</td>
</tr>
<tr>
<td>clutter-actor-set-width</td>
<td>28</td>
</tr>
<tr>
<td>clutter-actor-set-x</td>
<td>29</td>
</tr>
<tr>
<td>clutter-actor-set-y</td>
<td>30</td>
</tr>
<tr>
<td>clutter-actor-set-y-align</td>
<td>23</td>
</tr>
<tr>
<td>clutter-actor-set-y</td>
<td>30</td>
</tr>
<tr>
<td>clutter-actor-should-pick-paint</td>
<td>17</td>
</tr>
<tr>
<td>clutter-actor-show</td>
<td>14</td>
</tr>
<tr>
<td>clutter-actor-unmap</td>
<td>17</td>
</tr>
<tr>
<td>clutter-actor-unset-flags</td>
<td>14</td>
</tr>
<tr>
<td>clutter-actor-unrealize</td>
<td>15</td>
</tr>
<tr>
<td>clutter-actor-unset-flags</td>
<td>14</td>
</tr>
<tr>
<td>clutter-align-constraint-get-factor</td>
<td>64</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clutter-align-constraint-get-source</td>
<td>63</td>
</tr>
<tr>
<td>clutter-align-constraint-new</td>
<td>63</td>
</tr>
<tr>
<td>clutter-align-constraint-set-factor</td>
<td>64</td>
</tr>
<tr>
<td>clutter-align-constraint-set-source</td>
<td>63</td>
</tr>
<tr>
<td>clutter-alpha-get-alpha</td>
<td>67</td>
</tr>
<tr>
<td>clutter-alpha-get-mode</td>
<td>67</td>
</tr>
<tr>
<td>clutter-alpha-get-timeline</td>
<td>66</td>
</tr>
<tr>
<td>clutter-alpha-new</td>
<td>66</td>
</tr>
<tr>
<td>clutter-alpha-register-timeline</td>
<td>66</td>
</tr>
<tr>
<td>clutter-alpha-set-timeline</td>
<td>66</td>
</tr>
<tr>
<td>clutter-alpha-set-mode</td>
<td>66</td>
</tr>
<tr>
<td>clutter-alpha-set-timeline</td>
<td>66</td>
</tr>
<tr>
<td>clutter-animateable-find-property</td>
<td>68</td>
</tr>
<tr>
<td>clutter-animateable-set-final-state</td>
<td>68</td>
</tr>
<tr>
<td>clutter-animation-bind</td>
<td>73</td>
</tr>
<tr>
<td>clutter-animation-bind-interval</td>
<td>73</td>
</tr>
<tr>
<td>clutter-animation-completed</td>
<td>73</td>
</tr>
<tr>
<td>clutter-animation-get-duration</td>
<td>72</td>
</tr>
<tr>
<td>clutter-animation-get-duration-override</td>
<td></td>
</tr>
<tr>
<td>clutter-animation-get-interval</td>
<td>75</td>
</tr>
<tr>
<td>clutter-animation-get-loop</td>
<td>72</td>
</tr>
<tr>
<td>clutter-animation-get-mode</td>
<td>71</td>
</tr>
<tr>
<td>clutter-animation-get-object</td>
<td>71</td>
</tr>
<tr>
<td>clutter-animation-get-timeline</td>
<td>72</td>
</tr>
<tr>
<td>clutter-animation-has-property</td>
<td>74</td>
</tr>
<tr>
<td>clutter-animation-new</td>
<td>70</td>
</tr>
<tr>
<td>clutter-animation-set-duration</td>
<td>71</td>
</tr>
<tr>
<td>clutter-animation-set-loop</td>
<td>72</td>
</tr>
<tr>
<td>clutter-animation-set-mode</td>
<td>71</td>
</tr>
<tr>
<td>clutter-animation-set-object</td>
<td>70</td>
</tr>
<tr>
<td>clutter-animation-set-timeline</td>
<td>72</td>
</tr>
<tr>
<td>clutter-animation-unbind-property</td>
<td>74</td>
</tr>
<tr>
<td>clutter-animation-update</td>
<td>74</td>
</tr>
<tr>
<td>clutter-animation-update-interval</td>
<td>74</td>
</tr>
<tr>
<td>clutter-animator-compute-value</td>
<td>79</td>
</tr>
<tr>
<td>clutter-animator-get-duration</td>
<td>80</td>
</tr>
<tr>
<td>clutter-animator-get-keys</td>
<td>78</td>
</tr>
<tr>
<td>clutter-animator-get-timeline</td>
<td>79</td>
</tr>
<tr>
<td>clutter-animator-key-get-mode</td>
<td>80</td>
</tr>
<tr>
<td>clutter-animator-key-get-object</td>
<td>80</td>
</tr>
<tr>
<td>clutter-animator-key-get-progress</td>
<td>80</td>
</tr>
<tr>
<td>clutter-animator-key-get-property-type</td>
<td>266</td>
</tr>
<tr>
<td>clutter-animator-key-get-value</td>
<td>81</td>
</tr>
<tr>
<td>clutter-animator-new</td>
<td>77</td>
</tr>
<tr>
<td>clutter-animator-remove-key</td>
<td>78</td>
</tr>
<tr>
<td>clutter-animator-set-duration</td>
<td>80</td>
</tr>
<tr>
<td>clutter-animator-set-key</td>
<td>77</td>
</tr>
<tr>
<td>clutter-animator-set-timeline</td>
<td>79</td>
</tr>
<tr>
<td>clutter-animator-start</td>
<td>78</td>
</tr>
<tr>
<td>clutter-backend-get-font-options</td>
<td>266</td>
</tr>
<tr>
<td>clutter-backend-get-resolution</td>
<td>82</td>
</tr>
<tr>
<td>clutter-backend-set-font-options</td>
<td>82</td>
</tr>
<tr>
<td>clutter-bin-layout-add</td>
<td>86</td>
</tr>
<tr>
<td>clutter-bin-layout-get-alignment</td>
<td>85</td>
</tr>
<tr>
<td>clutter-bin-layout-new</td>
<td>85</td>
</tr>
<tr>
<td>clutter-bin-layout-set-alignment</td>
<td>85</td>
</tr>
<tr>
<td>clutter-bind-constraint-get-offset</td>
<td>89</td>
</tr>
<tr>
<td>clutter-bind-constraint-get-source</td>
<td>88</td>
</tr>
<tr>
<td>clutter-bind-constraint-new</td>
<td>88</td>
</tr>
<tr>
<td>clutter-bind-constraint-set-offset</td>
<td>88</td>
</tr>
<tr>
<td>clutter-bind-constraint-set-source</td>
<td>88</td>
</tr>
<tr>
<td>clutter-binding-pool-activate</td>
<td>93</td>
</tr>
<tr>
<td>clutter-binding-pool-block-action</td>
<td>92</td>
</tr>
<tr>
<td>clutter-binding-pool-find</td>
<td>91</td>
</tr>
<tr>
<td>clutter-binding-pool-find-action</td>
<td>92</td>
</tr>
<tr>
<td>clutter-binding-pool-get-for-class</td>
<td>91</td>
</tr>
<tr>
<td>clutter-binding-pool-new</td>
<td>91</td>
</tr>
<tr>
<td>clutter-binding-pool-remove-action</td>
<td>92</td>
</tr>
<tr>
<td>clutter-binding-pool-unblock-action</td>
<td>92</td>
</tr>
<tr>
<td>clutter-blur-effect-new</td>
<td>94</td>
</tr>
<tr>
<td>clutter-box-layout-get-alignment</td>
<td>98</td>
</tr>
<tr>
<td>clutter-box-layout-get-easing-mode</td>
<td>99</td>
</tr>
<tr>
<td>clutter-box-layout-get-expand</td>
<td>98</td>
</tr>
<tr>
<td>clutter-box-layout-get-fill</td>
<td>99</td>
</tr>
<tr>
<td>clutter-box-layout-get-homogeneous</td>
<td>97</td>
</tr>
<tr>
<td>clutter-box-layout-get-pack-start</td>
<td>96</td>
</tr>
<tr>
<td>clutter-box-layout-get-spacing</td>
<td>96</td>
</tr>
<tr>
<td>clutter-box-layout-get-vertical</td>
<td>97</td>
</tr>
<tr>
<td>clutter-box-layout-new</td>
<td>95</td>
</tr>
<tr>
<td>clutter-box-layout-pack</td>
<td>97</td>
</tr>
<tr>
<td>clutter-box-layout-set-alignment</td>
<td>98</td>
</tr>
<tr>
<td>clutter-box-layout-set-easing-mode</td>
<td>99</td>
</tr>
<tr>
<td>clutter-box-layout-set-expand</td>
<td>98</td>
</tr>
<tr>
<td>clutter-box-layout-set-fill</td>
<td>99</td>
</tr>
<tr>
<td>clutter-box-layout-set-homogeneous</td>
<td>97</td>
</tr>
<tr>
<td>clutter-box-layout-set-pack-start</td>
<td>96</td>
</tr>
<tr>
<td>clutter-box-layout-set-spacing</td>
<td>96</td>
</tr>
<tr>
<td>clutter-box-layout-set-vertical</td>
<td>96</td>
</tr>
<tr>
<td>clutter-brightness-contrast-effect-get-brightness</td>
<td>266</td>
</tr>
<tr>
<td>clutter-brightness-contrast-effect-get-contrast</td>
<td>266</td>
</tr>
<tr>
<td>clutter-cairo-set-source-color</td>
<td>102</td>
</tr>
<tr>
<td>clutter-cairo-texture-clear</td>
<td>102</td>
</tr>
<tr>
<td>clutter-cairo-texture-get-surface-size</td>
<td>267</td>
</tr>
<tr>
<td>clutter-cairo-texture-invalidate</td>
<td>101</td>
</tr>
<tr>
<td>clutter-cairo-texture-new</td>
<td>101</td>
</tr>
<tr>
<td>clutter-canvas-new</td>
<td>103</td>
</tr>
<tr>
<td>clutter-canvas-set-size</td>
<td>103</td>
</tr>
<tr>
<td>clutter-check-version</td>
<td>264</td>
</tr>
<tr>
<td>clutter-check-windowing-backend</td>
<td>83</td>
</tr>
<tr>
<td>clutter-child-meta-get-actor</td>
<td>104</td>
</tr>
<tr>
<td>clutter-child-meta-get-container</td>
<td>104</td>
</tr>
<tr>
<td>clutter-click-action-get-button</td>
<td>106</td>
</tr>
<tr>
<td>clutter-click-action-get-coords</td>
<td>106</td>
</tr>
<tr>
<td>clutter-click-action-get-state</td>
<td>106</td>
</tr>
<tr>
<td>clutter-click-action-new</td>
<td>106</td>
</tr>
<tr>
<td>clutter-click-action-release</td>
<td>107</td>
</tr>
<tr>
<td>clutter-click-action-release</td>
<td>107</td>
</tr>
<tr>
<td>clutter-click-action-release</td>
<td>107</td>
</tr>
<tr>
<td>clutter-click-action-release</td>
<td>107</td>
</tr>
<tr>
<td>clutter-clamp-node-new</td>
<td>178</td>
</tr>
<tr>
<td>clutter-clone-get-source</td>
<td>108</td>
</tr>
<tr>
<td>clutter-clone-new</td>
<td>108</td>
</tr>
<tr>
<td>clutter-clone-set-source</td>
<td>108</td>
</tr>
<tr>
<td>clutter-color-add</td>
<td>111</td>
</tr>
<tr>
<td>clutter-color-darken</td>
<td>112</td>
</tr>
<tr>
<td>clutter-color-from-hls</td>
<td>110</td>
</tr>
<tr>
<td>clutter-color-from-pixel</td>
<td>111</td>
</tr>
<tr>
<td>clutter-color-from-string</td>
<td>109</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clutter-color-get-static</td>
<td>109</td>
</tr>
<tr>
<td>clutter-color-interpolate</td>
<td>112</td>
</tr>
<tr>
<td>clutter-color-lighten</td>
<td>112</td>
</tr>
<tr>
<td>clutter-color-new</td>
<td>109</td>
</tr>
<tr>
<td>clutter-color-node-new</td>
<td>178</td>
</tr>
<tr>
<td>clutter-color-shade</td>
<td>112</td>
</tr>
<tr>
<td>clutter-color-subtract</td>
<td>111</td>
</tr>
<tr>
<td>clutter-color-to-hls</td>
<td>111</td>
</tr>
<tr>
<td>clutter-color-to-pixel</td>
<td>111</td>
</tr>
<tr>
<td>clutter-color-to-string</td>
<td>110</td>
</tr>
<tr>
<td>clutter-colorize-effect-get-tint</td>
<td>114</td>
</tr>
<tr>
<td>clutter-colorize-effect-get-new</td>
<td>114</td>
</tr>
<tr>
<td>clutter-colorize-effect-set-tint</td>
<td>114</td>
</tr>
<tr>
<td>clutter-container-child-notify</td>
<td>117</td>
</tr>
<tr>
<td>clutter-container-create-child-meta</td>
<td>117</td>
</tr>
<tr>
<td>clutter-container-get-child-meta</td>
<td>117</td>
</tr>
<tr>
<td>clutter-content-get-preferred-size</td>
<td>119</td>
</tr>
<tr>
<td>clutter-content-invalidate</td>
<td>119</td>
</tr>
<tr>
<td>clutter-deform-effect-get-n-tiles</td>
<td>120</td>
</tr>
<tr>
<td>clutter-deform-effect-invalidate</td>
<td>121</td>
</tr>
<tr>
<td>clutter-deform-effect-set-n-tiles</td>
<td>120</td>
</tr>
<tr>
<td>clutter-desaturate-effect-new</td>
<td>122</td>
</tr>
<tr>
<td>clutter-device-manager-get-device</td>
<td>123</td>
</tr>
<tr>
<td>clutter-device-manager-list-devices</td>
<td>123</td>
</tr>
<tr>
<td>clutter-device-manager-peek-devices</td>
<td>123</td>
</tr>
<tr>
<td>clutter-do-event</td>
<td>166</td>
</tr>
<tr>
<td>clutter-drag-action-get-drag-axis</td>
<td>126</td>
</tr>
<tr>
<td>clutter-drag-action-get-drag-handle</td>
<td>126</td>
</tr>
<tr>
<td>clutter-drag-action-get-drag-threshold</td>
<td>267</td>
</tr>
<tr>
<td>clutter-drag-action-get-motion-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-drag-action-get-press-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-drag-action-new</td>
<td>125</td>
</tr>
<tr>
<td>clutter-drag-action-set-drag-axis</td>
<td>126</td>
</tr>
<tr>
<td>clutter-drag-action-set-drag-handle</td>
<td>125</td>
</tr>
<tr>
<td>clutter-drop-action-new</td>
<td>127</td>
</tr>
<tr>
<td>clutter-effect-queue-repaint</td>
<td>131</td>
</tr>
<tr>
<td>clutter-event-get</td>
<td>135</td>
</tr>
<tr>
<td>clutter-event-get-button</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-get-click-count</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-get-coords</td>
<td>132</td>
</tr>
<tr>
<td>clutter-event-get-device</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-get-device-id</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-get-device-type</td>
<td>140</td>
</tr>
<tr>
<td>clutter-event-get-event-sequence</td>
<td>135</td>
</tr>
<tr>
<td>clutter-event-get-flags</td>
<td>134</td>
</tr>
<tr>
<td>clutter-event-get-key-code</td>
<td>137</td>
</tr>
<tr>
<td>clutter-event-get-key-symbol</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-get-key-unicode</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-get-related</td>
<td>138</td>
</tr>
<tr>
<td>clutter-event-get-scroll-delta</td>
<td>138</td>
</tr>
<tr>
<td>clutter-event-get-source</td>
<td>134</td>
</tr>
<tr>
<td>clutter-event-get-source-device</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-get-state</td>
<td>134</td>
</tr>
<tr>
<td>clutter-event-get-state-device</td>
<td>133</td>
</tr>
<tr>
<td>clutter-event-get-time</td>
<td>133</td>
</tr>
<tr>
<td>clutter-event-new</td>
<td>132</td>
</tr>
<tr>
<td>clutter-event-peek</td>
<td>135</td>
</tr>
<tr>
<td>clutter-event-put</td>
<td>135</td>
</tr>
<tr>
<td>clutter-event-set-button</td>
<td>135</td>
</tr>
<tr>
<td>clutter-event-set-coords</td>
<td>132</td>
</tr>
<tr>
<td>clutter-event-set-device</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-set-flags</td>
<td>134</td>
</tr>
<tr>
<td>clutter-event-set-key-code</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-set-key-symbol</td>
<td>136</td>
</tr>
<tr>
<td>clutter-event-set-key-unicode</td>
<td>137</td>
</tr>
<tr>
<td>clutter-event-set-related</td>
<td>138</td>
</tr>
<tr>
<td>clutter-event-set-scroll-delta</td>
<td>138</td>
</tr>
<tr>
<td>clutter-event-set-scroll-direction</td>
<td>138</td>
</tr>
<tr>
<td>clutter-event-set-source</td>
<td>133</td>
</tr>
<tr>
<td>clutter-event-set-source-device</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-set-source-device</td>
<td>139</td>
</tr>
<tr>
<td>clutter-event-set-stage</td>
<td>134</td>
</tr>
<tr>
<td>clutter-event-set-stale</td>
<td>133</td>
</tr>
<tr>
<td>clutter-event-set-time</td>
<td>133</td>
</tr>
<tr>
<td>clutter-event-type</td>
<td>132</td>
</tr>
<tr>
<td>clutter-events-pending</td>
<td>135</td>
</tr>
<tr>
<td>clutter-feature-available</td>
<td>141</td>
</tr>
<tr>
<td>clutter-feature-get-all</td>
<td>141</td>
</tr>
<tr>
<td>clutter-fixed-layout-new</td>
<td>142</td>
</tr>
<tr>
<td>clutter-flow-layout-get-column-width</td>
<td>267</td>
</tr>
<tr>
<td>clutter-flow-layout-get-homogeneous</td>
<td>144</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-height</td>
<td>145</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>clutter-flow-layout-get-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>clutter-geometry-intersects</td>
<td>59</td>
</tr>
<tr>
<td>clutter-geometry-union</td>
<td>58</td>
</tr>
<tr>
<td>clutter-gesture-action-get-motion-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-gesture-action-get-motion-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-gesture-action-get-press-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-gesture-action-get-release-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-gesture-action-get-release-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-get-accessibility-enabled</td>
<td>164</td>
</tr>
<tr>
<td>clutter-get-current-event</td>
<td>140</td>
</tr>
<tr>
<td>clutter-get-current-event-time</td>
<td>140</td>
</tr>
<tr>
<td>clutter-get-default-backend</td>
<td>82</td>
</tr>
<tr>
<td>clutter-get-default-frame-rate</td>
<td>164</td>
</tr>
<tr>
<td>clutter-get-default-text-direction</td>
<td>164</td>
</tr>
<tr>
<td>clutter-get-font-map</td>
<td>164</td>
</tr>
<tr>
<td>clutter-get-keyboard-grab</td>
<td>165</td>
</tr>
<tr>
<td>clutter-get-pointer-grab</td>
<td>165</td>
</tr>
<tr>
<td>clutter-get-script-id</td>
<td>195</td>
</tr>
<tr>
<td>clutter-grab-keyboard</td>
<td>165</td>
</tr>
<tr>
<td>clutter-grab-pointer</td>
<td>165</td>
</tr>
<tr>
<td>clutter-image-new</td>
<td>147</td>
</tr>
<tr>
<td>clutter-input-device-get-axis</td>
<td>149</td>
</tr>
<tr>
<td>clutter-input-device-get-axis-value</td>
<td>150</td>
</tr>
<tr>
<td>clutter-input-device-get-device-coords</td>
<td>267</td>
</tr>
<tr>
<td>clutter-input-device-get-device-id</td>
<td>148</td>
</tr>
<tr>
<td>clutter-input-device-get-enabled</td>
<td>148</td>
</tr>
<tr>
<td>clutter-input-device-get-has-cursor</td>
<td>148</td>
</tr>
<tr>
<td>clutter-input-device-get-key</td>
<td>149</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clutter-input-device-get-n-axes</td>
<td>149</td>
</tr>
<tr>
<td>clutter-input-device-get-n-keys</td>
<td>149</td>
</tr>
<tr>
<td>clutter-input-device-get-slave-devices</td>
<td>268</td>
</tr>
<tr>
<td>clutter-input-device-grab</td>
<td>150</td>
</tr>
<tr>
<td>clutter-input-device-keycode-to-evdev</td>
<td>268</td>
</tr>
<tr>
<td>clutter-input-device-set-enabled</td>
<td>148</td>
</tr>
<tr>
<td>clutter-input-device-set-key</td>
<td>149</td>
</tr>
<tr>
<td>clutter-input-device-ungrab</td>
<td>151</td>
</tr>
<tr>
<td>clutter-interval-clone</td>
<td>152</td>
</tr>
<tr>
<td>clutter-interval-compute</td>
<td>154</td>
</tr>
<tr>
<td>clutter-interval-get-final</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-final-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-initial</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-set</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-value-type</td>
<td>152</td>
</tr>
<tr>
<td>clutter-interval-get-value-type</td>
<td>152</td>
</tr>
<tr>
<td>clutter-interval-get-value-type</td>
<td>152</td>
</tr>
<tr>
<td>clutter-interval-get-final</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-final-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-initial</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-set</td>
<td>265</td>
</tr>
<tr>
<td>clutter-interval-get-set-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-get-set-value</td>
<td>153</td>
</tr>
<tr>
<td>clutter-interval-valide</td>
<td>154</td>
</tr>
<tr>
<td>clutter-keysym-to-unicode</td>
<td>137</td>
</tr>
<tr>
<td>clutter-layot-manager-allocate</td>
<td>160</td>
</tr>
<tr>
<td>clutter-layot-manager-get-preferred-height</td>
<td>268</td>
</tr>
<tr>
<td>clutter-layot-manager-get-preferred-width</td>
<td>268</td>
</tr>
<tr>
<td>clutter-layot-meta-get-manager</td>
<td>161</td>
</tr>
<tr>
<td>clutter-main</td>
<td>164</td>
</tr>
<tr>
<td>clutter-main-level</td>
<td>164</td>
</tr>
<tr>
<td>clutter-main.Quit</td>
<td>164</td>
</tr>
<tr>
<td>clutter-margin-new</td>
<td>24</td>
</tr>
<tr>
<td>clutter-media-get-audio-volume</td>
<td>169</td>
</tr>
<tr>
<td>clutter-media-get-buffer-fill</td>
<td>169</td>
</tr>
<tr>
<td>clutter-media-get-can-seek</td>
<td>169</td>
</tr>
<tr>
<td>clutter-media-get-duration</td>
<td>169</td>
</tr>
<tr>
<td>clutter-media-get-playing</td>
<td>167</td>
</tr>
<tr>
<td>clutter-media-get-progress</td>
<td>168</td>
</tr>
<tr>
<td>clutter-media-get-subtitle-uri</td>
<td>168</td>
</tr>
<tr>
<td>clutter-media-get-uri</td>
<td>167</td>
</tr>
<tr>
<td>clutter-media-get-set-audio-volume</td>
<td>168</td>
</tr>
<tr>
<td>clutter-media-get-set-file</td>
<td>169</td>
</tr>
<tr>
<td>clutter-media-get-set-play</td>
<td>167</td>
</tr>
<tr>
<td>clutter-media-get-set-progress</td>
<td>168</td>
</tr>
<tr>
<td>clutter-media-get-set-subtitle-uri</td>
<td>168</td>
</tr>
<tr>
<td>clutter-media-get-set-uri</td>
<td>167</td>
</tr>
<tr>
<td>clutter-offscreen-effect-get-target-size</td>
<td>268</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-angle</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-radius</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-new</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-set-angle</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-set-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-get-radius</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-angle</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-radius</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-angle</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-radius</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-angle</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-radius</td>
<td>176</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-angle</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-period</td>
<td>175</td>
</tr>
<tr>
<td>clutter-page-turn-effect-set-radius</td>
<td>176</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>----------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clutter-snap-constraint-get-edges</td>
<td>202</td>
</tr>
<tr>
<td>clutter-snap-constraint-get-offset</td>
<td>202</td>
</tr>
<tr>
<td>clutter-snap-constraint-get-source</td>
<td>201</td>
</tr>
<tr>
<td>clutter-snap-constraint-new</td>
<td>201</td>
</tr>
<tr>
<td>clutter-snap-constraint-set-edges</td>
<td>202</td>
</tr>
<tr>
<td>clutter-snap-constraint-set-offset</td>
<td>202</td>
</tr>
<tr>
<td>clutter-snap-constraint-set-source</td>
<td>201</td>
</tr>
<tr>
<td>clutter-stage-ensure-current</td>
<td>205</td>
</tr>
<tr>
<td>clutter-stage-ensure-redraw</td>
<td>206</td>
</tr>
<tr>
<td>clutter-stage-ensure-viewpoint</td>
<td>206</td>
</tr>
<tr>
<td>clutter-stage-event</td>
<td>206</td>
</tr>
<tr>
<td>clutter-stage-get-accept-focus</td>
<td>208</td>
</tr>
<tr>
<td>clutter-stage-get-actor-at-pos</td>
<td>205</td>
</tr>
<tr>
<td>clutter-stage-get-fullscreen</td>
<td>205</td>
</tr>
<tr>
<td>clutter-stage-get-key-focus</td>
<td>206</td>
</tr>
<tr>
<td>clutter-stage-get-minimum-size</td>
<td>207</td>
</tr>
<tr>
<td>clutter-stage-get-no-clear-hint</td>
<td>208</td>
</tr>
<tr>
<td>clutter-stage-get-perspective</td>
<td>209</td>
</tr>
<tr>
<td>clutter-stage-get-title</td>
<td>209</td>
</tr>
<tr>
<td>clutter-stage-get-use-alpha</td>
<td>207</td>
</tr>
<tr>
<td>clutter-stage-get-user-resizable</td>
<td>210</td>
</tr>
<tr>
<td>clutter-stage-hide-cursor</td>
<td>205</td>
</tr>
<tr>
<td>clutter-stage-manager-get-default</td>
<td>203</td>
</tr>
<tr>
<td>clutter-stage-manager-get-default-list-stages</td>
<td>203</td>
</tr>
<tr>
<td>clutter-stage-manager-peek-stages</td>
<td>203</td>
</tr>
<tr>
<td>clutter-stage-new</td>
<td>204</td>
</tr>
<tr>
<td>clutter-stage-set-accept-focus</td>
<td>208</td>
</tr>
<tr>
<td>clutter-stage-set-fullscreen</td>
<td>204</td>
</tr>
<tr>
<td>clutter-stage-set-key-focus</td>
<td>206</td>
</tr>
<tr>
<td>clutter-stage-set-minimum-size</td>
<td>207</td>
</tr>
<tr>
<td>clutter-stage-set-no-clear-hint</td>
<td>208</td>
</tr>
<tr>
<td>clutter-stage-set-perspective</td>
<td>209</td>
</tr>
<tr>
<td>clutter-stage-set-title</td>
<td>209</td>
</tr>
<tr>
<td>clutter-stage-set-use-alpha</td>
<td>207</td>
</tr>
<tr>
<td>clutter-stage-set-user-resizable</td>
<td>209</td>
</tr>
<tr>
<td>clutter-stage-show-cursor</td>
<td>205</td>
</tr>
<tr>
<td>clutter-state-get-animator</td>
<td>218</td>
</tr>
<tr>
<td>clutter-state-get-duration</td>
<td>216</td>
</tr>
<tr>
<td>clutter-state-get-keys</td>
<td>216</td>
</tr>
<tr>
<td>clutter-state-get-state</td>
<td>214</td>
</tr>
<tr>
<td>clutter-state-get-timeline</td>
<td>217</td>
</tr>
<tr>
<td>clutter-state-key-get-mode</td>
<td>218</td>
</tr>
<tr>
<td>clutter-state-key-get-object</td>
<td>218</td>
</tr>
<tr>
<td>clutter-state-key-get-post-delay</td>
<td>219</td>
</tr>
<tr>
<td>clutter-state-key-get-pre-delay</td>
<td>219</td>
</tr>
<tr>
<td>clutter-state-key-get-property-name</td>
<td>218</td>
</tr>
<tr>
<td>clutter-state-key-get-property-type</td>
<td>219</td>
</tr>
<tr>
<td>clutter-state-key-get-value</td>
<td>219</td>
</tr>
<tr>
<td>clutter-state-new</td>
<td>214</td>
</tr>
<tr>
<td>clutter-state-remove-key</td>
<td>217</td>
</tr>
<tr>
<td>clutter-state-set-animator</td>
<td>217</td>
</tr>
<tr>
<td>clutter-state-set-duration</td>
<td>215</td>
</tr>
<tr>
<td>clutter-state-set-key</td>
<td>215</td>
</tr>
<tr>
<td>clutter-state-set-state</td>
<td>214</td>
</tr>
<tr>
<td>clutter-state-warp-to-state</td>
<td>214</td>
</tr>
<tr>
<td>clutter-swipe-action-new</td>
<td>220</td>
</tr>
<tr>
<td>clutter-table-layout-get-alignment</td>
<td>222</td>
</tr>
<tr>
<td>clutter-table-layout-get-expand</td>
<td>223</td>
</tr>
<tr>
<td>clutter-table-layout-get-fill</td>
<td>223</td>
</tr>
<tr>
<td>clutter-table-layout-get-row-count</td>
<td>222</td>
</tr>
<tr>
<td>clutter-table-layout-get-span</td>
<td>224</td>
</tr>
<tr>
<td>clutter-table-layout-get新</td>
<td>221</td>
</tr>
<tr>
<td>clutter-table-layout-get-pack</td>
<td>222</td>
</tr>
<tr>
<td>clutter-table-layout-get-new</td>
<td>222</td>
</tr>
<tr>
<td>clutter-table-layout-get-set-alignment</td>
<td>222</td>
</tr>
<tr>
<td>clutter-table-layout-get-set-expand</td>
<td>223</td>
</tr>
<tr>
<td>clutter-table-layout-get-set-fill</td>
<td>223</td>
</tr>
<tr>
<td>clutter-table-layout-set-span</td>
<td>224</td>
</tr>
<tr>
<td>clutter-table-stage-new</td>
<td>203</td>
</tr>
<tr>
<td>clutter-table-stage-get-title</td>
<td>208</td>
</tr>
<tr>
<td>clutter-table-stage-get-use-alpha</td>
<td>207</td>
</tr>
<tr>
<td>clutter-table-stage-get-user-resizable</td>
<td>210</td>
</tr>
<tr>
<td>clutter-table-stage-hide-cursor</td>
<td>205</td>
</tr>
<tr>
<td>clutter-table-stage-manager-get-default</td>
<td>203</td>
</tr>
<tr>
<td>clutter-table-stage-manager-get-default-list-stages</td>
<td>203</td>
</tr>
<tr>
<td>clutter-table-stage-manager-get-default-peek-stages</td>
<td>203</td>
</tr>
<tr>
<td>clutter-table-snap-constraint-set-edges</td>
<td>202</td>
</tr>
<tr>
<td>clutter-table-snap-constraint-set-offset</td>
<td>202</td>
</tr>
<tr>
<td>clutter-table-snap-constraint-set-source</td>
<td>201</td>
</tr>
<tr>
<td>clutter-text-new</td>
<td>203</td>
</tr>
<tr>
<td>clutter-text-get-active</td>
<td>242</td>
</tr>
<tr>
<td>clutter-text-get-buffer-new</td>
<td>225</td>
</tr>
<tr>
<td>clutter-text-get-buffer-new-with-text</td>
<td>225</td>
</tr>
<tr>
<td>clutter-text-get-attributes</td>
<td>231</td>
</tr>
<tr>
<td>clutter-text-get-buffer</td>
<td>229</td>
</tr>
<tr>
<td>clutter-text-get-color</td>
<td>231</td>
</tr>
<tr>
<td>clutter-text-get-cursor-color</td>
<td>240</td>
</tr>
<tr>
<td>clutter-text-get-editor</td>
<td>238</td>
</tr>
<tr>
<td>clutter-text-get-safe</td>
<td>242</td>
</tr>
<tr>
<td>clutter-text-get-size</td>
<td>232</td>
</tr>
<tr>
<td>clutter-text-get-selection</td>
<td>239</td>
</tr>
<tr>
<td>clutter-text-get-selection-bound</td>
<td>237</td>
</tr>
<tr>
<td>clutter-text-get-single-line-mode</td>
<td>234</td>
</tr>
<tr>
<td>clutter-text-get-single-line-wrap</td>
<td>234</td>
</tr>
<tr>
<td>clutter-text-get-single-line-wrap-modal</td>
<td>235</td>
</tr>
<tr>
<td>clutter-text-get-max-length</td>
<td>235</td>
</tr>
<tr>
<td>clutter-text-get-max-length-modal</td>
<td>235</td>
</tr>
<tr>
<td>clutter-text-get-password-char</td>
<td>233</td>
</tr>
<tr>
<td>clutter-text-get-selectable</td>
<td>236</td>
</tr>
<tr>
<td>clutter-text-get-selection-bound</td>
<td>236</td>
</tr>
<tr>
<td>clutter-text-get-selection-bound-bound</td>
<td>237</td>
</tr>
<tr>
<td>clutter-text-get-selection-color</td>
<td>240</td>
</tr>
<tr>
<td>clutter-text-get-single-line-mode</td>
<td>237</td>
</tr>
<tr>
<td>clutter-text-get-text</td>
<td>230</td>
</tr>
<tr>
<td>clutter-text-get-use-markup</td>
<td>238</td>
</tr>
<tr>
<td>clutter-text-insert-text</td>
<td>238</td>
</tr>
<tr>
<td>clutter-text-insert-unichar</td>
<td>239</td>
</tr>
<tr>
<td>clutter-text-new</td>
<td>228</td>
</tr>
<tr>
<td>clutter-text-new-full</td>
<td>228</td>
</tr>
<tr>
<td>clutter-text-new-with-buffer</td>
<td>229</td>
</tr>
<tr>
<td>clutter-text-new-with-text</td>
<td>228</td>
</tr>
<tr>
<td>Function Name</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>clutter-text-node-new</td>
<td>178</td>
</tr>
<tr>
<td>clutter-text-position-to-coords</td>
<td>242</td>
</tr>
<tr>
<td>clutter-text-set-activatable</td>
<td>230</td>
</tr>
<tr>
<td>clutter-text-set-attributes</td>
<td>231</td>
</tr>
<tr>
<td>clutter-text-set-buffer</td>
<td>229</td>
</tr>
<tr>
<td>clutter-text-set-color</td>
<td>231</td>
</tr>
<tr>
<td>clutter-text-set-cursor-color</td>
<td>240</td>
</tr>
<tr>
<td>clutter-text-set-cursor-position</td>
<td>241</td>
</tr>
<tr>
<td>clutter-text-set-cursor-size</td>
<td>241</td>
</tr>
<tr>
<td>clutter-text-set-cursor-visible</td>
<td>241</td>
</tr>
<tr>
<td>clutter-text-set-editable</td>
<td>238</td>
</tr>
<tr>
<td>clutter-text-set-ellipsize</td>
<td>232</td>
</tr>
<tr>
<td>clutter-text-set-font-description</td>
<td>233</td>
</tr>
<tr>
<td>clutter-text-set-font-name</td>
<td>232</td>
</tr>
<tr>
<td>clutter-text-set-justify</td>
<td>233</td>
</tr>
<tr>
<td>clutter-text-set-line-alignment</td>
<td>234</td>
</tr>
<tr>
<td>clutter-text-set-line-wrap</td>
<td>234</td>
</tr>
<tr>
<td>clutter-text-set-line-wrap-mode</td>
<td>235</td>
</tr>
<tr>
<td>clutter-text-set-markup</td>
<td>229</td>
</tr>
<tr>
<td>clutter-text-set-max-length</td>
<td>235</td>
</tr>
<tr>
<td>clutter-text-set-password-char</td>
<td>233</td>
</tr>
<tr>
<td>clutter-text-set-preedit-string</td>
<td>243</td>
</tr>
<tr>
<td>clutter-text-set-selectable</td>
<td>236</td>
</tr>
<tr>
<td>clutter-text-set-selection</td>
<td>236</td>
</tr>
<tr>
<td>clutter-text-set-selection-bound</td>
<td>236</td>
</tr>
<tr>
<td>clutter-text-set-selection-color</td>
<td>240</td>
</tr>
<tr>
<td>clutter-text-set-single-line-mode</td>
<td>237</td>
</tr>
<tr>
<td>clutter-text-set-text</td>
<td>229</td>
</tr>
<tr>
<td>clutter-text-set-use-markup</td>
<td>237</td>
</tr>
<tr>
<td>clutter-texture-get-base-size</td>
<td>245</td>
</tr>
<tr>
<td>clutter-texture-get-load-async</td>
<td>246</td>
</tr>
<tr>
<td>clutter-texture-get-load-data-async</td>
<td>247</td>
</tr>
<tr>
<td>clutter-texture-get-max-tile-waste</td>
<td>245</td>
</tr>
<tr>
<td>clutter-texture-get-pick-with-alpha</td>
<td>247</td>
</tr>
<tr>
<td>clutter-texture-get-repeat</td>
<td>246</td>
</tr>
<tr>
<td>clutter-texture-get-sync-size</td>
<td>245</td>
</tr>
<tr>
<td>clutter-texture-new</td>
<td>244</td>
</tr>
<tr>
<td>clutter-texture-new-from-file</td>
<td>244</td>
</tr>
<tr>
<td>clutter-texture-set-filter-quality</td>
<td>245</td>
</tr>
<tr>
<td>clutter-texture-set-from-file</td>
<td>244</td>
</tr>
<tr>
<td>clutter-texture-set-load-async</td>
<td>246</td>
</tr>
<tr>
<td>clutter-texture-set-load-data-async</td>
<td>247</td>
</tr>
<tr>
<td>clutter-texture-set-pick-with-alpha</td>
<td>247</td>
</tr>
<tr>
<td>clutter-texture-set-repeat</td>
<td>246</td>
</tr>
<tr>
<td>clutter-texture-set-sync-size</td>
<td>246</td>
</tr>
<tr>
<td>clutter-timeline-add-marker-at-time</td>
<td>255</td>
</tr>
<tr>
<td>clutter-timeline-advance</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-advance-to-marker</td>
<td>256</td>
</tr>
<tr>
<td>clutter-timeline-get-auto-reverse</td>
<td>252</td>
</tr>
<tr>
<td>clutter-timeline-get-counter-repeat</td>
<td>253</td>
</tr>
<tr>
<td>clutter-timeline-get-delay</td>
<td>251</td>
</tr>
<tr>
<td>clutter-timeline-get-direction</td>
<td>251</td>
</tr>
<tr>
<td>clutter-timeline-get-duration</td>
<td>250</td>
</tr>
<tr>
<td>clutter-timeline-get-duration-hint</td>
<td>253</td>
</tr>
<tr>
<td>clutter-timeline-get-elapsed-time</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-get-progress</td>
<td>255</td>
</tr>
<tr>
<td>clutter-timeline-get-progress-mode</td>
<td>253</td>
</tr>
<tr>
<td>clutter-timeline-get-repeat-count</td>
<td>251</td>
</tr>
<tr>
<td>clutter-timeline-has-marker</td>
<td>256</td>
</tr>
<tr>
<td>clutter-timeline-is-playing</td>
<td>255</td>
</tr>
<tr>
<td>clutter-timeline-new</td>
<td>250</td>
</tr>
<tr>
<td>clutter-timeline-pause</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-remove-marker</td>
<td>256</td>
</tr>
<tr>
<td>clutter-timeline-rewind</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-set-auto-reverse</td>
<td>252</td>
</tr>
<tr>
<td>clutter-timeline-set-delay</td>
<td>251</td>
</tr>
<tr>
<td>clutter-timeline-set-direction</td>
<td>251</td>
</tr>
<tr>
<td>clutter-timeline-set-duration</td>
<td>250</td>
</tr>
<tr>
<td>clutter-timeline-set-progress-mode</td>
<td>253</td>
</tr>
<tr>
<td>clutter-timeline-set-repeat-count</td>
<td>250</td>
</tr>
<tr>
<td>clutter-timeline-set-repeat-count</td>
<td>250</td>
</tr>
<tr>
<td>clutter-timeline-set-skip</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-start</td>
<td>254</td>
</tr>
<tr>
<td>clutter-timeline-stop</td>
<td>254</td>
</tr>
<tr>
<td>clutter-transition-get-animatable</td>
<td>258</td>
</tr>
<tr>
<td>clutter-transition-get-interval</td>
<td>257</td>
</tr>
<tr>
<td>clutter-transition-get-animatable</td>
<td>257</td>
</tr>
<tr>
<td>clutter-transition-get-interval</td>
<td>257</td>
</tr>
<tr>
<td>clutter-transition-get-animatable</td>
<td>257</td>
</tr>
<tr>
<td>clutter-transition-get-interval</td>
<td>257</td>
</tr>
<tr>
<td>clutter-transition-get-interval</td>
<td>257</td>
</tr>
<tr>
<td>clutter-ungrab-keyboard</td>
<td>165</td>
</tr>
<tr>
<td>clutter-ungrab-pointer</td>
<td>165</td>
</tr>
<tr>
<td>clutter-ungrab-pointer</td>
<td>165</td>
</tr>
<tr>
<td>clutter-unicode-to-keysym</td>
<td>137</td>
</tr>
<tr>
<td>clutter-units-from-cm</td>
<td>259</td>
</tr>
<tr>
<td>clutter-units-from-em</td>
<td>259</td>
</tr>
<tr>
<td>clutter-units-from-em-for-font</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-from-mm</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-from-pixels</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-from-pt</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-from-string</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-get-unit-type</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-get-unit-value</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-get-unit-value</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-get-unit-value</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-get-unit-value</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-to-pixels</td>
<td>260</td>
</tr>
<tr>
<td>clutter-units-to-string</td>
<td>261</td>
</tr>
<tr>
<td>clutter-units-to-string</td>
<td>261</td>
</tr>
<tr>
<td>clutter-value-get-color</td>
<td>113</td>
</tr>
<tr>
<td>clutter-value-get-color</td>
<td>112</td>
</tr>
<tr>
<td>clutter-value-get-color</td>
<td>112</td>
</tr>
<tr>
<td>clutter-vertex-equal</td>
<td>58</td>
</tr>
<tr>
<td>clutter-vertex-init</td>
<td>58</td>
</tr>
<tr>
<td>clutter-vertex-new</td>
<td>58</td>
</tr>
<tr>
<td>clutter-vertex-new</td>
<td>58</td>
</tr>
<tr>
<td>clutter-vertex-new</td>
<td>73</td>
</tr>
<tr>
<td>compute</td>
<td>154</td>
</tr>
<tr>
<td>compute-value</td>
<td>79</td>
</tr>
<tr>
<td>constraint-new</td>
<td>179</td>
</tr>
<tr>
<td>contains</td>
<td>43</td>
</tr>
<tr>
<td>continue-paint</td>
<td>16</td>
</tr>
<tr>
<td>coords-to-position</td>
<td>242</td>
</tr>
<tr>
<td>create-child-meta</td>
<td>117</td>
</tr>
<tr>
<td>create-pango-context</td>
<td>48</td>
</tr>
<tr>
<td>create-pango-layout</td>
<td>48</td>
</tr>
</tbody>
</table>

**D**

- delete-chars                                      | 239  |
- delete-selection                                  | 239  |
- delete-text                                       | 227, 239 |
- destroy                                           | 16   |
- destroy-all-children                              | 41   |
- detach-animation                                  | 75   |
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ensure-current</td>
<td>205</td>
</tr>
<tr>
<td>ensure-objects</td>
<td>194</td>
</tr>
<tr>
<td>ensure-redraw</td>
<td>206</td>
</tr>
<tr>
<td>ensure-viewport</td>
<td>206</td>
</tr>
<tr>
<td>event</td>
<td>17, 206</td>
</tr>
<tr>
<td>find-action</td>
<td>92</td>
</tr>
<tr>
<td>find-property</td>
<td>68</td>
</tr>
<tr>
<td>get-accept-focus</td>
<td>208</td>
</tr>
<tr>
<td>get-accessible</td>
<td>49</td>
</tr>
<tr>
<td>get-action</td>
<td>51</td>
</tr>
<tr>
<td>get-actions</td>
<td>50</td>
</tr>
<tr>
<td>get-activatable</td>
<td>230</td>
</tr>
<tr>
<td>get-actor</td>
<td>4, 104</td>
</tr>
<tr>
<td>get-actor-at-pos</td>
<td>205</td>
</tr>
<tr>
<td>get-alignment</td>
<td>85, 98, 222</td>
</tr>
<tr>
<td>get-allocation-box</td>
<td>20</td>
</tr>
<tr>
<td>get-alpha</td>
<td>67</td>
</tr>
<tr>
<td>get-anchor-point</td>
<td>34</td>
</tr>
<tr>
<td>get-angle</td>
<td>176</td>
</tr>
<tr>
<td>get-animated</td>
<td>258</td>
</tr>
<tr>
<td>get-animation</td>
<td>75</td>
</tr>
<tr>
<td>get-animator</td>
<td>218</td>
</tr>
<tr>
<td>get-attributes</td>
<td>231</td>
</tr>
<tr>
<td>get-audio-volume</td>
<td>169</td>
</tr>
<tr>
<td>get-auto-reverse</td>
<td>252</td>
</tr>
<tr>
<td>get-background-color</td>
<td>27</td>
</tr>
<tr>
<td>get-base-size</td>
<td>245</td>
</tr>
<tr>
<td>get-buffer</td>
<td>229</td>
</tr>
<tr>
<td>get-buffer-fill</td>
<td>169</td>
</tr>
<tr>
<td>get-button</td>
<td>106, 136</td>
</tr>
<tr>
<td>get-bytes</td>
<td>226</td>
</tr>
<tr>
<td>get-can-seek</td>
<td>169</td>
</tr>
<tr>
<td>get-chars</td>
<td>239</td>
</tr>
<tr>
<td>get-child-at-index</td>
<td>42</td>
</tr>
<tr>
<td>get-child-meta</td>
<td>117</td>
</tr>
<tr>
<td>get-children</td>
<td>43</td>
</tr>
<tr>
<td>get-click-count</td>
<td>136</td>
</tr>
<tr>
<td>get-cancel</td>
<td>38</td>
</tr>
<tr>
<td>get-color</td>
<td>231</td>
</tr>
<tr>
<td>get-constraint</td>
<td>52</td>
</tr>
<tr>
<td>get-constraints</td>
<td>52</td>
</tr>
<tr>
<td>get-container</td>
<td>104</td>
</tr>
<tr>
<td>get-content</td>
<td>36</td>
</tr>
<tr>
<td>get-content-box</td>
<td>37</td>
</tr>
<tr>
<td>get-content-gravity</td>
<td>37</td>
</tr>
<tr>
<td>get-coords</td>
<td>106, 132</td>
</tr>
<tr>
<td>get-current-repeat</td>
<td>253</td>
</tr>
<tr>
<td>get-cursor-color</td>
<td>240</td>
</tr>
<tr>
<td>get-cursor-position</td>
<td>241</td>
</tr>
<tr>
<td>get-cursor-size</td>
<td>242</td>
</tr>
<tr>
<td>get-cursor-visible</td>
<td>241</td>
</tr>
<tr>
<td>get-delay</td>
<td>251</td>
</tr>
<tr>
<td>get-delta</td>
<td>255</td>
</tr>
<tr>
<td>get-depth</td>
<td>31</td>
</tr>
<tr>
<td>get-description</td>
<td>186</td>
</tr>
<tr>
<td>get-device</td>
<td>123, 139</td>
</tr>
<tr>
<td>get-device-id</td>
<td>139</td>
</tr>
<tr>
<td>get-device-type</td>
<td>140</td>
</tr>
<tr>
<td>get-direction</td>
<td>251</td>
</tr>
<tr>
<td>get-drag-axis</td>
<td>126</td>
</tr>
<tr>
<td>get-drag-handle</td>
<td>126</td>
</tr>
<tr>
<td>get-duration</td>
<td>72, 80, 169, 216, 250</td>
</tr>
<tr>
<td>get-duration-hint</td>
<td>253</td>
</tr>
<tr>
<td>get-easing-delay</td>
<td>45</td>
</tr>
<tr>
<td>get-easing-duration</td>
<td>44</td>
</tr>
<tr>
<td>get-easing-mode</td>
<td>45, 99</td>
</tr>
<tr>
<td>get-edges</td>
<td>202</td>
</tr>
<tr>
<td>get-editable</td>
<td>238</td>
</tr>
<tr>
<td>get-effect</td>
<td>54</td>
</tr>
<tr>
<td>get-effects</td>
<td>54</td>
</tr>
<tr>
<td>get-elapsed-time</td>
<td>254</td>
</tr>
<tr>
<td>get-ellipsis</td>
<td>232</td>
</tr>
<tr>
<td>get-enabled</td>
<td>4</td>
</tr>
<tr>
<td>get-event-sequence</td>
<td>135</td>
</tr>
<tr>
<td>get-expand</td>
<td>98, 223</td>
</tr>
<tr>
<td>get-factor</td>
<td>64</td>
</tr>
<tr>
<td>get-fill</td>
<td>99, 223</td>
</tr>
<tr>
<td>get-final-value</td>
<td>153</td>
</tr>
<tr>
<td>get-first-child</td>
<td>41</td>
</tr>
<tr>
<td>get-flags</td>
<td>14, 134</td>
</tr>
<tr>
<td>get-font-name</td>
<td>232</td>
</tr>
<tr>
<td>get-fullscreen</td>
<td>205</td>
</tr>
<tr>
<td>get-height</td>
<td>29</td>
</tr>
<tr>
<td>get-homogeneous</td>
<td>97, 144</td>
</tr>
<tr>
<td>get-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>get-interval</td>
<td>75, 257</td>
</tr>
<tr>
<td>get-justify</td>
<td>233</td>
</tr>
<tr>
<td>get-key-code</td>
<td>137</td>
</tr>
<tr>
<td>get-key-focus</td>
<td>206</td>
</tr>
<tr>
<td>get-key-symbol</td>
<td>136</td>
</tr>
<tr>
<td>get-key-unicode</td>
<td>137</td>
</tr>
<tr>
<td>get-keys</td>
<td>78, 216</td>
</tr>
<tr>
<td>get-last-child</td>
<td>42</td>
</tr>
<tr>
<td>get-layout</td>
<td>234</td>
</tr>
<tr>
<td>get-layout-manager</td>
<td>26</td>
</tr>
<tr>
<td>get-layout-offsets</td>
<td>243</td>
</tr>
<tr>
<td>get-length</td>
<td>187, 226</td>
</tr>
<tr>
<td>get-line-alignment</td>
<td>234</td>
</tr>
<tr>
<td>get-line-wrap</td>
<td>234</td>
</tr>
<tr>
<td>get-line-wrap-mode</td>
<td>235</td>
</tr>
<tr>
<td>get-load-async</td>
<td>246</td>
</tr>
<tr>
<td>get-load-data-async</td>
<td>247</td>
</tr>
<tr>
<td>get-loop</td>
<td>72</td>
</tr>
<tr>
<td>get-manager</td>
<td>161</td>
</tr>
<tr>
<td>get-margin</td>
<td>24</td>
</tr>
<tr>
<td>get-margin-bottom</td>
<td>25</td>
</tr>
<tr>
<td>get-margin-left</td>
<td>26</td>
</tr>
<tr>
<td>get-margin-right</td>
<td>25</td>
</tr>
<tr>
<td>get-margin-top</td>
<td>24</td>
</tr>
</tbody>
</table>
Function Index 279

get-max-length ........................................ 227, 235
get-max-tile-waste ...................................... 245
get-minimum-size ...................................... 207
get-mode .................................................. 67, 71
get-n-children .......................................... 43
get-n-nodes ............................................. 185
get-n-tiles .............................................. 120
get-name ................................................. 3, 14
get-next-sibling ....................................... 42
get-no-clear-hint ...................................... 208
get-node .................................................. 185
get-nodes ................................................. 185
get-object ............................................... 71, 193
get-offset ............................................... 89, 180, 202
get-opacity ............................................. 38
get-pack-start ......................................... 96
get-paint-box .......................................... 36
get-paint-opacity ..................................... 35
get-paint-visibility ................................... 36
get-pango-context ..................................... 47
get-parent ............................................. 43
get-password-char .................................... 233
get-path .................................................. 179
get-period ................................................ 175
get-perspective ....................................... 209
get-pick-with-alpha .................................... 247
get-playing .............................................. 167
get-position ............................................ 28, 187
get-preferred-height .................................. 22
get-preferred-size .................................... 21, 119
get-preferred-width .................................. 21
get-previous-sibling .................................. 42
get-progress ........................................... 168, 255
get-progress-mode .................................... 253
get-radius ................................................ 176
get-reactive .......................................... 47
get-related ............................................ 138
get-repeat .............................................. 246
get-repeat-count ...................................... 251
get-request-mode ..................................... 22
get-resolution .......................................... 82
get-rotation ........................................... 33
get-row-count .......................................... 222
get-row-height ......................................... 145
get-row-spacing ....................................... 144
get-scale .................................................. 32
get-scale-center ....................................... 32
get-scale-gravity ...................................... 32
get-scroll-delta ....................................... 138
get-selectable ......................................... 236
get-selection .......................................... 236
get-selection-bound .................................. 237
get-selection-color ................................... 240
get-single-line-mode .................................. 237
get-size ................................................... 27
get-source ............................................. 63, 88, 108, 134, 201
get-source-device ..................................... 139
get-spacing ............................................. 96
get-span ................................................... 224
get-stage ................................................ 44, 134
get-state ............................................... 106, 133, 214
get-states .............................................. 195, 216
get-subtitle-uri ....................................... 168
get-sync-size .......................................... 245
get-text .................................................. 226, 230
get-text-direction ...................................... 49
get-time .................................................. 133
get-timeline ............................................. 66, 72, 79, 217
get-tint .................................................. 114
get-title .................................................. 209
get-transformed-size .................................. 3.5
get-transition .......................................... .45
get-type-from-name .................................... 195
get-uri .................................................. 167
get-use-alpha .......................................... 207
get-use-markup ......................................... 238
get-user-resizable ..................................... 210
get-value-type ......................................... 152
get-vertical ............................................ 97
get-width ............................................... 28
get-x ...................................................... 29
get-x-align ............................................. 23
get-y ...................................................... 30
get-y-align ............................................. 24
grab-key-focus ......................................... 47

H

has-actions .............................................. 50
has-allocation ......................................... 23
has-clip ................................................ 38
has-constraints ....................................... 52
has-effects ............................................. 53
has-key-focus ......................................... 47
has-marker ............................................ 256
has-overlaps .......................................... 18
has-pointer ............................................ 47
has-property .......................................... 74
hide ....................................................... 15
hide-cursor ............................................ 205

I

insert-child-above ..................................... 39
insert-child-at-index .................................. 40
insert-child-below .................................... 40
insert-node ............................................ 185
insert-text ............................................. 227, 238
insert-unichar ........................................ 239
invalidate ............................................ 101, 119, 121
is-in-clone-paint .................................... 39
is-playing .............................................. 255
is-rotated ............................................. 33
is-scaled .............................................. 32
### Function Index

<table>
<thead>
<tr>
<th>L</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>list-devices</td>
<td>123</td>
</tr>
<tr>
<td>list-objects</td>
<td>194</td>
</tr>
<tr>
<td>list-stages</td>
<td>203</td>
</tr>
<tr>
<td>load-from-data</td>
<td>192</td>
</tr>
<tr>
<td>load-from-file</td>
<td>193</td>
</tr>
<tr>
<td>load-from-resource</td>
<td>193</td>
</tr>
<tr>
<td>lookup-filename</td>
<td>193</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>17</td>
</tr>
<tr>
<td>move-anchor-point</td>
<td>34</td>
</tr>
<tr>
<td>move-by</td>
<td>30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>pack</td>
<td>97, 222</td>
</tr>
<tr>
<td>paint</td>
<td>15</td>
</tr>
<tr>
<td>pause</td>
<td>254</td>
</tr>
<tr>
<td>peek-devices</td>
<td>123</td>
</tr>
<tr>
<td>peek-stages</td>
<td>203</td>
</tr>
<tr>
<td>position-to-coords</td>
<td>242</td>
</tr>
<tr>
<td>put</td>
<td>135</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Q</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>queue-redraw</td>
<td>16</td>
</tr>
<tr>
<td>queue-relayout</td>
<td>16</td>
</tr>
<tr>
<td>queue-repaint</td>
<td>131</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>realize</td>
<td>15</td>
</tr>
<tr>
<td>release</td>
<td>107</td>
</tr>
<tr>
<td>remove-action</td>
<td>50, 92</td>
</tr>
<tr>
<td>remove-action-by-name</td>
<td>50</td>
</tr>
<tr>
<td>remove-all-children</td>
<td>41</td>
</tr>
<tr>
<td>remove-child</td>
<td>40</td>
</tr>
<tr>
<td>remove-clip</td>
<td>38</td>
</tr>
<tr>
<td>remove-constraint</td>
<td>51</td>
</tr>
<tr>
<td>remove-effect</td>
<td>53</td>
</tr>
<tr>
<td>remove-effect-by-name</td>
<td>53</td>
</tr>
<tr>
<td>remove-key</td>
<td>78, 217</td>
</tr>
<tr>
<td>remove-marker</td>
<td>256</td>
</tr>
<tr>
<td>remove-node</td>
<td>185</td>
</tr>
<tr>
<td>remove-transition</td>
<td>46</td>
</tr>
<tr>
<td>replace-child</td>
<td>40</td>
</tr>
<tr>
<td>replace-node</td>
<td>186</td>
</tr>
<tr>
<td>restore-easing-state</td>
<td>44</td>
</tr>
<tr>
<td>rewind</td>
<td>254</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>S</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>save-easing-state</td>
<td>44</td>
</tr>
<tr>
<td>set-accept-focus</td>
<td>208</td>
</tr>
<tr>
<td>set-activatable</td>
<td>230</td>
</tr>
<tr>
<td>set-alignment</td>
<td>85, 98, 222</td>
</tr>
<tr>
<td>set-allocation</td>
<td></td>
</tr>
<tr>
<td>set-anchor-point</td>
<td>19</td>
</tr>
<tr>
<td>set-angle</td>
<td>175</td>
</tr>
<tr>
<td>set-animatable</td>
<td>257</td>
</tr>
<tr>
<td>set-animator</td>
<td>217</td>
</tr>
<tr>
<td>set-attributes</td>
<td>231</td>
</tr>
<tr>
<td>set-audio-volume</td>
<td>168</td>
</tr>
<tr>
<td>set-auto-reverse</td>
<td>252</td>
</tr>
<tr>
<td>set-background-color</td>
<td>26</td>
</tr>
<tr>
<td>set-buffer</td>
<td>229</td>
</tr>
<tr>
<td>set-button</td>
<td>135</td>
</tr>
<tr>
<td>set-child-at-index</td>
<td>43</td>
</tr>
<tr>
<td>set-clip</td>
<td>37</td>
</tr>
<tr>
<td>set-closure</td>
<td>67</td>
</tr>
<tr>
<td>set-color</td>
<td>231</td>
</tr>
<tr>
<td>set-content</td>
<td>36</td>
</tr>
<tr>
<td>set-content-gravity</td>
<td>37</td>
</tr>
<tr>
<td>set-coords</td>
<td>132</td>
</tr>
<tr>
<td>set-cursor-color</td>
<td>240</td>
</tr>
<tr>
<td>set-cursor-position</td>
<td>241</td>
</tr>
<tr>
<td>set-cursor-size</td>
<td>241</td>
</tr>
<tr>
<td>set-cursor-visible</td>
<td>241</td>
</tr>
<tr>
<td>set-delay</td>
<td>251</td>
</tr>
<tr>
<td>set-depth</td>
<td>31</td>
</tr>
<tr>
<td>set-description</td>
<td>186</td>
</tr>
<tr>
<td>set-device</td>
<td>139</td>
</tr>
<tr>
<td>set-direction</td>
<td>251</td>
</tr>
<tr>
<td>set-drag-axis</td>
<td>126</td>
</tr>
<tr>
<td>set-drag-handle</td>
<td>125</td>
</tr>
<tr>
<td>set-duration</td>
<td>71, 80, 215, 250</td>
</tr>
<tr>
<td>set-easing-delay</td>
<td>45</td>
</tr>
<tr>
<td>set-easing-duration</td>
<td>44</td>
</tr>
<tr>
<td>set-easing-mode</td>
<td>45, 99</td>
</tr>
<tr>
<td>set-edges</td>
<td>202</td>
</tr>
<tr>
<td>set-editable</td>
<td>238</td>
</tr>
<tr>
<td>set-ellipse</td>
<td>232</td>
</tr>
<tr>
<td>set-enabled</td>
<td>3</td>
</tr>
<tr>
<td>set-expand</td>
<td>98, 223</td>
</tr>
<tr>
<td>set-factor</td>
<td>64</td>
</tr>
<tr>
<td>set-filename</td>
<td>169</td>
</tr>
<tr>
<td>set-fill</td>
<td>99, 223</td>
</tr>
<tr>
<td>set-filter-quality</td>
<td>245</td>
</tr>
<tr>
<td>set-final-state</td>
<td>68</td>
</tr>
<tr>
<td>set-final-value</td>
<td>153</td>
</tr>
<tr>
<td>set-flags</td>
<td>13, 134</td>
</tr>
<tr>
<td>set-font-description</td>
<td>233</td>
</tr>
<tr>
<td>set-font-name</td>
<td>232</td>
</tr>
<tr>
<td>set-font-options</td>
<td>82</td>
</tr>
<tr>
<td>set-from-file</td>
<td>244</td>
</tr>
<tr>
<td>set-fullscreen</td>
<td>204</td>
</tr>
<tr>
<td>set-height</td>
<td>29</td>
</tr>
<tr>
<td>set-homogeneous</td>
<td>97, 143</td>
</tr>
<tr>
<td>set-initial-value</td>
<td>153</td>
</tr>
<tr>
<td>set-interval</td>
<td>257</td>
</tr>
<tr>
<td>set-justify</td>
<td>233</td>
</tr>
<tr>
<td>set-key</td>
<td>77, 215</td>
</tr>
<tr>
<td>set-key-code</td>
<td>136</td>
</tr>
<tr>
<td>Function</td>
<td>Page</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------</td>
</tr>
<tr>
<td>set-key-focus</td>
<td>206</td>
</tr>
<tr>
<td>set-key-symbol</td>
<td>136</td>
</tr>
<tr>
<td>set-key-unicode</td>
<td>137</td>
</tr>
<tr>
<td>set-layout-manager</td>
<td>26</td>
</tr>
<tr>
<td>set-line-alignment</td>
<td>234</td>
</tr>
<tr>
<td>set-line-wrap</td>
<td>234</td>
</tr>
<tr>
<td>set-line-wrap-mode</td>
<td>235</td>
</tr>
<tr>
<td>set-load-async</td>
<td>246</td>
</tr>
<tr>
<td>set-load-data-async</td>
<td>247</td>
</tr>
<tr>
<td>set-loop</td>
<td>72</td>
</tr>
<tr>
<td>set-margin</td>
<td>24</td>
</tr>
<tr>
<td>set-margin-bottom</td>
<td>25</td>
</tr>
<tr>
<td>set-margin-left</td>
<td>25</td>
</tr>
<tr>
<td>set-margin-right</td>
<td>25</td>
</tr>
<tr>
<td>set-margin-top</td>
<td>24</td>
</tr>
<tr>
<td>set-markup</td>
<td>229</td>
</tr>
<tr>
<td>set-max-length</td>
<td>226</td>
</tr>
<tr>
<td>set-minimum-size</td>
<td>207</td>
</tr>
<tr>
<td>set-mode</td>
<td>66, 71</td>
</tr>
<tr>
<td>set-n-tiles</td>
<td>120</td>
</tr>
<tr>
<td>set-name</td>
<td>3, 14, 177</td>
</tr>
<tr>
<td>set-no-clear-hint</td>
<td>208</td>
</tr>
<tr>
<td>set-object</td>
<td>70</td>
</tr>
<tr>
<td>set-offset</td>
<td>88, 180, 202</td>
</tr>
<tr>
<td>set-opacity</td>
<td>38</td>
</tr>
<tr>
<td>set-orientation</td>
<td>144</td>
</tr>
<tr>
<td>set-pack-start</td>
<td>96</td>
</tr>
<tr>
<td>set-password-char</td>
<td>233</td>
</tr>
<tr>
<td>set-path</td>
<td>179</td>
</tr>
<tr>
<td>set-period</td>
<td>175</td>
</tr>
<tr>
<td>set-perspective</td>
<td>209</td>
</tr>
<tr>
<td>set-pick-with-alpha</td>
<td>247</td>
</tr>
<tr>
<td>set-playing</td>
<td>167</td>
</tr>
<tr>
<td>set-position</td>
<td>27</td>
</tr>
<tr>
<td>set-preedit-string</td>
<td>243</td>
</tr>
<tr>
<td>set-progress</td>
<td>168</td>
</tr>
<tr>
<td>set-progress-mode</td>
<td>253</td>
</tr>
<tr>
<td>set-radius</td>
<td>176</td>
</tr>
<tr>
<td>set-reactive</td>
<td>47</td>
</tr>
<tr>
<td>set-related</td>
<td>438</td>
</tr>
<tr>
<td>set-repeat</td>
<td>246</td>
</tr>
<tr>
<td>set-repeat-count</td>
<td>250</td>
</tr>
<tr>
<td>set-request-mode</td>
<td>22</td>
</tr>
<tr>
<td>set-rotation</td>
<td>32</td>
</tr>
<tr>
<td>set-row-height</td>
<td>145</td>
</tr>
<tr>
<td>set-row-spacing</td>
<td>144</td>
</tr>
<tr>
<td>set-scale</td>
<td>31</td>
</tr>
<tr>
<td>set-scale-full</td>
<td>31</td>
</tr>
<tr>
<td>set-scroll-delta</td>
<td>138</td>
</tr>
<tr>
<td>set-scroll-direction</td>
<td>138</td>
</tr>
<tr>
<td>set-selectable</td>
<td>236</td>
</tr>
<tr>
<td>set-selection</td>
<td>236</td>
</tr>
<tr>
<td>set-selection-bound</td>
<td>236</td>
</tr>
<tr>
<td>set-selection-color</td>
<td>240</td>
</tr>
<tr>
<td>set-single-line-mode</td>
<td>237</td>
</tr>
<tr>
<td>set-size</td>
<td>27, 103</td>
</tr>
<tr>
<td>set-source</td>
<td>63, 88, 108, 133, 201</td>
</tr>
<tr>
<td>set-source-device</td>
<td>139</td>
</tr>
<tr>
<td>set-spacing</td>
<td>96</td>
</tr>
<tr>
<td>set-span</td>
<td>224</td>
</tr>
<tr>
<td>set-stage</td>
<td>134</td>
</tr>
<tr>
<td>set-state</td>
<td>133, 214</td>
</tr>
<tr>
<td>set-subtitle-uri</td>
<td>168</td>
</tr>
<tr>
<td>set-sync-size</td>
<td>246</td>
</tr>
<tr>
<td>set-text</td>
<td>225, 229</td>
</tr>
<tr>
<td>set-text-direction</td>
<td>48</td>
</tr>
<tr>
<td>set-time</td>
<td>133</td>
</tr>
<tr>
<td>set-timeline</td>
<td>66, 72, 79</td>
</tr>
<tr>
<td>set-tint</td>
<td>114</td>
</tr>
<tr>
<td>set-title</td>
<td>209</td>
</tr>
<tr>
<td>set-uri</td>
<td>167</td>
</tr>
<tr>
<td>set-use-alpha</td>
<td>207</td>
</tr>
<tr>
<td>set-use-markup</td>
<td>237</td>
</tr>
<tr>
<td>set-user-resizable</td>
<td>209</td>
</tr>
<tr>
<td>set-vertical</td>
<td>96</td>
</tr>
<tr>
<td>set-width</td>
<td>28</td>
</tr>
<tr>
<td>set-x</td>
<td>29</td>
</tr>
<tr>
<td>set-x-align</td>
<td>23</td>
</tr>
<tr>
<td>set-y</td>
<td>30</td>
</tr>
<tr>
<td>set-y-align</td>
<td>23</td>
</tr>
<tr>
<td>should-pick-paint</td>
<td>17</td>
</tr>
<tr>
<td>show</td>
<td>14</td>
</tr>
<tr>
<td>show-cursor</td>
<td>205</td>
</tr>
<tr>
<td>skip</td>
<td>254</td>
</tr>
<tr>
<td>start</td>
<td>78, 254</td>
</tr>
<tr>
<td>stop</td>
<td>254</td>
</tr>
</tbody>
</table>

**T**

to cairo-path                                   186
transform-stage-point                           34
type                                             132

**U**

unbind-property                                  74
unbind-action                                    92
unmap                                            17
unmerge-objects                                  194
unrealize                                        15
unset-objects                                    14
update                                           74
update-interval                                  74

**V**

validate                                         154

**W**

warp-to-state                                    214