Table of Contents

TextureNodeGroup(TextureNode)

base classes — bpy_struct, Node, NodeInternal, TextureNode

class bpy.types.**TextureNodeGroup(TextureNode)**

node_tree

TYPE:

NodeTree

classmethod is_registered_node_type()

True if a registered node type

RETURNS:

Result

RETURN TYPE:

boolean

classmethod input_template(index)

Input socket template

PARAMETERS:

index (*int in [0, inf]*) – Index

RETURNS:

result

RETURN TYPE:

NodeInternalSocketTemplate

classmethod output_template(index)

Output socket template

PARAMETERS:

index (*int in [0, inf]*) – Index

RETURNS:

result

RETURN TYPE:

NodeInternalSocketTemplate

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found

The class of default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- Node.type
- Node.location
- Node.location_absolute
- Node.width
- Node.height
- Node.dimensions
- Node.name
- Node.label
- Node.inputs
- Node.outputs
- Node.internal_links
- Node.parent
- Node.warning_propagation
- Node.use_custom_color
- Node.color
- Node.color_tag
- Node.select
- Node.show_options
- Node.show_preview
- Node.hide
- Node.mute
- Node.show_texture
- Node.bl_idname
- Node.bl_label
- Node.bl_description
- Node.bl_icon
- Node.bl_static_type
- Node.bl_width_default
- Node.bl_width_min
- Node.bl_width_max
- Node.bl_height_default
- Node.bl_height_min
- Node.bl_height_max

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- Node.poll_instance
- Node.update
- Node.insert_link
- Node.init
- Node.copy
- Node.free
- Node.draw_buttons
- Node.draw_buttons_ext
- Node.draw_label
- Node.debug_zone_body_lazy_function_graph
- Node.debug_zone_lazy_function_graph
- Node.poll
- Node.bl_rna_get_subclass
- Node.bl_rna_get_subclass_py
- NodeInternal.poll
- NodeInternal.poll_instance
- NodeInternal.update
- NodeInternal.draw_buttons
- NodeInternal.draw_buttons_ext
- NodeInternal.bl_rna_get_subclass
- NodeInternal.bl_rna_get_subclass_py

- bpy_struct.values
- Node.socket_value_update
- Node.is_registered_node_type
- Node.poll

- TextureNode.poll
- TextureNode.bl_rna_get_subclass
- TextureNode.bl_rna_get_subclass_py

Previous
TextureNodeDistance(TextureNode)

Copyright © Blender Authors
Made with Furo

Ne
TextureNodeHueSaturation(TextureNoc

Report issue on this page

UILayout(bpy_struct)

base class — bpy_struct

class bpy.types.**UILayout(bpy_struct)**

User interface layout in a panel or header

activate_init

When true, buttons defined in popups will be activated on first display (use so you can type into a field without having to click on it first)

TYPE:

boolean, default False

active

TYPE:

boolean, default False

active_default

When true, an operator button defined after this will be activated when pressing return(use with popup dialogs)

TYPE:

boolean, default False

alert

TYPE:

boolean, default False

alignment

TYPE:

enum in ['EXPAND', 'LEFT', 'CENTER', 'RIGHT'], default 'EXPAND'

direction

TYPE:

enum in ['HORIZONTAL', 'VERTICAL'], default 'HORIZONTAL', (readonly)

emboss

- NORMAL Regular – Draw standard button emboss style.
- NONE None – Draw only text and icons.
- PULLDOWN_MENU Pulldown Menu – Draw pulldown menu style.
- RADIAL_MENU Pie Menu – Draw radial menu style.
- NONE_OR_STATUS None or Status – Draw with no emboss unless the button has a coloring status like an animation state.

TYPE:

enum in ['NORMAL', 'NONE', 'PULLDOWN_MENU', 'RADIAL_MENU', 'NONE_OR_STATUS'], default 'NORMAL'

enabled

When false, this (sub)layout is grayed out

TYPE:

boolean, default False

operator_context

TYPE:

enum in Operator Context Items, default 'INVOKE_DEFAULT'

enum in Operator Context Items, default 'INVOKE_DEFAULT'

scale_x

Scale factor along the X for items in this (sub)layout

TYPE:

float in [0, inf], default 0.0

scale_y

Scale factor along the Y for items in this (sub)layout

TYPE:

float in [0, inf], default 0.0

ui_units_x

Fixed size along the X for items in this (sub)layout

TYPE:

float in [0, inf], default 0.0

ui_units_y

Fixed size along the Y for items in this (sub)layout

TYPE:

float in [0, inf], default 0.0

use_property_decorate

TYPE:

boolean, default False

use_property_split

TYPE:

boolean, default False

row(*, align=False, heading='', heading_ctxt='', translate=True)

Sub-layout. Items placed in this sublayout are placed next to each other in a row.

PARAMETERS:

- **align** (*boolean, (optional)*) – Align buttons to each other
- **heading** (*string, (optional, never None)*) – Heading, Label to insert into the layout for this sub-layout
- **heading_ctxt** (*string, (optional, never None)*) – Override automatic translation context of the given heading
- **translate** (*boolean, (optional)*) – Translate the given heading, when UI translation is enabled

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

column(*, align=False, heading='', heading_ctxt='', translate=True)

Sub-layout. Items placed in this sublayout are placed under each other in a column.

PARAMETERS:

- **align** (*boolean, (optional)*) – Align buttons to each other
- **heading** (*string, (optional, never None)*) – Heading, Label to insert into the layout for this sub-layout
- **heading_ctxt** (*string, (optional, never None)*) – Override automatic translation context of the given heading
- **translate** (*boolean, (optional)*) – Translate the given heading, when UI translation is enabled

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

panel(idname, *, default_closed=False)

Creates a collapsable panel. Whether it is open or closed is stored in the region using the given idname. This can only be used when the panel has the full width of the panel region available to it. So it can't be used in e.g. in a box or columns.

PARAMETERS:

- **idname** (*string, (never None)*) – Identifier of the panel
- **default_closed** (*boolean, (optional)*) – Open by Default, When true, the panel will be open the first time it is shown

RETURNS:

layout_header, Sub-layout to put items in, UILayout

layout_body, Sub-layout to put items in. Will be none if the panel is collapsed., UILayout

RETURN TYPE:

(UILayout, UILayout)

panel_prop(data, property)

Similar to *.panel(...)* but instead of storing whether it is open or closed in the region, it is stored in the provided boolean property. This should be used when multiple instances of the same panel can exist. For example one for every item in a collection property or list. This can only be used when the panel has the full width of the panel region available to it. So it can't be used in e.g. in a box or columns.

PARAMETERS:

- **data** (AnyType, (never None)) – Data from which to take the open-state property
- **property** (*string, (never None)*) – Identifier of the boolean property that determines whether the panel is open or closed

RETURNS:

layout_header, Sub-layout to put items in, UILayout

layout_body, Sub-layout to put items in. Will be none if the panel is collapsed., UILayout

RETURN TYPE:

(UILayout, UILayout)

column_flow(*, columns=0, align=False)

column_flow

PARAMETERS:

- **columns** (*int in [0, inf], (optional)*) – Number of columns, 0 is automatic
- **align** (*boolean, (optional)*) – Align buttons to each other

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

grid_flow(*, row_major=False, columns=0, even_columns=False, even_rows=False, align=False)

grid_flow

PARAMETERS:

- **row_major** (*boolean, (optional)*) – Fill row by row, instead of column by column
- **columns** (*int in [-inf, inf], (optional)*) – Number of columns, positive are absolute fixed numbers, 0 is automatic, negative are automati multiple numbers along major axis (e.g. -2 will only produce 2, 4, 6 etc. columns for row major layout, and 2, 4, 6 etc. rows for column

major layout).

- **even_columns** (*boolean, (optional)*) – All columns will have the same width
- **even_rows** (*boolean, (optional)*) – All rows will have the same height
- **align** (*boolean, (optional)*) – Align buttons to each other

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

box()

Sublayout (items placed in this sublayout are placed under each other in a column and are surrounded by a box)

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

split(*, factor=0.0, align=False)

split

PARAMETERS:

- **factor** (*float in [0, 1], (optional)*) – Percentage, Percentage of width to split at (leave unset for automatic calculation)
- **align** (*boolean, (optional)*) – Align buttons to each other

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

menu_pie()

Sublayout. Items placed in this sublayout are placed in a radial fashion around the menu center).

RETURNS:

Sub-layout to put items in

RETURN TYPE:

UILayout

classmethod icon(data)

Return the custom icon for this data, use it e.g. to get materials or texture icons.

PARAMETERS:

data (AnyType , (never None)) – Data from which to take the icon

RETURNS:

Icon identifier

RETURN TYPE:

int in [0, inf]

classmethod enum_item_name(data, property, identifier)

Return the UI name for this enum item

PARAMETERS:

- **data** (AnyType , (never None)) – Data from which to take property

- **property** (*string, (never None)*) – Identifier of property in data
- **identifier** (*string, (never None)*) – Identifier of the enum item

RETURNS:

UI name of the enum item

RETURN TYPE:

string, (never None)

classmethod enum_item_description(data, property, identifier)

Return the UI description for this enum item

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **identifier** (*string, (never None)*) – Identifier of the enum item

RETURNS:

UI description of the enum item

RETURN TYPE:

string, (never None)

classmethod enum_item_icon(data, property, identifier)

Return the icon for this enum item

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **identifier** (*string, (never None)*) – Identifier of the enum item

RETURNS:

Icon identifier

RETURN TYPE:

int in [0, inf]

prop(data, property, *, text='', text_ctxt='', translate=True, icon='NONE', placeholder='', expand=False, slider=False, toggle=-1, icon_only=False, event=False, full_event=False, emboss=True, index=-1, icon_value=0, invert_checkbox=False)

Item. Exposes an RNA item and places it into the layout.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **placeholder** (*string, (optional)*) – Hint describing the expected value when empty
- **expand** (*boolean, (optional)*) – Expand button to show more detail
- **slider** (*boolean, (optional)*) – Use slider widget for numeric values
- **toggle** (*int in [-1, 1], (optional)*) – Use toggle widget for boolean values, or a checkbox when disabled (the default is -1 which uses toggle only when an icon is displayed)
- **icon_only** (*boolean, (optional)*) – Draw only icons in buttons, no text
- **event** (*boolean, (optional)*) – Use button to input key events
- **full_event** (*boolean, (optional)*) – Use button to input full events including modifiers
- **emboss** (*boolean, (optional)*) – Draw the button itself, not just the icon/text. When false, corresponds to the 'NONE_OR_STATUS'

- **emboss** (*boolean, (optional)*) – Draw the button itself, not just the icon/text. When false, corresponds to the 'NONE_OR_STATUS' layout emboss type.
- **index** (*int in [-2, inf], (optional)*) – The index of this button, when set a single member of an array can be accessed, when set to -1 all array members are used
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item
- **invert_checkbox** (*boolean, (optional)*) – Draw checkbox value inverted

props_enum(data, property)

props_enum

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

prop_menu_enum(data, property, *, text='', text_ctxt='', translate=True, icon='NONE')

prop_menu_enum

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item

prop_with_popover(data, property, *, text='', text_ctxt='', translate=True, icon='NONE', icon_only=False, panel)

prop_with_popover

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_only** (*boolean, (optional)*) – Draw only icons in tabs, no text
- **panel** (*string, (never None)*) – Identifier of the panel

prop_with_menu(data, property, *, text='', text_ctxt='', translate=True, icon='NONE', icon_only=False, menu)

prop_with_menu

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_only** (*boolean, (optional)*) – Draw only icons in tabs, no text
- **menu** (*string, (never None)*) – Identifier of the menu

prop_tabs_enum(data, property, *, data_highlight=None, property_highlight='', icon_only=False)

prop_tabs_enum

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **data_highlight** (`AnyType` , (optional, never None)) – Data from which to take highlight property
- **property_highlight** (*string, (optional, never None)*) – Identifier of highlight property in data
- **icon_only** (*boolean, (optional)*) – Draw only icons in tabs, no text

prop_enum(data, property, value, *, text='', text_ctxt='', translate=True, icon='NONE')

prop_enum

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **value** (*string, (never None)*) – Enum property value
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item

prop_search(data, property, search_data, search_property, *, text='', text_ctxt='', translate=True, icon='NONE', results_are_suggestions=False)

prop_search

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **search_data** (`AnyType` , (never None)) – Data from which to take collection to search in
- **search_property** (*string, (never None)*) – Identifier of search collection property
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **results_are_suggestions** (*boolean, (optional)*) – Accept inputs that do not match any item

prop_decorator(data, property, *, index=-1)

prop_decorator

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **index** (*int in [-2, inf], (optional)*) – The index of this button, when set a single member of an array can be accessed, when set to -1 all array members are used

operator(operator, *, text='', text_ctxt='', translate=True, icon='NONE', emboss=True, depress=False, icon_value=0, search_weight=0.0)

Item. Places a button into the layout to call an Operator.

PARAMETERS:

- **operator** (*string, (never None)*) – Identifier of the operator
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item

- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **emboss** (*boolean, (optional)*) – Draw the button itself, not just the icon/text
- **depress** (*boolean, (optional)*) – Draw pressed in
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item
- **search_weight** (*float in [-inf, inf], (optional)*) – Search Weight, Influences the sorting when using menu-seach

RETURNS:

Operator properties to fill in

RETURN TYPE:

OperatorProperties

operator_menu_hold(operator, *, text='', text_ctxt='', translate=True, icon='NONE', emboss=True, depress=False, icon_value=0, menu)

Item. Places a button into the layout to call an Operator.

PARAMETERS:

- **operator** (*string, (never None)*) – Identifier of the operator
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **emboss** (*boolean, (optional)*) – Draw the button itself, not just the icon/text
- **depress** (*boolean, (optional)*) – Draw pressed in
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item
- **menu** (*string, (never None)*) – Identifier of the menu

RETURNS:

Operator properties to fill in

RETURN TYPE:

OperatorProperties

operator_enum(operator, property, *, icon_only=False)

operator_enum

PARAMETERS:

- **operator** (*string, (never None)*) – Identifier of the operator
- **property** (*string, (never None)*) – Identifier of property in operator
- **icon_only** (*boolean, (optional)*) – Draw only icons in buttons, no text

operator_menu_enum(operator, property, *, text='', text_ctxt='', translate=True, icon='NONE')

operator_menu_enum

PARAMETERS:

- **operator** (*string, (never None)*) – Identifier of the operator
- **property** (*string, (never None)*) – Identifier of property in operator
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item

RETURNS:

Operator properties to fill in

RETURN TYPE:

OperatorProperties

operator_properties

label(*, text='', text_ctxt='', translate=True, icon='NONE', icon_value=0)

Item. Displays text and/or icon in the layout.

PARAMETERS:

- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item

menu(menu, *, text='', text_ctxt='', translate=True, icon='NONE', icon_value=0)

menu

PARAMETERS:

- **menu** (*string, (never None)*) – Identifier of the menu
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item

menu_contents(menu)

menu_contents

PARAMETERS:

- **menu** (*string, (never None)*) – Identifier of the menu

popover(panel, *, text='', text_ctxt='', translate=True, icon='NONE', icon_value=0)

popover

PARAMETERS:

- **panel** (*string, (never None)*) – Identifier of the panel
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item

popover_group(space_type, region_type, context, category)

popover_group

PARAMETERS:

- **space_type** (enum in Space Type Items) – Space Type
- **region_type** (enum in Region Type Items) – Region Type
- **context** (*string, (never None)*) – panel type context
- **category** (*string, (never None)*) – panel type category

separator(*, factor=1.0, type='AUTO')

Item. Inserts empty space into the layout between items.

PARAMETERS:

- **factor** (*float in [0, inf], (optional)*) – Percentage, Percentage of width to space (leave unset for default space)
- **type** (*enum in ['AUTO', 'SPACE', 'LINE'], (optional)*)

- **type** (*enum in ['AUTO', 'SPACE', 'LINE'], (optional)*) –

  Type, The type of the separator

  - `AUTO` Auto – Best guess at what type of separator is needed..
  - `SPACE` Empty space – Horizontal or Vertical empty space, depending on layout direction..
  - `LINE` Line – Horizontal or Vertical line, depending on layout direction..

separator_spacer()

Item. Inserts horizontal spacing empty space into the layout between items.

progress(*, text='', text_ctxt='', translate=True, factor=0.0, type='BAR')

Progress indicator

PARAMETERS:

- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **factor** (*float in [0, 1], (optional)*) – Factor, Amount of progress from 0.0f to 1.0f
- **type** (*enum in ['BAR', 'RING'], (optional)*) – Type, The type of progress indicator

context_pointer_set(name, data)

context_pointer_set

PARAMETERS:

- **name** (*string, (never None)*) – Name, Name of entry in the context
- **data** (`AnyType`) – Pointer to put in context

context_string_set(name, value)

context_string_set

PARAMETERS:

- **name** (*string, (never None)*) – Name, Name of entry in the context
- **value** (*string, (never None)*) – Value, String to put in context

template_header()

Inserts common Space header UI (editor type selector)

template_ID(data, property, *, new='', open='', unlink='', filter='ALL', live_icon=False, text='', text_ctxt='', translate=True)

template_ID

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **new** (*string, (optional, never None)*) – Operator identifier to create a new ID block
- **open** (*string, (optional, never None)*) – Operator identifier to open a file for creating a new ID block
- **unlink** (*string, (optional, never None)*) – Operator identifier to unlink the ID block
- **filter** (*enum in ['ALL', 'AVAILABLE'], (optional)*) – Optionally limit the items which can be selected
- **live_icon** (*boolean, (optional)*) – Show preview instead of fixed icon
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_ID_preview(data, property, *, new='', open='', unlink='', rows=0, cols=0, filter='ALL', hide_buttons=False)

template_ID_preview

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **new** (*string, (optional, never None)*) – Operator identifier to create a new ID block
- **open** (*string, (optional, never None)*) – Operator identifier to open a file for creating a new ID block
- **unlink** (*string, (optional, never None)*) – Operator identifier to unlink the ID block
- **rows** (*int in [0, inf], (optional)*) – Number of thumbnail preview rows to display
- **cols** (*int in [0, inf], (optional)*) – Number of thumbnail preview columns to display
- **filter** (*enum in ['ALL', 'AVAILABLE'], (optional)*) – Optionally limit the items which can be selected
- **hide_buttons** (*boolean, (optional)*) – Show only list, no buttons

template_any_ID(data, property, type_property, *, text='', text_ctxt='', translate=True)

template_any_ID

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **type_property** (*string, (never None)*) – Identifier of property in data giving the type of the ID-blocks to use
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_ID_tabs(data, property, *, new='', menu='', filter='ALL')

template_ID_tabs

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **new** (*string, (optional, never None)*) – Operator identifier to create a new ID block
- **menu** (*string, (optional, never None)*) – Context menu identifier
- **filter** (*enum in ['ALL', 'AVAILABLE'], (optional)*) – Optionally limit the items which can be selected

template_action(id, *, new='', unlink='', text='', text_ctxt='', translate=True)

template_action

PARAMETERS:

- **id** (`ID` , (never None)) – The data-block for which to select an Action
- **new** (*string, (optional, never None)*) – Operator identifier to create a new ID block
- **unlink** (*string, (optional, never None)*) – Operator identifier to unlink the ID block
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_search(data, property, search_data, search_property, *, new='', unlink='', text='', text_ctxt='', translate=True)

template_search

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **search_data** (`AnyType` , (never None)) – Data from which to take collection to search in
- **search_property** (*string, (never None)*) – Identifier of search collection property
- **new** (*string, (optional, never None)*) – Operator identifier to create a new item for the collection

- **unlink** (*string, (optional, never None)*) – Operator identifier to unlink or delete the active item from the collection
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_search_preview(data, property, search_data, search_property, *, new='', unlink='', text='', text_ctxt='', translate=True, rows=0, cols=0)

template_search_preview

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **search_data** (`AnyType` , (never None)) – Data from which to take collection to search in
- **search_property** (*string, (never None)*) – Identifier of search collection property
- **new** (*string, (optional, never None)*) – Operator identifier to create a new item for the collection
- **unlink** (*string, (optional, never None)*) – Operator identifier to unlink or delete the active item from the collection
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **rows** (*int in [0, inf], (optional)*) – Number of thumbnail preview rows to display
- **cols** (*int in [0, inf], (optional)*) – Number of thumbnail preview columns to display

template_path_builder(data, property, root, *, text='', text_ctxt='', translate=True)

template_path_builder

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **root** (`ID`) – ID-block from which path is evaluated from
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_modifiers()

Generates the UI layout for the modifier stack

template_collection_exporters()

Generates the UI layout for collection exporters

template_constraints(*, use_bone_constraints=True)

Generates the panels for the constraint stack

PARAMETERS:

use_bone_constraints (*boolean, (optional)*) – Add panels for bone constraints instead of object constraints

template_shaderfx()

Generates the panels for the shader effect stack

template_greasepencil_color(data, property, *, rows=0, cols=0, scale=1.0, filter='ALL')

template_greasepencil_color

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

- **property** (*string, (never None)*) – Identifier of property in data
- **rows** (*int in [0, inf], (optional)*) – Number of thumbnail preview rows to display
- **cols** (*int in [0, inf], (optional)*) – Number of thumbnail preview columns to display
- **scale** (*float in [0.1, 1.5], (optional)*) – Scale of the image thumbnails
- **filter** (*enum in ['ALL', 'AVAILABLE'], (optional)*) – Optionally limit the items which can be selected

template_constraint_header(data)

Generates the header for constraint panels

PARAMETERS:

data (`Constraint` , (never None)) – Constraint data

template_preview(id, *, show_buttons=True, parent=None, slot=None, preview_id='')

Item. A preview window for materials, textures, lights or worlds.

PARAMETERS:

- **id** (`ID`) – ID data-block
- **show_buttons** (*boolean, (optional)*) – Show preview buttons?
- **parent** (`ID` , (optional)) – ID data-block
- **slot** (`TextureSlot` , (optional)) – Texture slot
- **preview_id** (*string, (optional, never None)*) – Identifier of this preview widget, if not set the ID type will be used (i.e. all previews of materials without explicit ID will have the same size…).

template_curve_mapping(data, property, *, type='NONE', levels=False, brush=False, use_negative_slope=False, show_tone=False)

Item. A curve mapping widget used for e.g falloff curves for lights.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **type** (*enum in ['NONE', 'VECTOR', 'COLOR', 'HUE'], (optional)*) – Type, Type of curves to display
- **levels** (*boolean, (optional)*) – Show black/white levels
- **brush** (*boolean, (optional)*) – Show brush options
- **use_negative_slope** (*boolean, (optional)*) – Use a negative slope by default
- **show_tone** (*boolean, (optional)*) – Show tone options

template_curveprofile(data, property)

A profile path editor used for custom profiles

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_color_ramp(data, property, *, expand=False)

Item. A color ramp widget.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **expand** (*boolean, (optional)*) – Expand button to show more detail

template_icon(icon_value, *, scale=1.0)

Display a large icon

PARAMETERS:

- **icon_value** (*int in [0, inf]*) – Icon to display
- **scale** (*float in [1, 100], (optional)*) – Scale, Scale the icon size (by the button size)

template_icon_view(data, property, *, show_labels=False, scale=6.0, scale_popup=5.0)

Enum. Large widget showing Icon previews.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **show_labels** (*boolean, (optional)*) – Show enum label in preview buttons
- **scale** (*float in [1, 100], (optional)*) – UI Units, Scale the button icon size (by the button size)
- **scale_popup** (*float in [1, 100], (optional)*) – Scale, Scale the popup icon size (by the button size)

template_histogram(data, property)

Item. A histogramm widget to analyze imaga data.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_waveform(data, property)

Item. A waveform widget to analyze imaga data.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_vectorscope(data, property)

Item. A vectorscope widget to analyze imaga data.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_layers(data, property, used_layers_data, used_layers_property, active_layer)

template_layers

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **used_layers_data** (`AnyType`) – Data from which to take property
- **used_layers_property** (*string, (never None)*) – Identifier of property in data
- **active_layer** (*int in [0, inf]*) – Active Layer

template_color_picker(data, property, *, value_slider=False, lock=False, lock_luminosity=False, cubic=False)

Item. A color wheel widget to pick colors.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **value_slider** (*boolean, (optional)*) – Display the value slider to the right of the color wheel
- **lock** (*boolean, (optional)*) – Lock the color wheel display to value 1.0 regardless of actual color
- **lock_luminosity** (*boolean, (optional)*) – Keep the color at its original vector length
- **cubic** (*boolean, (optional)*) – Cubic saturation for picking values close to white

template_palette(data, property, *, color=False)

Item. A palette used to pick colors.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **color** (*boolean, (optional)*) – Display the colors as colors or values

template_image_layers(image, image_user)

template_image_layers

template_image(data, property, image_user, *, compact=False, multiview=False)

Item(s). User interface for selecting images and their source paths.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **compact** (*boolean, (optional)*) – Use more compact layout
- **multiview** (*boolean, (optional)*) – Expose Multi-View options

template_image_settings(image_settings, *, color_management=False)

User interface for setting image format options

PARAMETERS:

color_management (*boolean, (optional)*) – Show color management settings

template_image_stereo_3d(stereo_3d_format)

User interface for setting image stereo 3d options

template_image_views(image_settings)

User interface for setting image views output options

template_movieclip(data, property, *, compact=False)

Item(s). User interface for selecting movie clips and their source paths.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **compact** (*boolean, (optional)*) – Use more compact layout

template_track(data, property)

Item. A movie-track widget to preview tracking image.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_marker(data, property, clip_user, track, *, compact=False)

Item. A widget to control single marker settings.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data
- **compact** (*boolean, (optional)*) – Use more compact layout

template_movieclip_information(data, property, clip_user)

Item. Movie clip information data.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_list(listtype_name, list_id, dataptr, propname, active_dataptr, active_propname, *, item_dyntip_propname='', rows=5, maxrows=5, type='DEFAULT', columns=9, sort_reverse=False, sort_lock=False)

Item. A list widget to display data, e.g. vertexgroups.

PARAMETERS:

- **listtype_name** (*string, (never None)*) – Identifier of the list type to use
- **list_id** (*string, (never None)*) – Identifier of this list widget. Necessary to tell apart different list widgets. Mandatory when using default "UI_UL_list" class. If this not an empty string, the uilist gets a custom ID, otherwise it takes the name of the class used to define the uilist (for example, if the class name is "OBJECT_UL_vgroups", and list_id is not set by the script, then bl_idname = "OBJECT_UL_vgroups
- **dataptr** (`AnyType`) – Data from which to take the Collection property
- **propname** (*string, (never None)*) – Identifier of the Collection property in data
- **active_dataptr** (`AnyType` , (never None)) – Data from which to take the integer property, index of the active item
- **active_propname** (*string, (never None)*) – Identifier of the integer property in active_data, index of the active item
- **item_dyntip_propname** (*string, (optional, never None)*) – Identifier of a string property in items, to use as tooltip content
- **rows** (*int in [0, inf], (optional)*) – Default and minimum number of rows to display
- **maxrows** (*int in [0, inf], (optional)*) – Default maximum number of rows to display
- **type** (enum in Uilist Layout Type Items, (optional)) – Type, Type of layout to use
- **columns** (*int in [0, inf], (optional)*) – Number of items to display per row, for GRID layout
- **sort_reverse** (*boolean, (optional)*) – Display items in reverse order by default
- **sort_lock** (*boolean, (optional)*) – Lock display order to default value

template_running_jobs()

template_running_jobs

template_operator_search()

template_operator_search

template_menu_search()

template_menu_search

template_header_3D_mode()

template_edit_mode_selection()

Inserts common 3DView Edit modes header UI (selector for selection mode)

template_reports_banner()

template_reports_banner

template_input_status()

template_input_status

template_status_info()

template_status_info

template_node_link(ntree, node, socket)

template_node_link

template_node_view(ntree, node, socket)

template_node_view

template_node_asset_menu_items(*, catalog_path='')

template_node_asset_menu_items

template_modifier_asset_menu_items(*, catalog_path='')

template_modifier_asset_menu_items

template_node_operator_asset_menu_items(*, catalog_path='')

template_node_operator_asset_menu_items

template_node_operator_asset_root_items()

template_node_operator_asset_root_items

template_texture_user()

template_texture_user

template_keymap_item_properties(item)

template_keymap_item_properties

template_component_menu(data, property, *, name='')

Item. Display expanded property in a popup menu

PARAMETERS:

- **data** (`AnyType`) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_colorspace_settings(data, property)

Item. A widget to control input color space settings.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_colormanaged_view_settings(data, property)

Item. A widget to control color managed view settings.

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_node_socket(*, color=(0.0, 0.0, 0.0, 1.0))

Node Socket Icon

PARAMETERS:

color (*float array of 4 items in [0, 1], (optional)*) – Color

template_cache_file(data, property)

Item(s). User interface for selecting cache files and their source paths

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_cache_file_velocity(data, property)

Show cache files velocity properties

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_cache_file_procedural(data, property)

Show cache files render procedural properties

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_cache_file_time_settings(data, property)

Show cache files time settings

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_cache_file_layers(data, property)

Show cache files override layers properties

PARAMETERS:

- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_recent_files(*, rows=5)

Show list of recently saved .blend files

PARAMETERS:

rows (*int in [1, inf], (optional)*) – Maximum number of items to show

RETURNS:

Number of items drawn

RETURN TYPE:

int in [0, inf]

template_file_select_path(params)

Item. A text button to set the active file browser path.

template_event_from_keymap_item(item, *, text='', text_ctxt='', translate=True)

Display keymap item as icons/text

PARAMETERS:

- **item** (`KeyMapItem` , (never None)) – Item
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled

template_asset_view(list_id, asset_library_dataptr, asset_library_propname, assets_dataptr, assets_propname, active_dataptr, active_propname, *, filter_id_types={}, display_options={}, activate_operator='', drag_operator='')

Item. A scrollable list of assets in a grid view

PARAMETERS:

PARAMETERS:

- **list_id** (*string, (never None)*) – Identifier of this asset view. Necessary to tell apart different asset views and to identify an asset view re from a .blend
- **asset_library_dataptr** (`AnyType` , (never None)) – Data from which to take the active asset library property
- **asset_library_propname** (*string, (never None)*) – Identifier of the asset library property
- **assets_dataptr** (`AnyType` , (never None)) – Data from which to take the asset list property
- **assets_propname** (*string, (never None)*) – Identifier of the asset list property
- **active_dataptr** (`AnyType` , (never None)) – Data from which to take the integer property, index of the active item
- **active_propname** (*string, (never None)*) – Identifier of the integer property in active_data, index of the active item
- **filter_id_types** (*enum set in {}, (optional)*) – filter_id_types
- **display_options** (*enum set in {'NO_NAMES', 'NO_FILTER', 'NO_LIBRARY'}, (optional)*) – Displaying options for the asset view
  - `NO_NAMES` Do not display the name of each asset underneath preview images.
  - `NO_FILTER` Do not display buttons for filtering the available assets.
  - `NO_LIBRARY` Do not display buttons to choose or refresh an asset library.
- **activate_operator** (*string, (optional, never None)*) – Name of a custom operator to invoke when activating an item
- **drag_operator** (*string, (optional, never None)*) – Name of a custom operator to invoke when starting to drag an item. Never invoked together with the `active_operator` (if set), it's either the drag or the activate one

RETURNS:

activate_operator_properties, Operator properties to fill in for the custom activate operator passed to the template, `OperatorProperties`

drag_operator_properties, Operator properties to fill in for the custom drag operator passed to the template, `OperatorProperties`

RETURN TYPE:

(`OperatorProperties` , `OperatorProperties`)

template_light_linking_collection(context_layout, data, property)

Visualization of a content of a light linking collection

PARAMETERS:

- **context_layout** (`UILayout` , (never None)) – Layout to set active list element as context properties
- **data** (`AnyType` , (never None)) – Data from which to take property
- **property** (*string, (never None)*) – Identifier of property in data

template_bone_collection_tree()

Show bone collections tree

template_grease_pencil_layer_tree()

View of the active Grease Pencil layer tree

template_node_tree_interface(interface)

Show a node tree interface

PARAMETERS:

interface (`NodeTreeInterface` , (never None)) – Node Tree Interface, Interface of a node tree to display

template_node_inputs(node)

Show a node settings and input socket values

PARAMETERS:

node (`Node` , (never None)) – Node, Display inputs of this node

template_asset_shelf_popover(asset_shelf, *, name='', icon='NONE', icon_value=0)

Create a button to open an asset shelf in a popover

PARAMETERS:

- **asset_shelf** (*string, (never None)*) – Identifier of the asset shelf to display (`bl_idname`)
- **name** (*string, (optional)*) – Optional name to indicate the active asset
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **icon_value** (*int in [0, inf], (optional)*) – Icon Value, Override automatic icon of the item

template_popup_confirm(operator, *, text='', text_ctxt='', translate=True, icon='NONE', cancel_text='', cancel_default=False)

Add confirm & cancel buttons into a popup which will close the popup when pressed

PARAMETERS:

- **operator** (*string, (never None)*) – Identifier of the operator
- **text** (*string, (optional)*) – Override automatic text of the item
- **text_ctxt** (*string, (optional)*) – Override automatic translation context of the given text
- **translate** (*boolean, (optional)*) – Translate the given text, when UI translation is enabled
- **icon** (enum in Icon Items, (optional)) – Icon, Override automatic icon of the item
- **cancel_text** (*string, (optional, never None)*) – Optional text to use for the cancel, not shown when an empty string
- **cancel_default** (*boolean, (optional)*) – Cancel button by default

RETURNS:

Operator properties to fill in

RETURN TYPE:

`OperatorProperties`

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

introspect()

Return a dictionary containing a textual representation of the UI layout.

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- AssetShelf.draw_context_menu
- Header.layout
- Menu.layout
- Node.draw_buttons
- Node.draw_buttons_ext
- NodeInternal.draw_buttons
- NodeInternal.draw_buttons_ext
- NodeSocket.draw
- NodeSocketStandard.draw
- NodeTreeInterfaceSocket.draw
- NodeTreeInterfaceSocketBool.draw
- NodeTreeInterfaceSocketCollection.draw
- NodeTreeInterfaceSocketColor.draw
- NodeTreeInterfaceSocketFloat.draw
- NodeTreeInterfaceSocketFloatAngle.draw
- NodeTreeInterfaceSocketFloatColorTemperature.draw
- NodeTreeInterfaceSocketFloatDistance.draw
- NodeTreeInterfaceSocketFloatFactor.draw
- NodeTreeInterfaceSocketFloatFrequency.draw
- NodeTreeInterfaceSocketFloatPercentage.draw
- NodeTreeInterfaceSocketFloatTime.draw
- NodeTreeInterfaceSocketFloatTimeAbsolute.draw
- NodeTreeInterfaceSocketFloatUnsigned.draw
- NodeTreeInterfaceSocketFloatWavelength.draw
- NodeTreeInterfaceSocketGeometry.draw
- NodeTreeInterfaceSocketImage.draw
- NodeTreeInterfaceSocketInt.draw
- NodeTreeInterfaceSocketIntFactor.draw
- NodeTreeInterfaceSocketIntPercentage.draw
- NodeTreeInterfaceSocketIntUnsigned.draw
- NodeTreeInterfaceSocketMaterial.draw
- NodeTreeInterfaceSocketMatrix.draw
- NodeTreeInterfaceSocketObject.draw
- NodeTreeInterfaceSocketRotation.draw
- NodeTreeInterfaceSocketShader.draw
- NodeTreeInterfaceSocketString.draw
- NodeTreeInterfaceSocketStringFilePath
- NodeTreeInterfaceSocketTexture.draw
- NodeTreeInterfaceSocketVector.draw
- NodeTreeInterfaceSocketVectorAccelera
- NodeTreeInterfaceSocketVectorDirectio
- NodeTreeInterfaceSocketVectorEuler.dr
- NodeTreeInterfaceSocketVectorTranslat
- NodeTreeInterfaceSocketVectorVelocity
- NodeTreeInterfaceSocketVectorXYZ.draw
- Operator.layout
- Panel.layout
- UILayout.box
- UILayout.column
- UILayout.column_flow
- UILayout.grid_flow
- UILayout.menu_pie
- UILayout.panel
- UILayout.panel
- UILayout.panel_prop
- UILayout.panel_prop
- UILayout.row
- UILayout.split
- UILayout.template_light_linking_colle
- UIList.draw_filter
- UIList.draw_item
- UIPieMenu.layout
- UIPopover.layout

- NodeTreeInterfaceSocketMenu.draw
- UIPopupMenu.layout

Copyright © Blender Authors
Made with Furo

Skip to content

UIList(bpy_struct)

Basic UIList Example

This script is the UIList subclass used to show material slots, with a bunch of additional commentaries.

Notice the name of the class, this naming convention is similar as the one for panels or menus.

> Note
>
> UIList subclasses must be registered for blender to use them.

```
import bpy


class MATERIAL_UL_matslots_example(bpy.types.UIList):
    # The draw_item function is called for each item of the collection that is visible in
    #   data is the RNA object containing the collection,
    #   item is the current drawn item of the collection,
    #   icon is the "computed" icon for the item (as an integer, because some objects like
    #   have custom icons ID, which are not available as enum items).
    #   active_data is the RNA object containing the active property for the collection (i
    #   active item of the collection).
    #   active_propname is the name of the active property (use 'getattr(active_data, acti
    #   index is index of the current item in the collection.
    #   flt_flag is the result of the filtering process for this item.
    #   Note: as index and flt_flag are optional arguments, you do not have to use/declare
    #         need them.
    def draw_item(self, context, layout, data, item, icon, active_data, active_propname):
        ob = data
        slot = item
        ma = slot.material
        # draw_item must handle the three layout types... Usually 'DEFAULT' and 'COMPACT'
        if self.layout_type in {'DEFAULT', 'COMPACT'}:
            # You should always start your row layout by a label (icon + text), or a non-e
            # this will also make the row easily selectable in the list! The later also en
            # We use icon_value of label, as our given icon is an integer value, not an en
            # Note "data" names should never be translated!
            if ma:
                layout.prop(ma, "name", text="", emboss=False, icon_value=icon)
            else:
                layout.label(text="", translate=False, icon_value=icon)
        # 'GRID' layout type should be as compact as possible (typically a single icon!).
        elif self.layout_type == 'GRID':
            layout.alignment = 'CENTER'
            layout.label(text="", icon_value=icon)


# And now we can use this list everywhere in Blender. Here is a small example panel.
class UIListPanelExample1(bpy.types.Panel):
    """Creates a Panel in the Object properties window"""
    bl_label = "UIList Example 1 Panel"
    bl_idname = "OBJECT_PT_ui_list_example_1"
    bl_space_type = 'PROPERTIES'
```

```
    bl_region_type = 'WINDOW'
    bl_context = "object"

    def draw(self, context):
        layout = self.layout

        obj = context.object

        # template_list now takes two new args.
        # The first one is the identifier of the registered UIList to use (if you want onl
        # with no custom draw code, use "UI_UL_list").
        layout.template_list("MATERIAL_UL_matslots_example", "", obj, "material_slots", ob

        # The second one can usually be left as an empty string.
        # It's an additional ID used to distinguish lists in case you use the same list se
        layout.template_list("MATERIAL_UL_matslots_example", "compact", obj, "material_slo
                             obj, "active_material_index", type='COMPACT')


def register():
    bpy.utils.register_class(MATERIAL_UL_matslots_example)
    bpy.utils.register_class(UIListPanelExample1)


def unregister():
    bpy.utils.unregister_class(UIListPanelExample1)
    bpy.utils.unregister_class(MATERIAL_UL_matslots_example)


if __name__ == "__main__":
    register()
```

Advanced UIList Example - Filtering and Reordering

This script is an extended version of the `UIList` subclass used to show vertex groups. It is not used 'as is', because iterating over all vertices in a 'draw' function is a very bad idea for UI performances! However, it's a good example of how to create/use filtering/reordering callbacks.

```
import bpy


class MESH_UL_vgroups_slow(bpy.types.UIList):
    # Constants (flags)
    # Be careful not to shadow FILTER_ITEM!
    VGROUP_EMPTY = 1 << 0

    # Custom properties, saved with .blend file.
    use_filter_empty: bpy.props.BoolProperty(
        name="Filter Empty",
        default=False,
        options=set(),
        description="Whether to filter empty vertex groups",
    )
    use_filter_empty_reverse: bpy.props.BoolProperty(
        name="Reverse Empty",
```

```
        default=False,
        options=set(),
        description="Reverse empty filtering",
    )
    use_filter_name_reverse: bpy.props.BoolProperty(
        name="Reverse Name",
        default=False,
        options=set(),
        description="Reverse name filtering",
    )

    # This allows us to have mutually exclusive options, which are also all disable-able!
    def _gen_order_update(name1, name2):
        def _u(self, ctxt):
            if (getattr(self, name1)):
                setattr(self, name2, False)
        return _u
    use_order_name: bpy.props.BoolProperty(
        name="Name", default=False, options=set(),
        description="Sort groups by their name (case-insensitive)",
        update=_gen_order_update("use_order_name", "use_order_importance"),
    )
    use_order_importance: bpy.props.BoolProperty(
        name="Importance",
        default=False,
        options=set(),
        description="Sort groups by their average weight in the mesh",
        update=_gen_order_update("use_order_importance", "use_order_name"),
    )

    # Usual draw item function.
    def draw_item(self, context, layout, data, item, icon, active_data, active_propname, i
        # Just in case, we do not use it here!
        self.use_filter_invert = False

        # assert(isinstance(item, bpy.types.VertexGroup)
        vgroup = item
        if self.layout_type in {'DEFAULT', 'COMPACT'}:
            # Here we use one feature of new filtering feature: it can pass data to draw_i
            # parameter, which contains exactly what filter_items set in its filter list f
            # In this case, we show empty groups grayed out.
            if flt_flag & self.VGROUP_EMPTY:
                col = layout.column()
                col.enabled = False
                col.alignment = 'LEFT'
                col.prop(vgroup, "name", text="", emboss=False, icon_value=icon)
            else:
                layout.prop(vgroup, "name", text="", emboss=False, icon_value=icon)
            icon = 'LOCKED' if vgroup.lock_weight else 'UNLOCKED'
            layout.prop(vgroup, "lock_weight", text="", icon=icon, emboss=False)
        elif self.layout_type == 'GRID':
            layout.alignment = 'CENTER'
            if flt_flag & self.VGROUP_EMPTY:
                layout.enabled = False
```

```
            layout.label(text="", icon_value=icon)

    def draw_filter(self, context, layout):
        # Nothing much to say here, it's usual UI code...
        row = layout.row()

        subrow = row.row(align=True)
        subrow.prop(self, "filter_name", text="")
        icon = 'ZOOM_OUT' if self.use_filter_name_reverse else 'ZOOM_IN'
        subrow.prop(self, "use_filter_name_reverse", text="", icon=icon)

        subrow = row.row(align=True)
        subrow.prop(self, "use_filter_empty", toggle=True)
        icon = 'ZOOM_OUT' if self.use_filter_empty_reverse else 'ZOOM_IN'
        subrow.prop(self, "use_filter_empty_reverse", text="", icon=icon)

        row = layout.row(align=True)
        row.label(text="Order by:")
        row.prop(self, "use_order_name", toggle=True)
        row.prop(self, "use_order_importance", toggle=True)
        icon = 'TRIA_UP' if self.use_filter_orderby_invert else 'TRIA_DOWN'
        row.prop(self, "use_filter_orderby_invert", text="", icon=icon)

    def filter_items_empty_vgroups(self, context, vgroups):
        # This helper function checks vgroups to find out whether they are empty, and what
        # TODO: This should be RNA helper actually (a vgroup prop like "raw_data: ((vidx,
        #       Too slow for python!
        obj_data = context.active_object.data
        ret = {vg.index: [True, 0.0] for vg in vgroups}
        if hasattr(obj_data, "vertices"):  # Mesh data
            if obj_data.is_editmode:
                import bmesh
                bm = bmesh.from_edit_mesh(obj_data)
                # only ever one deform weight layer
                dvert_lay = bm.verts.layers.deform.active
                fact = 1 / len(bm.verts)
                if dvert_lay:
                    for v in bm.verts:
                        for vg_idx, vg_weight in v[dvert_lay].items():
                            ret[vg_idx][0] = False
                            ret[vg_idx][1] += vg_weight * fact
            else:
                fact = 1 / len(obj_data.vertices)
                for v in obj_data.vertices:
                    for vg in v.groups:
                        ret[vg.group][0] = False
                        ret[vg.group][1] += vg.weight * fact
        elif hasattr(obj_data, "points"):  # Lattice data
            # XXX no access to lattice editdata?
            fact = 1 / len(obj_data.points)
            for v in obj_data.points:
                for vg in v.groups:
                    ret[vg.group][0] = False
                    ret[vg.group][1] += vg.weight * fact
        return ret
```

```
        return flt

    def filter_items(self, context, data, propname):
        # This function gets the collection property (as the usual tuple (data, propname))
        # * The first one is for filtering, it must contain 32bit integers were self.bitfl
        #   matching item as filtered (i.e. to be shown). The upper 16 bits (including sel
        #   reserved for internal use, the lower 16 bits are free for custom use. Here we
        #   VGROUP_EMPTY.
        # * The second one is for reordering, it must return a list containing the new inc
        #   gives us a mapping org_idx -> new_idx).
        # Please note that the default UI_UL_list defines helper functions for common task
        # If you do not make filtering and/or ordering, return empty list(s) (this will be
        # returning full lists doing nothing!).
        vgroups = getattr(data, propname)
        helper_funcs = bpy.types.UI_UL_list

        # Default return values.
        flt_flags = []
        flt_neworder = []

        # Pre-compute of vgroups data, CPU-intensive. :/
        vgroups_empty = self.filter_items_empty_vgroups(context, vgroups)

        # Filtering by name
        if self.filter_name:
            flt_flags = helper_funcs.filter_items_by_name(self.filter_name, self.bitflag_f
                                                          reverse=self.use_filter_name_rev
        if not flt_flags:
            flt_flags = [self.bitflag_filter_item] * len(vgroups)

        # Filter by emptiness.
        for idx, vg in enumerate(vgroups):
            if vgroups_empty[vg.index][0]:
                flt_flags[idx] |= self.VGROUP_EMPTY
                if self.use_filter_empty and self.use_filter_empty_reverse:
                    flt_flags[idx] &= ~self.bitflag_filter_item
            elif self.use_filter_empty and not self.use_filter_empty_reverse:
                flt_flags[idx] &= ~self.bitflag_filter_item

        # Reorder by name or average weight.
        if self.use_order_name:
            flt_neworder = helper_funcs.sort_items_by_name(vgroups, "name")
        elif self.use_order_importance:
            _sort = [(idx, vgroups_empty[vg.index][1]) for idx, vg in enumerate(vgroups)]
            flt_neworder = helper_funcs.sort_items_helper(_sort, lambda e: e[1], True)

        return flt_flags, flt_neworder


# Minimal code to use above UIList...
class UIListPanelExample2(bpy.types.Panel):
    """Creates a Panel in the Object properties window"""
    bl_label = "UIList Example 2 Panel"
    bl_idname = "OBJECT_PT_ui_list_example_2"
    bl_space_type = 'PROPERTIES'
```

```
    bl_region_type = 'WINDOW'
    bl_context = "object"

    def draw(self, context):
        layout = self.layout
        obj = context.object

        # template_list now takes two new args.
        # The first one is the identifier of the registered UIList to use (if you want onl
        # with no custom draw code, use "UI_UL_list").
        layout.template_list("MESH_UL_vgroups_slow", "", obj, "vertex_groups", obj.vertex_


def register():
    bpy.utils.register_class(MESH_UL_vgroups_slow)
    bpy.utils.register_class(UIListPanelExample2)


def unregister():
    bpy.utils.unregister_class(UIListPanelExample2)
    bpy.utils.unregister_class(MESH_UL_vgroups_slow)


if __name__ == "__main__":
    register()
```

base class — bpy_struct

subclasses — ASSETBROWSER_UL_metadata_tags, CLIP_UL_tracking_objects, CURVES_UL_attributes, DATA_UL_bone_collections, FILEBROWSER_UL_dir, GPENCIL_UL_annotation_layer, GPENCIL_UL_layer, GPENCIL_UL_masks, GPENCIL_UL_matslots, GREASE_PENCIL_UL_attributes, GREASE_PENCIL_UL_masks, IMAGE_UL_render_slots, IMAGE_UL_udim_tiles, MASK_UL_layers, MATERIAL_UL_matslots, MESH_UL_attributes, MESH_UL_color_attributes, MESH_UL_color_attributes_selector, MESH_UL_shape_keys, MESH_UL_uvmaps, MESH_UL_vgroups, PARTICLE_UL_particle_systems, PHYSICS_UL_dynapaint_surfaces, POINTCLOUD_UL_attributes, POSE_UL_selection_set, RENDER_UL_renderviews, SCENE_UL_gltf2_filter_action, SCENE_UL_keying_set_paths, TEXTURE_UL_texpaintslots, TEXTURE_UL_texslots, UI_UL_list, USERPREF_UL_asset_libraries, USERPREF_UL_extension_repos, VIEWLAYER_UL_aov, VIEWLAYER_UL_linesets, VOLUME_UL_grids, WORKSPACE_UL_addons_items

class bpy.types.**UIList(bpy_struct)**

UI list containing the elements of a collection

bitflag_filter_item

The value of the reserved bitflag “FILTER_ITEM” (in filter_flags values)

TYPE:

int in [0, inf], default 0, (readonly)

bl_idname

If this is set, the uilist gets a custom ID, otherwise it takes the name of the class used to define the uilist (for example, if the class name is “OBJECT_UL_vgroups”, and bl_idname is not set by the script, then bl_idname = “OBJECT_UL_vgroups”)

TYPE:

string, default “”, (never None)

filter_name

filter_name

Only show items matching this name (use '*' as wildcard)

TYPE:

string, default '', (never None)

layout_type

TYPE:

enum in Uilist Layout Type Items, default 'DEFAULT', (readonly)

list_id

Identifier of the list, if any was passed to the "list_id" parameter of "template_list()"

TYPE:

string, default '', (readonly, never None)

use_filter_invert

Invert filtering (show hidden items, and vice versa)

TYPE:

boolean, default False

use_filter_show

Show filtering options

TYPE:

boolean, default False

use_filter_sort_alpha

Sort items by their name

TYPE:

boolean, default False

use_filter_sort_lock

Lock the order of shown items (user cannot change it)

TYPE:

boolean, default False

use_filter_sort_reverse

Reverse the order of shown items

TYPE:

boolean, default False

draw_item(context, layout, data, item, icon, active_data, active_property, index, flt_flag)

Draw an item in the list (NOTE: when you define your own draw_item function, you may want to check given 'item' is of the right type…)

PARAMETERS:

- **layout** (`UILayout`, (never None)) – Layout to draw the item
- **data** (`AnyType`) – Data from which to take Collection property
- **item** (`AnyType`) – Item of the collection property
- **icon** (*int in [0, inf]*) – Icon of the item in the collection
- **active_data** (`AnyType`, (never None)) – Data from which to take property for the active element
- **active_property** (*string, (optional argument, never None)*) – Identifier of property in active_data, for the active element
- **index** (*int in [0, inf]*) – Index of the item in the collection

- **flt_flag** (*int in [0, inf]*) – The filter-flag result for this item

draw_filter(context, layout)

Draw filtering options

PARAMETERS:

layout (`UILayout`, (never None)) – Layout to draw the item

filter_items(context, data, property)

Filter and/or re-order items of the collection (output filter results in filter_flags, and reorder results in filter_neworder arrays)

PARAMETERS:

- **data** (`AnyType`) – Data from which to take Collection property
- **property** (*string, (never None)*) – Identifier of property in data, for the collection

RETURNS:

filter_flags, An array of filter flags, one for each item in the collection (NOTE: The upper 16 bits, including FILTER_ITEM, are reserve only use the lower 16 bits for custom usages), int array of 1 items in [0, inf]

filter_neworder, An array of indices, one for each item in the collection, mapping the org index to the new one, int array of 1 items in [0 inf]

RETURN TYPE:

(int array of 1 items in [0, inf], int array of 1 items in [0, inf])

classmethod append(draw_func)

Append a draw function to this menu, takes the same arguments as the menus draw function

classmethod is_extended()

classmethod prepend(draw_func)

Prepend a draw function to this menu, takes the same arguments as the menus draw function

classmethod remove(draw_func)

Remove a draw function that has been added to this menu

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

Copyright © Blender Authors
Made with Furo

Skip to content

UIPieMenu(bpy_struct)

base class — bpy_struct

class bpy.types.**UIPieMenu(bpy_struct)**

layout

TYPE:

UILayout , (readonly)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- WindowManager.piemenu_begin__internal
- WindowManager.piemenu_end__internal

Copyright © Blender Authors
Made with Furo

Skip to content

UIPopover(bpy_struct)

base class — bpy_struct

class bpy.types.**UIPopover(bpy_struct)**

layout

TYPE:

UILayout , (readonly)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- `WindowManager.popover_begin__internal`
- `WindowManager.popover_end__internal`

Copyright © Blender Authors
Made with Furo

Skip to content

UIPopupMenu(bpy_struct)

base class — bpy_struct

class bpy.types.**UIPopupMenu(bpy_struct)**

layout

TYPE:

UILayout , (readonly)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- `WindowManager.popmenu_begin__internal`
- `WindowManager.popmenu_end__internal`

Copyright © Blender Authors
Made with Furo

Report issue on this page

UnifiedPaintSettings(bpy_struct)

base class — bpy_struct

class bpy.types.**UnifiedPaintSettings(bpy_struct)**

Overrides for some of the active brush's settings

color

TYPE:

mathutils.Color of 3 items in [0, 1], default (0.0, 0.0, 0.0)

input_samples

Number of input samples to average together to smooth the brush stroke

TYPE:

int in [1, 64], default 1

secondary_color

TYPE:

mathutils.Color of 3 items in [0, 1], default (1.0, 1.0, 1.0)

size

Radius of the brush

TYPE:

int in [1, 5000], default 50

strength

How powerful the effect of the brush is when applied

TYPE:

float in [0, 10], default 0.5

unprojected_radius

Radius of brush in Blender units

TYPE:

float in [0.001, inf], default 0.29

use_locked_size

Measure brush size relative to the view or the scene

- VIEW View – Measure brush size relative to the view.
- SCENE Scene – Measure brush size relative to the scene.

TYPE:

enum in ['VIEW', 'SCENE'], default 'VIEW'

use_unified_color

Instead of per-brush color, the color is shared across brushes

TYPE:

boolean, default True

use_unified_input_samples

Instead of per-brush input samples, the value is shared across brushes

Instead of per-brush input samples, the value is shared across brushes

TYPE:
boolean, default False

use_unified_size

Instead of per-brush radius, the radius is shared across brushes

TYPE:
boolean, default True

use_unified_strength

Instead of per-brush strength, the strength is shared across brushes

TYPE:
boolean, default False

use_unified_weight

Instead of per-brush weight, the weight is shared across brushes

TYPE:
boolean, default False

weight

Weight to assign in vertex groups

TYPE:
float in [0, 1], default 0.5

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:
id (*str*) – The RNA type identifier.

RETURNS:
The RNA type or default when not found.

RETURN TYPE:
`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:
id (*str*) – The RNA type identifier.

RETURNS:
The class or default when not found.

RETURN TYPE:
type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.items`

- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- ToolSettings.unified_paint_settings

Copyright © Blender Authors
Made with Furo

Skip to content

UnitSettings(bpy_struct)

base class — bpy_struct

class bpy.types.**UnitSettings(bpy_struct)**

length_unit

Unit that will be used to display length values

TYPE:

enum in [‘DEFAULT’], default ‘DEFAULT’

mass_unit

Unit that will be used to display mass values

TYPE:

enum in [‘DEFAULT’], default ‘DEFAULT’

scale_length

Scale to use when converting between Blender units and dimensions. When working at microscopic or astronomical scale, a small or large uni scale respectively can be used to avoid numerical precision problems

TYPE:

float in [1e-09, inf], default 0.0

system

The unit system to use for user interface controls

TYPE:

enum in [‘NONE’, ‘METRIC’, ‘IMPERIAL’], default ‘NONE’

system_rotation

Unit to use for displaying/editing rotation values

- DEGREES Degrees – Use degrees for measuring angles and rotations.
- RADIANS Radians.

TYPE:

enum in [‘DEGREES’, ‘RADIANS’], default ‘DEGREES’

temperature_unit

Unit that will be used to display temperature values

TYPE:

enum in [‘DEFAULT’], default ‘DEFAULT’

time_unit

Unit that will be used to display time values

TYPE:

enum in [‘DEFAULT’], default ‘DEFAULT’

use_separate

Display units in pairs (e.g. 1m 0cm)

TYPE:

boolean, default False

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.id_properties_ui`
- `bpy_struct.is_property_hidden`
- `bpy_struct.is_property_overridable_library`
- `bpy_struct.is_property_readonly`
- `bpy_struct.is_property_set`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`
- `bpy_struct.pop`
- `bpy_struct.property_overridable_library_set`
- `bpy_struct.property_unset`
- `bpy_struct.type_recast`
- `bpy_struct.values`

References

- `Scene.unit_settings`

Copyright © Blender Authors
Made with Furo

UnknownType(bpy_struct)

base class — bpy_struct

class bpy.types.**UnknownType(bpy_struct)**

Stub RNA type used for pointers to unknown or internal data

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- ShapeKey.data

Copyright © Blender Authors
Made with Furo

Report issue on this page

USDHook(bpy_struct)

USD Hook Example

This example shows an implementation of `USDHook` to extend USD export and import functionality.

Callback Function API

One may optionally define any or all of the following callback functions in the `USDHook` subclass.

on_export

Called before the USD export finalizes, allowing modifications to the USD stage immediately before it is saved.

Args:

- `export_context` (USDSceneExportContext): Provides access to the stage and dependency graph

Returns:

- `True` on success or `False` if the operation was bypassed or otherwise failed to complete

on_material_export

Called for each material that is exported, allowing modifications to the USD material, such as shader generation.

Args:

- `export_context` (USDMaterialExportContext): Provides access to the stage and a texture export utility function
- `bl_material` (`bpy.types.Material`): The source Blender material
- `usd_material` (`pxr.UsdShade.Material`): The target USD material to be exported

Returns:

- `True` on success or `False` if the operation was bypassed or otherwise failed to complete

Note that the target USD material might already have connected shaders created by the USD exporter or by other material export hooks.

on_import

Called after the USD import finalizes.

Args:

- `import_context` (USDSceneImportContext): Provides access to the stage and a map associating USD prim paths and Blender IDs

Returns:

- `True` on success or `False` if the operation was bypassed or otherwise failed to complete

material_import_poll

Called to determine if the `USDHook` implementation can convert a given USD material.

Args:

- `import_context` (USDMaterialImportContext): Provides access to the stage and a texture import utility function
- `usd_material` (`pxr.UsdShade.Material`): The source USD material to be exported

Returns:

- `True` if the hook can convert the material or `False` otherwise

If any hook returns `True` from `material_import_poll`, the USD importer will skip standard `USD Preview Surface` or

`MaterialX` import and invoke the hook's on_material_import method to convert the material instead.

on_material_import

Called for each material that is imported, to allow converting the USD material to nodes on the Blender material. To ensure that this function gets called, the hook must also implement the `material_import_poll()` callback to return `True` for the given USD material.

Args:

- `import_context` (USDMaterialImportContext): Provides access to the stage and a texture import utility function
- `bl_material` (`bpy.types.Material`): The target Blender material with an empty node tree
- `usd_material` (`pxr.UsdShade.Material`): The source USD material to be imported

Returns:

- `True` on success or `False` if the conversion failed or otherwise did not complete

Context Classes

Instances of the following built-in classes are provided as arguments to the callbacks.

USDSceneExportContext

Argument for on_export.

Methods:

- `get_stage()` : returns the USD stage to be saved
- `get_depsgraph()` : returns the Blender scene dependency graph

USDMaterialExportContext

Argument for on_material_export.

Methods:

- `get_stage()` : returns the USD stage to be saved
- `export_texture(image: bpy.types.Image)` : Returns the USD asset path for the given texture image

The `export_texture` function will save in-memory images and may copy texture assets, depending on the current USD export options. For example, by default calling `export_texture(/foo/bar.png)` will copy the file to a `textures` directory next to the exported USD and will return the relative path `./textures/bar.png`.

USDSceneImportContext

Argument for on_import.

Methods:

- `get_prim_map()` returns a `dict` where the key is an imported USD Prim path and the value a `list` of the IDs created by the imported prim.
- `get_stage()` returns the USD stage which was imported.

USDMaterialImportContext

Argument for material_import_poll and on_material_import.

Methods:

- `get_stage()` : returns the USD stage to be saved.
- `import_texture(asset_path: str)` : for the given USD texture asset path, returns a `tuple[str, bool]`, containing the asset local path and a bool indicating whether the path references a temporary file.

The `import_texture` function may copy the texture to the local file system if the given asset path is a package-relative path for a USDZ archive, depending on the current USD `Import Textures` options. When the `Import Textures` mode is `Packed`, the texture is saved to a

depending on the current USD `Import Textures` options. When the `Import Textures` mode is `Packed`, the texture is saved to a temporary location and the second element of the returned tuple is `True`, indicating that the file is temporary, in which case it may be necessary to pacl the image. The original asset path will be returned unchanged if it's already a local file or if it could not be copied to a local destination.

Errors

Exceptions raised by these functions will be reported in Blender with the exception details printed to the console.

Example Code

The `USDHookExample` class in the example below implements the following functions:

- `on_export()` function to add custom data to the stage's root layer.
- `on_material_export()` function to create a simple `MaterialX` shader on the given USD material.
- `on_import()` function to create a text object to display the stage's custom layer data.
- `material_import_poll()` returns `True` if the given USD material has an `mtlx` context.
- `on_material_import()` function to convert a simple `MaterialX` shader with a `base_color` input.

```
bl_info = {
    "name": "USD Hook Example",
    "blender": (4, 4, 0),
}

import bpy
import bpy.types
import textwrap

# Make `pxr` module available, for running as `bpy` PIP package.
bpy.utils.expose_bundled_modules()

import pxr.Gf as Gf
import pxr.Sdf as Sdf
import pxr.Usd as Usd
import pxr.UsdShade as UsdShade


class USDHookExample(bpy.types.USDHook):
    """Example implementation of USD IO hooks"""
    bl_idname = "usd_hook_example"
    bl_label = "Example"

    @staticmethod
    def on_export(export_context):
        """ Include the Blender filepath in the root layer custom data.
        """

        stage = export_context.get_stage()

        if stage is None:
            return False
        data = bpy.data
        if data is None:
            return False

        # Set the custom data.
        rootLayer = stage.GetRootLayer()
```

```
        rootLayer = stage.GetRootLayer()
        customData = rootLayer.customLayerData
        customData["blenderFilepath"] = data.filepath
        rootLayer.customLayerData = customData

        return True

    @staticmethod
    def on_material_export(export_context, bl_material, usd_material):
        """ Create a simple MaterialX shader on the exported material.
        """

        stage = export_context.get_stage()

        # Create a MaterialX standard surface shader
        mtl_path = usd_material.GetPrim().GetPath()
        shader = UsdShade.Shader.Define(stage, mtl_path.AppendPath("mtlxstandard_surface")
        shader.CreateIdAttr("ND_standard_surface_surfaceshader")

        # Connect the shader.  MaterialX materials use "mtlx" renderContext
        usd_material.CreateSurfaceOutput("mtlx").ConnectToSource(shader.ConnectableAPI(),

        # Set the color to the Blender material's viewport display color.
        col = bl_material.diffuse_color
        shader.CreateInput("base_color", Sdf.ValueTypeNames.Color3f).Set(Gf.Vec3f(col[0],

        return True

    @staticmethod
    def on_import(import_context):
        """ Create a text object to display the stage's custom data.
        """
        stage = import_context.get_stage()

        if stage is None:
            return False

        # Get the custom data.
        rootLayer = stage.GetRootLayer()
        customData = rootLayer.customLayerData

        # Create a text object to display the stage path
        # and custom data dictionary entries.

        bpy.ops.object.text_add()
        ob = bpy.context.view_layer.objects.active

        if (ob is None) or (ob.data is None):
            return False

        ob.name = "layer_data"
        ob.data.name = "layer_data"

        # The stage root path is the first line.
        text = rootLayer.realPath
```

```
        # Append key/value strings, enforcing text wrapping.
        for item in customData.items():
            print(item)
            text += '\n'
            line = str(item[0]) + ': ' + str(item[1])
            text += textwrap.fill(line, width=80)

        ob.data.body = text

        return True

    @staticmethod
    def material_import_poll(import_context, usd_material):
        """
        Return True if the given USD material can be converted.
        Return False otherwise.
        """
        # We can convert MaterialX.
        surf_output = usd_material.GetSurfaceOutput("mtlx")
        return bool(surf_output)

    @staticmethod
    def on_material_import(import_context, bl_material, usd_material):
        """
        Import a simple mtlx material.  Just handle the base_color input
        of a ND_standard_surface_surfaceshader.
        """

        # We must confirm that we can handle this material.
        surf_output = usd_material.GetSurfaceOutput("mtlx")
        if not surf_output:
            return False

        if not surf_output.HasConnectedSource():
            return False

        # Get the connected surface output source.
        source = surf_output.GetConnectedSource()
        # Get the shader prim from the source
        shader = UsdShade.Shader(source[0])
        shader_id = shader.GetShaderId()
        if shader_id != "ND_standard_surface_surfaceshader":
            return False

        color_attr = shader.GetInput("base_color")
        if color_attr is None:
            return False

        # Create the node tree
        bl_material.use_nodes = True
        node_tree = bl_material.node_tree
        nodes = node_tree.nodes
        bsdf = nodes.get("Principled BSDF")
        assert bsdf
```

```
        bsdf_base_color_input = bsdf.inputs['Base Color']

        # Try to set the default color value.
        # Get the authored default value
        color = color_attr.Get()

        if color is None:
            return False

        bsdf_base_color_input.default_value = (color[0], color[1], color[2], 1)

        return True


def register():
    bpy.utils.register_class(USDHookExample)


def unregister():
    bpy.utils.unregister_class(USDHookExample)


if __name__ == "__main__":
    register()
```

base class — bpy_struct

class bpy.types.**USDHook(bpy_struct)**

Defines callback functions to extend USD IO

bl_description

A short description of the USD hook

TYPE:

string, default “”, (never None)

bl_idname

TYPE:

string, default “”, (never None)

bl_label

TYPE:

string, default “”, (never None)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

Copyright © Blender Authors
Made with Furo

UserAssetLibrary(bpy_struct)

base class — `bpy_struct`

class bpy.types.**UserAssetLibrary(bpy_struct)**

Settings to define a reusable library for Asset Browsers to use

import_method

Determine how the asset will be imported, unless overridden by the Asset Browser

- `LINK` Link – Import the assets as linked data-block.
- `APPEND` Append – Import the assets as copied data-block, with no link to the original asset data-block.
- `APPEND_REUSE` Append (Reuse Data) – Import the assets as copied data-block while avoiding multiple copies of nested, typically heavy data. For example the textures of a material asset, or the mesh of an object asset, don't have to be copied every time this asset is imported. The instances of the asset share the data instead..

TYPE:

enum in ['LINK', 'APPEND', 'APPEND_REUSE'], default 'APPEND_REUSE'

name

Identifier (not necessarily unique) for the asset library

TYPE:

string, default "", (never None)

path

Path to a directory with .blend files to use as an asset library

TYPE:

string, default "", (never None)

use_relative_path

Use relative path when linking assets from this asset library

TYPE:

boolean, default True

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- AssetLibraryCollection.new
- AssetLibraryCollection.remove
- PreferencesFilePaths.asset_libraries

Copyright © Blender Authors
Made with Furo

UserExtensionRepo(bpy_struct)

base class — bpy_struct

class bpy.types.**UserExtensionRepo(bpy_struct)**

Settings to define an extension repository

access_token

Personal access token, may be required by some repositories

TYPE:

string, default “”, (never None)

custom_directory

The local directory containing extensions

TYPE:

string, default “”, (never None)

directory

The local directory containing extensions

TYPE:

string, default “”, (readonly, never None)

enabled

Enable the repository

TYPE:

boolean, default False

module

Unique module identifier

TYPE:

string, default “”, (never None)

name

Unique repository name

TYPE:

string, default “”, (never None)

remote_url

Remote URL to the extension repository, the file-system may be referenced using the file URI scheme: “file://”

TYPE:

string, default “”, (never None)

source

Select if the repository is in a user managed or system provided directory

- USER User – Repository managed by the user, stored in user directories.
- SYSTEM System – Read-only repository provided by the system.

TYPE:

enum in ['USER', 'SYSTEM'], default 'USER'

use_access_token

Repository requires an access token

TYPE:

boolean, default False

use_cache

Downloaded package files are deleted after installation

TYPE:

boolean, default False

use_custom_directory

Manually set the path for extensions to be stored. When disabled a user's extensions directory is created.

TYPE:

boolean, default False

use_remote_url

Synchronize the repository with a remote URL

TYPE:

boolean, default False

use_sync_on_startup

Allow Blender to check for updates upon launch

TYPE:

boolean, default False

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- PreferencesExtensions.repos
- UserExtensionRepoCollection.new
- UserExtensionRepoCollection.remove

Copyright © Blender Authors
Made with Furo

UserExtensionRepoCollection(bpy_struct)

base class — bpy_struct

class bpy.types.**UserExtensionRepoCollection(bpy_struct)**

Collection of user extension repositories

classmethod new(*, name='', module='', custom_directory='', remote_url='', source='USER')

Add a new repository

PARAMETERS:

- **name** (*string, (optional, never None)*) – Name
- **module** (*string, (optional, never None)*) – Module
- **custom_directory** (*string, (optional, never None)*) – Custom Directory
- **remote_url** (*string, (optional, never None)*) – Remote URL
- **source** (*enum in ['USER', 'SYSTEM'], (optional)*) –
  Source, How the repository is managed
  - `USER` User – Repository managed by the user, stored in user directories.
  - `SYSTEM` System – Read-only repository provided by the system.

RETURNS:

Newly added repository

RETURN TYPE:

`UserExtensionRepo`

classmethod remove(repo)

Remove repos

PARAMETERS:

repo (`UserExtensionRepo`, (never None)) – Repository to remove

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- PreferencesExtensions.repos

Copyright © Blender Authors
Made with Furo

Skip to content

USERPREF_UL_asset_libraries(UIList)

base classes — bpy_struct , UIList

class bpy.types.**USERPREF_UL_asset_libraries(UIList)**

draw_item(_context, layout, _data, item, icon, _active_data, _active_propname, _index)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- UIList.bl_idname
- UIList.list_id
- UIList.layout_type
- UIList.use_filter_show
- UIList.filter_name
- UIList.use_filter_invert
- UIList.use_filter_sort_alpha
- UIList.use_filter_sort_reverse
- UIList.use_filter_sort_lock
- UIList.bitflag_filter_item

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- UIList.draw_item
- UIList.draw_filter
- UIList.filter_items
- UIList.append
- UIList.is_extended
- UIList.prepend

- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- UIList.remove
- UIList.bl_rna_get_subclass
- UIList.bl_rna_get_subclass_py

Copyright © Blender Authors
Made with Furo

Skip to content

USERPREF_UL_extension_repos(UIList)

base classes — bpy_struct , UIList

class bpy.types.**USERPREF_UL_extension_repos(UIList)**

draw_item(_context, layout, _data, item, icon, _active_data, _active_propname, _index)

filter_items(_context, data, propname)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- UIList.bl_idname
- UIList.list_id
- UIList.layout_type
- UIList.use_filter_show
- UIList.filter_name
- UIList.use_filter_invert
- UIList.use_filter_sort_alpha
- UIList.use_filter_sort_reverse
- UIList.use_filter_sort_lock
- UIList.bitflag_filter_item

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- UIList.draw_item
- UIList.draw_filter
- UIList.filter_items
- UIList.append

- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- UIList.is_extended
- UIList.prepend
- UIList.remove
- UIList.bl_rna_get_subclass
- UIList.bl_rna_get_subclass_py

Previous
USERPREF_UL_asset_libraries(UIList)
Copyright © Blender Authors
Made with Furo
N
UVLoopLayers(bpy_stru
Report issue on this page

Skip to content

UserSolidLight(bpy_struct)

base class — bpy_struct

class bpy.types.**UserSolidLight(bpy_struct)**

Light used for Studio lighting in solid shading mode

diffuse_color

Color of the light's diffuse highlight

TYPE:

mathutils.Color of 3 items in [0, inf], default (0.8, 0.8, 0.8)

direction

Direction that the light is shining

TYPE:

mathutils.Vector of 3 items in [-inf, inf], default (0.0, 0.0, 1.0)

smooth

Smooth the lighting from this light

TYPE:

float in [0, 1], default 0.5

specular_color

Color of the light's specular highlight

TYPE:

mathutils.Color of 3 items in [0, inf], default (0.8, 0.8, 0.8)

use

Enable this light in solid shading mode

TYPE:

boolean, default True

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- PreferencesSystem.solid_lights
- StudioLight.solid_lights

Copyright © Blender Authors
Made with Furo

UVLoopLayers(bpy_struct)

base class — bpy_struct

class bpy.types.**UVLoopLayers(bpy_struct)**

Collection of UV map layers

active

Active UV Map layer

TYPE:

MeshUVLoopLayer

active_index

Active UV map index

TYPE:

int in [0, inf], default 0

new(*, name='UVMap', do_init=True)

Add a UV map layer to Mesh

PARAMETERS:

- **name** (*string, (optional, never None)*) – UV map name
- **do_init** (*boolean, (optional)*) – Whether new layer's data should be initialized by copying current active one, or if none is active, with a default UVmap

RETURNS:

The newly created layer

RETURN TYPE:

MeshUVLoopLayer

remove(layer)

Remove a vertex color layer

PARAMETERS:

layer (MeshUVLoopLayer, (never None)) – The layer to remove

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- Mesh.uv_layers

Copyright © Blender Authors
Made with Furo

UVProjectModifier(Modifier)

base classes — bpy_struct, Modifier

class bpy.types.**UVProjectModifier(Modifier)**

UV projection modifier to set UVs from a projector

aspect_x

Horizontal aspect ratio (only used for camera projectors)

TYPE:

float in [1, inf], default 1.0

aspect_y

Vertical aspect ratio (only used for camera projectors)

TYPE:

float in [1, inf], default 1.0

projector_count

Number of projectors to use

TYPE:

int in [1, 10], default 1

projectors

TYPE:

bpy_prop_collection of UVProjector, (readonly)

scale_x

Horizontal scale (only used for camera projectors)

TYPE:

float in [0, inf], default 1.0

scale_y

Vertical scale (only used for camera projectors)

TYPE:

float in [0, inf], default 1.0

uv_layer

UV map name

TYPE:

string, default “”, (never None)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- Modifier.name
- Modifier.type
- Modifier.show_viewport
- Modifier.show_render
- Modifier.show_in_editmode
- Modifier.show_on_cage
- Modifier.show_expanded
- Modifier.is_active
- Modifier.use_pin_to_last
- Modifier.is_override_data
- Modifier.use_apply_on_spline
- Modifier.execution_time
- Modifier.persistent_uid

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- Modifier.bl_rna_get_subclass
- Modifier.bl_rna_get_subclass_py

Copyright © Blender Authors
Made with Furo

UVProjector(bpy_struct)

base class — bpy_struct

class bpy.types.**UVProjector(bpy_struct)**

UV projector used by the UV project modifier

object

Object to use as projector transform

TYPE:

Object

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- `UVProjectModifier.projectors`

Copyright © Blender Authors
Made with Furo

UvSculpt(bpy_struct)

base class — bpy_struct

class bpy.types.**UvSculpt(bpy_struct)**

curve_preset

TYPE:

enum in Brush Curve Preset Items, default 'CUSTOM'

size

TYPE:

int in [1, 5000], default 0

strength

TYPE:

float in [0, 1], default 0.0

strength_curve

TYPE:

CurveMapping, (readonly)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys

- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- ToolSettings.uv_sculpt

Copyright © Blender Authors
Made with Furo

Skip to content

UVWarpModifier(Modifier)

base classes — bpy_struct , Modifier

class bpy.types.**UVWarpModifier(Modifier)**

Add target position to UV coordinates

axis_u

Pole axis for rotation

TYPE:

enum in Axis Xyz Items, default 'X'

axis_v

Pole axis for rotation

TYPE:

enum in Axis Xyz Items, default 'Y'

bone_from

Bone defining offset

TYPE:

string, default "", (never None)

bone_to

Bone defining offset

TYPE:

string, default "", (never None)

center

Center point for rotate/scale

TYPE:

float array of 2 items in [-inf, inf], default (0.5, 0.5)

invert_vertex_group

Invert vertex group influence

TYPE:

boolean, default False

object_from

Object defining offset

TYPE:

Object

object_to

Object defining offset

TYPE:

Object

offset

2D Offset for the warp

TYPE:
float array of 2 items in [-inf, inf], default (0.0, 0.0)

rotation

2D Rotation for the warp

TYPE:
float in [-inf, inf], default 0.0

scale

2D Scale for the warp

TYPE:
float array of 2 items in [-inf, inf], default (1.0, 1.0)

uv_layer

UV map name

TYPE:
string, default "", (never None)

vertex_group

Vertex group name

TYPE:
string, default "", (never None)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:
id (*str*) – The RNA type identifier.

RETURNS:
The RNA type or default when not found.

RETURN TYPE:
`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:
id (*str*) – The RNA type identifier.

RETURNS:
The class or default when not found.

RETURN TYPE:
type

Inherited Properties

- `bpy_struct.id_data`
- `Modifier.name`
- `Modifier.type`
- `Modifier.show_viewport`
- `Modifier.show_render`
- `Modifier.show_in_editmode`
- `Modifier.show_expanded`
- `Modifier.is_active`
- `Modifier.use_pin_to_last`
- `Modifier.is_override_data`
- `Modifier.use_apply_on_spline`
- `Modifier.execution_time`

- Modifier.show_on_cage
- Modifier.persistent_uid

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- Modifier.bl_rna_get_subclass
- Modifier.bl_rna_get_subclass_py

Copyright © Blender Authors
Made with Furo

VectorFont(ID)

base classes — bpy_struct, ID

class bpy.types.**VectorFont(ID)**

Vector font for Text objects

filepath

TYPE:

string, default “”, (never None)

packed_file

TYPE:

PackedFile, (readonly)

pack()

Pack the font into the current blend file

unpack(*, method='USE_LOCAL')

Unpack the font to the samples filename

PARAMETERS:

method (enum in Unpack Method Items, (optional)) – method, How to unpack

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- ID.name
- ID.name_full
- ID.id_type
- ID.session_uid
- ID.is_evaluated
- ID.original
- ID.users
- ID.is_missing
- ID.is_runtime_data
- ID.is_editable
- ID.tag
- ID.is_library_indirect
- ID.library
- ID.library_weak_reference

- ID.users
- ID.use_fake_user
- ID.use_extra_user
- ID.is_embedded_data
- ID.asset_data
- ID.override_library
- ID.preview

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- ID.rename
- ID.evaluated_get
- ID.copy
- ID.asset_mark
- ID.asset_clear
- ID.asset_generate_preview
- ID.override_create
- ID.override_hierarchy_create
- ID.user_clear
- ID.user_remap
- ID.make_local
- ID.user_of_id
- ID.animation_data_create
- ID.animation_data_clear
- ID.update_tag
- ID.preview_ensure
- ID.bl_rna_get_subclass
- ID.bl_rna_get_subclass_py

References

- BlendData.fonts
- BlendDataFonts.load
- BlendDataFonts.remove
- GeometryNodeStringToCurves.font
- TextCurve.font
- TextCurve.font_bold
- TextCurve.font_bold_italic
- TextCurve.font_italic
- TextStrip.font

Copyright © Blender Authors
Made with Furo

VertexGroup(bpy_struct)

base class — bpy_struct

class bpy.types.**VertexGroup(bpy_struct)**

Group of vertices, used for armature deform and other purposes

index

Index number of the vertex group

TYPE:

int in [0, inf], default 0, (readonly)

lock_weight

Maintain the relative weights for the group

TYPE:

boolean, default False

name

Vertex group name

TYPE:

string, default "", (never None)

add(index, weight, type)

Add vertices to the group

PARAMETERS:

- **index** (*int array of 1 items in [-inf, inf]*) – List of indices
- **weight** (*float in [0, 1]*) – Vertex weight
- **type** (*enum in ['REPLACE', 'ADD', 'SUBTRACT']*) – Vertex assign mode
  - REPLACE Replace – Replace.
  - ADD Add – Add.
  - SUBTRACT Subtract – Subtract.

remove(index)

Remove vertices from the group

PARAMETERS:

index (*int array of 1 items in [-inf, inf]*) – List of indices

weight(index)

Get a vertex weight from the group

PARAMETERS:

index (*int in [0, inf]*) – Index, The index of the vertex

RETURNS:

Vertex weight

RETURN TYPE:

float in [0, 1]

classmethod bl_rna_get_subclass(id, default=None)

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

`bpy.types.Struct` subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- `bpy_struct.id_data`

Inherited Functions

- `bpy_struct.as_pointer`
- `bpy_struct.driver_add`
- `bpy_struct.driver_remove`
- `bpy_struct.get`
- `bpy_struct.id_properties_clear`
- `bpy_struct.id_properties_ensure`
- `bpy_struct.id_properties_ui`
- `bpy_struct.is_property_hidden`
- `bpy_struct.is_property_overridable_library`
- `bpy_struct.is_property_readonly`
- `bpy_struct.is_property_set`
- `bpy_struct.items`
- `bpy_struct.keyframe_delete`
- `bpy_struct.keyframe_insert`
- `bpy_struct.keys`
- `bpy_struct.path_from_id`
- `bpy_struct.path_resolve`
- `bpy_struct.pop`
- `bpy_struct.property_overridable_library_set`
- `bpy_struct.property_unset`
- `bpy_struct.type_recast`
- `bpy_struct.values`

References

- `Object.vertex_groups`
- `VertexGroups.active`
- `VertexGroups.new`
- `VertexGroups.remove`

Copyright © Blender Authors
Made with Furo

VertexGroupElement(bpy_struct)

base class — bpy_struct

class bpy.types.**VertexGroupElement(bpy_struct)**

Weight value of a vertex in a vertex group

group

TYPE:

int in [0, inf], default 0, (readonly)

weight

Vertex Weight

TYPE:

float in [0, 1], default 0.0

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset

- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.type_recast
- bpy_struct.values

References

- LatticePoint.groups
- MeshVertex.groups

Copyright © Blender Authors
Made with Furo

VertexGroups(bpy_struct)

base class — bpy_struct

class bpy.types.**VertexGroups(bpy_struct)**

Collection of vertex groups

active

Vertex groups of the object

TYPE:

VertexGroup

active_index

Active index in vertex group array

TYPE:

int in [0, inf], default 0

new(*, name='Group')

Add vertex group to object

PARAMETERS:

name (*string, (optional, never None)*) – Vertex group name

RETURNS:

New vertex group

RETURN TYPE:

VertexGroup

remove(group)

Delete vertex group from object

PARAMETERS:

group (VertexGroup, (never None)) – Vertex group to remove

clear()

Delete all vertex groups from object

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values

References

- Object.vertex_groups

Copyright © Blender Authors
Made with Furo

VertexPaint(Paint)

base classes — bpy_struct, Paint

class bpy.types.**VertexPaint(Paint)**

Properties of vertex and weight paint mode

radial_symmetry

Number of times to copy strokes across the surface

TYPE:

int array of 3 items in [1, 64], default (1, 1, 1)

use_group_restrict

Restrict painting to vertices in the group

TYPE:

boolean, default False

classmethod bl_rna_get_subclass(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The RNA type or default when not found.

RETURN TYPE:

bpy.types.Struct subclass

classmethod bl_rna_get_subclass_py(id, default=None)

PARAMETERS:

id (*str*) – The RNA type identifier.

RETURNS:

The class or default when not found.

RETURN TYPE:

type

Inherited Properties

- bpy_struct.id_data
- Paint.brush
- Paint.brush_asset_reference
- Paint.eraser_brush
- Paint.eraser_brush_asset_reference
- Paint.palette
- Paint.show_brush
- Paint.show_brush_on_surface
- Paint.show_low_resolution
- Paint.use_sculpt_delay_updates
- Paint.use_symmetry_x
- Paint.use_symmetry_y
- Paint.use_symmetry_z
- Paint.use_symmetry_feather
- Paint.cavity_curve
- Paint.use_cavity
- Paint.tile_offset
- Paint.tile_x
- Paint.tile_y
- Paint.tile_z

Inherited Functions

- bpy_struct.as_pointer
- bpy_struct.driver_add
- bpy_struct.driver_remove
- bpy_struct.get
- bpy_struct.id_properties_clear
- bpy_struct.id_properties_ensure
- bpy_struct.id_properties_ui
- bpy_struct.is_property_hidden
- bpy_struct.is_property_overridable_library
- bpy_struct.is_property_readonly
- bpy_struct.is_property_set
- bpy_struct.items
- bpy_struct.keyframe_delete
- bpy_struct.keyframe_insert
- bpy_struct.keys
- bpy_struct.path_from_id
- bpy_struct.path_resolve
- bpy_struct.pop
- bpy_struct.property_overridable_library_set
- bpy_struct.property_unset
- bpy_struct.type_recast
- bpy_struct.values
- Paint.bl_rna_get_subclass
- Paint.bl_rna_get_subclass_py

References

- ToolSettings.vertex_paint
- ToolSettings.weight_paint

Copyright © Blender Authors
Made with Furo