1 wxErlang User's Guide

The wxErlang application is an api for writing graphical user interfaces with wxWidgets.

1.1 wx the erlang binding of wxWidgets

The wx application is an erlang binding of wxWidgets. This document describes the erlang mapping to wxWidgets and it’s implementation. It is not a complete users guide to wxWidgets. If you need that, you will have to read the wxWidgets documentation instead. wx tries to keep a one-to-one mapping with the original API so that the original documentation and examples shall be as easy as possible to use.

wxErlang examples and test suite can be found in the erlang src release. They can also provide some help on how to use the API.

This is currently a very brief introduction to wx. The application is still under development, which means the interface may change, and the test suite currently have a poor coverage ratio.

1.1.1 Contents

• Introduction
• Multiple processes and memory handling
• Event Handling
• Acknowledgments

1.1.2 Introduction

The original wxWidgets is an object-oriented (C++) API and that is reflected in the erlang mapping. In most cases each class in wxWidgets is represented as a module in erlang. This gives the wx application a huge interface, spread over several modules, and it all starts with the wx module. The wx module contains functions to create and destroy the GUI, i.e. wx:new/0, wx:destroy/0, and some other useful functions.

Objects or object references in wx should be seen as erlang processes rather than erlang terms. When you operate on them they can change state, e.g. they are not functional objects as erlang terms are. Each object has a type or rather a class, which is manipulated with the corresponding module or by sub-classes of that object. Type checking is done so that a module only operates on it’s objects or inherited classes.

An object is created with new and destroyed with destroy. Most functions in the classes are named the same as their C++ counterpart, except that for convenience, in erlang they start with a lowercase letter and the first argument is the object reference. Optional arguments are last and expressed as tagged tuples in any order.

For example the wxWindow C++ class is implemented in the wxWindow erlang module and the member wxWindow::CenterOnParent is thus wxWindow:centerOnParent. The following C++ code:

```cpp
wxWindow MyWin = new wxWindow();
MyWin.CenterOnParent(wxVERTICAL);
...
delete MyWin;
```

would in erlang look like:
When you are reading wxWidgets documentation or the examples, you will notice that some of the most basic classes are missing in wx, they are directly mapped to corresponding erlang terms:

wxPoint is represented by \{Xcoord,Ycoord\}
wxSize is represented by \{Width,Height\}
xwRect is represented by \{Xcoord,Ycoord,Width,Height\}
xwColour is represented by \{Red,Green,Blue[,Alpha]\}
xwPoint is represented by \{Xcoord,Ycoord\}
xwString is represented by unicode:charlist()
xwGBPosition is represented by \{Row,Column\}
xwGBSpan is represented by \{RowSpan,ColumnSpan\}
xwGridCellCoords is represented by \{Row,Column\}

In the places where the erlang API differs from the original one it should be obvious from the erlang documentation which representation has been used. E.g. the C++ arrays and/or lists are sometimes represented as erlang lists and sometimes as tuples.

Colours are represented with \{Red,Green,Blue[,Alpha]\}, the Alpha value is optional when used as an argument to functions, but it will always be returned from wx functions.

Defines, enumerations and global variables exists in wx.hrl as defines. Most of these defines are constants but not all. Some are platform dependent and therefore the global variables must be instantiated during runtime. These will be acquired from the driver with a call, so not all defines can be used in matching statements. Class local enumerations will be prefixed with the class name and a underscore as in ClassName_Enum.

Additionally some global functions, i.e. non-class functions, exist in the wx_misc module.

wxErlang is implemented as a (threaded) driver and a rather direct interface to the C++ API, with the drawback that if the erlang programmer does an error, it might crash the emulator.

Since the driver is threaded it requires a smp enabled emulator, that provides a thread safe interface to the driver.

### 1.1.3 Multiple processes and memory handling

The intention is that each erlang application calls wx:new() once to setup it's GUI which creates an environment and a memory mapping. To be able to use wx from several processes in your application, you must share the environment. You can get the active environment with wx:get_env/0 and set it in the new processes with wx:set_env/1.

Two processes or applications which have both called wx:new() will not be able use each others objects.

```
xw:new(),
MyWin = wxFrame:new(wx:null(), 42, "Example", []),
Env = wx:get_env(),
spawn(fun() ->
    wx:set_env(Env),
    %% Here you can do wx calls from your helper process.
    ...
end);
...
```

When wx:destroy/0 is invoked or when all processes in the application have died, the memory is deleted and all windows created by that application are closed.
1.1 wx the erlang binding of wxWidgets

The wx application never cleans or garbage collects memory as long as the user application is alive. Most of the objects are deleted when a window is closed, or at least all the objects which have a parent argument that is non null. By using `wxCLASS:destroy/1` when possible you can avoid an increasing memory usage. This is especially important when `wxWidgets` assumes or recommends that you (or rather the C++ programmer) have allocated the object on the stack since that will never be done in the erlang binding. For example `wxDC` class or its sub-classes or `wxSizerFlags`.

Currently the dialogs show modal function freezes `wxWidgets` until the dialog is closed. That is intended but in erlang where you can have several GUI applications running at the same time it causes trouble. This will hopefully be fixed in future `wxWidgets` releases.

1.1.4 Event Handling

Event handling in `wx` differs most from the original API. You must specify every event you want to handle in `wxWidgets`, that is the same in the erlang binding but you can choose to receive the events as messages or handle them with callback funs.

Otherwise the event subscription is handled as `wxWidgets` dynamic event-handler connection. You subscribe to events of a certain type from objects with an ID or within a range of IDs. The callback fun is optional, if not supplied the event will be sent to the process that called `connect/2`. Thus, a handler is a callback fun or a process which will receive an event message.

Events are handled in order from bottom to top, in the widgets hierarchy, by the last subscribed handler first. Depending on if `wxEvent:skip()` is called the event will be handled by the other handler(s) afterwards. Most of the events have default event handler(s) installed.

Message events looks like `#wx{id=integer(), obj=wx:wxObject(), userData=term(), event=Rec}`. The id is the identifier of the object that received the event. The obj field contains the object that you used `connect` on. The userData field contains a user supplied term, this is an option to `connect`. And the event field contains a record with event type dependent information. The first element in the event record is always the type you subscribed to. For example if you subscribed to `key_up` events you will receive the `#wx{event=Event}` where Event will be a `wxKey` event record where `Event#wxKey.type = key_up`.

In `wxWidgets` the developer has to call `wxEvent:skip()` if he wants the event to be processed by other handlers. You can do the same in `wx` if you use callbacks. If you want the event as messages you just don't supply a callback and you can set the skip option in `connect` call to true or false, the default it is false. True means that you get the message but let the subsequent handlers also handle the event. If you want to change this behavior dynamically you must use callbacks and call `wxEvent:skip()`.

Callback event handling is done by using the optional callback fun/2 when attaching the handler. The fun(#wx{},wxObject()) must take two arguments where the first is the same as with message events described above and the second is an object reference to the actual event object. With the event object you can call `wxEvent:skip()` and access all the data. When using callbacks you must call `wxEvent:skip()` by yourself if you want any of the events to be forwarded to the following handlers. The actual event objects are deleted after the fun returns.

The callbacks are always invoked by another process and have exclusive usage of the GUI when invoked. This means that a callback fun cannot use the process dictionary and should not make calls to other processes. Calls to another process inside a callback fun may cause a deadlock if the other process is waiting on completion of his call to the GUI.

1.1.5 Acknowledgments

Mats-Ola Persson wrote the initial `wxWidgets` binding as part of his master thesis. The current version is a total re-write but many ideas have been reused. The reason for the re-write was mostly due to the limited requirements he had been given by us.

Also thanks to the `wxWidgets` team that develops and supports it so we have something to use.
2 Reference Manual

The wxErlang application is an api for writing graphical user interfaces with wxWidgets.
A port of **wxWidgets**.

This is the base api of **wxWidgets**. This module contains functions for starting and stopping the wx-server, as well as other utility functions.

wxWidgets is object oriented, and not functional. Thus, in wxErlang a module represents a class, and the object created by this class has an own type, `wxCLASS()`. This module represents the base class, and all other `wxMODULE`'s are sub-classes of this class.

Objects of a class are created with `wxCLASS:new(...)` and destroyed with `wxCLASS:destroy()`. Member functions are called with `wxCLASS:member(Object, ...)` instead of as in C++ `Object.member(…)`.

Sub class modules inherit (non static) functions from their parents. The inherited functions are not documented in the sub-classes.

This erlang port of wxWidgets tries to be a one-to-one mapping with the original wxWidgets library. Some things are different though, as the optional arguments use property lists and can be in any order. The main difference is the event handling which is different from the original library. See `wxEvtHandler`.

The following classes are implemented directly as erlang types:

-.wxPoint={x,y},wxSize={w,h},wxRect={x,y,w,h},wxColour={r,g,b   [,a]},     wxString=unicode:chardata(),
- wxGBPosition={r,c},wxGBSpan={rs,cs},wxGridCellCoords={r,c}.

wxWidgets uses a process specific environment, which is created by `wx:new/0`. To be able to use the environment from other processes, call `get_env/0` to retrieve the environment and `set_env/1` to assign the environment in the other process.

Global (classless) functions are located in the `wx_misc` module.

**DATA TYPES**

- `wx_colour() = [R::byte(), G::byte(), B::byte()] | wx_colour4()`
- `wx_colour4() = [R::byte(), G::byte(), B::byte(), A::byte()]`
- `wx_datetime() = [{Year::integer(), Month::integer(), Day::integer()}, {Hour::integer(), Minute::integer(), Second::integer()}]`

In Local Timezone

- `wx_enum() = integer()` Constant defined in wx.hrl
- `wx_env() = #wx_env{}` Opaque process environment
- `wx_memory() = binary() | #wx_mem{}` Opaque memory reference
- `wx_object() = #wx_ref{}` Opaque object reference
- `wx_wxHtmlLinkInfo() = #wxHtmlLinkInfo{href=unicode:chardata(), target=unicode:chardata()}`
wx_wxMouseState() = #wxMouseState{x=integer(), y=integer(), leftDown=boolean(), middleDown=boolean(),
rightDown=boolean(), controlDown=boolean(), shiftDown=boolean(), altDown=boolean(), metaDown=boolean(),
cmdDown=boolean()}

See #wxMouseState{} defined in wx.hrl

Exports

parent_class(X1) -> term()

new() -> wx_object()
Starts a wx server.

new(Options::[Option]) -> wx_object()
Types:
    Option = {debug, list() | atom()} | {silent_start, boolean()}
Starts a wx server. Option may be {debug, Level}, see debug/1. Or {silent_start, Bool}, which causes error messages
at startup to be suppressed. The latter can be used as a silent test of whether wx is properly installed or not.

destroy() -> ok
Stop a wx server.

get_env() -> wx_env()
Gets this process's current wx environment. Can be sent to other processes to allow them use this process wx
environment.
See also: set_env/1.

set_env(Wx_env::wx_env()) -> ok
Sets the process wx environment, allows this process to use another process wx environment.

null() -> wx_object()
Returns the null object

is_null(Wx_ref::wx_object()) -> boolean()
Returns true if object is null, false otherwise

equal(Wx_ref::wx_object(), X2::wx_object()) -> boolean()
Returns true if both arguments references the same object, false otherwise

ggetObjectType(Wx_ref::wx_object()) -> atom()
Returns the object type
typeCast(Old::wx_object(), NewType::atom()) -> wx_object()
Casts the object to class NewType. It is needed when using functions like wxWindow:findWindow/2, which returns a generic wxObject type.

batch(Fun::function()) -> term()
Batches all wx commands used in the fun. Improves performance of the command processing by grabbing the wxWidgets thread so that no event processing will be done before the complete batch of commands is invoked.
See also: foldl/3, foldr/3, foreach/2, map/2.

foreach(Fun::function(), List::list()) -> ok
Behaves like lists:foreach/2 but batches wx commands. See batch/1.

map(Fun::function(), List::list()) -> list()
Behaves like lists:map/2 but batches wx commands. See batch/1.

foldl(Fun::function(), Acc::term(), List::list()) -> term()
Behaves like lists:foldl/3 but batches wx commands. See batch/1.

foldr(Fun::function(), Acc::term(), List::list()) -> term()
Behaves like lists:foldr/3 but batches wx commands. See batch/1.

create_memory(Size::integer()) -> wx_memory()
Creates a memory area (of Size in bytes) which can be used by an external library (i.e. opengl). It is up to the client to keep a reference to this object so it does not get garbage collected by erlang while still in use by the external library. This is far from erlang’s intentional usage and can crash the erlang emulator. Use it carefully.

get_memory_bin(Wx_mem::wx_memory()) -> binary()
Returns the memory area as a binary.

retain_memory(Wx_mem::wx_memory()) -> ok
Saves the memory from deletion until release_memory/1 is called. If release_memory/1 is not called the memory will not be garbage collected.

release_memory(Wx_mem::wx_memory()) -> ok

debug(Debug::Level | [Level]) -> ok
Types:
  Level = none | verbose | trace | driver | integer()
Sets debug level. If debug level is 'verbose' or 'trace' each call is printed on console. If Level is 'driver' each allocated object and deletion is printed on the console.

demo() -> ok | {error, atom()}
Starts a wxErlang demo if examples directory exists and is compiled
wx_object

Erlang module

wx_object - Generic wx object behaviour

This is a behaviour module that can be used for "sub classing" wx objects. It works like a regular gen_server module and creates a server per object.

NOTE: Currently no form of inheritance is implemented.

The user module should export:

init(Args) should return
{wxObject, State} | {wxObject, State, Timeout} | ignore | {stop, Reason}

Asynchronous window event handling:

handle_event(#wx{}, State) should return
{noreply, State} | {noreply, State, Timeout} | {stop, Reason, State}

The user module can export the following callback functions:

handle_call(Msg, {From, Tag}, State) should return
{reply, Reply, State} | {reply, Reply, State, Timeout} | {noreply, State} | {noreply, State, Timeout} | {stop, Reason, Reply, State}

handle_cast(Msg, State) should return
{noreply, State} | {noreply, State, Timeout} | {stop, Reason, State}

If the above are not exported but called, the wx_object process will crash. The user module can also export:

Info is message e.g. {'EXIT', P, R}, {nodedown, N}, ...

handle_info(Info, State) should return , ...

{noreply, State} | {noreply, State, Timeout} | {stop, Reason, State}

If a message is sent to the wx_object process when handle_info is not exported, the message will be dropped and ignored.

When stop is returned in one of the functions above with Reason = normal | shutdown | Term, terminate(State) is called. It lets the user module clean up, it is always called when server terminates or when wx_object() in the driver is deleted. If the Parent process terminates the Module:terminate/2 function is called.

terminate(Reason, State)

Example:
wx_object

-module(myDialog).
-export([new/2, show/1, destroy/1]).  %% API
-export([init/1, handle_call/3, handle_event/2, handle_info/2, code_change/3, terminate/2]).
新/2, showModal/1, destroy/1]).  %% Callbacks
%% Client API
new(Parent, Msg) ->
    wx_object:start(?MODULE, [Parent,Id], []).
show(Dialog) ->
    wx_object:call(Dialog, show_modal).
destroy(Dialog) ->
    wx_object:call(Dialog, destroy).
%% Server Implementation ala gen_server
init([Parent, Str]) ->
    Dialog = wxDialog:new(Parent, 42, "Testing", []),
    ...
    wxDialog:connect(Dialog, command_button_clicked),
    {Dialog, MyState}.
handle_call(show, From, State) ->
    wxDialog:show(State#state.win),
    {reply, ok, State};
    ...
handle_event(#wx{}, State) ->
    io:format("Users clicked button~n",[]),
    {noreply, State};
    ...

DATA TYPES
request_id() = term()
server_ref() = wx:wx_object() | atom() | pid()

Exports
start(Name, Mod, Args, Options) -> wxWindow:wxWindow() | {error, term()}
Types:
    Name = {local, atom()}
    Mod = atom()
    Args = term()
    Flag = trace | log | {logfile, string()} | statistics | debug
    Options = [{timeout, timeout()} | {debug, [Flag]}]
Starts a generic wx_object server and invokes Mod:init(Args) in the new process.

start_link(Mod, Args, Options) -> wxWindow:wxWindow() | {error, term()}
Types:
    Mod = atom()
    Args = term()
    Flag = trace | log | {logfile, string()} | statistics | debug
Options = [{timeout, timeout()} | {debug, [Flag]}]
Starts a generic \texttt{wx\_object} server and invokes \texttt{Mod:init(Arg)} in the new process.

\texttt{start\_link(Name, Mod, Args, Options) -> \texttt{wxWindow:wxWindow()}} | \{error, term()\}
Types:
- \texttt{Name = \{local, atom()\}}
- \texttt{Mod = atom()}
- \texttt{Args = term()}
- \texttt{Flag = trace | log | \{logfile, string()\} | statistics | debug}
- \texttt{Options = [{timeout, timeout()} | {debug, [Flag]}]}
Starts a generic \texttt{wx\_object} server and invokes \texttt{Mod:init(Arg)} in the new process.

\texttt{stop(Obj) -> ok}
Types:
- \texttt{Obj = \texttt{wx:wx\_object()}} | atom() | \texttt{pid()}
Stops a generic \texttt{wx\_object} server with reason 'normal'. Invokes \texttt{terminate(Reason,State)} in the server. The call waits until the process is terminated. If the process does not exist, an exception is raised.

\texttt{stop(Obj, Reason, Timeout) -> ok}
Types:
- \texttt{Obj = \texttt{wx:wx\_object()}} | atom() | \texttt{pid()}
- \texttt{Reason = term()}
- \texttt{Timeout = \texttt{timeout()}}
Stops a generic \texttt{wx\_object} server with the given \texttt{Reason}. Invokes \texttt{terminate(Reason,State)} in the server. The call waits until the process is terminated. If the call times out, or if the process does not exist, an exception is raised.

\texttt{call(Obj, Request) -> term()}
Types:
- \texttt{Obj = \texttt{wx:wx\_object()}} | atom() | \texttt{pid()}
- \texttt{Request = term()}
Make a call to a \texttt{wx\_object} server. The call waits until it gets a result. Invokes \texttt{handle\_call(Request, From, State)} in the server.

\texttt{call(Obj, Request, Timeout) -> term()}
Types:
- \texttt{Obj = \texttt{wx:wx\_object()}} | atom() | \texttt{pid()}
- \texttt{Request = term()}
- \texttt{Timeout = \texttt{integer()}}
Make a call to a \texttt{wx\_object} server with a timeout. Invokes \texttt{handle\_call(Request, From, State)} in server.

\texttt{send\_request(Obj, Request::term()) -> request\_id()}
Types:
- \texttt{Obj = \texttt{wx:wx\_object()}} | atom() | \texttt{pid()}

wx_object

Make an send_request to a generic server. and return a RequestId which can/should be used with wait_response/[1|2]. Invokes handle_call(Request, From, State) in server.

wait_response(RequestId::request_id()) -> {reply, Reply::term()} | {error, {term(), server_ref()}}
Wait infinitely for a reply from a generic server.

wait_response(Key::request_id(), Timeout::timeout()) -> {reply, Reply::term()} | timeout | {error, {term(), server_ref()}}
Wait 'timeout' for a reply from a generic server.

check_response(Msg::term(), Key::request_id()) -> {reply, Reply::term()} | false | {error, {term(), server_ref()}}
Check if a received message was a reply to a RequestId

cast(Obj, Request) -> ok
Types:
   Obj = wx:wx_object() | atom() | pid()
   Request = term()
Make a cast to a wx_object server. Invokes handle_cast(Request, State) in the server

get_pid(Obj) -> pid()
Types:
   Obj = wx:wx_object() | atom() | pid()
Get the pid of the object handle.

set_pid(Obj, Pid::pid()) -> wx:wx_object()
Types:
   Obj = wx:wx_object() | atom() | pid()
Sets the controlling process of the object handle.

reply(X1::{pid(), Tag::term()}, Reply::term()) -> pid()
Get the pid of the object handle.
wxAcceleratorEntry

Erlang module

See external documentation: wxAcceleratorEntry.

DATA TYPES

wxAcceleratorEntry()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxAcceleratorEntry()
Equivalent to new([]).

ew(Options::[Option]) -> wxAcceleratorEntry()
Types:

        Option = {flags, integer()} | {keyCode, integer()} | {cmd, integer()} | 
                      {item, wxMenuItem:wxMenuItem()}

See external documentation.
Also:
new(Entry) -> wxAcceleratorEntry() when
Entry::wxAcceleratorEntry().

getCommand(This) -> integer()
Types:

        This = wxAcceleratorEntry()

See external documentation.

getFlags(This) -> integer()
Types:

        This = wxAcceleratorEntry()

See external documentation.

getKeyCode(This) -> integer()
Types:

        This = wxAcceleratorEntry()

See external documentation.

set(This, Flags, KeyCode, Cmd) -> ok
Types:

        This = wxAcceleratorEntry()
wxAcceleratorEntry

Flags = integer()
KeyCode = integer()
Cmd = integer()

Equivalent to \texttt{set(This, Flags, KeyCode, Cmd, []).}

\texttt{set(This, Flags, KeyCode, Cmd, Options::[Option]) -> ok}

Types:
\begin{itemize}
  \item This = \texttt{wxAcceleratorEntry()}
  \item Flags = integer()
  \item KeyCode = integer()
  \item Cmd = integer()
  \item Option = \{item, \texttt{wxMenuItem:wxMenuItem()}\}
\end{itemize}

See \texttt{external documentation}.

\texttt{destroy(This::wxAcceleratorEntry()) -> ok}

Destroys this object, do not use object again
wxAcceleratorTable

Erlang module

See external documentation: `wxAcceleratorTable`.

DATA TYPES

**wxAcceleratorTable()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxAcceleratorTable()
See external documentation.

new(N, Entries) -> wxAcceleratorTable()
Types:

\[ N = \text{integer}\() \]
\[ \text{Entries} = [\text{wxAcceleratorEntry}:\text{wxAcceleratorEntry}()] \]
See external documentation.

ok(This) -> boolean()
Types:

\[ \text{This} = \text{wxAcceleratorTable}\() \]
See external documentation.

destroy(This::wxAcceleratorTable()) -> ok
Destroys this object, do not use object again
wxActivateEvent

Erlang module

See external documentation: 

Use `wxEvtHandler:connect/3` with EventType:

`activate, activate_app, hibernate`

See also the message variant `#wxActivate/1` event record type.

This class is derived (and can use functions) from:

`wxEvent`

DATA TYPES

wxActivateEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`getActive(This) -> boolean()`

Types:

This = wxActivateEvent()  

See external documentation.
wxArtProvider

Erlang module

See external documentation: wxArtProvider.

DATA TYPES

wxArtProvider()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

getBitmap(Id) -> wxBitmap:wxBitmap()
Types:

  \( \text{Id} = \text{unicode:chardata() \times} \)

Equivalent to getBitmap(Id, []).

getBitmap(Id, Options::[Option]) -> wxBitmap:wxBitmap()
Types:

  \( \text{Id} = \text{unicode:chardata()} \times \)
  \( \text{Option} = \{\text{client, unicode:chardata() \times}, \text{size, \{W::integer(), H::integer()\}} \times\} \)

See external documentation.

getIcon(Id) -> wxIcon:wxIcon()
Types:

  \( \text{Id} = \text{unicode:chardata()} \times\)

Equivalent to getIcon(Id, []).

getIcon(Id, Options::[Option]) -> wxIcon:wxIcon()
Types:

  \( \text{Id} = \text{unicode:chardata()} \times \)
  \( \text{Option} = \{\text{client, unicode:chardata() \times}, \text{size, \{W::integer(), H::integer()\}} \times\} \)

See external documentation.
wxAuiDockArt

Erlang module

See external documentation: wxAuiDockArt.

DATA TYPES

wxAuiDockArt()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

colour(This, Id) -> wx:wx_colour4()
Types:
  This = wxAuiDockArt()
  Id = integer()
See external documentation.

getFont(This, Id) -> wxFont:wxFont()
Types:
  This = wxAuiDockArt()
  Id = integer()
See external documentation.

gtMetric(This, Id) -> integer()
Types:
  This = wxAuiDockArt()
  Id = integer()
See external documentation.

setColour(This, Id, Colour) -> ok
Types:
  This = wxAuiDockArt()
  Id = integer()
  Colour = wx:wx_colour()
See external documentation.

setFont(This, Id, Font) -> ok
Types:
  This = wxAuiDockArt()
  Id = integer()
  Font = wxFont:wxFont()
See external documentation.

```erlang
setMetric(This, Id, New_val) -> ok
Types:
    This = wxAuiDockArt()  
    Id = integer()           
    New_val = integer()      
See external documentation.
```
wxAuiManager

Erlang module

See external documentation: wxAuiManager.

This class is derived (and can use functions) from:
wxEvtHandler

DATA TYPES

wxAuiManager()

An object reference. The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxAuiManager()
Equivalent to new([]).

new(Options::[Option]) -> wxAuiManager()
Types:
    Option = {managed_wnd, wxWindow:wxWindow()} | {flags, integer()}
See external documentation.

addPane(This, Window) -> boolean()
Types:
    This = wxAuiManager()
    Window = wxWindow:wxWindow()
Equivalent to addPane(This, Window, []).

addPane(This, Window, Options::[Option]) -> boolean()
Types:
    This = wxAuiManager()
    Window = wxWindow:wxWindow()
    Option = {direction, integer()} | {caption, unicode:chardata()}
See external documentation.
Also:
addPane(This, Window, Pane_info) -> boolean() when
This::wxAuiManager(), Window::wxWindow:wxWindow(), Pane_info::wxAuiPaneInfo:wxAuiPaneInfo().

addPane(This, Window, Pane_info, Drop_pos) -> boolean()
Types:
    This = wxAuiManager()
    Window = wxWindow:wxWindow()
Pane_info = wxAuiPaneInfo:wxAuiPaneInfo()
Drop_pos = {X::integer(), Y::integer()}

See external documentation.

detachPane(This, Window) -> boolean()

Types:
  This = wxAuiManager()
  Window = wxWindow:wxWindow()

See external documentation.

getAllPanes(This) -> [wxAuiPaneInfo:wxAuiPaneInfo()]

Types:
  This = wxAuiManager()

See external documentation.

getArtProvider(This) -> wxAuiDockArt:wxAuiDockArt()

Types:
  This = wxAuiManager()

See external documentation.

getDockSizeConstraint(This) -> {Width_pct::number(), Height_pct::number()}

Types:
  This = wxAuiManager()

See external documentation.

getFlags(This) -> integer()

Types:
  This = wxAuiManager()

See external documentation.

getManagedWindow(This) -> wxWindow:wxWindow()

Types:
  This = wxAuiManager()

See external documentation.

getManager(Window) -> wxAuiManager()

Types:
  Window = wxWindow:wxWindow()

See external documentation.

getPane(This, Name) -> wxAuiPaneInfo:wxAuiPaneInfo()

Types:
  This = wxAuiManager()
**wxAuiManager**

- **Name** = `unicode:chardata()`

  See external documentation.

  Also:
  
  `getPane(This, Window) -> wxAuiPaneInfo:wxAuiPaneInfo()`
  
  `This::wxAuiManager(), Window::wxWindow::wxWindow().`

  `hideHint(This) -> ok`

  Types:
  
  ```
  This = wxAuiManager()
  ```

  See external documentation.

  `insertPane(This, Window, Insert_location) -> boolean()`

  Types:
  
  ```
  This = wxAuiManager()
  Window = wxWindow::wxWindow()
  Insert_location = wxAuiPaneInfo:wxAuiPaneInfo()
  ```

  Equivalent to `insertPane(This, Window, Insert_location, []).`

  `insertPane(This, Window, Insert_location, Options::[Option]) -> boolean()`

  Types:
  
  ```
  This = wxAuiManager()
  Window = wxWindow::wxWindow()
  Insert_location = wxAuiPaneInfo:wxAuiPaneInfo()
  Option = {insert_level, integer()}
  ```

  See external documentation.

  `loadPaneInfo(This, Pane_part, Pane) -> ok`

  Types:
  
  ```
  This = wxAuiManager()
  Pane_part = unicode:chardata()
  Pane = wxAuiPaneInfo:wxAuiPaneInfo()
  ```

  See external documentation.

  `loadPerspective(This, Perspective) -> boolean()`

  Types:
  
  ```
  This = wxAuiManager()
  Perspective = unicode:chardata()
  ```

  Equivalent to `loadPerspective(This, Perspective, [])`.

  `loadPerspective(This, Perspective, Options::[Option]) -> boolean()`

  Types:
  
  ```
  This = wxAuiManager()
  Perspective = unicode:chardata()
  ```
Option = {update, boolean()}
See external documentation.

savePaneInfo(This, Pane) -> unicode:charlist()
Types:
This = wxAuiManager()
Pane = wxAuiPaneInfo:wxAuiPaneInfo()
See external documentation.

savePerspective(This) -> unicode:charlist()
Types:
This = wxAuiManager()
See external documentation.

setArtProvider(This, Art_provider) -> ok
Types:
This = wxAuiManager()
Art_provider = wxAuiDockArt:wxAuiDockArt()
See external documentation.

setDockSizeConstraint(This, Width_pct, Height_pct) -> ok
Types:
This = wxAuiManager()
Width_pct = number()
Height_pct = number()
See external documentation.

setFlags(This, Flags) -> ok
Types:
This = wxAuiManager()
Flags = integer()
See external documentation.

setManagedWindow(This, Managed_wnd) -> ok
Types:
This = wxAuiManager()
Managed_wnd = wxWindow:wxWindow()
See external documentation.

showHint(This, Rect) -> ok
Types:
This = wxAuiManager()
Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
wxAuiManager

See external documentation.

unInit(This) -> ok
Types:
    This = wxAuiManager()
See external documentation.

update(This) -> ok
Types:
    This = wxAuiManager()
See external documentation.

destroy(This::wxAuiManager()) -> ok
Destroys this object, do not use object again
wxAuiManagerEvent

Erlang module

See external documentation: wxAuiManagerEvent.

Use wxEvtHandler:connect/3 with EventType:

    aui_pane_button, aui_pane_close, aui_pane_maximize, aui_pane_restore, aui_pane_activated,
    aui_render, aui_find_manager

See also the message variant #wxAuiManager/1 event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxAuiManagerEvent()

    An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

setManager(This, Mgr) -> ok

Types:

    This = wxAuiManagerEvent()
    Mgr = wxAuiManager:wxAuiManager()

See external documentation.

getManager(This) -> wxAuiManager:wxAuiManager()

Types:

    This = wxAuiManagerEvent()

See external documentation.

setPane(This, P) -> ok

Types:

    This = wxAuiManagerEvent()
    P = wxAuiPaneInfo:wxAuiPaneInfo()

See external documentation.

getPane(This) -> wxAuiPaneInfo:wxAuiPaneInfo()

Types:

    This = wxAuiManagerEvent()

See external documentation.
wxAuiManagerEvent

setButton(This, B) -> ok
Types:
  This = wxAuiManagerEvent()
  B = integer()
See external documentation.

getButton(This) -> integer()
Types:
  This = wxAuiManagerEvent()
See external documentation.

setDC(This, Pdc) -> ok
Types:
  This = wxAuiManagerEvent()
  Pdc = wxDC:wxDC()
See external documentation.

getDC(This) -> wxDC:wxDC()
Types:
  This = wxAuiManagerEvent()
See external documentation.

veto(This) -> ok
Types:
  This = wxAuiManagerEvent()
Equivalent to veto(This, []).

veto(This, Options::[Option]) -> ok
Types:
  This = wxAuiManagerEvent()
  Option = {veto, boolean()}
See external documentation.

getVeto(This) -> boolean()
Types:
  This = wxAuiManagerEvent()
See external documentation.

setCanVeto(This, Can_veto) -> ok
Types:
  This = wxAuiManagerEvent()
  Can_veto = boolean()
See external documentation.
canVeto(This) -> boolean()

Types:

    This = wxAuiManagerEvent()

See external documentation.
wxAuiNotebook

Erlang module

See external documentation: wxAuiNotebook.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxAuiNotebook()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxAuiNotebook()
See external documentation.

ew(Parent) -> wxAuiNotebook()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxAuiNotebook()
Types:
    Parent = wxWindow:wxWindow()
    Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

addPage(This, Page, Caption) -> boolean()
Types:
    This = wxAuiNotebook()
    Page = wxWindow:wxWindow()
    Caption = unicode:chardata()
Equivalent to addPage(This, Page, Caption, []).

addPage(This, Page, Caption, Options::[Option]) -> boolean()
Types:
    This = wxAuiNotebook()
    Page = wxWindow:wxWindow()
Caption = unicode:chardata()
Option = {select, boolean()} | {bitmap, wxBitmap:wxBitmap()}

See external documentation.

create(This, Parent) -> boolean()
Types:
   This = wxAuiNotebook()
   Parent = wxWindow:wxWindow()
Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()
Types:
   This = wxAuiNotebook()
   Parent = wxWindow:wxWindow()
   Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

deletePage(This, Page) -> boolean()
Types:
   This = wxAuiNotebook()
   Page = integer()
See external documentation.

getArtProvider(This) -> wxAuiTabArt:wxAuiTabArt()
Types:
   This = wxAuiNotebook()
See external documentation.

getPage(This, Page_idx) -> wxWindow:wxWindow()
Types:
   This = wxAuiNotebook()
   Page_idx = integer()
See external documentation.

getPageBitmap(This, Page_idx) -> wxBitmap:wxBitmap()
Types:
   This = wxAuiNotebook()
   Page_idx = integer()
See external documentation.

getPageCount(This) -> integer()
Types:
This = wxAuiNotebook()
See external documentation.

getPageIndex(This, Page_wnd) -> integer()
Types:
   This = wxAuiNotebook()
   Page_wnd = wxWindow:wxWindow()
See external documentation.

getPageText(This, Page_idx) -> unicode:charlist()
Types:
   This = wxAuiNotebook()
   Page_idx = integer()
See external documentation.

getSelection(This) -> integer()
Types:
   This = wxAuiNotebook()
See external documentation.

insertPage(This, Page_idx, Page, Caption) -> boolean()
Types:
   This = wxAuiNotebook()
   Page_idx = integer()
   Page = wxWindow:wxWindow()
   Caption = unicode:chardata()
Equivalent to insertPage(This, Page_idx, Page, Caption, []).

insertPage(This, Page_idx, Page, Caption, Options::[Option]) -> boolean()
Types:
   This = wxAuiNotebook()
   Page_idx = integer()
   Page = wxWindow:wxWindow()
   Caption = unicode:chardata()
   Option = {select, boolean()} | {bitmap, wxBitmap:wxBitmap()}
See external documentation.

removePage(This, Page) -> boolean()
Types:
   This = wxAuiNotebook()
   Page = integer()
See external documentation.
setArtProvider(This, Art) -> ok
Types:
   This = wxAuiNotebook()
   Art = wxAuiTabArt:wxAuiTabArt()
See external documentation.

setFont(This, Font) -> boolean()
Types:
   This = wxAuiNotebook()
   Font = wxFont:wxFont()
See external documentation.

setPageBitmap(This, Page, Bitmap) -> boolean()
Types:
   This = wxAuiNotebook()
   Page = integer()
   Bitmap = wxBitmap:wxBitmap()
See external documentation.

setPageText(This, Page, Text) -> boolean()
Types:
   This = wxAuiNotebook()
   Page = integer()
   Text = unicode:chardata()
See external documentation.

setSelection(This, New_page) -> integer()
Types:
   This = wxAuiNotebook()
   New_page = integer()
See external documentation.

setTabCtrlHeight(This, Height) -> ok
Types:
   This = wxAuiNotebook()
   Height = integer()
See external documentation.

setUniformBitmapSize(This, Size) -> ok
Types:
   This = wxAuiNotebook()
   Size = {W::integer(), H::integer()}
See external documentation.
wxAuiNotebook

destroy(This::wxAuiNotebook()) -> ok
Destroys this object, do not use object again
wxAuiNotebookEvent

Erlang module

See external documentation: wxAuiNotebookEvent.

Use wxEvtHandler:connect/3 with EventType:

- command_aui_notebook_page_close,
- command_aui_notebook_page_changing,
- command_aui_notebook_begin_drag,
- command_aui_notebook_drag_motion,
- command_aui_notebook_tab_middle_down,
- command_aui_notebook_tab_right_down,
- command_aui_notebook_page_closed,
- command_aui_notebook_bg_dclick

See also the message variant #wxAuiNotebook[] event record type.

This class is derived (and can use functions) from:
wxNotifyEvent
wxCommandEvent
wxEvent

DATA TYPES

wxAuiNotebookEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

setSelection(This, S) -> ok
Types:
  This = wxAuiNotebookEvent()
  S = integer()

getSelection(This) -> integer()
Types:
  This = wxAuiNotebookEvent()

setOldSelection(This, S) -> ok
Types:
  This = wxAuiNotebookEvent()
  S = integer()

See external documentation.
wxAuiNotebookEvent

getOldSelection(This) -> integer()
Types:
    This = wxAuiNotebookEvent()
See external documentation.

setDragSource(This, S) -> ok
Types:
    This = wxAuiNotebookEvent()
    S = wxAuiNotebook:wxAuiNotebook()
See external documentation.

getDragSource(This) -> wxAuiNotebook:wxAuiNotebook()
Types:
    This = wxAuiNotebookEvent()
See external documentation.
wxAuiPaneInfo

Erlang module

See external documentation: wxAuiPaneInfo.

DATA TYPES

wxAuiPaneInfo()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxAuiPaneInfo()
See external documentation.

new(C) -> wxAuiPaneInfo()
Types:
  C = wxAuiPaneInfo()
See external documentation.

bestSize(This, Size) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Size = {W::integer(), H::integer()}
See external documentation.

bestSize(This, X, Y) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  X = integer()
  Y = integer()
See external documentation.

bottom(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
See external documentation.

bottomDockable(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
Equivalent to bottomDockable(This, []).
wxAuiPaneInfo

bottomDockable(This, Options:::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {b, boolean()}
See external documentation.

caption(This, C) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  C = unicode:chardata()
See external documentation.

captionVisible(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
Equivalent to captionVisible(This, []).

captionVisible(This, Options:::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {visible, boolean()}
See external documentation.

centre(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
See external documentation.

centrePane(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
See external documentation.

closeButton(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
Equivalent to closeButton(This, []).

closeButton(This, Options:::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {visible, boolean()}
See external documentation.
defaultPane(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.

destroyOnClose(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to destroyOnClose(This, []).

destroyOnClose(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {b, boolean()}
See external documentation.

direction(This, Direction) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Direction = integer()
See external documentation.

dock(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.

dockable(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to dockable(This, []).

dockable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {b, boolean()}
See external documentation.

fixed(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.
wxAuiPaneInfo

float(This) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
See external documentation.

floatable(This) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
Equivalent to floatable(This, []).

floatable(This, Options::[Option]) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
        Option = {b, boolean()}
See external documentation.

floatingPosition(This, Pos) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
        Pos = {X::integer(), Y::integer()}
See external documentation.

floatingPosition(This, X, Y) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
        X = integer()   
        Y = integer()   
See external documentation.

floatingSize(This, Size) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
        Size = {W::integer(), H::integer()}
See external documentation.

floatingSize(This, X, Y) -> wxAuiPaneInfo()  
Types:   
        This = wxAuiPaneInfo()   
        X = integer()   
        Y = integer()   
See external documentation.
gripper(This) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
Equivalent to gripper(This, []).

gripper(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    Option = {visible, boolean()}
See external documentation.

gripperTop(This) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
Equivalent to gripperTop(This, []).

gripperTop(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    Option = {attop, boolean()}
See external documentation.

hasBorder(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

hasCaption(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

hasCloseButton(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

hasFlag(This, Flag) -> boolean()
Types:
    This = wxAuiPaneInfo()
    Flag = integer()
See external documentation.
hasGripper(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

hasGripperTop(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

hasMaximizeButton(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

hasMinimizeButton(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

hasPinButton(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

hide(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.

isBottomDockable(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

isDocked(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

isFixed(This) -> boolean()
Types:
   This = wxAuiPaneInfo()
See external documentation.

isFloatable(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isFloating(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isLeftDockable(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isMovable(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isOk(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isResizable(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isRightDockable(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.

isShown(This) -> boolean()
Types:
    This = wxAuiPaneInfo()
See external documentation.
isToolbar(This) -> boolean()
Types:
  This = wxAuiPaneInfo()
See external documentation.

isTopDockable(This) -> boolean()
Types:
  This = wxAuiPaneInfo()
See external documentation.

layer(This, Layer) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Layer = integer()
See external documentation.

left(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
See external documentation.

leftDockable(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
Equivalent to leftDockable(This, []).

leftDockable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {b, boolean()}
See external documentation.

maxSize(This, Size) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Size = {W::integer(), H::integer()}
See external documentation.

maxSize(This, X, Y) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  X = integer()
  Y = integer()
See external documentation.

maximizeButton(This) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
Equivalent to maximizeButton(This, []).

maximizeButton(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    Option = {visible, boolean()}
See external documentation.

minSize(This, Size) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    Size = {W::integer(), H::integer()}
See external documentation.

minSize(This, X, Y) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    X = integer()
    Y = integer()
See external documentation.

minimizeButton(This) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
Equivalent to minimizeButton(This, []).

minimizeButton(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
    Option = {visible, boolean()}
See external documentation.

movable(This) -> wxAuiPaneInfo()
Types:
    This = wxAuiPaneInfo()
Equivalent to movable(This, []).
**wxAuiPaneInfo**

movable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = (b, boolean())

See external documentation.

name(This, N) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   N = unicode:chardata()

See external documentation.

paneBorder(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to paneBorder(This, []).

paneBorder(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {visible, boolean()}

See external documentation.

pinButton(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to pinButton(This, []).

pinButton(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {visible, boolean()}

See external documentation.

position(This, Pos) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Pos = integer()

See external documentation.

resizable(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to `resizable(This, [])`.

```erlang
resizable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {resizable, boolean()}
See external documentation.
```

```erlang
right(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
See external documentation.
```

```erlang
rightDockable(This) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
Equivalent to `rightDockable(This, [])`.
```

```erlang
rightDockable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Option = {b, boolean()}
See external documentation.
```

```erlang
row(This, Row) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Row = integer()
See external documentation.
```

```erlang
safeSet(This, Source) -> ok
Types:
  This = wxAuiPaneInfo()
  Source = wxAuiPaneInfo()
See external documentation.
```

```erlang
setFlag(This, Flag, Option_state) -> wxAuiPaneInfo()
Types:
  This = wxAuiPaneInfo()
  Flag = integer()
  Option_state = boolean()
See external documentation.
```
show(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to show(This, []).

show(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {show, boolean()}
See external documentation.

toolbarPane(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.

top(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
See external documentation.

topDockable(This) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
Equivalent to topDockable(This, []).

topDockable(This, Options::[Option]) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   Option = {b, boolean()}
See external documentation.

window(This, W) -> wxAuiPaneInfo()
Types:
   This = wxAuiPaneInfo()
   W = wxWindow:wxWindow()
See external documentation.

getWindow(This) -> wxWindow:wxWindow()
Types:
   This = wxAuiPaneInfo()
See external documentation.
getFrame(This) -> wxFrame:wxFrame()
Types:
   This = wxAuiPaneInfo()
See external documentation.

getDirection(This) -> integer()
Types:
   This = wxAuiPaneInfo()
See external documentation.

getLayer(This) -> integer()
Types:
   This = wxAuiPaneInfo()
See external documentation.

getRow(This) -> integer()
Types:
   This = wxAuiPaneInfo()
See external documentation.

getPosition(This) -> integer()
Types:
   This = wxAuiPaneInfo()
See external documentation.

getFloatingPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxAuiPaneInfo()
See external documentation.

getFloatingSize(This) -> {W::integer(), H::integer()}
Types:
   This = wxAuiPaneInfo()
See external documentation.

destroy(This::wxAuiPaneInfo()) -> ok
Destroys this object, do not use object again
wxAuiSimpleTabArt

Erlang module

See external documentation: *wxAuiSimpleTabArt*.

This class is derived (and can use functions) from:
*wxAuiTabArt*

**DATA TYPES**

wxAuiSimpleTabArt()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxAuiSimpleTabArt()

See external documentation.

destroy(This::wxAuiSimpleTabArt()) -> ok

Destroys this object, do not use object again
wxAuiTabArt

Erlang module

See external documentation: wxAuiTabArt.

DATA TYPES

wxAuiTabArt()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

setFlags(This, Flags) -> ok
Types:
  This = wxAuiTabArt()
  Flags = integer()
See external documentation.

setMeasuringFont(This, Font) -> ok
Types:
  This = wxAuiTabArt()
  Font = wxFont:wxFont()
See external documentation.

setNormalFont(This, Font) -> ok
Types:
  This = wxAuiTabArt()
  Font = wxFont:wxFont()
See external documentation.

setSelectedFont(This, Font) -> ok
Types:
  This = wxAuiTabArt()
  Font = wxFont:wxFont()
See external documentation.

setColour(This, Colour) -> ok
Types:
  This = wxAuiTabArt()
  Colour = wx:wx_colour()
See external documentation.
setActiveColour(This, Colour) -> ok

Types:

This = wxUiTabArt()

Colour = wx:wx_colour()

See external documentation.
wxBitmap

Erlang module

See external documentation: wxBitmap.

DATA TYPES

wxBitmap()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxBitmap()
See external documentation.

new(Filename) -> wxBitmap()
Types:
Filename = unicode:chardata()
See external documentation.
Also:
new(Image) -> wxBitmap() when
Image::wxImage::wxImage().

new(Width, Height) -> wxBitmap()
Types:
Width = integer()
Height = integer()
See external documentation.
Also:
new(Filename, [Option]) -> wxBitmap() when
Filename::unicode:chardata(),
Option :: [{type, wx:wx_enum()};
(Image, [Option]) -> wxBitmap() when
wxBitmap

Image::wxImage:wxImage(),
Option :: {'depth', integer()}.  
Type = ?wxBITMAP_TYPE_INVALID | ?wxBITMAP_TYPE_BMP | ?wxBITMAP_TYPE_BMP_RESOURCE |
  ?wxBITMAP_TYPE_RESOURCE | ?wxBITMAP_TYPE_ICO | ?wxBITMAP_TYPE_ICO_RESOURCE |
  ?wxBITMAP_TYPE_CUR | ?wxBITMAP_TYPE_CUR_RESOURCE | ?wxBITMAP_TYPE_XBM |
  ?wxBITMAP_TYPE_XBM_DATA | ?wxBITMAP_TYPE_XPM | ?wxBITMAP_TYPE_XPM_DATA |
  ?wxBITMAP_TYPE_TIF | ?wxBITMAP_TYPE_TIFFRESOURCE | ?wxBITMAP_TYPE_GIF |
  ?wxBITMAP_TYPE_GIFRESOURCE | ?wxBITMAP_TYPE_PNG | ?wxBITMAP_TYPE_PNGRESOURCE |
  ?wxBITMAP_TYPE_JPEG | ?wxBITMAP_TYPE_JPEGRSOURCE | ?wxBITMAP_TYPE_PNM|
  ?wxBITMAP_TYPE_PNMRSCOURCE | ?wxBITMAP_TYPE_PCX | ?wxBITMAP_TYPE_PCXRESOURCE |
  ?wxBITMAP_TYPE_PICT | ?wxBITMAP_TYPE_PICTRESOURCE | ?wxBITMAP_TYPE_ICON |
  ?wxBITMAP_TYPE_ICONRESOURCE | ?wxBITMAP_TYPE_ANI | ?wxBITMAP_TYPE_IFF |
  ?wxBITMAP_TYPE_TGA | ?wxBITMAP_TYPE_MACCURSOR |
  ?wxBITMAP_TYPE_MACCURSORRESOURCE | ?wxBITMAP_TYPE_ANY

new(Bits, Width, Height) -> wxBitmap()
Types:
  Bits = binary()
  Width = integer()
  Height = integer()
See external documentation.
Also:
new(Width, Height, [Option]) -> wxBitmap() when
Width::integer(), Height::integer(),
Option :: {'depth', integer()}.  
new(Bits, Width, Height, Options::[Option] ) -> wxBitmap()
Types:
  Bits = binary()
  Width = integer()
  Height = integer()
  Option = {depth, integer()}

See external documentation.

convertToImage(This) -> wxImage:wxImage()
Types:
  This = wxBitmap()
See external documentation.

copyFromIcon(This, Icon) -> boolean()
Types:
  This = wxBitmap()
  Icon = wxIcon:wxIcon()
See external documentation.
create(This, Width, Height) -> boolean()
Types:
  This = wxBitmap()
  Width = integer()
  Height = integer()
Equivalent to create(This, Width, Height, []).

create(This, Width, Height, Options::[Option]) -> boolean()
Types:
  This = wxBitmap()
  Width = integer()
  Height = integer()
  Option = {depth, integer()}
See external documentation.

getDepth(This) -> integer()
Types:
  This = wxBitmap()
See external documentation.

ggetHeight(This) -> integer()
Types:
  This = wxBitmap()
See external documentation.

getPalette(This) -> wxPalette:wxPalette()
Types:
  This = wxBitmap()
See external documentation.

getMask(This) -> wxMask:wxMask()
Types:
  This = wxBitmap()
See external documentation.

getWidth(This) -> integer()
Types:
  This = wxBitmap()
See external documentation.

getSubBitmap(This, Rect) -> wxBitmap()
Types:
  This = wxBitmap()
Rect = \{(X::integer(), Y::integer(), W::integer(), H::integer())\}

See external documentation.

`loadFile(This, Name) -> boolean()`

Types:
- `This = wxBitmap()`
- `Name = unicode:chardata()`

Equivalent to `loadFile(This, Name, [])`.

`loadFile(This, Name, Options::[Option]) -> boolean()`

Types:
- `This = wxBitmap()`
- `Name = unicode:chardata()`
- `Option = \{type, wx:wx_enum()\}`

See external documentation.


`ok(This) -> boolean()`

Types:
- `This = wxBitmap()`

See external documentation.

`saveFile(This, Name, Type) -> boolean()`

Types:
- `This = wxBitmap()`
- `Name = unicode:chardata()`
- `Type = wx:wx_enum()`

Equivalent to `saveFile(This, Name, Type, [])`.

`saveFile(This, Name, Type, Options::[Option]) -> boolean()`

Types:
- `This = wxBitmap()`
- `Name = unicode:chardata()`
- `Type = wx:wx_enum()`
Option = {palette, wxPalette:wxPalette()}

See external documentation.
Type = ?wxBITMAP_TYPE_INVALID | ?wxBITMAP_TYPE_BMP | ?wxBITMAP_TYPE_BMP_RESOURCE |
| ?wxBITMAP_TYPERESOURCE | ?wxBITMAP_TYPE ICO | ?wxBITMAP_TYPE ICO_RESOURCE |
| ?wxBITMAP_TYPE_CUR | ?wxBITMAP_TYPE_CUR_RESOURCE | ?wxBITMAP_TYPE_XBM | |
wxBITMAP_TYPE_XBM_DATA | ?wxBITMAP_TYPE_XPM | ?wxBITMAP_TYPE_XPM DATA | |
wxBITMAP_TYPE_TIF | ?wxBITMAP_TYPE_TIF_RESOURCE | ?wxBITMAP_TYPE_GIF | |
wxBITMAP_TYPE_GIF_RESOURCE | ?wxBITMAP_TYPE_PNG | ?wxBITMAP_TYPE_PNG_RESOURCE |
| ?wxBITMAP_TYPE_JPEG | ?wxBITMAP_TYPE_JPEG_RESOURCE | ?wxBITMAP_TYPE_PNM | |
wxBITMAP_TYPE_PNM_RESOURCE | ?wxBITMAP_TYPE_PCX | ?wxBITMAP_TYPE_PCX_RESOURCE |
| ?wxBITMAP_TYPE_PICT | ?wxBITMAP_TYPE_PICT_RESOURCE | ?wxBITMAP_TYPE_ICON | |
wxBITMAP_TYPE_ICONRESOURCE | ?wxBITMAP_TYPE_ANI | ?wxBITMAP_TYPE_IFF | |
wxBITMAP_TYPE_TGA | | ?wxBITMAP_TYPE_MACCURSOR | |
wxBITMAP_TYPE_MACCURSOR_RESOURCE | ?wxBITMAP_TYPE_ANY

setDepth(This, Depth) -> ok
Types:
   This = wxBitmap()
   Depth = integer()
See external documentation.

setHeight(This, Height) -> ok
Types:
   This = wxBitmap()
   Height = integer()
See external documentation.

setMask(This, Mask) -> ok
Types:
   This = wxBitmap()
   Mask = wxMask:wxMask()
See external documentation.

setPalette(This, Palette) -> ok
Types:
   This = wxBitmap()
   Palette = wxPalette:wxPalette()
See external documentation.

setWidth(This, Width) -> ok
Types:
   This = wxBitmap()
   Width = integer()
See external documentation.
**wxBitmap**

`destroy(This::wxBitmap()) -> ok`

Destroys this object, do not use object again
wxBitmapButton

Erlang module

See external documentation: wxBitmapButton.
This class is derived (and can use functions) from:
wxButton
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxBitmapButton()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxBitmapButton()
See external documentation.

new(Parent, Id, Bitmap) -> wxBitmapButton()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Bitmap = wxBitmap:wxBitmap()

Equivalent to new(Parent, Id, Bitmap, []).

new(Parent, Id, Bitmap, Options::[Option]) -> wxBitmapButton()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Bitmap = wxBitmap:wxBitmap()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Bitmap) -> boolean()

Types:
  This = wxBitmapButton()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Bitmap = wxBitmap:wxBitmap()
wxBitmapButton

Equivalent to `create(This, Parent, Id, Bitmap, []).`

`create(This, Parent, Id, Bitmap, Options::[Option]) -> boolean()`

Types:
- `This = wxBitmapButton()`
- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Bitmap = wxBitmap:wxBitmap()`
- `Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}`

See external documentation.

`getBitmapDisabled(This) -> wxBitmap:wxBitmap()`

Types:
- `This = wxBitmapButton()`

See external documentation.

`getBitmapFocus(This) -> wxBitmap:wxBitmap()`

Types:
- `This = wxBitmapButton()`

See external documentation.

`getBitmapLabel(This) -> wxBitmap:wxBitmap()`

Types:
- `This = wxBitmapButton()`

See external documentation.

`getBitmapSelected(This) -> wxBitmap:wxBitmap()`

Types:
- `This = wxBitmapButton()`

See external documentation.

`setBitmapDisabled(This, Disabled) -> ok`

Types:
- `This = wxBitmapButton()`
- `Disabled = wxBitmap:wxBitmap()`

See external documentation.

`setBitmapFocus(This, Focus) -> ok`

Types:
- `This = wxBitmapButton()`
- `Focus = wxBitmap:wxBitmap()`

See external documentation.
setBitmapLabel(This, Bitmap) -> ok
Types:
   This = wxBitmapButton()
   Bitmap = wxBitmap:wxBitmap()
See external documentation.

setBitmapSelected(This, Sel) -> ok
Types:
   This = wxBitmapButton()
   Sel = wxBitmap:wxBitmap()
See external documentation.

destroy(This::wxBitmapButton()) -> ok
Destroys this object, do not use object again
wxBitmapDataObject

Erlang module

See external documentation: `wxBitmapDataObject`.

This class is derived (and can use functions) from:

`wxDataObject`

**DATA TYPES**

`wxBitmapDataObject()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxBitmapDataObject()`

Equivalent to `new([])`.

`new(Options::[Option]) -> wxBitmapDataObject()`

Types:

```
Option = {bitmap, wxBitmap:wxBitmap()}
```

See external documentation.

Also:

`new(Bitmap) -> wxBitmapDataObject()` when `Bitmap::wxBitmap:wxBitmap()`.

`getBitmap(This) -> wxBitmap:wxBitmap()`

Types:

```
This = wxBitmapDataObject()
```

See external documentation.

`setBitmap(This, Bitmap) -> ok`

Types:

```
This = wxBitmapDataObject()
Bitmap = wxBitmap:wxBitmap()
```

See external documentation.

`destroy(This::wxBitmapDataObject()) -> ok`

Destroys this object, do not use object again.
wxBoxSizer

Erlang module

See external documentation: `wxBoxSizer`.

This class is derived (and can use functions) from: `wxSizer`

**DATA TYPES**

`wxBoxSizer()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new(Orient) -> wxBoxSizer()`

Types:

`Orient = integer()`

See external documentation.

`getOrientation(This) -> integer()`

Types:

`This = wxBoxSizer()`

See external documentation.

`destroy(This::wxBoxSizer()) -> ok`

Destroys this object, do not use object again
wxBrush

Erlang module

See external documentation: *wxBrush*.

**DATA TYPES**

*wxBrush()*

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

*new()* -> *wxBrush()*

See external documentation.

*new(Colour)* -> *wxBrush()*

Types:

```
Colour = wx:wx_colour()
```

See external documentation.

Also:

new(StippleBitmap) -> wxBrush() when StippleBitmap::wxBitmap:wxBitmap().

*new(Colour, Options::[Option])* -> *wxBrush()*

Types:

```
Colour = wx:wx_colour()
Option = {style, integer()}
```

See external documentation.

*getColour(This)* -> *wx:wx_colour4()*

Types:

```
This = wxBrush()
```

See external documentation.

*getStipple(This)* -> *wxBitmap:wxBitmap()*

Types:

```
This = wxBrush()
```

See external documentation.

*getStyle(This)* -> *integer()*

Types:

```
This = wxBrush()
```
See external documentation.

`isHatch(This) -> boolean()`
Types:

- `This = wxBrush()`

See external documentation.

`isOk(This) -> boolean()`
Types:

- `This = wxBrush()`

See external documentation.

`setColour(This, Col) -> ok`
Types:

- `This = wxBrush()`
- `Col = wx:wx_colour()`

See external documentation.

`setColour(This, R, G, B) -> ok`
Types:

- `This = wxBrush()`
- `R = integer()`
- `G = integer()`
- `B = integer()`

See external documentation.

`setStipple(This, Stipple) -> ok`
Types:

- `This = wxBrush()`
- `Stipple = wxBitmap:wxBitmap()`

See external documentation.

`setStyle(This, Style) -> ok`
Types:

- `This = wxBrush()`
- `Style = integer()`

See external documentation.

`destroy(This::wxBrush()) -> ok`
Destroys this object, do not use object again
wxBufferedDC

wxBufferedDC

Erlang module

See external documentation: `wxBufferedDC`.

This class is derived (and can use functions) from:

- `wxMemoryDC`
- `wxDC`

**DATA TYPES**

- `wxBufferedDC()`
  
  An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

- `new() -> wxBufferedDC()`
  
  See external documentation.

- `new(Dc) -> wxBufferedDC()`
  
  Types:
  
  - `Dc = wxDC:wxDC()`
  
  Equivalent to `new(Dc, [])`.

- `new(Dc, Area) -> wxBufferedDC()`
  
  Types:
  
  - `Dc = wxDC:wxDC()`
  - `Area = {W::integer(), H::integer()}`
  
  See external documentation.

  Also:

  `new(Dc, [Option]) -> wxBufferedDC() when Dc::wxDC:wxDC(), Option :: [buffer, wxBitmap:wxBitmap()] | [style, integer()].`

- `new(Dc, Area, Options::[Option]) -> wxBufferedDC()`
  
  Types:
  
  - `Dc = wxDC:wxDC()`
  - `Area = {W::integer(), H::integer()}`
  - `Option = {style, integer()}`
  
  See external documentation.

- `init(This, Dc) -> ok`
  
  Types:
This = wxBufferedDC()
Dc = wxDC:wxDC()

Equivalent to init(This, Dc, []).

init(This, Dc, Area) -> ok
Types:
    This = wxBufferedDC()
    Dc = wxDC:wxDC()
    Area = {W::integer(), H::integer()}

See external documentation.

Also:
init(This, Dc, [Option]) -> 'ok' when
This::wxBufferedDC(), Dc::wxDC:wxDC(),
Option :: [{'buffer', wxBitmap:wxBitmap()} | {'style', integer()}].

init(This, Dc, Area, Options::[Option]) -> ok
Types:
    This = wxBufferedDC()
    Dc = wxDC:wxDC()
    Area = {W::integer(), H::integer()}
    Option = {style, integer()}

See external documentation.

destroy(This::wxBufferedDC()) -> ok
Destroys this object, do not use object again
wxBufferedPaintDC

Erlang module

See external documentation: wxBufferedPaintDC.

This class is derived (and can use functions) from:
wxBufferedDC
wxMemoryDC
wxDC

DATA TYPES

wxBufferedPaintDC()

An object reference, The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new(Window) -> wxBufferedPaintDC()
Types:

Window = wxWindow:wxWindow()
Equivalent to new(Window, []).

new(Window, Buffer) -> wxBufferedPaintDC()
Types:

Window = wxWindow:wxWindow()
Buffer = wxBitmap:wxBitmap()
See external documentation.
Also:
new(Window, [Option]) -> wxBufferedPaintDC() when
Window::wxWindow:wxWindow(),
Option :: {'style', integer()}.

new(Window, Buffer, Options::[Option]) -> wxBufferedPaintDC()
Types:

Window = wxWindow:wxWindow()
Buffer = wxBitmap:wxBitmap()
Option = {style, integer()}
See external documentation.

destroy(This::wxBufferedPaintDC()) -> ok
Destroys this object, do not use object again
wxButton

Erlang module

See external documentation: wxButton.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxButton()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparsion stored on disc or distributed for use on other nodes.

Exports

call|new() -> wxButton()

See external documentation.

call|new(Parent, Id) -> wxButton()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()

Equivalent to new(Parent, Id, []).

call|new(Parent, Id, Options::[Option]) -> wxButton()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {label, unicode:chardata()} | {pos, {X::integer(), Y::integer()}}
  | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator,
  wx:wx_object()}

See external documentation.

create(This, Parent, Id) -> boolean()

Types:
  This = wxButton()
  Parent = wxWindow:wxWindow()
  Id = integer()

Equivalent to create(This, Parent, Id, []).
create(This, Parent, Id, Options::[Option]) -> boolean()

Types:

  This = wxButton()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {label, unicode:chardata()} | {pos, {X::integer(), Y::integer()}}
         | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

defaultSize() -> {W::integer(), H::integer()}

See external documentation.

setDefault(This) -> ok

Types:

  This = wxButton()

See external documentation.

setLabel(This, Label) -> ok

Types:

  This = wxButton()
  Label = unicode:chardata()

See external documentation.

destroy(This::wxButton()) -> ok

Destroys this object, do not use object again
wxCalendarCtrl

Erlang module

See external documentation: `wxCalendarCtrl`.

This class is derived (and can use functions) from:

- `wxControl`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

- `wxCalendarCtrl()`
  
  An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

- `new() -> wxCalendarCtrl()`
  
  See external documentation.

- `new(Parent, Id) -> wxCalendarCtrl()`
  
  Types:
  
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`
  
  Equivalent to `new(Parent, Id, [])`.

- `new(Parent, Id, Options::[Option]) -> wxCalendarCtrl()`
  
  Types:
  
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`
  - `Option = {date, wx:wx_datetime()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

  See external documentation.

- `create(This, Parent, Id) -> boolean()`
  
  Types:
  
  - `This = wxCalendarCtrl()`
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`

  Equivalent to `create(This, Parent, Id, [])`.

- `create(This, Parent, Id, Options::[Option]) -> boolean()`
  
  Types:
wxCalendarCtrl

This = wxCalendarCtrl()
Parent = wxWindow:wxWindow()
Id = integer()
Option = {date, wx:wx_datetime()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

setDate(This, Date) -> boolean()
Types:
   This = wxCalendarCtrl()
   Date = wx:wx_datetime()

See external documentation.

dateGet(This) -> wx:wx_datetime()
Types:
   This = wxCalendarCtrl()

See external documentation.

enableYearChange(This) -> ok
Types:
   This = wxCalendarCtrl()

This function is deprecated: not available in wxWidgets-2.9 and later
Equivalent to enableYearChange(This, []).

enableYearChange(This, Options::[Option]) -> ok
Types:
   This = wxCalendarCtrl()
   Option = {enable, boolean()}

This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

enableMonthChange(This) -> ok
Types:
   This = wxCalendarCtrl()

Equivalent to enableMonthChange(This, []).

enableMonthChange(This, Options::[Option]) -> ok
Types:
   This = wxCalendarCtrl()
   Option = {enable, boolean()}

See external documentation.
enableHolidayDisplay(This) -> ok
Types:
    This = wxCalendarCtrl()
Equivalent to enableHolidayDisplay(This, []).

enableHolidayDisplay(This, Options::[Option]) -> ok
Types:
    This = wxCalendarCtrl()
    Option = {display, boolean()}
See external documentation.

setHeaderColours(This, ColFg, ColBg) -> ok
Types:
    This = wxCalendarCtrl()
    ColFg = wx:wx_colour()
    ColBg = wx:wx_colour()
See external documentation.

generateHeaderColourFg(This) -> wx:wx_colour4()
Types:
    This = wxCalendarCtrl()
See external documentation.

generateHeaderColourBg(This) -> wx:wx_colour4()
Types:
    This = wxCalendarCtrl()
See external documentation.

setHighlightColours(This, ColFg, ColBg) -> ok
Types:
    This = wxCalendarCtrl()
    ColFg = wx:wx_colour()
    ColBg = wx:wx_colour()
See external documentation.

getHighlightColourFg(This) -> wx:wx_colour4()
Types:
    This = wxCalendarCtrl()
See external documentation.

generateHighlightColourBg(This) -> wx:wx_colour4()
Types:
    This = wxCalendarCtrl()
See external documentation.
setHolidayColours(This, ColFg, ColBg) -> ok

Types:
   This = wxCalendarCtrl()
   ColFg = wx:wx_colour()
   ColBg = wx:wx_colour()

See external documentation.

getHolidayColourFg(This) -> wx:wx_colour4()

Types:
   This = wxCalendarCtrl()

See external documentation.

getHolidayColourBg(This) -> wx:wx_colour4()

Types:
   This = wxCalendarCtrl()

See external documentation.

getAttr(This, Day) -> wxCalendarDateAttr:wxCalendarDateAttr()

Types:
   This = wxCalendarCtrl()
   Day = integer()

See external documentation.

setAttr(This, Day, Attr) -> ok

Types:
   This = wxCalendarCtrl()
   Day = integer()
   Attr = wxCalendarDateAttr:wxCalendarDateAttr()

See external documentation.

setHoliday(This, Day) -> ok

Types:
   This = wxCalendarCtrl()
   Day = integer()

See external documentation.

resetAttr(This, Day) -> ok

Types:
   This = wxCalendarCtrl()
   Day = integer()

See external documentation.
hitTest(This, Pos) -> Result

Types:

Result = {Res::wx:wx_enum(), Date::wx:wx_datetime(), Wd::wx:wx_enum()}

This = wxCalendarCtrl()

Pos = {X::integer(), Y::integer()}

See external documentation.

Wd = ?wxDateTime_Sun | ?wxDateTime_Mon | ?wxDateTime_Tue | ?wxDateTime_Wed | ?wxDateTime_Thu | ?wxDateTime_Fri | ?wxDateTime_Sat | ?wxDateTime_Inv_WeekDay


destroy(This::wxCalendarCtrl()) -> ok

Destroys this object, do not use object again
wxCalendarDateAttr

Erlang module

See external documentation: `wxCalendarDateAttr`.

DATA TYPES

`wxCalendarDateAttr()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`new() -> wxCalendarDateAttr()`

See external documentation.

`new(Border) -> wxCalendarDateAttr()`

Types:

```
Border = wx:wx_enum()
```

See external documentation.
Also:
```
new(ColText) -> wxCalendarDateAttr() when
ColText::wx:wx_colour().
Border = ?wxCAL_BORDER_NONE | ?wxCAL_BORDER_SQUARE | ?wxCAL_BORDER_ROUND
```

`new(Border, Options::[Option]) -> wxCalendarDateAttr()`

Types:

```
Border = wx:wx_enum()
Option = {colBorder, wx:wx_colour()}
```

See external documentation.
Also:
```
new(ColText, [Option]) -> wxCalendarDateAttr() when
ColText::wx:wx_colour(),
Option :: [{colBack, wx:wx_colour()},
             {colBorder, wx:wx_colour()},
             {font, wxFont:wxFont()},
             {border, wx:wx_enum()}.
Border = ?wxCAL_BORDER_NONE | ?wxCAL_BORDER_SQUARE | ?wxCAL_BORDER_ROUND
```

`setTextColour(This, ColText) -> ok`

Types:

```
This = wxCalendarDateAttr()
ColText = wx:wx_colour()
```

See external documentation.
setBackgroundColour(This, ColBack) -> ok
Types:
  This = wxCalendarDateAttr()
  ColBack = wx:wx_colour()
See external documentation.

setBorderColour(This, Col) -> ok
Types:
  This = wxCalendarDateAttr()
  Col = wx:wx_colour()
See external documentation.

setFont(This, Font) -> ok
Types:
  This = wxCalendarDateAttr()
  Font = wxFont:wxFont()
See external documentation.

setBorder(This, Border) -> ok
Types:
  This = wxCalendarDateAttr()
  Border = wx:wx_enum()
See external documentation.
Border = ?wxCAL_BORDER_NONE | ?wxCAL_BORDER_SQUARE | ?wxCAL_BORDER_ROUND

setHoliday(This, Holiday) -> ok
Types:
  This = wxCalendarDateAttr()
  Holiday = boolean()
See external documentation.

hasTextColour(This) -> boolean()
Types:
  This = wxCalendarDateAttr()
See external documentation.

hasBackgroundColour(This) -> boolean()
Types:
  This = wxCalendarDateAttr()
See external documentation.

hasBorderColour(This) -> boolean()
Types:
wxCalendarDateAttr

This = wxCalendarDateAttr()
See external documentation.

hasFont(This) -> boolean()
Types:
  This = wxCalendarDateAttr()
See external documentation.

hasBorder(This) -> boolean()
Types:
  This = wxCalendarDateAttr()
See external documentation.

isHoliday(This) -> boolean()
Types:
  This = wxCalendarDateAttr()
See external documentation.

ggetTextColour(This) -> wx:wx_colour4()
Types:
  This = wxCalendarDateAttr()
See external documentation.

getBackgroundColour(This) -> wx:wx_colour4()
Types:
  This = wxCalendarDateAttr()
See external documentation.

getBorderColour(This) -> wx:wx_colour4()
Types:
  This = wxCalendarDateAttr()
See external documentation.

gSetFont(This) -> wxFont:wxFont()
Types:
  This = wxCalendarDateAttr()
See external documentation.

getBorder(This) -> wx:wx_enum()
Types:
  This = wxCalendarDateAttr()
See external documentation.

Res = ?wxCAL_BORDER_NONE | ?wxCAL_BORDER_SQUARE | ?wxCAL_BORDER_ROUND
destroy(This::wxCalendarDateAttr()) -> ok

Destroys this object, do not use object again
wxCalendarEvent

Erlang module

See external documentation: wxCalendarEvent.

Use wxEvtHandler:connect/3 with EventType:

- calendar_sel_changed, calendar_day_changed, calendar_month_changed, calendar_year_changed,
- calendar_doubleclicked, calendar_weekday_clicked

See also the message variant #wxCalendar{} event record type.

This class is derived (and can use functions) from:

wxDateEvent
wxCommandEvent
wxEvent

DATA TYPES

wxCalendarEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getWeekDay(This) -> wx:wx_enum()

Types:

  This = wxCalendarEvent()

See external documentation.

Res = ?wxDateTime_Sun | ?wxDateTime_Mon | ?wxDateTime_Tue | ?wxDateTime_Wed | ?wxDateTime_Thu | ?wxDateTime_Fri | ?wxDateTime_Sat | ?wxDateTime_Inv_WeekDay
wxCaret
Erlang module

See external documentation: wxCaret.

DATA TYPES

wxCaret()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Window, Size) -> wxCaret()
Types:
   Window = wxWindow:wxWindow()
   Size = {W::integer(), H::integer()}
See external documentation.

ew(Window, Width, Height) -> wxCaret()
Types:
   Window = wxWindow:wxWindow()
   Width = integer()
   Height = integer()
See external documentation.

create(This, Window, Size) -> boolean()
Types:
   This = wxCaret()
   Window = wxWindow:wxWindow()
   Size = {W::integer(), H::integer()}
See external documentation.

create(This, Window, Width, Height) -> boolean()
Types:
   This = wxCaret()
   Window = wxWindow:wxWindow()
   Width = integer()
   Height = integer()
See external documentation.

getBlinkTime() -> integer()
See external documentation.
getPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxCaret()
See external documentation.

getSize(This) -> {W::integer(), H::integer()}
Types:
   This = wxCaret()
See external documentation.

getWindow(This) -> wxWindow:wxWindow()
Types:
   This = wxCaret()
See external documentation.

hide(This) -> ok
Types:
   This = wxCaret()
See external documentation.

isOk(This) -> boolean()
Types:
   This = wxCaret()
See external documentation.

isVisible(This) -> boolean()
Types:
   This = wxCaret()
See external documentation.

move(This, Pt) -> ok
Types:
   This = wxCaret()
   Pt = {X::integer(), Y::integer()}
See external documentation.

move(This, X, Y) -> ok
Types:
   This = wxCaret()
   X = integer()
   Y = integer()
See external documentation.
setBlinkTime(Milliseconds) -> ok
Types:
  Milliseconds = integer()
See external documentation.

setSize(This, Size) -> ok
Types:
  This = wxCaret()
  Size = {W::integer(), H::integer()}
See external documentation.

setSize(This, Width, Height) -> ok
Types:
  This = wxCaret()
  Width = integer()
  Height = integer()
See external documentation.

show(This) -> ok
Types:
  This = wxCaret()
Equivalent to show(This, []).

show(This, Options::[Option]) -> ok
Types:
  This = wxCaret()
  Option = {show, boolean()}
See external documentation.

destroy(This::wxCaret()) -> ok
Destroys this object, do not use object again
wxCheckBox

Erlang module

See external documentation: `wxCheckBox`.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

**DATA TYPES**

`wxCheckBox()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxCheckBox()`

See external documentation.

`new(Parent, Id, Label) -> wxCheckBox()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Label = unicode:chardata()`

Equivalent to `new(Parent, Id, Label, [])`.

`new(Parent, Id, Label, Options::[Option]) -> wxCheckBox()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Label = unicode:chardata()`
- `Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}`

See external documentation.

`create(This, Parent, Id, Label) -> boolean()`

Types:

- `This = wxCheckBox()`
- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Label = unicode:chardata()`

Equivalent to `create(This, Parent, Id, Label, [])`. 

82 | Ericsson AB. All Rights Reserved.: wxErlang
create(This, Parent, Id, Label, Options::[Option]) -> boolean()
Types:

  This = wxCheckBox()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Label = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getValue(This) -> boolean()
Types:

  This = wxCheckBox()

See external documentation.

get3StateValue(This) -> wx:wx_enum()
Types:

  This = wxCheckBox()

See external documentation.

Res = ?wxCHK_UNCHECKED | ?wxCHK_CHECKED | ?wxCHK_UNDETERMINED

is3rdStateAllowedForUser(This) -> boolean()
Types:

  This = wxCheckBox()

See external documentation.

is3State(This) -> boolean()
Types:

  This = wxCheckBox()

See external documentation.

isChecked(This) -> boolean()
Types:

  This = wxCheckBox()

See external documentation.

setValue(This, State) -> ok
Types:

  This = wxCheckBox()
  State = boolean()

See external documentation.
set3StateValue(This, State) -> ok
Types:
    This = wxCheckBox()
    State = wx:wx_enum()
See external documentation.
State = ?wxCHK_UNCHECKED | ?wxCHK_CHECKED | ?wxCHK_UNDETERMINED

destroy(This::wxCheckBox()) -> ok
Destroys this object, do not use object again
wxCheckListBox

Erlang module

See external documentation: wxCheckListBox.

This class is derived (and can use functions) from:
wxListBox
wxControlWithItems
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxCheckListBox()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxCheckListBox()

See external documentation.

new(Parent, Id) -> wxCheckListBox()

Types:

    Parent = wxWindow:wxWindow()
    Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxCheckListBox()

Types:

    Parent = wxWindow:wxWindow()
    Id = integer()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [unicode:chardata()]} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

check(This, Index) -> ok

Types:

    This = wxCheckListBox()
    Index = integer()

Equivalent to check(This, Index, []).
wxCheckListBox

check(This, Index, Options::[Option]) -> ok
Types:
  This = wxCheckListBox()
  Index = integer()
  Option = {check, boolean()}
See external documentation.

isChecked(This, Index) -> boolean()
Types:
  This = wxCheckListBox()
  Index = integer()
See external documentation.

destroy(This::wxCheckListBox()) -> ok
Destroys this object, do not use object again
wxChildFocusEvent

Erlang module

See external documentation: wxChildFocusEvent.

Use wxEvtHandler:connect/3 with EventType:

    child_focus

See also the message variant #wxChildFocus event record type.

This class is derived (and can use functions) from:
wxCommandEvent
wxEvent

DATA TYPES

wxChildFocusEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparision stored on disc or distributed for use on other nodes.

Exports

getWindow(This) -> wxWindow:wxWindow()

Types:

    This = wxChildFocusEvent()
wxChoice

Erlang module

See external documentation: *wxChoice*.

This class is derived (and can use functions) from:

wxControlWithItems
wxControl
wxWindow
wxEvtHandler

**DATA TYPES**

**wxChoice()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxChoice()

See external documentation.

new(Parent, Id) -> wxChoice()

Types:

Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxChoice()

Types:

Parent = wxWindow:wxWindow()
Id = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [unicode:chardata()]} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Pos, Size, Choices) -> boolean()

Types:

This = wxChoice()
Parent = wxWindow:wxWindow()
Id = integer()
Pos = {X::integer(), Y::integer()}
Size = {W::integer(), H::integer()}
Choices = [unicode:chardata()]
Equivalent to `create(This, Parent, Id, Pos, Size, Choices, [])`.

```erlang
create(This, Parent, Id, Pos, Size, Choices, Options::[Option]) -> boolean()
```

Types:

- `This = wxChoice()`
- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Pos = {X::integer(), Y::integer()}`
- `Size = {W::integer(), H::integer()}`
- `Choices = [unicode:chardata()]`
- `Option = {style, integer()} | {validator, wx:wx_object()}`

See external documentation.

```erlang
delete(This, N) -> ok
```

Types:

- `This = wxChoice()`
- `N = integer()`

See external documentation.

```erlang
columns(This) -> integer()
```

Types:

- `This = wxChoice()`

See external documentation.

```erlang
columns(This) -> ok
```

Types:

- `This = wxChoice()`

Equivalent to `setColumns(This, [])`.

```erlang
columns(This, Options::[Option]) -> ok
```

Types:

- `This = wxChoice()`
- `Option = {n, integer()}`

See external documentation.

```erlang
destroy(This::wxChoice()) -> ok
```

Destroys this object, do not use object again

Destroys this object, do not use object again
wxChoicebook

Erlang module

See external documentation: wxChoicebook.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES
wxChoicebook()

An object reference, The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxChoicebook()

See external documentation.

new(Parent, Id) -> wxChoicebook()
Types:
   Parent = wxWindow:wxWindow()
   Id = integer()
Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxChoicebook()
Types:
   Parent = wxWindow:wxWindow()
   Id = integer()
   Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(),
      H::integer()}} | {style, integer()}
See external documentation.

addPage(This, Page, Text) -> boolean()
Types:
   This = wxChoicebook()
   Page = wxWindow:wxWindow()
   Text = unicode:chardata()
Equivalent to addPage(This, Page, Text, []).

addPage(This, Page, Text, Options::[Option]) -> boolean()
Types:
This = wxChoicebook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

advanceSelection(This) -> ok
Types:
  This = wxChoicebook()
Equivalent to advanceSelection(This, []).

advanceSelection(This, Options::[Option]) -> ok
Types:
  This = wxChoicebook()
  Option = {forward, boolean()}
See external documentation.

assignImageList(This, ImageList) -> ok
Types:
  This = wxChoicebook()
  ImageList = wxImageList:wxImageList()
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
  This = wxChoicebook()
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
  This = wxChoicebook()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

deleteAllPages(This) -> boolean()
Types:
  This = wxChoicebook()
See external documentation.
wxChoicebook

deletePage(This, N) -> boolean()
Types:
   This = wxChoicebook()
   N = integer()
See external documentation.

removePage(This, N) -> boolean()
Types:
   This = wxChoicebook()
   N = integer()
See external documentation.

currentPage(This) -> wxWindow:wxWindow()
Types:
   This = wxChoicebook()
See external documentation.

getImageList(This) -> wxImageList:wxImageList()
Types:
   This = wxChoicebook()
See external documentation.

getPage(This, N) -> wxWindow:wxWindow()
Types:
   This = wxChoicebook()
   N = integer()
See external documentation.

count(This) -> integer()
Types:
   This = wxChoicebook()
See external documentation.

getPageImage(This, N) -> integer()
Types:
   This = wxChoicebook()
   N = integer()
See external documentation.

getPageText(This, N) -> unicode:charlist()
Types:
   This = wxChoicebook()
   N = integer()
See external documentation.

getSelection(This) -> integer()
Types:
   This = wxChoicebook()
See external documentation.

hitTest(This, Pt) -> Result
Types:
   Result = {Res::integer(), Flags::integer()}
   This = wxChoicebook()
   Pt = {X::integer(), Y::integer()}
See external documentation.

insertPage(This, N, Page, Text) -> boolean()
Types:
   This = wxChoicebook()
   N = integer()
   Page = wxWindow:wxWindow()
   Text = unicode:chardata()
Equivalent to insertPage(This, N, Page, Text, []).

insertPage(This, N, Page, Text, Options::[Option]) -> boolean()
Types:
   This = wxChoicebook()
   N = integer()
   Page = wxWindow:wxWindow()
   Text = unicode:chardata()
   Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

setImageList(This, ImageList) -> ok
Types:
   This = wxChoicebook()
   ImageList = wxImageList:wxImageList()
See external documentation.

setPageSize(This, Size) -> ok
Types:
   This = wxChoicebook()
   Size = {W::integer(), H::integer()}
See external documentation.
wxChoicebook

setPageImage(This, N, ImageId) -> boolean()

Types:
   This = wxChoicebook()
   N = integer()
   ImageId = integer()

See external documentation.

setPageText(This, N, StrText) -> boolean()

Types:
   This = wxChoicebook()
   N = integer()
   StrText = unicode:chardata()

See external documentation.

setSelection(This, N) -> integer()

Types:
   This = wxChoicebook()
   N = integer()

See external documentation.

changeSelection(This, N) -> integer()

Types:
   This = wxChoicebook()
   N = integer()

See external documentation.

destroy(This::wxChoicebook()) -> ok

Destroys this object, do not use object again
wxClientDC

Erlang module

See external documentation: wxClientDC.
This class is derived (and can use functions) from:
wxWindowDC
wxDC

DATA TYPES
wxClientDC()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxClientDC()

*This function is deprecated:* not available in wxWidgets-2.9 and later
See external documentation.

new(Win) -> wxClientDC()
Types:

Win = wxWindow:wxWindow()
See external documentation.

destroy(This::wxClientDC()) -> ok
Destroys this object, do not use object again
wxClipboard

Erlang module

See external documentation: wxClipboard.

DATA TYPES
wxClipboard()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxClipboard()
See external documentation.

addData(This, Data) -> boolean()
Types:
  This = wxClipboard()
  Data = wxDataObject:wxDataObject()
See external documentation.

clear(This) -> ok
Types:
  This = wxClipboard()
See external documentation.

close(This) -> ok
Types:
  This = wxClipboard()
See external documentation.

flush(This) -> boolean()
Types:
  This = wxClipboard()
See external documentation.

getData(This, Data) -> boolean()
Types:
  This = wxClipboard()
  Data = wxDataObject:wxDataObject()
See external documentation.
isOpened(This) -> boolean()
Types:
   This = wxClipboard()
See external documentation.

open(This) -> boolean()
Types:
   This = wxClipboard()
See external documentation.

setData(This, Data) -> boolean()
Types:
   This = wxClipboard()
   Data = wxDataObject:wxDataObject()
See external documentation.

usePrimarySelection(This) -> ok
Types:
   This = wxClipboard()
Equivalent to usePrimarySelection(This, []).

usePrimarySelection(This, Options::[Option]) -> ok
Types:
   This = wxClipboard()
   Option = {primary, boolean()}
See external documentation.

isSupported(This, Format) -> boolean()
Types:
   This = wxClipboard()
   Format = wx:wx_enum()
See external documentation.

get() -> wxClipboard()
See external documentation.

destroy(This::wxClipboard()) -> ok
Destroys this object, do not use object again
wxClipboardTextEvent

Erlang module

See external documentation: **wxClipboardTextEvent**.

Use `wxEvtHandler:connect/3` with EventType:

```
    command_text_copy, command_text_cut, command_text_paste
```

See also the message variant `#wxClipboardText/[]` event record type.

This class is derived (and can use functions) from:

- `wxCommandEvent`
- `wxEvent`

**DATA TYPES**

`wxClipboardTextEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxCloseEvent

Erlang module

See external documentation: wxCloseEvent.

Use wxEvtHandler:connect/3 with EventType:

   close_window, end_session, query_end_session

See also the message variant #wxCloseEvent event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxCloseEvent()

   An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

canVeto(This) -> boolean()

Types:
   This = wxCloseEvent()

See external documentation.

getLoggingOff(This) -> boolean()

Types:
   This = wxCloseEvent()

See external documentation.

setCanVeto(This, CanVeto) -> ok

Types:
   This = wxCloseEvent()
   CanVeto = boolean()

See external documentation.

setLoggingOff(This, LogOff) -> ok

Types:
   This = wxCloseEvent()
   LogOff = boolean()

See external documentation.

veto(This) -> ok

Types:
This = wxCloseEvent()

Equivalent to veto(This, []).

veto(This, Options::[Option]) -> ok

Types:

  This = wxCloseEvent()
  Option = {veto, boolean()}

See external documentation.
wxColourData

Erlang module

See external documentation: wxColourData.

DATA TYPES

wxColourData()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxColourData()
See external documentation.

ew(Data) -> wxColourData()
Types:
    Data = wxColourData()
See external documentation.

getChooseFull(This) -> boolean()
Types:
    This = wxColourData()
See external documentation.

getColour(This) -> wx:wx_colour4()
Types:
    This = wxColourData()
See external documentation.

getCustomColour(This, I) -> wx:wx_colour4()
Types:
    This = wxColourData()
    I = integer()
See external documentation.

setChooseFull(This, Flag) -> ok
Types:
    This = wxColourData()
    Flag = boolean()
See external documentation.
setColour(This, Colour) -> ok
Types:
   This = wxColourData()  
   Colour = wx:wx_colour()
See external documentation.

setCustomColour(This, I, Colour) -> ok
Types:
   This = wxColourData()  
   I = integer()  
   Colour = wx:wx_colour()
See external documentation.

destroy(This::wxColourData()) -> ok
Destroys this object, do not use object again
wxColourDialog

Erlang module

See external documentation: wxColourDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxColourDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxColourDialog()
See external documentation.

new(Parent) -> wxColourDialog()
Types:

Parent = wxWindow:wxWindow()

Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxColourDialog()
Types:

Parent = wxWindow:wxWindow()
Option = {data, wxColourData:wxColourData()}

See external documentation.

create(This, Parent) -> boolean()
Types:

This = wxColourDialog()
Parent = wxWindow:wxWindow()

Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()
Types:

This = wxColourDialog()
Parent = wxWindow:wxWindow()
Option = {data, wxColourData:wxColourData()}
wxColourDialog

See external documentation.

getColourData(This) -> wxColourData:wxColourData()
Types:

    This = wxColourDialog()

See external documentation.

destroy(This::wxColourDialog()) -> ok
Destroys this object, do not use object again
wxColourPickerCtrl

Erlang module

See external documentation: wxColourPickerCtrl.

This class is derived (and can use functions) from:
wxPickerBase
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxColourPickerCtrl()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxColourPickerCtrl()
See external documentation.

ew(Parent, Id) -> wxColourPickerCtrl()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxColourPickerCtrl()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {col, wx:wx_colour()}
  * {pos, {X::integer(), Y::integer()}} |
  * {size, {W::integer(), H::integer()}} | {style, integer()} | {validator,
  * wx:wx_object()}
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
  This = wxColourPickerCtrl()
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to create(This, Parent, Id, []).
create(This, Parent, Id, Options::[Option]) -> boolean()

Types:

  This = wxColourPickerCtrl()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {col, wx:wx_colour()} | {pos, {X::integer(), Y::integer()}} | 
          {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, 
          wx:wx_object()}

See external documentation.

getColour(This) -> wx:wx_colour4()

Types:

  This = wxColourPickerCtrl()

See external documentation.

setColour(This, Text) -> boolean()

Types:

  This = wxColourPickerCtrl()
  Text = unicode:chardata()

See external documentation.

Also:
setColour(This, Col) -> 'ok' when
This::wxColourPickerCtrl(), Col::wx:wx_colour().

destroy(This::wxColourPickerCtrl()) -> ok

Destroys this object, do not use object again
wxColourPickerEvent

Erlang module

See external documentation: `wxColourPickerEvent`.

Use `wxEvtHandler:connect/3` with EventType:

```
command_colourpicker_changed
```

See also the message variant `#wxColourPicker[]` event record type.

This class is derived (and can use functions) from:

- `wxCommandEvent`
- `wxEvent`

**DATA TYPES**

`wxColourPickerEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`getColour(This) -> wx:wx_colour4()`

Types:

```
This = wxColourPickerEvent()
```

See external documentation.
wxComboBox

Erlang module

See external documentation: wxComboBox.

This class is derived (and can use functions) from:
wxControlWithItems
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxComboBox()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxComboBox()

See external documentation.

new(Parent, Id) -> wxComboBox()

Types:
Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxComboBox()

Types:
Parent = wxWindow:wxWindow()
Id = integer()
Option = {value, unicode:chardata()} | {pos, {X::integer(), Y::integer()}}
| {size, {W::integer(), H::integer()}} | {choices, [unicode:chardata()]} |
{style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Value, Pos, Size, Choices) -> boolean()

Types:
This = wxComboBox()
Parent = wxWindow:wxWindow()
Id = integer()
Value = unicode:chardata()
Pos = {X::integer(), Y::integer()}
Size = {W::integer(), H::integer()}

108 | Ericsson AB. All Rights Reserved.: wxErlang
Choices = [unicode:chardata()]
Equivalent to create(This, Parent, Id, Value, Pos, Size, Choices, []).

create(This, Parent, Id, Value, Pos, Size, Choices, Options::[Option]) -> boolean()
Types:
  This = wxComboBox()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Value = unicode:chardata()
  Pos = {X::integer(), Y::integer()}
  Size = {W::integer(), H::integer()}
  Choices = [unicode:chardata()]
  Option = {style, integer()} | {validator, wx:wx_object()}

See external documentation.

canCopy(This) -> boolean()
Types:
  This = wxComboBox()

See external documentation.

canCut(This) -> boolean()
Types:
  This = wxComboBox()

See external documentation.

canPaste(This) -> boolean()
Types:
  This = wxComboBox()

See external documentation.

canRedo(This) -> boolean()
Types:
  This = wxComboBox()

See external documentation.

canUndo(This) -> boolean()
Types:
  This = wxComboBox()

See external documentation.

copy(This) -> ok
Types:
This = wxComboBox()
See external documentation.

cut(This) -> ok
Types:
  This = wxComboBox()
See external documentation.

getInsertionPoint(This) -> integer()
Types:
  This = wxComboBox()
See external documentation.

getLastPosition(This) -> integer()
Types:
  This = wxComboBox()
See external documentation.

getValue(This) -> unicode:charlist()
Types:
  This = wxComboBox()
See external documentation.

paste(This) -> ok
Types:
  This = wxComboBox()
See external documentation.

redo(This) -> ok
Types:
  This = wxComboBox()
See external documentation.

replace(This, From, To, Value) -> ok
Types:
  This = wxComboBox()
  From = integer()
  To = integer()
  Value = unicode:chardata()
See external documentation.

remove(This, From, To) -> ok
Types:
This = wxComboBox()
From = integer()
To = integer()
See external documentation.

setInsertionPoint(This, Pos) -> ok
Types:
    This = wxComboBox()
    Pos = integer()
See external documentation.

setInsertionPointEnd(This) -> ok
Types:
    This = wxComboBox()
See external documentation.

setSelection(This, N) -> ok
Types:
    This = wxComboBox()
    N = integer()
See external documentation.

setSelection(This, From, To) -> ok
Types:
    This = wxComboBox()
    From = integer()
    To = integer()
See external documentation.

setValue(This, Value) -> ok
Types:
    This = wxComboBox()
    Value = unicode:chardata()
See external documentation.

undo(This) -> ok
Types:
    This = wxComboBox()
See external documentation.

destroy(This::wxComboBox()) -> ok
Destroy this object, do not use object again
wxCommandEvent

Erlang module

See external documentation: **wxCommandEvent**.

Use **wxEvtHandler:connect/3** with EventType:

```
command_button_clicked, command_checkbox_clicked, command_choice_selected,
command_listbox_selected, command_listbox_doubleclicked, command_text_updated,
command_text_enter, command_menu_selected, command_slider_updated,
command_radiobox_selected, command_radiobutton_selected, command_scrollbar_updated,
command_vlbox_selected, command_combobox_selected, command_tool_rclicked, command_tool_enter,
command_checklistbox_toggled, command_togglebutton_clicked, command_left_click,
command_left_dclick, command_right_click, command_set_focus, command_kill_focus, command_enter
```

See also the message variant **#wxCommandEvent** event record type.

This class is derived (and can use functions) from:

**wxEvent**

**DATA TYPES**

**wxCommandEvent()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

**getClientData(This) -> term()**

Types:

```
This = wxCommandEvent()
```

See external documentation.

**getExtraLong(This) -> integer()**

Types:

```
This = wxCommandEvent()
```

See external documentation.

**getInt(This) -> integer()**

Types:

```
This = wxCommandEvent()
```

See external documentation.

**getSelection(This) -> integer()**

Types:

```
This = wxCommandEvent()
```

See external documentation.
getString(This) -> unicode:charlist()

Types:
    This = wxCommandEvent()

See external documentation.

isChecked(This) -> boolean()

Types:
    This = wxCommandEvent()

See external documentation.

isSelection(This) -> boolean()

Types:
    This = wxCommandEvent()

See external documentation.

setInt(This, I) -> ok

Types:
    This = wxCommandEvent()
    I = integer()

See external documentation.

setString(This, S) -> ok

Types:
    This = wxCommandEvent()
    S = unicode:chardata()

See external documentation.
wxContextMenuEvent

Erlang module

See external documentation: wxContextMenuEvent.

Use wxEvtHandler:connect/3 with EventType:

  context_menu

See also the message variant #wxContextMenu{} event record type.

This class is derived (and can use functions) from:

wxCommandEvent
wxEvent

DATA TYPES

wxContextMenuEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

getPosition(This) -> {X::integer(), Y::integer()}
Types:
  This = wxContextMenuEvent()
See external documentation.

setPosition(This, Pos) -> ok
Types:
  This = wxContextMenuEvent()
  Pos = {X::integer(), Y::integer()}
See external documentation.
wxControl

Erlang module

See external documentation: wxControl.

This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES

wxControl()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getLabel(This) -> unicode:charlist()
Types:

  This = wxControl()

See external documentation.

setLabel(This, Label) -> ok
Types:

  This = wxControl()

  Label = unicode:charlist()

See external documentation.
wxControlWithItems

Erlang module

See external documentation: **wxControlWithItems**.

This class is derived (and can use functions) from:

*wxControl*
*wxWindow*
*wxEvtHandler*

**DATA TYPES**

**wxControlWithItems()**

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

**append(This, Item) -> integer()**

Types:

- *This* = **wxControlWithItems()**
- *Item* = **unicode:chardata()**

See external documentation.

**append(This, Item, ClientData) -> integer()**

Types:

- *This* = **wxControlWithItems()**
- *Item* = **unicode:chardata()**
- *ClientData* = **term()**

See external documentation.

**appendStrings(This, Strings) -> ok**

Types:

- *This* = **wxControlWithItems()**
- *Strings* = [**unicode:chardata()**]

See external documentation.

**clear(This) -> ok**

Types:

- *This* = **wxControlWithItems()**

See external documentation.

**delete(This, N) -> ok**

Types:
This = wxControlWithItems()
N = integer()

See external documentation.

findString(This, S) -> integer()
Types:
   This = wxControlWithItems()
   S = unicode:chardata()

Equivalent to findString(This, S, []).

findString(This, S, Options::[Option]) -> integer()
Types:
   This = wxControlWithItems()
   S = unicode:chardata()
   Option = {bCase, boolean()}

See external documentation.

clientData(This, N) -> term()
Types:
   This = wxControlWithItems()
   N = integer()

See external documentation.

setClientData(This, N, ClientData) -> ok
Types:
   This = wxControlWithItems()
   N = integer()
   ClientData = term()

See external documentation.

count(This) -> integer()
Types:
   This = wxControlWithItems()

See external documentation.

selection(This) -> integer()
Types:
   This = wxControlWithItems()

See external documentation.

getString(This, N) -> unicode:charlist()
Types:
   This = wxControlWithItems()
wxControlWithItems

N = integer()
See external documentation.

getStringSelection(This) -> unicode:charlist()
Types:
   This = wxControlWithItems()
See external documentation.

insert(This, Item, Pos) -> integer()
Types:
   This = wxControlWithItems()
   Item = unicode:chardata()
   Pos = integer()
See external documentation.

insert(This, Item, Pos, ClientData) -> integer()
Types:
   This = wxControlWithItems()
   Item = unicode:chardata()
   Pos = integer()
   ClientData = term()
See external documentation.

isEmpty(This) -> boolean()
Types:
   This = wxControlWithItems()
See external documentation.

select(This, N) -> ok
Types:
   This = wxControlWithItems()
   N = integer()
See external documentation.

setSelection(This, N) -> ok
Types:
   This = wxControlWithItems()
   N = integer()
See external documentation.

setString(This, N, S) -> ok
Types:
   This = wxControlWithItems()
N = integer()
S = unicode:chardata()

See external documentation.

setStringSelection(This, S) -> boolean()

Types:
   This = wxControlWithItems()
   S = unicode:chardata()

See external documentation.
wxCursor

Erlang module

See external documentation: wxCursor.

This class is derived (and can use functions) from:
wxBitmap

DATA TYPES

wxCursor()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxCursor()

See external documentation.

new(CursorId) -> wxCursor()

Types:
   CursorId = integer()

See external documentation.

Also:
new(Image) -> wxCursor() when Image::wxImage:wxImage().

new(Bits, Width, Height) -> wxCursor()

Types:
   Bits = binary()
   Width = integer()
   Height = integer()

This function is deprecated: not available in wxWidgets-2.9 and later

Equivalent to new(Bits, Width, Height, []).

new(Bits, Width, Height, Options::[Option]) -> wxCursor()

Types:
   Bits = binary()
   Width = integer()
   Height = integer()
   Option = {hotSpotX, integer()} | {hotSpotY, integer()}

This function is deprecated: not available in wxWidgets-2.9 and later

See external documentation.
ok(This) -> boolean()

Types:

This = wxCursor()

See external documentation.

destroy(This::wxCursor()) -> ok

Destroy this object, do not use object again
wxDC

Erlang module

See external documentation: wxDC.

DATA TYPES

wxDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

blit(This, DestPt, Sz, Source, SrcPt) -> boolean()

Types:

This = wxDC()
DestPt = {X::integer(), Y::integer()}
Sz = {W::integer(), H::integer()}
Source = wxDC()
SrcPt = {X::integer(), Y::integer()}

Equivalent to blit(This, DestPt, Sz, Source, SrcPt, []).

blit(This, DestPt, Sz, Source, SrcPt, Options::[Option]) -> boolean()

Types:

This = wxDC()
DestPt = {X::integer(), Y::integer()}
Sz = {W::integer(), H::integer()}
Source = wxDC()
SrcPt = {X::integer(), Y::integer()}
Option = {rop, wx:wx_enum()} | {useMask, boolean()} | {srcPtMask, {X::integer(), Y::integer()}}

See external documentation.

Rop = integer

calcBoundingBox(This, X, Y) -> ok

Types:

This = wxDC()
X = integer()
Y = integer()

See external documentation.

clear(This) -> ok

Types:
This = wxDC()
See external documentation.

computeScaleAndOrigin(This) -> ok
Types:
    This = wxDC()
This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

crossHair(This, Pt) -> ok
Types:
    This = wxDC()
    Pt = {X::integer(), Y::integer()}
See external documentation.

destroyClippingRegion(This) -> ok
Types:
    This = wxDC()
See external documentation.

deviceToLogicalX(This, X) -> integer()
Types:
    This = wxDC()
    X = integer()
See external documentation.

deviceToLogicalXRel(This, X) -> integer()
Types:
    This = wxDC()
    X = integer()
See external documentation.

deviceToLogicalY(This, Y) -> integer()
Types:
    This = wxDC()
    Y = integer()
See external documentation.

deviceToLogicalYRel(This, Y) -> integer()
Types:
    This = wxDC()
    Y = integer()
See external documentation.

drawArc(This, Pt1, Pt2, Centre) -> ok
Types:
  This = wxDC()
  Pt1 = {X::integer(), Y::integer()}
  Pt2 = {X::integer(), Y::integer()}
  Centre = {X::integer(), Y::integer()}
See external documentation.

drawBitmap(This, Bmp, Pt) -> ok
Types:
  This = wxDC()
  Bmp = wxBitmap:wxBitmap()
  Pt = {X::integer(), Y::integer()}
Equivalent to drawBitmap(This, Bmp, Pt, []).

drawBitmap(This, Bmp, Pt, Options::[Option]) -> ok
Types:
  This = wxDC()
  Bmp = wxBitmap:wxBitmap()
  Pt = {X::integer(), Y::integer()}
  Option = {useMask, boolean()}
See external documentation.

drawCheckMark(This, Rect) -> ok
Types:
  This = wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
See external documentation.

drawCircle(This, Pt, Radius) -> ok
Types:
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
  Radius = integer()
See external documentation.

drawEllipse(This, Rect) -> ok
Types:
  This = wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
See external documentation.
drawEllipse(This, Pt, Sz) -> ok
Types:
  This = wxDC()
  Pt = (X::integer(), Y::integer())
  Sz = (W::integer(), H::integer())
See external documentation.

drawEllipticArc(This, Pt, Sz, Sa, Ea) -> ok
Types:
  This = wxDC()
  Pt = (X::integer(), Y::integer())
  Sz = (W::integer(), H::integer())
  Sa = number()
  Ea = number()
See external documentation.

drawIcon(This, Icon, Pt) -> ok
Types:
  This = wxDC()
  Icon = wxIcon:wxIcon()
  Pt = (X::integer(), Y::integer())
See external documentation.

drawLabel(This, Text, Rect) -> ok
Types:
  This = wxDC()
  Text = unicode:chardata()
  Rect = (X::integer(), Y::integer(), W::integer(), H::integer())
Equivalent to drawLabel(This, Text, Rect, []).

drawLabel(This, Text, Rect, Options::[Option]) -> ok
Types:
  This = wxDC()
  Text = unicode:chardata()
  Rect = (X::integer(), Y::integer(), W::integer(), H::integer())
  Option = {alignment, integer()} | {indexAccel, integer()}
See external documentation.

drawLine(This, Pt1, Pt2) -> ok
Types:
  This = wxDC()
  Pt1 = (X::integer(), Y::integer())
  Pt2 = (X::integer(), Y::integer())
See external documentation.

drawLines(This, Points) -> ok
Types:
   This = wxDC()
   Points = [{X::integer(), Y::integer()}]
Equivalent to drawLines(This, Points, []).

drawLines(This, Points, Options::[Option]) -> ok
Types:
   This = wxDC()
   Points = [{X::integer(), Y::integer()}]
   Option = {xoffset, integer()} | {yoffset, integer()}
See external documentation.

drawPolygon(This, Points) -> ok
Types:
   This = wxDC()
   Points = [{X::integer(), Y::integer()}]
Equivalent to drawPolygon(This, Points, []).

drawPolygon(This, Points, Options::[Option]) -> ok
Types:
   This = wxDC()
   Points = [{X::integer(), Y::integer()}]
   Option = {xoffset, integer()} | {yoffset, integer()} | {fillStyle, w:wx_enum()}
See external documentation.
FillStyle = integer

drawPoint(This, Pt) -> ok
Types:
   This = wxDC()
   Pt = {X::integer(), Y::integer()}
See external documentation.

drawRectangle(This, Rect) -> ok
Types:
   This = wxDC()
   Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
See external documentation.
drawRectangle(This, Pt, Sz) -> ok
Types:
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
  Sz = {W::integer(), H::integer()}
See external documentation.

drawRotatedText(This, Text, Pt, Angle) -> ok
Types:
  This = wxDC()
  Text = unicode:chardata()
  Pt = {X::integer(), Y::integer()}
  Angle = number()
See external documentation.

drawRoundedRectangle(This, R, Radius) -> ok
Types:
  This = wxDC()
  R = {X::integer(), Y::integer(), W::integer(), H::integer()}
  Radius = number()
See external documentation.

drawRoundedRectangle(This, Pt, Sz, Radius) -> ok
Types:
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
  Sz = {W::integer(), H::integer()}
  Radius = number()
See external documentation.

drawText(This, Text, Pt) -> ok
Types:
  This = wxDC()
  Text = unicode:chardata()
  Pt = {X::integer(), Y::integer()}
See external documentation.

endDoc(This) -> ok
Types:
  This = wxDC()
See external documentation.
endPage(This) -> ok
Types:
  This = wxDC()
See external documentation.

floodFill(This, Pt, Col) -> boolean()
Types:
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
  Col = wx:wx_colour()
Equivalent to floodFill(This, Pt, Col, []).

floodFill(This, Pt, Col, Options::[Option]) -> boolean()
Types:
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
  Col = wx:wx_colour()
  Option = {style, wx:wx_enum()}
See external documentation.
Style = integer

getBackground(This) -> wxBrush:wxBrush()
Types:
  This = wxDC()
See external documentation.

getBackgroundMode(This) -> integer()
Types:
  This = wxDC()
See external documentation.

getBrush(This) -> wxBrush:wxBrush()
Types:
  This = wxDC()
See external documentation.

getCharHeight(This) -> integer()
Types:
  This = wxDC()
See external documentation.

getCharWidth(This) -> integer()
Types:
This = wxDC()
See external documentation.

getClippingBox(This) -> Result
Types:
   Result = {X::integer(), Y::integer(), W::integer(), H::integer()}
   This = wxDC()
See external documentation.

g.getFont(This) -> wxFont:wxFont()
Types:
   This = wxDC()
See external documentation.

getLayoutDirection(This) -> wx:wx_enum()
Types:
   This = wxDC()
See external documentation.
Res = ?wxLayout_Default | ?wxLayout_LeftToRight | ?wxLayout_RightToLeft

getLogicalFunction(This) -> integer()
Types:
   This = wxDC()
See external documentation.

getMapMode(This) -> integer()
Types:
   This = wxDC()
See external documentation.

getMultiLineTextExtent(This, String) -> {W::integer(), H::integer()}
Types:
   This = wxDC()
   String = unicode:chardata()
See external documentation.

getMultiLineTextExtent(This, String, Options::[Option]) -> {Width::integer(), Height::integer(), HeightLine::integer()}
Types:
   This = wxDC()
   String = unicode:chardata()
   Option = {font, wxFont:wxFont()}
See external documentation.
getPartialTextExtents(This, Text) -> Result
Types:
  Result = {Res::boolean(), Widths::[integer()]}  
  This = wxDC()
  Text = unicode:chardata()
See external documentation.

getPen(This) -> wxPen:wxPen()
Types:
  This = wxDC()
See external documentation.

getPixel(This, Pt) -> Result
Types:
  Result = {Res::boolean(), Col::wx:wx_colour4()}
  This = wxDC()
  Pt = {X::integer(), Y::integer()}
See external documentation.

getPPI(This) -> {W::integer(), H::integer()}
Types:
  This = wxDC()
See external documentation.

getSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxDC()
See external documentation.

getSizeMM(This) -> {W::integer(), H::integer()}
Types:
  This = wxDC()
See external documentation.

getTextBackground(This) -> wx:wx_colour4()
Types:
  This = wxDC()
See external documentation.

getTextExtent(This, String) -> {W::integer(), H::integer()}
Types:
  This = wxDC()
  String = unicode:chardata()}
See external documentation.

getTextExtent(This, String, Options::[Option]) -> Result
Types:
  Result = {X::integer(), Y::integer(), Descent::integer(),
            ExternalLeading::integer()}
  This = wxDC()
  String = unicode:chardata()
  Option = {theFont, wxFont:wxFont()}
See external documentation.

getTextForeground(This) -> wx:wx_colour4()
Types:
  This = wxDC()
See external documentation.

getUserScale(This) -> {X::number(), Y::number()}
Types:
  This = wxDC()
See external documentation.

gradientFillConcentric(This, Rect, InitialColour, DestColour) -> ok
Types:
  This = wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
  InitialColour = wx:wx_colour()
  DestColour = wx:wx_colour()
See external documentation.

gradientFillConcentric(This, Rect, InitialColour, DestColour, CircleCenter) -> ok
Types:
  This = wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
  InitialColour = wx:wx_colour()
  DestColour = wx:wx_colour()
  CircleCenter = {X::integer(), Y::integer()}
See external documentation.

gradientFillLinear(This, Rect, InitialColour, DestColour) -> ok
Types:
  This = wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}

InitialColour = wx:wx_colour()
DestColour = wx:wx_colour()

Equivalent to gradientFillLinear(This, Rect, InitialColour, DestColour, []).

gradientFillLinear(This, Rect, InitialColour, DestColour, Options::[Option]) -> ok

Types:
   This = wxDC()
   Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
   InitialColour = wx:wx_colour()
   DestColour = wx:wx_colour()
   Option = {nDirection, wx:wx_enum()}

See external documentation.

           | ?wxWEST | ?wxEAST | ?wxALL

logicalToDeviceX(This, X) -> integer()
Types:
   This = wxDC()
   X = integer()

See external documentation.

logicalToDeviceXRel(This, X) -> integer()
Types:
   This = wxDC()
   X = integer()

See external documentation.

logicalToDeviceY(This, Y) -> integer()
Types:
   This = wxDC()
   Y = integer()

See external documentation.

logicalToDeviceYRel(This, Y) -> integer()
Types:
   This = wxDC()
   Y = integer()

See external documentation.

maxX(This) -> integer()
Types:
   This = wxDC()
See external documentation.

maxY(This) -> integer()
Types:
  This = wxDC()
See external documentation.

minX(This) -> integer()
Types:
  This = wxDC()
See external documentation.

minY(This) -> integer()
Types:
  This = wxDC()
See external documentation.

isOk(This) -> boolean()
Types:
  This = wxDC()
See external documentation.

resetBoundingBox(This) -> ok
Types:
  This = wxDC()
See external documentation.

setAxisOrientation(This, XLeftRight, YBottomUp) -> ok
Types:
  This = wxDC()
  XLeftRight = boolean()
  YBottomUp = boolean()
See external documentation.

setBackground(This, Brush) -> ok
Types:
  This = wxDC()
  Brush = wxBrush:wxBrush()
See external documentation.

setBackgroundMode(This, Mode) -> ok
Types:
  This = wxDC()
**Mode** = integer()

See external documentation.

`setBrush(This, Brush) -> ok`

Types:

```
This = wxDC()
Brush = wxBrush:wxBush()
```

See external documentation.

`setClippingRegion(This, Region) -> ok`

Types:

```
This = wxDC()
Region = wxRegion:wxBRegion()
```

See external documentation.

Also:

`setClippingRegion(This, Rect) -> 'ok' when
This::wxDC(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.`

`setClippingRegion(This, Pt, Sz) -> ok`

Types:

```
This = wxDC()
Pt = {X::integer(), Y::integer()}
Sz = {W::integer(), H::integer()}
```

See external documentation.

`setDeviceOrigin(This, X, Y) -> ok`

Types:

```
This = wxDC()
X = integer()
Y = integer()
```

See external documentation.

`setFont(This, Font) -> ok`

Types:

```
This = wxDC()
Font = wxFont:wxFon()t
```

See external documentation.

`setLayoutDirection(This, Dir) -> ok`

Types:

```
This = wxDC()
Dir = wx:wx_enum()
```

See external documentation.
Dir = ?wxLayout_Default | ?wxLayout_LeftToRight | ?wxLayout_RightToLeft

setLogicalFunction(This, Function) -> ok
Types:
   This = wxDC()
   Function = wx:wx_enum()
See external documentation.
Function = integer

setMapMode(This, Mode) -> ok
Types:
   This = wxDC()
   Mode = wx:wx_enum()
See external documentation.
Mode = integer

setPalette(This, Palette) -> ok
Types:
   This = wxDC()
   Palette = wxPalette:wxPalette()
See external documentation.

setPen(This, Pen) -> ok
Types:
   This = wxDC()
   Pen = wxPen:wxPen()
See external documentation.

setTextBackground(This, Colour) -> ok
Types:
   This = wxDC()
   Colour = wx:wx_colour()
See external documentation.

setTextForeground(This, Colour) -> ok
Types:
   This = wxDC()
   Colour = wx:wx_colour()
See external documentation.

setUserScale(This, X, Y) -> ok
Types:
   This = wxDC()
wxDC

\[
X = \text{number}()
\]
\[
Y = \text{number}()
\]

See external documentation.

\[\text{startDoc}(This, \text{Message}) \rightarrow \text{boolean}()\]

Types:

\[
\text{This} = \text{wxDC}()
\]
\[
\text{Message} = \text{unicode:chardata}()
\]

See external documentation.

\[\text{startPage}(This) \rightarrow \text{ok}\]

Types:

\[
\text{This} = \text{wxDC}()
\]

See external documentation.
wxDCOverlay

Erlang module

See external documentation: wxDCOverlay.

DATA TYPES

wxDCOverlay()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Overlay, Dc) -> wxDCOverlay()
Types:

Overlay = wxOverlay:wxOverlay()
Dc = wxWindowDC:wxWindowDC()

See external documentation.

new(Overlay, Dc, X, Y, Width, Height) -> wxDCOverlay()
Types:

Overlay = wxOverlay:wxOverlay()
Dc = wxWindowDC:wxWindowDC()
X = integer()
Y = integer()
Width = integer()
Height = integer()

See external documentation.

clear(This) -> ok
Types:

This = wxDCOverlay()

See external documentation.

destroy(This::wxDCOverlay()) -> ok

Destroys this object, do not use object again
wxDataObject

Erlang module

See external documentation: `wxDataObject`.

**DATA TYPES**

`wxDataObject()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxDateEvent

Erlang module

See external documentation: wxDateEvent.

Use \texttt{wxEvtHandler:connect/3} with \texttt{EventType}:

\texttt{date\_changed}

See also the message variant \texttt{#wxDate[]} event record type.

This class is derived (and can use functions) from:
\texttt{wxCommandEvent}
\texttt{wxEvent}

\textbf{DATA TYPES}

\texttt{wxDateEvent()}

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

\textbf{Exports}

\texttt{getDateTime(This) -> wx:wx\_datetime()}

Types:

\texttt{This = wxDateEvent()}

See external documentation.
wxDatePickerCtrl

Erlang module

See external documentation: wxDatePickerCtrl.

This class is derived (and can use functions) from:
wxPickerBase
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxDatePickerCtrl()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxDatePickerCtrl()

See external documentation.

ew(Parent, Id) -> wxDatePickerCtrl()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxDatePickerCtrl()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {date, wx:wx_datetime()} | {pos, {X::integer(), Y::integer()}} |
            {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getRange(This, Dt1, Dt2) -> boolean()

Types:
  This = wxDatePickerCtrl()
  Dt1 = wx:wx_datetime()
  Dt2 = wx:wx_datetime()

See external documentation.
getValue(This) -> wx:wx_datetime()

Types:
  This = wxDatePickerCtrl()

See external documentation.

setRange(This, Dt1, Dt2) -> ok

Types:
  This = wxDatePickerCtrl()
  Dt1 = wx:wx_datetime()
  Dt2 = wx:wx_datetime()

See external documentation.

setValue(This, Date) -> ok

Types:
  This = wxDatePickerCtrl()
  Date = wx:wx_datetime()

See external documentation.

destroy(This::wxDatePickerCtrl()) -> ok

Destroys this object, do not use object again.
wxDialog

Erlang module

See external documentation: `wxDialog`.

This class is derived (and can use functions) from:

`wxTopLevelWindow`
`wxWindow`
`wxEvtHandler`

**DATA TYPES**

`wxDialog()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxDialog()`

See external documentation.

`new(Parent, Id, Title) -> wxDialog()`

Types:

```erlang
Parent = wxWindow:wxWindow()
Id = integer()
Title = unicode:chardata()
```

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent, Id, Title, Options::[Option]) -> wxDialog()`

Types:

```erlang
Parent = wxWindow:wxWindow()
Id = integer()
Title = unicode:chardata()
Option = #pos{X::integer(), Y::integer()} | #size{W::integer(), H::integer()} | #style{integer()}
```

See external documentation.

`create(This, Parent, Id, Title) -> boolean()`

Types:

```erlang
This = wxDialog()
Parent = wxWindow:wxWindow()
Id = integer()
Title = unicode:chardata()
```

Equivalent to `create(This, Parent, Id, Title, [])`. 
create(This, Parent, Id, Title, Options::[Option]) -> boolean()
Types:
    This = wxDialog()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Title = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(),
    H::integer()}} | {style, integer()}
See external documentation.

createButtonSizer(This, Flags) -> wxSizer:wxSizer()
Types:
    This = wxDialog()
    Flags = integer()
See external documentation.

createStdDialogButtonSizer(This, Flags) ->
wxStdDialogButtonSizer:wxStdDialogButtonSizer()
Types:
    This = wxDialog()
    Flags = integer()
See external documentation.

diModal(This, RetCode) -> ok
Types:
    This = wxDialog()
    RetCode = integer()
See external documentation.

getAffirmativeId(This) -> integer()
Types:
    This = wxDialog()
See external documentation.

getReturnCode(This) -> integer()
Types:
    This = wxDialog()
See external documentation.

isModal(This) -> boolean()
Types:
    This = wxDialog()
See external documentation.
wxDialog

setAffirmativeId(This, AffirmativeId) -> ok
Types:
    This = wxDialog()
    AffirmativeId = integer()
See external documentation.

setReturnCode(This, ReturnCode) -> ok
Types:
    This = wxDialog()
    ReturnCode = integer()
See external documentation.

show(This) -> boolean()
Types:
    This = wxDialog()
Equivalent to show(This, []).

show(This, Options::[Option]) -> boolean()
Types:
    This = wxDialog()
    Option = {show, boolean()}
See external documentation.

showModal(This) -> integer()
Types:
    This = wxDialog()
See external documentation.

destroy(This::wxDialog()) -> ok
Destroys this object, do not use object again
wxDirDialog

Erlang module

See external documentation: wxDirDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxDirDialog()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Parent) -> wxDirDialog()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxDirDialog()
Types:
    Parent = wxWindow:wxWindow()
    Option = {title, unicode:chardata()} | {defaultPath, unicode:chardata()} |
    {style, integer()} | {pos, {X::integer(), Y::integer()}} | {sz, 
    {W::integer(), H::integer()}}
See external documentation.

getPath(This) -> unicode:charlist()
Types:
    This = wxDirDialog()
See external documentation.

getMessage(This) -> unicode:charlist()
Types:
    This = wxDirDialog()
See external documentation.

setMessage(This, Message) -> ok
Types:
    This = wxDirDialog()
wxDirDialog

Message = \texttt{unicode:chardata()}

See external documentation.

\texttt{setPath(This, Path) \rightarrow ok}

Types:
\begin{itemize}
\item \texttt{This} = \texttt{wxDirDialog()}
\item \texttt{Path} = \texttt{unicode:chardata()}
\end{itemize}

See external documentation.

\texttt{destroy(This::wxDirDialog()) \rightarrow ok}

Destroys this object, do not use object again.
wxDirPickerCtrl

Erlang module

See external documentation: wxDirPickerCtrl.

This class is derived (and can use functions) from:
wxPickerBase
wxControl
wxWindow
wxEvtHandler

DATA TYPES
wxDirPickerCtrl()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxDirPickerCtrl()
See external documentation.

ew(Parent, Id) -> wxDirPickerCtrl()
Types:

Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxDirPickerCtrl()
Types:

Parent = wxWindow:wxWindow()
Id = integer()
Option = {path, unicode:chardata()} | {message, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id) -> boolean()
Types:

This = wxDirPickerCtrl()
Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to create(This, Parent, Id, []).
**wxDirPickerCtrl**

**create(This, Parent, Id, Options::[Option]) -> boolean()**  
Types:  

- **This** = wxDirPickerCtrl()  
- **Parent** = wxWindow::wxWindow()  
- **Id** = integer()  
- **Option** = {path, unicode:chardata()} | {message, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx::wx_object()}

See external documentation.

**getPath(This) -> unicode:charlist()**  
Types:  

- **This** = wxDirPickerCtrl()  

See external documentation.

**setPath(This, Str) -> ok**  
Types:  

- **This** = wxDirPickerCtrl()  
- **Str** = unicode:chardata()  

See external documentation.

**destroy(This::wxDirPickerCtrl()) -> ok**  
Destroy this object. do not use object again
wxDisplay
Erlang module

See external documentation: wxDisplay.

DATA TYPES
wxDisplay()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

call(1, []) -> wxDisplay()
Equivalent to new([]).

call(1, Options::[Option]) -> wxDisplay()
Types:
   Option = {n, integer()}
See external documentation.

call(1, []) -> boolean()
Types:
   This = wxDisplay()
See external documentation.

call(1, [], []) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
   This = wxDisplay()
See external documentation.

call(1, [], []) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
   This = wxDisplay()
See external documentation.

call(1, [], []) -> unicode:charlist()
Types:
   This = wxDisplay()
See external documentation.
isPrimary(This) -> boolean()
Types:
   This = wxDisplay()
See external documentation.

getCount() -> integer()
See external documentation.

getFromPoint(Pt) -> integer()
Types:
   Pt = {X::integer(), Y::integer()}
See external documentation.

getFromWindow(Window) -> integer()
Types:
   Window = wxWindow:wxWindow()
See external documentation.

getPPI(This) -> {W::integer(), H::integer()}
Types:
   This = wxDisplay()
See external documentation.

destroy(This::wxDisplay()) -> ok
Destroys this object, do not use object again
wxDisplayChangedEvent

Erlang module

See external documentation: `wxDisplayChangedEvent`.

Use `wxEvtHandler:connect/3` with EventType:

`display_changed`

See also the message variant `#wxDisplayChanged/1` event record type.

This class is derived (and can use functions) from: `wxEvent`

**DATA TYPES**

`wxDisplayChangedEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxDropFilesEvent

Erlang module

See external documentation: `wxDropFilesEvent`.

Use `wxEvtHandler:connect/3` with EventType:

`drop_files`

See also the message variant `wxDropFilesEvent` event record type.

This class is derived (and can use functions) from:

`wxEvent`

DATA TYPES

`wxDropFilesEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`getPosition(This) -> {X::integer(), Y::integer()}`

Types:

`This = wxDropFilesEvent()`

See external documentation.

`getNumberOfFiles(This) -> integer()`

Types:

`This = wxDropFilesEvent()`

See external documentation.

`getFiles(This) -> [unicode:charlist()]`

Types:

`This = wxDropFilesEvent()`

See external documentation.
wxEraseEvent

Erlang module

See external documentation: wxEraseEvent.

Use wxEvtHandler:connect/3 with EventType:

    erase_background

See also the message variant #wxErase{} event record type.

This class is derived (and can use functions) from:

wxEvent

DATA TYPES

wxEraseEvent()

    An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

getDC(This) -> wxDC:wxDC()

Types:

    This = wxEraseEvent()

See external documentation.
wxEvent

Erlang module

See external documentation: wxEvent.

DATA TYPES

wxEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getId(This) -> integer()
Types:
   This = wxEvent()
See external documentation.

getSkipped(This) -> boolean()
Types:
   This = wxEvent()
See external documentation.

getTimestamp(This) -> integer()
Types:
   This = wxEvent()
See external documentation.

isCommandEvent(This) -> boolean()
Types:
   This = wxEvent()
See external documentation.

resumePropagation(This, PropagationLevel) -> ok
Types:
   This = wxEvent()
   PropagationLevel = integer()
See external documentation.

shouldPropagate(This) -> boolean()
Types:
   This = wxEvent()
See external documentation.
wxEvent

skip(This) -> ok
Types:
   This = wxEvent()
Equivalent to skip(This, []).

skip(This, Options::[Option]) -> ok
Types:
   This = wxEvent()
   Option = {skip, boolean()}
See external documentation.

stopPropagation(This) -> integer()
Types:
   This = wxEvent()
See external documentation.
The Event handler.

To get events from wxwidgets objects you subscribe to them by calling connect/[2-3]. Events are sent as messages, if no callback was supplied these messages will be #wx/1 where EventRecord is a record that depends on the event type. The records are defined in: wx/include/wx.hrl.

If a callback was supplied to connect, the callback will be invoked (in another process) to handle the event. The callback should be of arity 2. fun(EventRecord::wx(), EventObject::wxObject()). Beware that the callback will be in executed in new process each time.

The original documentation.

DATA TYPES

event() = wxActivate() | wxAuiManager() | wxAuiNotebook() | wxCalendar() | wxChildFocus() | wxClipboardText() |
wxClose() | wxColourPicker() | wxCommand() | wxContextMenu() | wxDate() | wxDisplayChanged() |
xwDropFiles() | wxErase() | wxFileDirPicker() | wxFocus() | wxFontPicker() | wxGrid() | wxHelp() | wxHtmlLink() |
xIconize() | wxIdle() | wxInitDialog() | wxJoystick() | wxKey() | wxList() | wxMaximize() | wxMenu() | wxMouse() |
xMouseCaptureChanged() | wxMouseCaptureLost() | wxMove() | wxNavigationKey() | wxNotebook() | wxPaint() |
xPaletteChanged() | wxQueryNewPalette() | wxSash() | wxScroll() | wxScrollWin() | wxSetCursor() | wxShow() |
xSize() | wxSpin() | wxSplitter() | wxStyleText() | wxSysColourChanged() | wxTaskBarIcon() | wxTree() |
xUpdateUI() | wxTWindowCreate() | wxTWindowDestroy()

wx() = #wx{id=integer(), obj= wx:wx_object(), userData=term(), event=event()}

wxActivate() = #wxActivate{type=wxActivateEventType(), active=boolean()}

wxActivateEventType() = activate | activate_app | hibernate

wxAuiManager() = #wxAuiManager{type=wxAuiManagerEventType(), manager=wxAuiManager:wxAuiManager(), pane=wxAuiPaneInfo:wxAuiPaneInfo(), button=integer(), veto_flag=boolean(), canveto_flag=boolean(), dc=wxDC:wxDC()}

wxAuiManagerEventType() = aui_pane_button | aui_pane_close | aui_pane_maximize | aui_pane_restore | aui_pane_activated | aui_render | aui_find_manager

wxAuiNotebook() = #wxAuiNotebook{type=wxAuiNotebookEventType(), old_selection=integer(), selection=integer(), drag_source=wxAuiNotebook:wxAuiNotebook()}

wxAuiNotebookEventType() = command_aunotebook_page_close | command_aunotebook_page_changed | command_aunotebook_page_changing | command_aunotebook_button | command_aunotebook_begin_drag | command_aunotebook_end_drag | command_aunotebook_drag_motion | command_aunotebook_allow_dnd | command_aunotebook_tab_middle_down | command_aunotebook_tab_middle_up | command_aunotebook_tab_right_down | command_aunotebook_tab_right_up | command_aunotebook_page_closed | command_aunotebook_drag_done | command_aunotebook_bg_delclick

wxCalendar() = #wxCalendar{type=wxCalendarEventType(), wday= wx:wx_enum(), date=wx:wx_datetime()}

wxCalendarEventType() = calendar_sel_changed | calendar_day_changed | calendar_month_changed | calendar_year_changed | calendar_doubleclicked | calendar_weekday_clicked

DATA TYPES

event() = wxActivate() | wxAuiManager() | wxAuiNotebook() | wxCalendar() | wxChildFocus() | wxClipboardText() |
wxClose() | wxColourPicker() | wxCommand() | wxContextMenu() | wxDate() | wxDisplayChanged() |
xwDropFiles() | wxErase() | wxFileDirPicker() | wxFocus() | wxFontPicker() | wxGrid() | wxHelp() | wxHtmlLink() |
xIconize() | wxIdle() | wxInitDialog() | wxJoystick() | wxKey() | wxList() | wxMaximize() | wxMenu() | wxMouse() |
xMouseCaptureChanged() | wxMouseCaptureLost() | wxMove() | wxNavigationKey() | wxNotebook() | wxPaint() |
xPaletteChanged() | wxQueryNewPalette() | wxSash() | wxScroll() | wxScrollWin() | wxSetCursor() | wxShow() |
xSize() | wxSpin() | wxSplitter() | wxStyleText() | wxSysColourChanged() | wxTaskBarIcon() | wxTree() |
xUpdateUI() | wxTWindowCreate() | wxTWindowDestroy()
wxChildFocus() = #wxChildFocus{type=wxChildFocusEventType()}
wxChildFocusEventType() = child_focus
wxClipboardText() = #wxClipboardText{type=wxClipboardTextEventType()}
wxClipboardTextEventType() = command_text_copy | command_text_cut | command_text_paste
wxClose() = #wxClose{type=wxCloseEventType()}
wxCloseEventType() = close_window | end_session | query_end_session
wxColourPicker() = #wxColourPicker{type=wxColourPickerEventType(), colour=wx:wx_colour()}
wxColourPickerEventType() = command_colourpicker_changed
wxCommand() = #wxCommand{type=wxCommandEventType(), cmdString=unicode:chardata(), commandInt=integer(), extraLong=integer()}
wxCommandEventType() = command_button_clicked | command_checkbox_clicked | command_choice_selected | command_listbox_selected | command_listbox_doubleclicked | command_text_updated | command_text_enter | command_menu_selected | command_slider_updated | command_radiobox_selected | command_radiobutton_selected | command_scrollbar_updated | command_vbox_selected | command_combobox_selected | command_tool_rclicked | command_tool_enter | command_checklistbox_toggled | command.togglebutton_clicked | command_left_click | command_left_dclick | command_right_click | command_set_focus | command_kill_focus | command_enter
wxContextMenu() = #wxContextMenu{type=wxContextMenuEventType(), pos={X::integer(), Y::integer()}}
wxContextMenuEventType() = context_menu
wxDate() = #wxDate{type=wxDateEventType(), date=wx:wx_datetime()}
wxDateEventType() = date_changed
wxDisplayChanged() = #wxDisplayChanged{type=wxDisplayChangedEventType()}
wxDisplayChangedEventType() = display_changed
wxDropFiles() = #wxDropFiles{type=wxDropFilesEventType(), noFiles=integer(), pos={X::integer(), Y::integer()}, files=[unicode:chardata()]} wxDropFilesEventType() = drop_files
wxErase() = #wxErase{type=wxEraseEventType(), dc=wxDC:wxDC()}
wxEraseEventType() = erase_background
wxEventType() = wxActivateEventType() | wxAuiManagerEventType() | wxAuiNotebookEventType() | wxCalendarEventType() | wxChildFocusEventType() | wxClipboardTextEventType() | wxCloseEventType() | wxColourPickerEventType() | wxCommandEventType() | wxContextMenuEventType() | wxDateEventType() | wxDisplayChangedEventType() | wxDropFilesEventType() | wxEraseEventType() | wxFileDirPickerEventType() | wxFocusEventType() | wxFontPickerEventType() | wxGridEventType() | wxHelpEventType() | wxHtmlLinkEventType() | wxHideEvent() | wxIdleEventType() | wxInitDialogEventType() | wxJoystickEventType() | wxKeyEventType() | wxListEventType() | wxMaximizeEventType() | wxMenuEventType()
wxEvtHandler

wxMouseCaptureChangedEventType() | wxMouseCaptureLostEventType() | wxMouseEventType() | wxMoveEventType() | wxNavigationKeyEventType() | wxNotebookEventType() | wxPaintEventType() | wxPaletteChangedEventType() | wxQueryNewPaletteEventType() | wxSashEventType() | wxScrollEventType() | wxScrollWinEventType() | wxSetCursorEventType() | wxShowEventType() | wxSizeEventType() | wxSplitterEventType() | wxStyleTextEventType() | wxSysColourChangedEventType() | wxTaskBarIconEventType() | wxTreeEventType() | wxUpdateUIEventType() | wxWindowCreateEventType() | wxWindowDestroyEventType()

wxEvtHandler() = wx:wx_object()
wxFileDirPicker() = #wxFileDirPicker{type=wxFileDirPickerEventType(), path=unicode:chardata()}
wxFileDirPickerEventType() = command_filepicker_changed | command_dirpicker_changed
wxFocus() = #wxFocus{type=wxFocusEventType(), win=wxWindow:wxWindow()}
wxFocusEventType() = set_focus | kill_focus
wxFontPicker() = #wxFontPicker{type=wxFontPickerEventType(), font=wxFont:wxFont()}
wxFontPickerEventType() = command_fontpicker_changed
wxGrid() = #wxGrid{type=wxGridEventType(), row=integer(), col=integer(), x=integer(), y=integer(), selecting=boolean(), control=boolean(), meta=boolean(), shift=boolean(), alt=boolean(), selecting=boolean(), control=boolean(), meta=boolean(), shift=boolean(), alt=boolean()}
wxGridEventType() = grid_cell_left_click | grid_cell_right_click | grid_cell_left_dclick | grid_cell_right_dclick | grid_label_left_click | grid_label_right_click | grid_label_left_dclick | grid_label_right_dclick | grid_row_size | grid_col_size | grid_range_select | grid_cell_change | grid_select_cell | grid_editor_shown | grid_editor_hidden | grid_editor_created | grid_cell_begin_drag
wxHelp() = #wxHelp{type=wxHelpEventType()}
wxHelpEventType() = help | detailed_help
wxHtmlLink() = #wxHtmlLink{type=wxHtmlLinkEventType(), linkInfo=wx:wxHtmlLinkInfo()}
wxHtmlLinkEventType() = command_html_link_clicked
wxIconize() = #wxIconize{type=wxIconizeEventType(), iconized=boolean()}
wxIconizeEventType() = iconize
wxIdle() = #wxIdle{type=wxIdleEventType()}
wxIdleEventType() = idle
wxInitDialog() = #wxInitDialog{type=wxInitDialogEventType()}
wxInitDialogEventType() = init_dialog
wxJoystick() = #wxJoystick{type=wxJoystickEventType(), pos={X::integer(), Y::integer()}, zPosition=integer(), buttonChange=integer(), buttonState=integer(), joystick=integer()}
wxJoystickEventType() = joy_button_down | joy_button_up | joy_move | joy_zmove
wxKey() = #wxKey{type=wxKeyEvent(), x=integer(), y=integer(), keyCode=integer(),
controlDown=boolean(), shiftDown=boolean(), altDown=boolean(), metaDown=boolean(), scanCode=boolean(),
uniChar=integer(), rawCode=integer(), rawFlags=integer()}

wxKeyEvent() = char | char_hook | key_down | key_up

wxList() = #wxList{type=wxListEvent(), code=integer(), oldItemIndex=integer(), itemIndex=integer(),
col=integer(), pointDrag={X::integer(), Y::integer()}}

wxListEvent() = command_list_begin_drag | command_list_begin_rdrag | command_list_begin_label_edit |
command_list_end_label_edit | command_list_delete_item | command_list_delete_all_items |
command_list_key_down | command_list_insert_item | command_list_click | command_list_right_click |
command_list_col_begin_drag | command_list_col_dragging | command_list_col_end_drag |
command_list_item_selected | command_list_item_deselected | command_list_item_right_click |
command_list_item_middle_click | command_list_item_activated | command_list_itemFocused |
command_list_cache_hint

wxMaximize() = #wxMaximize{type=wxMaximizeEvent()}

wxMaximizeEvent() = maximize

wxMenu() = #wxMenu{type=wxMenuEvent(), menuId=integer(), menu=wxMenu:wxMenu()}

wxMenuEvent() = menu_open | menu_close | menu_highlight

wxMouse() = #wxMouse{type=wxMouseEvent(), x=integer(), y=integer(), leftDown=boolean(),
middleDown=boolean(), rightDown=boolean(), controlDown=boolean(), shiftDown=boolean(), altDown=boolean(),
metaDown=boolean(), wheelRotation=integer(), wheelDelta=integer(), linesPerAction=integer()}

wxMouseCaptureChanged() = #wxMouseCaptureChanged{type=wxMouseCaptureChangedEvent()}

wxMouseCaptureChangedEvent() = mouse_capture_change

wxMouseCaptureLost() = #wxMouseCaptureLost{type=wxMouseCaptureLostEvent()}

wxMouseCaptureLostEvent() = mouse_capture_lost

wxMouseEvent() = left_down | left_up | middle_down | middle_up | right_down | right_up | motion |
enter_window | leave_window | left_dclick | middle_dclick | right_dclick | mousewheel

wxMove() = #wxMove{type=wxMoveEvent(), pos={X::integer(), Y::integer()}, rect={X::integer(),
Y::integer(), W::integer(), H::integer()}}

wxMoveEvent() = move

wxNavigationKey() = #wxNavigationKey{type=wxNavigationKeyEvent(), flags=integer(),
focus=wxWindow:wxWindow()}

wxNavigationKeyEvent() = navigation_key

wxNotebook() = #wxNotebook{type=wxNotebookEvent(), nSel=integer(), nOldSel=integer()}

wxNotebookEvent() = command_notebook_page_change | command_notebook_page_changing

wxPaint() = #wxPaint{type=wxPaintEvent()}

Ericsson AB. All Rights Reserved.: wxErlang | 159
wxEventHandler

wxPaintEventType() = paint
wxPaletteChanged() = #wxPaletteChanged{type=wxPaletteChangedEventType()}
wxPaletteChangedEventType() = palette_changed
wxQueryNewPalette() = #wxQueryNewPalette{type=wxQueryNewPaletteEventType()}
wxQueryNewPaletteEventType() = query_new_palette
wxSash() = #wxSash{type=wxSashEventType(), edge=wx:wx_enum(), dragRect={X::integer(), Y::integer(), W::integer(), H::integer()}, dragStatus=wx:wx_enum()}
wxSashEventType() = sash_dragged
wxScroll() = #wxScroll{type=wxScrollEventType(), commandInt=integer(), extraLong=integer()}
wxScrollEventType() = scroll_top | scroll_bottom | scroll_lineup | scroll_linedown | scroll_pageup | scroll_pagedown | scroll_thumbtrack | scroll_thumbrelease | scroll_changed
wxScrollWin() = #wxScrollWin{type=wxScrollWinEventType(), commandInt=integer(), extraLong=integer()}
wxScrollWinEventType() = scrollwin_top | scrollwin_bottom | scrollwin_lineup | scrollwin_linedown | scrollwin_pageup | scrollwin_pagedown | scrollwin_thumbtrack | scrollwin_thumbrelease
wxSetCursor() = #wxSetCursor{type=wxSetCursorEventType(), x=integer(), y=integer(), cursor=wxCursor:wxCursor()}
wxSetCursorEventType() = set_cursor
wxShow() = #wxShow{type=wxShowEventType(), show=boolean()}
wxShowEventType() = show
wxSize() = #wxSize{type=wxSizeEventType(), size={W::integer(), H::integer()}, rect={X::integer(), Y::integer(), W::integer(), H::integer()}}
wxSizeEventType() = size
wxSpin() = #wxSpin{type=wxSpinEventType(), commandInt=integer()}
wxSpinEventType() = command_spinctrl_updated | spin_up | spin_down | spin
wxSplitter() = #wxSplitter{type=wxSplitterEventType()}
wxSplitterEventType() = command_splitter_sash_pos_changed | command_splitter_sash_pos_changing | command_splitter_doubleclicked | command_splitter_unsplit
wxStyledText() = #wxStyledText{type=wxStyledTextEventType(), position=integer(), key=integer(), modifiers=integer(), modificationType=integer(), text=unicode:chardata(), length=integer(), linesAdded=integer(), line=integer(), foldLevelNow=integer(), foldLevelPrev=integer(), margin=integer(), message=integer(), wParam=integer(), lParam=integer(), listType=integer(), x=integer(), y=integer(), dragText=unicode:chardata(), dragAllowMove=boolean(), dragResult=wx:wx_enum()}
wxStyledTextEventType() = stc_change | stc_styleneeded | stc_charadded | stc_savepointreached | stc_savepointleft | stc_romodifyattempt | stc_key | stc_doubleclick | stc_updateui | stc_modified | stc_macrecord | stc_marginclick

160 | Ericsson AB. All Rights Reserved.: wxErlang
| stc_needshown | stc_painted | stc_userlistselection | stc_uridropped | stc_dwellstart | stc_dwellend | stc_start_drag | stc_drag_over | stc_do_drop | stc_zoom | stc_hotspot_click | stc_hotspot_dclick | stc_calltip_click | stc_autocomp_selection |
wxSysColourChanged() = #wxSysColourChanged{type=wxSysColourChangedEventType()}
wxSysColourChangedEventType() = sys_colour_changed
wxTaskBarIcon() = #wxTaskBarIcon{type=wxTaskBarIconEventType()}
wxTaskBarIconEventType() = taskbar_move | taskbar_left_down | taskbar_left_up | taskbar_right_down | taskbar_right_up | taskbar_left_dclick | taskbar_right_dclick
wxTree() = #wxTree{type=wxTreeEventType(), item=integer(), itemOld=integer(), pointDrag={X::integer(), Y::integer()}}
wxTreeEventType() = command_tree_begin_drag | command_tree_begin_rdrag | command_tree_begin_label_edit | command_tree_end_label_edit | command_tree_delete_item | command_tree_get_info | command_tree_set_info | command_tree_item_expanded | command_tree_item_expanding | command_tree_itemCollapsed | command_tree_item_collapsing | command_tree_sel_changed | command_tree_sel_changing | command_tree_item_activated | command_tree_item_right_click | command_tree_item_middle_click | command_tree_end_drag | command_tree_state_image_click | command_tree_item_gettooltip | command_tree_item_menu
wxUpdateUI() = #wxUpdateUI{type=wxUpdateUIEventType()}
wxUpdateUIEventType() = update_ui
wxWindowCreate() = #wxWindowCreate{type=wxWindowCreateEventType()}
wxWindowCreateEventType() = create
wxWindowDestroy() = #wxWindowDestroy{type=wxWindowDestroyEventType()}
wxWindowDestroyEventType() = destroy

Exports

connect(This::wxEvtHandler(), EventType::wxEventType()) -> ok
Equivalent to connect(This, EventType, [])

connect(This::wxEvtHandler(), EventType::wxEventType(), Options::[Option]) -> ok
Types:
  Option = {id, integer()} | {lastId, integer()} | {skip, boolean()} | {callback, function()} | {userData, term()}
This function subscribes the to events of EventType, in the range id, lastId. The events will be received as messages if no callback is supplied.
Options: {id, integer()}. The identifier (or first of the identifier range) to be associated with this event handler. Default ?wxID_ANY {lastId, integer()}. The second part of the identifier range. If used 'id' must be set as the starting identifier range. Default ?wxID_ANY {skip, boolean()}. If skip is true further event_handlers will be called. This is
not used if the 'callback' option is used. Default false. {callback, function()} Use a callback fun(EventRecord::wx(), EventObject::wxObject()) to process the event. Default not specified i.e. a message will be delivered to the process calling this function. {userData, term()} An erlang term that will be sent with the event. Default: [].

disconnect(This::wxEvtHandler()) -> boolean()
Equivalent to disconnect(This, null, []) Can also have an optional callback Fun() as an additional last argument.

disconnect(This::wxEvtHandler(), EventType::wxEventType()) -> boolean()
Equivalent to disconnect(This, EventType, [])

disconnect(This::wxEvtHandler(), EventType::wxEventType(), Opts::[Option]) ->
boolean()
Types:
    Option = {id, integer()} | {lastId, integer()} | {callback, function()}
See external documentation This function unsubscribes the process or callback fun from the event handler. EventType may be the atom 'null' to match any eventtype. Notice that the options skip and userdata is not used to match the eventhandler.
wxFileDataObject

Erlang module

See external documentation: wxFileDataObject.
This class is derived (and can use functions) from: wxDataObject

DATA TYPES
wxFileDataObject()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFileDataObject()
See external documentation.

addFile(This, Filename) -> ok
Types:
    This = wxFileDataObject()
    Filename = unicode:chardata()
See external documentation.

getFilenames(This) -> [unicode:charlist()]
Types:
    This = wxFileDataObject()
See external documentation.

destroy(This::wxFileDataObject()) -> ok
Destroys this object, do not use object again
wxFileDialog

Erlang module

See external documentation: `wxFileDialog`.

This class is derived (and can use functions) from:

* `wxDialog`
* `wxTopLevelWindow`
* `wxWindow`
* `wxEvtHandler`

**DATA TYPES**

`wxFileDialog()`

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new(Parent) -> wxFileDialog()`

Types:

* `Parent = ` `wxWindow:wxWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxFileDialog()`

Types:

* `Parent = ` `wxWindow:wxWindow()`
* `Option = ` `{message, `unicode:chardata()`} | `{defaultDir, `unicode:chardata()`} |
  `{defaultFile, `unicode:chardata()`} | `{wildCard, `unicode:chardata()`} |
  `{style, integer()} | `{pos, `{X::integer(), Y::integer()}} | `{sz, `{W::integer(), H::integer()}}`

See external documentation.

`getDirectory(This) -> unicode:charlist()`

Types:

* `This = ` `wxFileDialog()`

See external documentation.

`getFilename(This) -> unicode:charlist()`

Types:

* `This = ` `wxFileDialog()`

See external documentation.

`getFilenames(This) -> [unicode:charlist()]`

Types:
This = wxFileDialog()
See external documentation.

getFilterIndex(This) -> integer()
Types:
  This = wxFileDialog()
See external documentation.

getMessage(This) -> unicode:charlist()
Types:
  This = wxFileDialog()
See external documentation.

getPath(This) -> unicode:charlist()
Types:
  This = wxFileDialog()
See external documentation.

getPaths(This) -> [unicode:charlist()]
Types:
  This = wxFileDialog()
See external documentation.

getWildcard(This) -> unicode:charlist()
Types:
  This = wxFileDialog()
See external documentation.

setDirectory(This, Dir) -> ok
Types:
  This = wxFileDialog()
  Dir = unicode:chardata()
See external documentation.

setFilename(This, Name) -> ok
Types:
  This = wxFileDialog()
  Name = unicode:chardata()
See external documentation.

setFilterIndex(This, FilterIndex) -> ok
Types:
  This = wxFileDialog()
FilterIndex = integer()
See external documentation.

setMessage(This, Message) -> ok
Types:
    This = wxFileDialog()
    Message = unicode:chardata()
See external documentation.

setPath(This, Path) -> ok
Types:
    This = wxFileDialog()
    Path = unicode:chardata()
See external documentation.

setWildcard(This, WildCard) -> ok
Types:
    This = wxFileDialog()
    WildCard = unicode:chardata()
See external documentation.

destroy(This::wxFileDialog()) -> ok
Destroys this object, do not use object again
wxFileDirPickerEvent

Erlang module

See external documentation: `wxFileDirPickerEvent`.

Use `wxEvtHandler:connect/3` with EventType:

```erlang
    command_filepicker_changed, command_dirpicker_changed
```

See also the message variant `#wxFileDirPicker{}` event record type.

This class is derived (and can use functions) from:

- `wxCommandEvent`
- `wxEvent`

**DATA TYPES**

`wxFileDirPickerEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

```erlang
getPath(This) -> unicode:charlist()
```

Types:

```erlang
    This = wxFileDirPickerEvent()
```

See external documentation.
wxFilePickerCtrl

Erlang module

See external documentation: wxFilePickerCtrl.

This class is derived (and can use functions) from:
wxPickerBase
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxFilePickerCtrl()

An object reference. The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFilePickerCtrl()
See external documentation.

ew(Parent, Id) -> wxFilePickerCtrl()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxFilePickerCtrl()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {path, unicode:chardata()} | {message, unicode:chardata()} | {wildcard, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
  This = wxFilePickerCtrl()
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to create(This, Parent, Id, []).
create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
  This = wxFilePickerCtrl()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {path, unicode:chardata()} | {message, unicode:chardata()} | 
            {wildcard, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | 
            {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, 
            wx:wx_object()}
See external documentation.

getPath(This) -> unicode:charlist()
Types:
  This = wxFilePickerCtrl()
See external documentation.

setPath(This, Str) -> ok
Types:
  This = wxFilePickerCtrl()
  Str = unicode:chardata()
See external documentation.

destroy(This::wxFilePickerCtrl()) -> ok
Destroys this object, do not use object again
wxFindReplaceData

Erlang module

See external documentation: *wxFindReplaceData*.

**DATA TYPES**

*wxFindReplaceData()*

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxFindReplaceData()

See external documentation.

new(Flags) -> wxFindReplaceData()

Types:

\[ Flags = integer() \]

See external documentation.

getFindString(This) -> unicode:charlist()

Types:

\[ This = wxFindReplaceData() \]

See external documentation.

getReplaceString(This) -> unicode:charlist()

Types:

\[ This = wxFindReplaceData() \]

See external documentation.

getFlags(This) -> integer()

Types:

\[ This = wxFindReplaceData() \]

See external documentation.

setFlags(This, Flags) -> ok

Types:

\[ This = wxFindReplaceData() \]
\[ Flags = integer() \]

See external documentation.
setFindString(This, Str) -> ok
Types:
   This = wxFindReplaceData()
   Str = unicode:chardata()
See external documentation.

setReplaceString(This, Str) -> ok
Types:
   This = wxFindReplaceData()
   Str = unicode:chardata()
See external documentation.

destroy(This::wxFindReplaceData()) -> ok
Destroys this object, do not use object again
wxFindReplaceDialog

Erlang module

See external documentation: wxFindReplaceDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES
wxFindReplaceDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFindReplaceDialog()

See external documentation.

new(Parent, Data, Title) -> wxFindReplaceDialog()
Types:
  Parent = wxWindow:wxWindow()
  Data = wxFindReplaceData:wxFindReplaceData()
  Title = unicode:chardata()
Equivalent to new(Parent, Data, Title, []).

new(Parent, Data, Title, Options::[Option]) -> wxFindReplaceDialog()
Types:
  Parent = wxWindow:wxWindow()
  Data = wxFindReplaceData:wxFindReplaceData()
  Title = unicode:chardata()
  Option = {style, integer()}
See external documentation.

create(This, Parent, Data, Title) -> boolean()
Types:
  This = wxFindReplaceDialog()
  Parent = wxWindow:wxWindow()
  Data = wxFindReplaceData:wxFindReplaceData()
  Title = unicode:chardata()
Equivalent to create(This, Parent, Data, Title, []).
create(This, Parent, Data, Title, Options::[Option]) -> boolean()
Types:
    This = wxFindReplaceDialog()
    Parent = wxWindow:wxWindow()
    Data = wxFindReplaceData:wxFindReplaceData()
    Title = unicode:chardata()
    Option = {style, integer()}
See external documentation.

dataGet(This) -> wxFindReplaceData:wxFindReplaceData()
Types:
    This = wxFindReplaceDialog()
See external documentation.

destroy(This::wxFindReplaceDialog()) -> ok
Destroys this object, do not use object again
wxFlexGridSizer

Erlang module

See external documentation: \texttt{wxFlexGridSizer}.

This class is derived (and can use functions) from:
\texttt{wxGridSizer}
\texttt{wxSizer}

\textbf{DATA TYPES}

\texttt{wxFlexGridSizer()}

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

\textbf{Exports}

\texttt{new(Cols) -> wxFlexGridSizer()}

Types:
\[
\text{Cols} = \text{integer()}
\]

Equivalent to \texttt{new(Cols, [])}.

\texttt{new(Cols, Options::[Option]) -> wxFlexGridSizer()}

Types:
\[
\text{Cols} = \text{integer()}
\]
\[
\text{Option} = \{\text{vgap}, \text{integer()}) \mid \{\text{hgap}, \text{integer()})
\]

See external documentation.

\texttt{new(Rows, Cols, Vgap, Hgap) -> wxFlexGridSizer()}

Types:
\[
\text{Rows} = \text{integer()}
\]
\[
\text{Cols} = \text{integer()}
\]
\[
\text{Vgap} = \text{integer()}
\]
\[
\text{Hgap} = \text{integer()}
\]

See external documentation.

\texttt{addGrowableCol(This, Idx) -> ok}

Types:
\[
\text{This} = \text{wxFlexGridSizer()}
\]
\[
\text{Idx} = \text{integer()}
\]

Equivalent to \texttt{addGrowableCol(This, Idx, [])}.

\texttt{addGrowableCol(This, Idx, Options::[Option]) -> ok}

Types:
This = wxFlexGridSizer()
Idx = integer()
Option = {proportion, integer()}
See external documentation.

addGrowableRow(This, Idx) -> ok
Types:
  This = wxFlexGridSizer()
  Idx = integer()
Equivalent to addGrowableRow(This, Idx, []).

addGrowableRow(This, Idx, Options::[Option]) -> ok
Types:
  This = wxFlexGridSizer()
  Idx = integer()
  Option = {proportion, integer()}
See external documentation.

getFlexibleDirection(This) -> integer()
Types:
  This = wxFlexGridSizer()
See external documentation.

getNonFlexibleGrowMode(This) -> wx:wx_enum()
Types:
  This = wxFlexGridSizer()
See external documentation.
Res = ?wxFLEX_GROWMODE_NONE | ?wxFLEX_GROWMODE_SPECIFIED | ?wxFLEX_GROWMODE_ALL

removeGrowableCol(This, Idx) -> ok
Types:
  This = wxFlexGridSizer()
  Idx = integer()
See external documentation.

removeGrowableRow(This, Idx) -> ok
Types:
  This = wxFlexGridSizer()
  Idx = integer()
See external documentation.

setFlexibleDirection(This, Direction) -> ok
Types:
wxFlexGridSizer

    This = wxFlexGridSizer()
    Direction = integer()

See external documentation.

setNonFlexibleGrowMode(This, Mode) -> ok
Types:
    This = wxFlexGridSizer()
    Mode = wx:wx_enum()

See external documentation.

Mode = ?wxFLEX_GROWMODE_NONE | ?wxFLEX_GROWMODE_SPECIFIED | ?wxFLEX_GROWMODE_ALL

destroy(This::wxFlexGridSizer()) -> ok

Destroys this object, do not use object again
wxFocusEvent
Erlang module

See external documentation: wxFocusEvent.
Use WXEvtHandler:connect/3 with EventType:
  set_focus, kill_focus
See also the message variant #wxFocus{} event record type.
This class is derived (and can use functions) from:
wxEvent

DATA TYPES
wxFocusEvent()
  An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

getWindow(This) -> wxWindow:wxWindow()
Types:
  This = wxFocusEvent()
See external documentation.
wxFont

Erlang module

See external documentation: \textit{wxFont}.

\textbf{DATA TYPES}

\texttt{wxFont()}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

\textbf{Exports}

\texttt{new() \rightarrow wxFont()}

See external documentation.

\texttt{new(Fontname) \rightarrow wxFont()}

Types:

\texttt{Fontname = unicode:chardata()}

See external documentation.

\texttt{new(Size, Family, Style, Weight) \rightarrow wxFont()}

Types:

\begin{itemize}
  \item \texttt{Size = integer()}
  \item \texttt{Family = wx:wx_enum()}
  \item \texttt{Style = wx:wx_enum()}
  \item \texttt{Weight = integer()}
\end{itemize}

Equivalent to \texttt{new(Size, Family, Style, Weight, []).}

\texttt{new(Size, Family, Style, Weight, Options::[Option]) \rightarrow wxFont()}

Types:

\begin{itemize}
  \item \texttt{Size = integer()}
  \item \texttt{Family = wx:wx_enum()}
  \item \texttt{Style = wx:wx_enum()}
  \item \texttt{Weight = integer()}
  \item \texttt{Option = \{underlined, boolean()\} \mid \{face, unicode:chardata()\} \mid \{encoding, \texttt{wx:wx_enum()\}}
\end{itemize}

See external documentation.

\begin{itemize}
  \item \texttt{Encoding = \texttt{wxFONTENCODING_SYSTEM} \mid \texttt{wxFONTENCODING_DEFAULT} \mid \texttt{wxFONTENCODING_ISO8859_1} \mid \texttt{wxFONTENCODING_ISO8859_2} \mid \texttt{wxFONTENCODING_ISO8859_3} \mid \texttt{wxFONTENCODING_ISO8859_4} \mid \texttt{wxFONTENCODING_ISO8859_5} \mid \texttt{wxFONTENCODING_ISO8859_6} \mid \texttt{wxFONTENCODING_ISO8859_7} \mid \texttt{wxFONTENCODING_ISO8859_8} \mid \texttt{wxFONTENCODING_ISO8859_9} \mid \texttt{wxFONTENCODING_ISO8859_10} \mid \texttt{wxFONTENCODING_ISO8859_11} \mid \texttt{wxFONTENCODING_ISO8859_12} \mid \texttt{wxFONTENCODING_ISO8859_13} \mid \texttt{wxFONTENCODING_ISO8859_14} \mid \texttt{wxFONTENCODING_ISO8859_15}}
\end{itemize}
wxFont

| ?wxFONTENCODING_ISO8859_MAX | ?wxFONTENCODING_KOI8 | ?wxFONTENCODING_KOI8_U |
| ?wxFONTENCODING_ALTERNATIVE | ?wxFONTENCODING_BULGARIAN | ? |
wxFONTENCODING_CP437 | ?wxFONTENCODING_CP850 | ?wxFONTENCODING_CP852 |
wxFONTENCODING_CP855 | ?wxFONTENCODING_CP866 | ?wxFONTENCODING_CP874 |
wxFONTENCODING_CP932 | ?wxFONTENCODING_CP936 | ?wxFONTENCODING_CP949 |
wxFONTENCODING_CP950 | ?wxFONTENCODING_CP1250 | ?wxFONTENCODING_CP1251 |
wxFONTENCODING_CP1252 | ?wxFONTENCODING_CP1253 | ?wxFONTENCODING_CP1254 |
wxFONTENCODING_CP1255 | ?wxFONTENCODING_CP1256 | ?wxFONTENCODING_CP1257 |
wxFONTENCODING_CP12_MAX | ?wxFONTENCODING_UTF7 | ?wxFONTENCODING_UTF8 |
wxFONTENCODING_UTF32BE | ?wxFONTENCODING_UTF32LE | ?wxFONTENCODING_MACROMAN |
?wxFONTENCODING_MACJAPANESE | ?wxFONTENCODING_MACMACHINESETRAD | |
wxFONTENCODING_MACKOREAN | ?wxFONTENCODING_MACARABIC | |
wxFONTENCODING_MACHEBREW | ?wxFONTENCODING_MACGREEK | |
wxFONTENCODING_MACCYRILLIC | ?wxFONTENCODING_MACDEVANAGARI | |
wxFONTENCODING_MACGUURMUKHI | ?wxFONTENCODING_MACGUJARATI | |
wxFONTENCODING_MACORIYA | ?wxFONTENCODING_MACBENGALI | |
wxFONTENCODING_MACTAMIIL | ?wxFONTENCODING_MACTELUGU | |
wxFONTENCODING_MACKANNADA | ?wxFONTENCODING_MACMALAIJALAM | |
wxFONTENCODING_MACSINHALESE | ?wxFONTENCODING_MACBURMESE | |
wxFONTENCODING_MACKHMER | ?wxFONTENCODING_MACTHAI | |
wxFONTENCODING_MACLAOTIAN | ?wxFONTENCODING_MACGEORGIAN | |
wxFONTENCODING_MACARMENIAN | ?wxFONTENCODING_MACCHINESESIMP | |
wxFONTENCODING_MACTIBETAN | ?wxFONTENCODING_MACMONGOLIAN | |
wxFONTENCODING_MACETHIOPIC | ?wxFONTENCODING_MACCENTRALEUR | |
wxFONTENCODING_MACVIATNAMESE | ?wxFONTENCODING_MACARABICEXT | |
wxFONTENCODING_MACSYMBOL | ?wxFONTENCODING_MACDINGBATS | |
wxFONTENCODING_MACTURKISH | ?wxFONTENCODING_MACCROATIAN | |
wxFONTENCODING_MACICELANDIC | ?wxFONTENCODING_MACROMANIAN | |
wxFONTENCODING_MACGELTIC | ?wxFONTENCODING_MACGAELEIC | |
wxFONTENCODING_MACKEYBOARD | ?wxFONTENCODING_MAX | ?wxFONTENCODING_MACMIN |
?wxFONTENCODING_MACMAC | ?wxFONTENCODING_UTF16 | ?wxFONTENCODING_UTF32 | |
xFontEncoding_UNICODE | ?wxFONTENCODING_GB2312 | ?wxFONTENCODING_BIG5 | |
xFontEncoding_SHIFT_JIS |

Family = ?wxFONTFAMILY_DEFAULT | ?wxFONTFAMILY_DECORATIVE | ?wxFONTFAMILY_ROMAN |
?wxFONTFAMILY_SCRIPT | ?wxFONTFAMILY_SWISS | ?wxFONTFAMILY_MODERN | |
xFontFamily_TELETYP | ?wxFONTFAMILY_MAX | ?wxFONTFAMILY_UNKNOWN |

Style = ?wxFONTSTYLE_NORMAL | ?wxFONTSTYLE_ITALIC | ?wxFONTSTYLE_SLANT | |
xFontStyle_MAX |

isFixedWidth(This) -> boolean()

Types:

This = wxFont()

See external documentation.

ggetDefaultEncoding() -> wx:wx_enum()

See external documentation.
wxFont

getFaceName(This) -> unicode:charlist()

Types:

This = wxFont()

See external documentation.

getFamily(This) -> wx:wx_enum()

Types:

This = wxFont()

See external documentation.


180 | Ericsson AB. All Rights Reserved.: wxErlang
getNativeFontInfoDesc(This) -> unicode:charlist()
Types:
   This = wxFont()
See external documentation.

generateNativeFontInfoUserDesc(This) -> unicode:charlist()
Types:
   This = wxFont()
See external documentation.

generatePointSize(This) -> integer()
Types:
   This = wxFont()
See external documentation.

generateStyle(This) -> wx:wx_enum()
Types:
   This = wxFont()
See external documentation.

getUnderlined(This) -> boolean()
Types:
   This = wxFont()
See external documentation.

getWeight(This) -> integer()
Types:
   This = wxFont()
See external documentation.

ok(This) -> boolean()
Types:
   This = wxFont()
See external documentation.

setDefaultEncoding(Encoding) -> ok
Types:
   Encoding = wx:wx_enum()
See external documentation.

setFaceName(This, FaceName) -> boolean()

Types:

This = wxFont()

FaceName = unicode:chardata()

See external documentation.

setFamily(This, Family) -> ok

Types:

This = wxFont()

Family = wx:wx_enum()

See external documentation.
Family = wxFONTFAMILY_DEFAULT | wxFONTFAMILY_DECORATIVE | wxFONTFAMILY_ROMAN | wxFONTFAMILY_SCRIPT | wxFONTFAMILY_SWISS | wxFONTFAMILY_MODERN | wxFONTFAMILY_TELETYPE | wxFONTFAMILY_MAX | wxFONTFAMILY_UNKNOWN

setPointSize(This, PointSize) -> ok
Types:
   This = wxFont()
   PointSize = integer()
See external documentation.

setStyle(This, Style) -> ok
Types:
   This = wxFont()
   Style = wx:wx_enum()
See external documentation.
Style = wxFONTSTYLE_NORMAL | wxFONTSTYLE_ITALIC | wxFONTSTYLE_SLANT | wxFONTSTYLE_MAX

setUnderlined(This, Underlined) -> ok
Types:
   This = wxFont()
   Underlined = boolean()
See external documentation.

setWeight(This, Weight) -> ok
Types:
   This = wxFont()
   Weight = integer()
See external documentation.

destroy(This::wxFont()) -> ok
Destroys this object, do not use object again
wxFontData

Erlang module

See external documentation: wxFontData.

DATA TYPES

wxFontData()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFontData()
See external documentation.

data(Data) -> wxFontData()
Types:
  Data = wxFontData()
See external documentation.

enableEffects(This, Flag) -> ok
Types:
  This = wxFontData()
  Flag = boolean()
See external documentation.

getAllowSymbols(This) -> boolean()
Types:
  This = wxFontData()
See external documentation.

getColour(This) -> wx:wx_colour4()
Types:
  This = wxFontData()
See external documentation.

getChosenFont(This) -> wxFont:wxFont()
Types:
  This = wxFontData()
See external documentation.
getEnableEffects(This) -> boolean()
Types:
    This = wxFontData()
See external documentation.

getInitialFont(This) -> wxFont:wxFont()
Types:
    This = wxFontData()
See external documentation.

getShowHelp(This) -> boolean()
Types:
    This = wxFontData()
See external documentation.

setAllowSymbols(This, Flag) -> ok
Types:
    This = wxFontData()
    Flag = boolean()
See external documentation.

setChosenFont(This, Font) -> ok
Types:
    This = wxFontData()
    Font = wxFont:wxFont()
See external documentation.

setColour(This, Colour) -> ok
Types:
    This = wxFontData()
    Colour = wx:wx_colour()
See external documentation.

setInitialFont(This, Font) -> ok
Types:
    This = wxFontData()
    Font = wxFont:wxFont()
See external documentation.

setRange(This, MinRange, MaxRange) -> ok
Types:
    This = wxFontData()
    MinRange = integer()
MaxRange = integer()

See external documentation.

setShowHelp(This, Flag) -> ok

Types:
This = wxFontData()
Flag = boolean()

See external documentation.

destroy(This::wxFontData()) -> ok

Destroys this object, do not use object again
wxFontDialog

Erlang module

See external documentation: wxFontDialog.
This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES
wxFontDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFontDialog()
See external documentation.

ew(Parent, Data) -> wxFontDialog()
Types:
Parent = wxWindow:wxWindow()
Data = wxFontData:wxFontData()
See external documentation.

create(This, Parent, Data) -> boolean()
Types:
This = wxFontDialog()
Parent = wxWindow:wxWindow()
Data = wxFontData:wxFontData()
See external documentation.

gETCHDesigned(This) -> wxFontData:wxFontData()
Types:
This = wxFontDialog()
See external documentation.

destroy(This::wxFontDialog()) -> ok
Destroys this object, do not use object again
wxFontPickerCtrl

Erlang module

See external documentation: **wxFontPickerCtrl**.

This class is derived (and can use functions) from:
- **wxPickerBase**
- **wxControl**
- **wxWindow**
- **wxEvtHandler**

**DATA TYPES**

**wxFontPickerCtrl()**

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxFontPickerCtrl()

See external documentation.

new(Parent, Id) -> wxFontPickerCtrl()

Types:
- **Parent** = *wxWindow:*wxWindow()
- **Id** = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxFontPickerCtrl()

Types:
- **Parent** = *wxWindow:*wxWindow()
- **Id** = integer()
- **Option** = {initial, *wxFont:*wxFont()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, *wx:*wx_object()}

See external documentation.

create(This, Parent, Id) -> boolean()

Types:
- **This** = wxFontPickerCtrl()
- **Parent** = *wxWindow:*wxWindow()
- **Id** = integer()

Equivalent to create(This, Parent, Id, []).
create(This, Parent, Id, Options::[Option]) -> boolean()

Types:
  This = wxFontPickerCtrl()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {initial, wxFont:wxFont()} | {pos, {X::integer(), Y::integer()}}
       | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx::wx_object()}

See external documentation.

getSelectedFont(This) -> wxFont:wxFont()

Types:
  This = wxFontPickerCtrl()

See external documentation.

setSelectedFont(This, F) -> ok

Types:
  This = wxFontPickerCtrl()
  F = wxFont:wxFont()

See external documentation.

getMaxPointSize(This) -> integer()

Types:
  This = wxFontPickerCtrl()

See external documentation.

setMaxPointSize(This, Max) -> ok

Types:
  This = wxFontPickerCtrl()
  Max = integer()

See external documentation.

destroy(This::wxFontPickerCtrl()) -> ok

Destroys this object, do not use object again
wxFontPickerEvent

Erlang module

See external documentation: wxFontPickerEvent.

Use wxEvtHandler:connect/3 with EventType:

    command_fontpicker_changed

See also the message variant #wxFontPicker{} event record type.

This class is derived (and can use functions) from:

wxCommandEvent
wxEvent

DATA TYPES

wxFontPickerEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

gSetFont(This) -> wxFont:wxFont()

Types:

    This = wxFontPickerEvent()

See external documentation.
wxFrame

Erlang module

See external documentation: wxFrame.

This class is derived (and can use functions) from:
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxFrame()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxFrame()

See external documentation.

ew(Parent, Id, Title) -> wxFrame()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()

Equivalent to new(Parent, Id, Title, []).

new(Parent, Id, Title, Options::[Option]) -> wxFrame()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

create(This, Parent, Id, Title) -> boolean()

Types:
  This = wxFrame()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()

Equivalent to create(This, Parent, Id, Title, []).
create(This, Parent, Id, Title, Options::[Option]) -> boolean()

Types:
  This = wxFrame()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

createStatusBar(This) -> wxStatusBar:wxStatusBar()

Types:
  This = wxFrame()

Equivalent to createStatusBar(This, []).

createStatusBar(This, Options::[Option]) -> wxStatusBar:wxStatusBar()

Types:
  This = wxFrame()
  Option = {number, integer()} | {style, integer()} | {id, integer()}

See external documentation.

createToolBar(This) -> wxToolBar:wxToolBar()

Types:
  This = wxFrame()

Equivalent to createToolBar(This, []).

createToolBar(This, Options::[Option]) -> wxToolBar:wxToolBar()

Types:
  This = wxFrame()
  Option = {style, integer()} | {id, integer()}

See external documentation.

clientAreaOrigin(This) -> {X::integer(), Y::integer()}

Types:
  This = wxFrame()

See external documentation.

getMenuBar(This) -> wxMenuBar:wxMenuBar()

Types:
  This = wxFrame()

See external documentation.
getStatusBar(This) -> wxStatusBar:wxStatusBar()
Types:
   This = wxFrame()
See external documentation.

getStatusBarPane(This) -> integer()
Types:
   This = wxFrame()
See external documentation.

getToolBar(This) -> wxToolBar:wxToolBar()
Types:
   This = wxFrame()
See external documentation.

processCommand(This, Winid) -> boolean()
Types:
   This = wxFrame()
   Winid = integer()
See external documentation.

sendSizeEvent(This) -> ok
Types:
   This = wxFrame()
See external documentation.

setMenuBar(This, Menubar) -> ok
Types:
   This = wxFrame()
   Menubar = wxMenuBar:wxMenuBar()
See external documentation.

setStatusBar(This, Statbar) -> ok
Types:
   This = wxFrame()
   Statbar = wxStatusBar:wxStatusBar()
See external documentation.

setStatusBarPane(This, N) -> ok
Types:
   This = wxFrame()
   N = integer()
See external documentation.
setStatusText(This, Text) -> ok
Types:
   This = wxFrame()
   Text = unicode:chardata()
Equivalent to setStatusText(This, Text, []).

setStatusText(This, Text, Options::[Option]) -> ok
Types:
   This = wxFrame()
   Text = unicode:chardata()
   Option = {number, integer()}
See external documentation.

setStatusWidths(This, Widths_field) -> ok
Types:
   This = wxFrame()
   Widths_field = [integer()]
See external documentation.

setToolBar(This, Toolbar) -> ok
Types:
   This = wxFrame()
   Toolbar = wxToolBar:wxToolBar()
See external documentation.

destroy(This::wxFrame()) -> ok
Destroys this object, do not use object again
wxGBSizerItem

Erlang module

See external documentation: wxGBSizerItem.

This class is derived (and can use functions) from:
wxSizerItem

DATA TYPES

wxGBSizerItem()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxGCDC

Erlang module

See external documentation: wxGCDC.

This class is derived (and can use functions) from:
wxDC

DATA TYPES

wxGCDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGCDC()
See external documentation.

new(Dc) -> wxGCDC()
Types:
  Dc = wxWindowDC:wxWindowDC()
See external documentation.

getGraphicsContext(This) -> wxGraphicsContext:wxGraphicsContext()
Types:
  This = wxGCDC()
See external documentation.

setGraphicsContext(This, Ctx) -> ok
Types:
  This = wxGCDC()
  Ctx = wxGraphicsContext:wxGraphicsContext()
See external documentation.

destroy(This::wxGCDC()) -> ok
Destroys this object, do not use object again
wxGLCanvas

Erlang module

See external documentation: wxGLCanvas.

This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES

wxGLCanvas()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Parent) -> wxGLCanvas()
Types:
  Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Shared) -> wxGLCanvas()
Types:
  Parent = wxWindow:wxWindow()
  Shared = wx:wx_object() | wxGLCanvas()
See external documentation.
Also:
new(Parent, [Option]) -> wxGLCanvas() when
Parent::wxWindow:wxWindow(),
Option :: [{'id', integer()}]
  | [{'pos', [X::integer(), Y::integer()]}]
  | [{'size', [W::integer(), H::integer()]}]
  | [{'style', integer()}]
  | [{'name', unicode:chardata()}]
  | [{'attribList', [integer()]}]
  | [{'palette', wxPalette:wxPalette()}].

new(Parent, Shared, Options::[Option]) -> wxGLCanvas()
Types:
  Parent = wxWindow:wxWindow()
  Shared = wx:wx_object() | wxGLCanvas()
  Option = {id, integer()} | {pos, [X::integer(), Y::integer()]}|
  | {size, [W::integer(), H::integer()]} | {style, integer()} | |
  | {name, unicode:chardata()} | {attribList, [integer()]} | {palette, wxPalette:wxPalette()}

Ericsson AB. All Rights Reserved.: wxErlang | 197
See external documentation.

`getContext(This) -> wx:wx_object()`
Types:

     This = `wxGLCanvas()`
See external documentation.

`setCurrent(This) -> ok`
Types:

     This = `wxGLCanvas()`
See external documentation.

`swapBuffers(This) -> ok`
Types:

     This = `wxGLCanvas()`
See external documentation.

`destroy(This::wxGLCanvas()) -> ok`
Destroys this object, do not use object again
wxGauge

Erlang module

See external documentation: wxGauge.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxGauge()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGauge()

See external documentation.

new(Parent, Id, Range) -> wxGauge()
Types:

Parent = wxWindow:wxWindow()
Id = integer()
Range = integer()

Equivalent to new(Parent, Id, Range, []).

new(Parent, Id, Range, Options::[Option]) -> wxGauge()
Types:

Parent = wxWindow:wxWindow()
Id = integer()
Range = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Range) -> boolean()
Types:

This = wxGauge()
Parent = wxWindow:wxWindow()
Id = integer()
Range = integer()

Equivalent to create(This, Parent, Id, Range, []).
wxGauge

create(This, Parent, Id, Range, Options::[Option]) -> boolean()
Types:
   This = wxGauge()
   Parent = wxWindow:wxWindow()
   Id = integer()
   Range = integer()
   Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}
See external documentation.

getRange(This) -> integer()
Types:
   This = wxGauge()
See external documentation.

getValue(This) -> integer()
Types:
   This = wxGauge()
See external documentation.

isVertical(This) -> boolean()
Types:
   This = wxGauge()
See external documentation.

setRange(This, R) -> ok
Types:
   This = wxGauge()
   R = integer()
See external documentation.

setValue(This, Pos) -> ok
Types:
   This = wxGauge()
   Pos = integer()
See external documentation.

pulse(This) -> ok
Types:
   This = wxGauge()
See external documentation.
destroy(This::wxGauge()) -> ok
Destroys this object, do not use object again
wxGenericDirCtrl

Erlang module

See external documentation: wxGenericDirCtrl.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxGenericDirCtrl()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGenericDirCtrl()

See external documentation.

new(Parent) -> wxGenericDirCtrl()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxGenericDirCtrl()
Types:
    Parent = wxWindow:wxWindow()
    Option = {id, integer()} | {dir, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} |
    {style, integer()} | {filter, unicode:chardata()} | {defaultFilter, integer()}
See external documentation.

create(This, Parent) -> boolean()
Types:
    This = wxGenericDirCtrl()
    Parent = wxWindow:wxWindow()
Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()
Types:
    This = wxGenericDirCtrl()
    Parent = wxWindow:wxWindow()
Option = \{id, integer()\} \mid \{dir, unicode:chardata()\} \mid \{pos, X::integer(), Y::integer()\}\mid \{size, W::integer(), H::integer()\} \mid \{style, integer()\} \mid \{filter, unicode:chardata()\} \mid \{defaultFilter, integer()\}

See external documentation.

init(This) -> ok
Types:
   This = wxGenericDirCtrl()

See external documentation.

collapseTree(This) -> ok
Types:
   This = wxGenericDirCtrl()

See external documentation.

expandPath(This, Path) -> boolean()
Types:
   This = wxGenericDirCtrl()
   Path = unicode:chardata()

See external documentation.

getDefaultPath(This) -> unicode:charlist()
Types:
   This = wxGenericDirCtrl()

See external documentation.

getPath(This) -> unicode:charlist()
Types:
   This = wxGenericDirCtrl()

See external documentation.

getFilePath(This) -> unicode:charlist()
Types:
   This = wxGenericDirCtrl()

See external documentation.

getFilter(This) -> unicode:charlist()
Types:
   This = wxGenericDirCtrl()

See external documentation.
getFilterIndex(This) -> integer()
Types:
   This = wxGenericDirCtrl()
See external documentation.

ggetRootId(This) -> integer()
Types:
   This = wxGenericDirCtrl()
See external documentation.

getTreeCtrl(This) -> wxTreeCtrl:wxTreeCtrl()
Types:
   This = wxGenericDirCtrl()
See external documentation.

reCreateTree(This) -> ok
Types:
   This = wxGenericDirCtrl()
See external documentation.

setDefaultPath(This, Path) -> ok
Types:
   This = wxGenericDirCtrl()
   Path = unicode:chardata()
See external documentation.

setFilter(This, Filter) -> ok
Types:
   This = wxGenericDirCtrl()
   Filter = unicode:chardata()
See external documentation.

setFilterIndex(This, N) -> ok
Types:
   This = wxGenericDirCtrl()
   N = integer()
See external documentation.

setPath(This, Path) -> ok
Types:
   This = wxGenericDirCtrl()
   Path = unicode:chardata()
See external documentation.
destroy(This::wxGenericDirCtrl()) -> ok

Destroys this object, do not use object again
wxGraphicsBrush

See external documentation: wxGraphicsBrush.

This class is derived (and can use functions) from:
wxGraphicsObject

DATA TYPES

wxGraphicsBrush()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxGraphicsContext
Erlang module

See external documentation: wxGraphicsContext.

This class is derived (and can use functions) from:
wxGraphicsObject

DATA TYPES
wxGraphicsContext()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

create() -> wxGraphicsContext()
See external documentation.

create(Dc) -> wxGraphicsContext()
Types:
   Dc = wxWindowDC:wxWindowDC() | wxWindow:wxWindow()
See external documentation.

createPen(This, Pen) -> wxGraphicsPen:wxGraphicsPen()
Types:
   This = wxGraphicsContext()
   Pen = wxPen:wxPen()
See external documentation.

createBrush(This, Brush) -> wxGraphicsBrush:wxGraphicsBrush()
Types:
   This = wxGraphicsContext()
   Brush = wxBrush:wxBrush()
See external documentation.

createRadialGradientBrush(This, Xo, Yo, Xc, Yc, Radius, OColor, CColor) ->
wxGraphicsBrush:wxGraphicsBrush()
Types:
   This = wxGraphicsContext()
   Xo = number()
   Yo = number()
   Xc = number()
   Yc = number()
Radius = number()
OColor = wx:wx_colour()
CColor = wx:wx_colour()

See external documentation.

cREATELINEARBRUSH(This, X1, Y1, X2, Y2, C1, C2) -> wxGraphicsBrush:wxGraphicsBrush()
Types:
  This = wxGraphicsContext()
  X1 = number()
  Y1 = number()
  X2 = number()
  Y2 = number()
  C1 = wx:wx_colour()
  C2 = wx:wx_colour()

See external documentation.

cREATEFONT(This, Font) -> wxGraphicsFont:wxGraphicsFont()
Types:
  This = wxGraphicsContext()
  Font = wxFont:wxFont()

Equivalent to createFont(This, Font, []).

cREATEFONT(This, Font, Options::[Option]) -> wxGraphicsFont:wxGraphicsFont()
Types:
  This = wxGraphicsContext()
  Font = wxFont:wxFont()
  Option = {col, wx:wx_colour()}

See external documentation.

createMatrix(This) -> wxGraphicsMatrix:wxGraphicsMatrix()
Types:
  This = wxGraphicsContext()

Equivalent to createMatrix(This, []).

cREATEMATRIX(This, Options::[Option]) -> wxGraphicsMatrix:wxGraphicsMatrix()
Types:
  This = wxGraphicsContext()
  Option = {a, number()} | {b, number()} | {c, number()} | {d, number()} | {tx, number()} | {ty, number()}

See external documentation.
createPath(This) -> wxGraphicsPath:wxGraphicsPath()
Types:
    This = wxGraphicsContext()
See external documentation.

clip(This, Region) -> ok
Types:
    This = wxGraphicsContext()
    Region = wxRegion:wxRegion()
See external documentation.

clip(This, X, Y, W, H) -> ok
Types:
    This = wxGraphicsContext()
    X = number()
    Y = number()
    W = number()
    H = number()
See external documentation.

resetClip(This) -> ok
Types:
    This = wxGraphicsContext()
See external documentation.

drawBitmap(This, Bmp, X, Y, W, H) -> ok
Types:
    This = wxGraphicsContext()
    Bmp = wxBitmap:wxBitmap()
    X = number()
    Y = number()
    W = number()
    H = number()
See external documentation.

drawEllipse(This, X, Y, W, H) -> ok
Types:
    This = wxGraphicsContext()
    X = number()
    Y = number()
    W = number()
    H = number()
See external documentation.
wxGraphicsContext

drawIcon(This, Icon, X, Y, W, H) -> ok
Types:
  This = wxGraphicsContext()
  Icon = wxIcon:wxIcon()
  X = number()
  Y = number()
  W = number()
  H = number()
See external documentation.

drawLines(This, Points) -> ok
Types:
  This = wxGraphicsContext()
  Points = [{X::float(), Y::float()}]
Equivalent to drawLines(This, Points, []).

drawLines(This, Points, Options::*[Option]) -> ok
Types:
  This = wxGraphicsContext()
  Points = [{X::float(), Y::float()}]
  Option = {fillStyle, wx:wx_enum()}
See external documentation.
FillStyle = integer

drawPath(This, Path) -> ok
Types:
  This = wxGraphicsContext()
  Path = wxGraphicsPath:wxGraphicsPath()
Equivalent to drawPath(This, Path, []).

drawPath(This, Path, Options::*[Option]) -> ok
Types:
  This = wxGraphicsContext()
  Path = wxGraphicsPath:wxGraphicsPath()
  Option = {fillStyle, wx:wx_enum()}
See external documentation.
FillStyle = integer

drawRectangle(This, X, Y, W, H) -> ok
Types:
  This = wxGraphicsContext()
  X = number()
  Y = number()
W = number()
H = number()

See external documentation.

drawRoundedRectangle(This, X, Y, W, H, Radius) -> ok
Types:
   This = wxGraphicsContext()
   X = number()
   Y = number()
   W = number()
   H = number()
   Radius = number()

See external documentation.

drawText(This, Str, X, Y) -> ok
Types:
   This = wxGraphicsContext()
   Str = unicode:chardata()
   X = number()
   Y = number()

See external documentation.

drawText(This, Str, X, Y, Angle) -> ok
Types:
   This = wxGraphicsContext()
   Str = unicode:chardata()
   X = number()
   Y = number()
   Angle = number()

See external documentation.

Also:

Also:
drawText(This, Str, X, Y, BackgroundBrush) -> 'ok' when
This::wxGraphicsContext(), Str::unicode:chardata(), X::number(), Y::number(),
BackgroundBrush::wxGraphicsBrush:wxGraphicsBrush().

drawText(This, Str, X, Y, Angle, BackgroundBrush) -> ok
Types:
   This = wxGraphicsContext()
   Str = unicode:chardata()
   X = number()
   Y = number()
   Angle = number()
   BackgroundBrush = wxGraphicsBrush:wxGraphicsBrush()
See external documentation.

`fillPath(This, Path) -> ok`

Types:

- `This = wxGraphicsContext()`
- `Path = wxGraphicsPath:wxGraphicsPath()`

Equivalent to `fillPath(This, Path, [])`.

`fillPath(This, Path, Options::[Option]) -> ok`

Types:

- `This = wxGraphicsContext()`
- `Path = wxGraphicsPath:wxGraphicsPath()`
- `Option = {fillStyle, wx:wx_enum()}`

See external documentation.

FillStyle = integer

`strokePath(This, Path) -> ok`

Types:

- `This = wxGraphicsContext()`
- `Path = wxGraphicsPath:wxGraphicsPath()`

See external documentation.

`getPartialTextExtents(This, Text) -> [number()]

Types:

- `This = wxGraphicsContext()`
- `Text = unicode:chardata()`

See external documentation.

`getTextExtent(This, Text) -> Result`

Types:

- `Result = {Width::number(), Height::number(), Descent::number(), ExternalLeading::number()}`
- `This = wxGraphicsContext()`
- `Text = unicode:chardata()`

See external documentation.

`rotate(This, Angle) -> ok`

Types:

- `This = wxGraphicsContext()`
- `Angle = number()`

See external documentation.
scale(This, XScale, YScale) -> ok
Types:
   This = wxGraphicsContext()
   XScale = number()
   YScale = number()
See external documentation.

translate(This, Dx, Dy) -> ok
Types:
   This = wxGraphicsContext()
   Dx = number()
   Dy = number()
See external documentation.

getTransform(This) -> wxGraphicsMatrix:wxGraphicsMatrix()
Types:
   This = wxGraphicsContext()
See external documentation.

setTransform(This, Matrix) -> ok
Types:
   This = wxGraphicsContext()
   Matrix = wxGraphicsMatrix:wxGraphicsMatrix()
See external documentation.

concatTransform(This, Matrix) -> ok
Types:
   This = wxGraphicsContext()
   Matrix = wxGraphicsMatrix:wxGraphicsMatrix()
See external documentation.

setBrush(This, Brush) -> ok
Types:
   This = wxGraphicsContext()
See external documentation.

setFont(This, Font) -> ok
Types:
   This = wxGraphicsContext()
   Font = wxGraphicsFont:wxGraphicsFont()
See external documentation.
wxGraphicsContext

setFont(This, Font, Colour) -> ok
Types:
   This = wxGraphicsContext()
   Font = wxFont:wxFont()
   Colour = wx:wx_colour()
See external documentation.

setPen(This, Pen) -> ok
Types:
   This = wxGraphicsContext()
See external documentation.

strokeLine(This, X1, Y1, X2, Y2) -> ok
Types:
   This = wxGraphicsContext()
   X1 = number()
   Y1 = number()
   X2 = number()
   Y2 = number()
See external documentation.

strokeLines(This, Points) -> ok
Types:
   This = wxGraphicsContext()
   Points = [{X::float(), Y::float()}]
See external documentation.

destroy(This::wxGraphicsContext()) -> ok
Destroys this object, do not use object again
wxGraphicsFont

Erlang module

See external documentation: wxGraphicsFont.

This class is derived (and can use functions) from: wxGraphicsObject

DATA TYPES

wxGraphicsFont()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxGraphicsMatrix

Erlang module

See external documentation: `wxGraphicsMatrix`.

This class is derived (and can use functions) from:

`wxGraphicsObject`

**DATA TYPES**

`wxGraphicsMatrix()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`concat(This, T) -> ok`

Types:

- `This = wxGraphicsMatrix()`
- `T = wxGraphicsMatrix()`

See external documentation.

`get(This) -> Result`

Types:

- `Result = {A::number(), B::number(), C::number(), D::number(), Tx::number(), Ty::number()}`
- `This = wxGraphicsMatrix()`

See external documentation.

`invert(This) -> ok`

Types:

- `This = wxGraphicsMatrix()`

See external documentation.

`isEqual(This, T) -> boolean()`

Types:

- `This = wxGraphicsMatrix()`
- `T = wxGraphicsMatrix()`

See external documentation.

`isIdentity(This) -> boolean()`

Types:

- `This = wxGraphicsMatrix()`

See external documentation.
rotate(This, Angle) -> ok
Types:
   This = wxGraphicsMatrix()
   Angle = number()
See external documentation.

scale(This, XScale, YScale) -> ok
Types:
   This = wxGraphicsMatrix()
   XScale = number()
   YScale = number()
See external documentation.

translate(This, Dx, Dy) -> ok
Types:
   This = wxGraphicsMatrix()
   Dx = number()
   Dy = number()
See external documentation.

set(This) -> ok
Types:
   This = wxGraphicsMatrix()
Equivalent to set(This, []).

set(This, Options::[Option]) -> ok
Types:
   This = wxGraphicsMatrix()
   Option = {a, number()} | {b, number()} | {c, number()} | {d, number()} |
           {tx, number()} | {ty, number()}
See external documentation.

transformPoint(This) -> {X::number(), Y::number()}
Types:
   This = wxGraphicsMatrix()
See external documentation.

transformDistance(This) -> {Dx::number(), Dy::number()}
Types:
   This = wxGraphicsMatrix()
See external documentation.
wxGraphicsObject

Erlang module

See external documentation: `wxGraphicsObject`.

DATA TYPES

wxGraphicsObject()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`getRenderer(This) -> wxGraphicsRenderer:wxGraphicsRenderer()`  
Types:

This = `wxGraphicsObject()`  
See external documentation.

`isNull(This) -> boolean()`  
Types:

This = `wxGraphicsObject()`  
See external documentation.

`destroy(This::wxGraphicsObject()) -> ok`  
Destroys this object, do not use object again
wxGraphicsPath

Erlang module

See external documentation: wxGraphicsPath.

This class is derived (and can use functions) from:
wxGraphicsObject

DATA TYPES

wxGraphicsPath()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparison stored on disc or distributed for use on other nodes.

Exports

moveToPoint(This, P) -> ok

Types:

This = wxGraphicsPath()

P = {X::float(), Y::float()}

See external documentation.

moveToPoint(This, X, Y) -> ok

Types:

This = wxGraphicsPath()

X = number()

Y = number()

See external documentation.

addArc(This, C, R, StartAngle, EndAngle, Clockwise) -> ok

Types:

This = wxGraphicsPath()

C = {X::float(), Y::float()}

R = number()

StartAngle = number()

EndAngle = number()

Clockwise = boolean()

See external documentation.

addArc(This, X, Y, R, StartAngle, EndAngle, Clockwise) -> ok

Types:

This = wxGraphicsPath()

X = number()

Y = number()
wxGraphicsPath

\[
\begin{align*}
R & = \text{number()} \\
\text{StartAngle} & = \text{number()} \\
\text{EndAngle} & = \text{number()} \\
\text{Clockwise} & = \text{boolean()}
\end{align*}
\]

See external documentation.

**addArcToPoint(This, X1, Y1, X2, Y2, R) -> ok**

Types:
\[
\begin{align*}
\text{This} & = \text{wxGraphicsPath()} \\
X1 & = \text{number()} \\
Y1 & = \text{number()} \\
X2 & = \text{number()} \\
Y2 & = \text{number()} \\
R & = \text{number()}
\end{align*}
\]

See external documentation.

**addCircle(This, X, Y, R) -> ok**

Types:
\[
\begin{align*}
\text{This} & = \text{wxGraphicsPath()} \\
X & = \text{number()} \\
Y & = \text{number()} \\
R & = \text{number()}
\end{align*}
\]

See external documentation.

**addCurveToPoint(This, C1, C2, E) -> ok**

Types:
\[
\begin{align*}
\text{This} & = \text{wxGraphicsPath()} \\
C1 & = \{X::float(), Y::float()\} \\
C2 & = \{X::float(), Y::float()\} \\
E & = \{X::float(), Y::float()\}
\end{align*}
\]

See external documentation.

**addCurveToPoint(This, Cx1, Cy1, Cx2, Cy2, X, Y) -> ok**

Types:
\[
\begin{align*}
\text{This} & = \text{wxGraphicsPath()} \\
Cx1 & = \text{number()} \\
Cy1 & = \text{number()} \\
Cx2 & = \text{number()} \\
Cy2 & = \text{number()} \\
X & = \text{number()} \\
Y & = \text{number()}
\end{align*}
\]

See external documentation.
addEllipse(This, X, Y, W, H) -> ok
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
    W = number()
    H = number()
See external documentation.

addLineToPoint(This, P) -> ok
Types:
    This = wxGraphicsPath()
    P = {X::float(), Y::float()}
See external documentation.

addLineToPoint(This, X, Y) -> ok
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
See external documentation.

addPath(This, Path) -> ok
Types:
    This = wxGraphicsPath()
    Path = wxGraphicsPath()
See external documentation.

addQuadCurveToPoint(This, Cx, Cy, X, Y) -> ok
Types:
    This = wxGraphicsPath()
    Cx = number()
    Cy = number()
    X = number()
    Y = number()
See external documentation.

addRectangle(This, X, Y, W, H) -> ok
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
    W = number()
H = number()
See external documentation.

addRoundedRectangle(This, X, Y, W, H, Radius) -> ok
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
    W = number()
    H = number()
    Radius = number()
See external documentation.

closeSubpath(This) -> ok
Types:
    This = wxGraphicsPath()
See external documentation.

contains(This, C) -> boolean()
Types:
    This = wxGraphicsPath()
    C = {X::float(), Y::float()}
Equivalent to contains(This, C, []).

contains(This, X, Y) -> boolean()
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
See external documentation.
Also:
contains(This, C, [Option]) -> boolean() when
This::wxGraphicsPath(), C::{X::float(), Y::float()},
Option :: [{'fillStyle', wx:wx_enum()}].
FillStyle = integer

contains(This, X, Y, Options::[Option]) -> boolean()
Types:
    This = wxGraphicsPath()
    X = number()
    Y = number()
    Option = {fillStyle, wx:wx_enum()}
See external documentation.
FillStyle = integer

getBox(This) -> \{X::float(), Y::float(), W::float(), H::float()\}
Types:
   \text{This} = \text{wxGraphicsPath()}
See external documentation.

getCurrentPoint(This) -> \{X::float(), Y::float()\}
Types:
   \text{This} = \text{wxGraphicsPath()}
See external documentation.

transform(This, Matrix) -> ok
Types:
   \text{This} = \text{wxGraphicsPath()}
   \text{Matrix} = \text{wxGraphicsMatrix:wxGraphicsMatrix()}
See external documentation.
wxGraphicsPen

Erlang module


This class is derived (and can use functions) from:
wxGraphicsObject

DATA TYPES

wxGraphicsPen()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxGraphicsRenderer

Erlang module

See external documentation: wxGraphicsRenderer.

DATA TYPES

wxGraphicsRenderer()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

gDefaultRenderer() -> wxGraphicsRenderer()
See external documentation.

createContext(This, Dc) -> wxGraphicsContext:wxGraphicsContext()
Types:
  This = wxGraphicsRenderer()
  Dc = wxWindowDC:wxWindowDC() | wxWindow:wxWindow()
See external documentation.

createPen(This, Pen) -> wxGraphicsPen:wxGraphicsPen()
Types:
  This = wxGraphicsRenderer()
  Pen = wxPen:wxPen()
See external documentation.

createBrush(This, Brush) -> wxGraphicsBrush:wxGraphicsBrush()
Types:
  This = wxGraphicsRenderer()
  Brush = wxBrush:wxBrush()
See external documentation.

createLinearGradientBrush(This, X1, Y1, X2, Y2, C1, C2) ->
wxGraphicsBrush:wxGraphicsBrush()
Types:
  This = wxGraphicsRenderer()
  X1 = number()
  Y1 = number()
  X2 = number()
  Y2 = number()
  C1 = wx:wx_colour()
This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

createRadialGradientBrush(This, Xo, Yo, Xc, Yc, Radius, OColor, CColor) -> wxGraphicsBrush:wxGraphicsBrush()
Types:
  This = wxGraphicsRenderer()
  Xo = number()
  Yo = number()
  Xc = number()
  Yc = number()
  Radius = number()
  OColor = wx:wx_colour()
  CColor = wx:wx_colour()
This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

createFont(This, Font) -> wxGraphicsFont:wxGraphicsFont()
Types:
  This = wxGraphicsRenderer()
  Font = wxFont:wxFont()
Equivalent to createFont(This, Font, []).

createFont(This, Font, Options::[Option]) -> wxGraphicsFont:wxGraphicsFont()
Types:
  This = wxGraphicsRenderer()
  Font = wxFont:wxFont()
  Option = {col, wx:wx_colour()}
See external documentation.

createMatrix(This) -> wxGraphicsMatrix:wxGraphicsMatrix()
Types:
  This = wxGraphicsRenderer()
Equivalent to createMatrix(This, []).

createMatrix(This, Options::[Option]) -> wxGraphicsMatrix:wxGraphicsMatrix()
Types:
  This = wxGraphicsRenderer()
  Option = {a, number()} | {b, number()} | {c, number()} | {d, number()} | {tx, number()} | {ty, number()}
See external documentation.
createPath(This) -> wxGraphicsPath:wxGraphicsPath()

Types:
   This = wxGraphicsRenderer()

See external documentation.
wxGrid

Erlang module

See external documentation: wxGrid.

This class is derived (and can use functions) from:
wxScrolledWindow
wxPanel
wxWindow
wxEvtHandler

DATA TYPES

wxGrid()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGrid()

See external documentation.

new(Parent, Id) -> wxGrid()

Types:
  Parent = wxWindow:wxWindow()
  Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, X, Y) -> wxGrid()

Types:
  Parent = wxWindow:wxWindow()
  X = integer()
  Y = integer()

See external documentation.

Also:

new(Parent, Id, [Option]) -> wxGrid() when
Parent::wxWindow:wxWindow(), Id::integer(),
Option :: ['pos', {X::integer(), Y::integer()}]
| ['size', {W::integer(), H::integer()}]
| ['style', integer()].

new(Parent, X, Y, Options::[Option]) -> wxGrid()

Types:
  Parent = wxWindow:wxWindow()
  X = integer()
Y = integer()
Option = {w, integer()} | {h, integer()} | {style, integer()}
See external documentation.

appendCols(This) -> boolean()
Types:
  This = wxGrid()
Equivalent to appendCols(This, []).

appendCols(This, Options::[Option]) -> boolean()
Types:
  This = wxGrid()
  Option = {numCols, integer()} | {updateLabels, boolean()}
See external documentation.

appendRows(This) -> boolean()
Types:
  This = wxGrid()
Equivalent to appendRows(This, []).

appendRows(This, Options::[Option]) -> boolean()
Types:
  This = wxGrid()
  Option = {numRows, integer()} | {updateLabels, boolean()}
See external documentation.

autoSize(This) -> ok
Types:
  This = wxGrid()
See external documentation.

autoSizeColumn(This, Col) -> ok
Types:
  This = wxGrid()
  Col = integer()
Equivalent to autoSizeColumn(This, Col, []).

autoSizeColumn(This, Col, Options::[Option]) -> ok
Types:
  This = wxGrid()
  Col = integer()
  Option = {setAsMin, boolean()}
See external documentation.
wxGrid

color::integer() -> {R::integer(), G::integer(), B::integer()}

Types:
- R, G, B :: integer()

See external documentation.

color::integer() when color::integer() -> ok

Types:
- R, G, B :: integer()

See external documentation.

colormap::integer() when colormap::integer() -> ok

Types:
- R, G, B :: integer()

See external documentation.

colormap::integer() when colormap::integer() -> ok

Types:
- R, G, B :: integer()

See external documentation.

colormap::integer() when colormap::integer() -> ok

Types:
- R, G, B :: integer()

See external documentation.

beginBatch(This) -> ok

Types:
- This = wxGrid()

See external documentation.

blockToDeviceRect(This, TopLeft, BottomRight) -> {X::integer(), Y::integer(), W::integer(), H::integer()}

Types:
- X, Y :: integer()
- W, H :: integer()

See external documentation.

autoSizeColumns(This) -> ok

Types:
- This = wxGrid()

Equivalent to autoSizeColumns(This, []).

autoSizeColumns(This, Options::[Option]) -> ok

Types:
- This = wxGrid()
- Option = {setAsMin, boolean()}

See external documentation.

autoSizeRow(This, Row) -> ok

Types:
- This = wxGrid()
- Row = integer()

Equivalent to autoSizeRow(This, Row, []).

autoSizeRow(This, Row, Options::[Option]) -> ok

Types:
- This = wxGrid()
- Row = integer()
- Option = {setAsMin, boolean()}

See external documentation.

autoSizeRows(This) -> ok

Types:
- This = wxGrid()

Equivalent to autoSizeRows(This, []).

autoSizeRows(This, Options::[Option]) -> ok

Types:
- This = wxGrid()
- Option = {setAsMin, boolean()}

See external documentation.

clearBackground() -> ok

Types:
- This = wxGrid()

See external documentation.
This = wxGrid()
TopLeft = {R::integer(), C::integer()}
BottomRight = {R::integer(), C::integer()}

See external documentation.

canDragColSize(This) -> boolean()
Types:
  This = wxGrid()
See external documentation.

canDragRowSize(This) -> boolean()
Types:
  This = wxGrid()
See external documentation.

canDragGridSize(This) -> boolean()
Types:
  This = wxGrid()
See external documentation.

canEnableCellControl(This) -> boolean()
Types:
  This = wxGrid()
See external documentation.

cellToRect(This, Coords) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
  This = wxGrid()
  Coords = {R::integer(), C::integer()}
See external documentation.

cellToRect(This, Row, Col) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
See external documentation.

clearGrid(This) -> ok
Types:
  This = wxGrid()
clearSelection(This) -> ok
Types:
   This = wxGrid()
See external documentation.

createGrid(This, NumRows, NumCols) -> boolean()
Types:
   This = wxGrid()
   NumRows = integer()
   NumCols = integer()
Equivalent to createGrid(This, NumRows, NumCols, []).

createGrid(This, NumRows, NumCols, Options::[Option]) -> boolean()
Types:
   This = wxGrid()
   NumRows = integer()
   NumCols = integer()
   Option = {selmode, wx:wx_enum()}
See external documentation.
Selmode = ?wxGrid_wxGridSelectCells | ?wxGrid_wxGridSelectRows | ?wxGrid_wxGridSelectColumns

deleteCols(This) -> boolean()
Types:
   This = wxGrid()
Equivalent to deleteCols(This, []).

deleteCols(This, Options::[Option]) -> boolean()
Types:
   This = wxGrid()
   Option = {pos, integer()} | {numCols, integer()} | {updateLabels, boolean()}
See external documentation.

deleteRows(This) -> boolean()
Types:
   This = wxGrid()
Equivalent to deleteRows(This, []).

deleteRows(This, Options::[Option]) -> boolean()
Types:
   This = wxGrid()
Option = {pos, integer()} | {numRows, integer()} | {updateLabels, boolean()}

See external documentation.

disableCellEditControl(This) -> ok
Types:
  This = wxGrid()

See external documentation.

disableDragColSize(This) -> ok
Types:
  This = wxGrid()

See external documentation.

disableDragGridSize(This) -> ok
Types:
  This = wxGrid()

See external documentation.

disableDragRowSize(This) -> ok
Types:
  This = wxGrid()

See external documentation.

enableCellEditControl(This) -> ok
Types:
  This = wxGrid()

Equivalent to enableCellEditControl(This, []).

enableCellEditControl(This, Options::[Option]) -> ok
Types:
  This = wxGrid()
  Option = {enable, boolean()}

See external documentation.

enableDragColSize(This) -> ok
Types:
  This = wxGrid()

Equivalent to enableDragColSize(This, []).

enableDragColSize(This, Options::[Option]) -> ok
Types:
  This = wxGrid()
Enable drag grid size

\[ enable\text{DragGridSize}(This) \rightarrow \text{ok} \]

**Types:**

- \( This = \text{wxGrid()} \)
- \( \text{Option} = \{\text{enable}, \text{boolean}()\} \)

See external documentation.

Enable drag row size

\[ enable\text{DragRowSize}(This) \rightarrow \text{ok} \]

**Types:**

- \( This = \text{wxGrid()} \)
- \( \text{Option} = \{\text{enable}, \text{boolean}()\} \)

See external documentation.

Enable editing

\[ enable\text{Editing}(This, \text{Edit}) \rightarrow \text{ok} \]

**Types:**

- \( This = \text{wxGrid()} \)
- \( \text{Edit} = \text{boolean}() \)

See external documentation.

Enable grid lines

\[ enable\text{GridLines}(This) \rightarrow \text{ok} \]

**Types:**

- \( This = \text{wxGrid()} \)

See external documentation.
endBatch(This) -> ok
Types:
  This = wxGrid()
See external documentation.

fit(This) -> ok
Types:
  This = wxGrid()
See external documentation.

forceRefresh(This) -> ok
Types:
  This = wxGrid()
See external documentation.

getBatchCount(This) -> integer()
Types:
  This = wxGrid()
See external documentation.

getCellAlignment(This, Row, Col) -> {Horiz::integer(), Vert::integer()}
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
See external documentation.

getCellBackgroundColour(This, Row, Col) -> wx:wx_colour4()
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
See external documentation.

gGetCellEditor(This, Row, Col) -> wxGridCellEditor:wxGridCellEditor()
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
See external documentation.

gGetCellFont(This, Row, Col) -> wxFont:wxFont()
Types:
wxGrid

    This = wxGrid()
    Row = integer()
    Col = integer()

See external documentation.

cellRenderer(This, Row, Col) -> wxGridCellRenderer:wxGridCellRenderer()
Types:
    This = wxGrid()
    Row = integer()
    Col = integer()

See external documentation.

cellTextColour(This, Row, Col) -> wx:wx_colour4()
Types:
    This = wxGrid()
    Row = integer()
    Col = integer()

See external documentation.

cellValue(This, Coords) -> unicode:charlist()
Types:
    This = wxGrid()
    Coords = {R::integer(), C::integer()}

See external documentation.

cellValue(This, Row, Col) -> unicode:charlist()
Types:
    This = wxGrid()
    Row = integer()
    Col = integer()

See external documentation.

colLabelAlignment(This) -> {Horiz::integer(), Vert::integer()}
Types:
    This = wxGrid()

See external documentation.

colLabelSize(This) -> integer()
Types:
    This = wxGrid()

See external documentation.
getColLabelValue(This, Col) -> unicode:charlist()
Types:
   This = wxGrid()
   Col = integer()
See external documentation.

getColMinimalAcceptableWidth(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getDefaultCellAlignment(This) -> {Horiz::integer(), Vert::integer()}
Types:
   This = wxGrid()
See external documentation.

getDefaultCellBackgroundColour(This) -> wx.wx_colour4()
Types:
   This = wxGrid()
See external documentation.

getDefaultCellFont(This) -> wxFont:wxFont()
Types:
   This = wxGrid()
See external documentation.

getDefaultCellTextColour(This) -> wx.wx_colour4()
Types:
   This = wxGrid()
See external documentation.

getDefaultColLabelSize(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getDefaultColSize(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getDefaultEditor(This) -> wxGridCellEditor:wxGridCellEditor()
Types:
```erlang
This = \texttt{wxGrid()}
See external documentation.

getDefaultEditorForCell(This, C) -> \texttt{wxGridCellEditor:wxGridCellEditor()}
Types:
  This = \texttt{wxGrid()}
  C = \{R::integer(), C::integer()\}
See external documentation.

getDefaultEditorForCell(This, Row, Col) -> \texttt{wxGridCellEditor:wxGridCellEditor()}
Types:
  This = \texttt{wxGrid()}
  Row = integer()
  Col = integer()
See external documentation.

getDefaultEditorForType(This, TypeName) -> \texttt{wxGridCellEditor:wxGridCellEditor()}
Types:
  This = \texttt{wxGrid()}
  TypeName = \texttt{unicode:chardata()}
See external documentation.

getDefaultRenderer(This) -> \texttt{wxGridCellRenderer:wxGridCellRenderer()}
Types:
  This = \texttt{wxGrid()}
See external documentation.

getDefaultRendererForCell(This, Row, Col) -> \texttt{wxGridCellRenderer:wxGridCellRenderer()}
Types:
  This = \texttt{wxGrid()}
  Row = integer()
  Col = integer()
See external documentation.

getDefaultRendererForType(This, TypeName) -> \texttt{wxGridCellRenderer:wxGridCellRenderer()}
Types:
  This = \texttt{wxGrid()}
  TypeName = \texttt{unicode:chardata()}
See external documentation.
```

238 | Ericsson AB. All Rights Reserved.: wxErlang
getDefaultRowLabelSize(This) -> integer()
Types:
    This = wxGrid()
See external documentation.

getDefaultRowSize(This) -> integer()
Types:
    This = wxGrid()
See external documentation.

getGridCursorCol(This) -> integer()
Types:
    This = wxGrid()
See external documentation.

getGridCursorRow(This) -> integer()
Types:
    This = wxGrid()
See external documentation.

getGridColumnColour(This) -> wx:wx_colour4()
Types:
    This = wxGrid()
See external documentation.

gridLinesEnabled(This) -> boolean()
Types:
    This = wxGrid()
See external documentation.

getLabelBackgroundColour(This) -> wx:wx_colour4()
Types:
    This = wxGrid()
See external documentation.

getLabelFont(This) -> wxFont:wxFont()
Types:
    This = wxGrid()
See external documentation.

getLabelTextColour(This) -> wx:wx_colour4()
Types:
    This = wxGrid()
wxGrid

See external documentation.

getNumberCols(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getNumberRows(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getOrCreateCellAttr(This, Row, Col) -> wxGridCellAttr:wxGridCellAttr()
Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
See external documentation.

getRowMinimalAcceptableHeight(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getRowLabelAlignment(This) -> {Horiz::integer(), Vert::integer()}
Types:
   This = wxGrid()
See external documentation.

getRowLabelSize(This) -> integer()
Types:
   This = wxGrid()
See external documentation.

getRowLabelValue(This, Row) -> unicode:charlist()
Types:
   This = wxGrid()
   Row = integer()
See external documentation.

getRowSize(This, Row) -> integer()
Types:
   This = wxGrid()
Row = integer()
See external documentation.

getScrollLineX(This) -> integer()
Types:
  This = wxGrid()
See external documentation.

getScrollLineY(This) -> integer()
Types:
  This = wxGrid()
See external documentation.

getSelectedCells(This) -> [{R::integer(), C::integer()}]
Types:
  This = wxGrid()
See external documentation.

getSelectedCols(This) -> [integer()]
Types:
  This = wxGrid()
See external documentation.

getSelectedRows(This) -> [integer()]
Types:
  This = wxGrid()
See external documentation.

getSelectionBackground(This) -> wx:wx_colour4()
Types:
  This = wxGrid()
See external documentation.

getSelectionBlockTopLeft(This) -> [{R::integer(), C::integer()}]
Types:
  This = wxGrid()
See external documentation.

getSelectionBlockBottomRight(This) -> [{R::integer(), C::integer()}]
Types:
  This = wxGrid()
See external documentation.
getSelectionForeground(This) -> wx:wx_colour4()
Types:
    This = wxGrid()
See external documentation.

getViewWidth(This) -> integer()
Types:
    This = wxGrid()
See external documentation.

getGridWindow(This) -> wxWindow:wxWindow()
Types:
    This = wxGrid()
See external documentation.

getGridRowLabelWindow(This) -> wxWindow:wxWindow()
Types:
    This = wxGrid()
See external documentation.

getGridColLabelWindow(This) -> wxWindow:wxWindow()
Types:
    This = wxGrid()
See external documentation.

getGridCornerLabelWindow(This) -> wxWindow:wxWindow()
Types:
    This = wxGrid()
See external documentation.

hideCellEditControl(This) -> ok
Types:
    This = wxGrid()
See external documentation.

insertCols(This) -> boolean()
Types:
    This = wxGrid()
Equivalent to insertCols(This, []).

insertCols(This, Options::*[Option]) -> boolean()
Types:
    This = wxGrid()
Option = {pos, integer()} | {numCols, integer()} | {updateLabels, boolean()}

See external documentation.

insertRows(This) -> boolean()

Types:
  This = wxGrid()

Equivalent to insertRows(This, []).

insertRows(This, Options::[Option]) -> boolean()

Types:
  This = wxGrid()
  Option = {pos, integer()} | {numRows, integer()} | {updateLabels, boolean()}

See external documentation.

isCellEditControlEnabled(This) -> boolean()

Types:
  This = wxGrid()

See external documentation.

isCurrentCellReadOnly(This) -> boolean()

Types:
  This = wxGrid()

See external documentation.

isEditable(This) -> boolean()

Types:
  This = wxGrid()

See external documentation.

isInSelection(This, Coords) -> boolean()

Types:
  This = wxGrid()
  Coords = {R::integer(), C::integer()}

See external documentation.

isInSelection(This, Row, Col) -> boolean()

Types:
  This = wxGrid()
  Row = integer()
  Col = integer()

See external documentation.
isReadOnly(This, Row, Col) -> boolean()

Types:
   This = wxGrid()
   Row = integer()
   Col = integer()

See external documentation.

isSelection(This) -> boolean()

Types:
   This = wxGrid()

See external documentation.

isVisible(This, Coords) -> boolean()

Types:
   This = wxGrid()
   Coords = {R::integer(), C::integer()}

Equivalent to isVisible(This, Coords, []).

isVisible(This, Row, Col) -> boolean()

Types:
   This = wxGrid()
   Row = integer()
   Col = integer()

See external documentation.

Also:
isVisible(This, Coords, [Option]) -> boolean() when
This::wxGrid(), Coords::{R::integer(), C::integer()},
Option :: {'wholeCellVisible', boolean()}.

isVisible(This, Row, Col, Options::[Option]) -> boolean()

Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
   Option = {wholeCellVisible, boolean()}

See external documentation.

makeCellVisible(This, Coords) -> ok

Types:
   This = wxGrid()
   Coords = {R::integer(), C::integer()}

See external documentation.
makeCellVisible(This, Row, Col) -> ok
Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
See external documentation.

moveCursorDown(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorLeft(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorRight(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorUp(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorDownBlock(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorLeftBlock(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.
moveCursorRightBlock(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

moveCursorUpBlock(This, ExpandSelection) -> boolean()
Types:
   This = wxGrid()
   ExpandSelection = boolean()
See external documentation.

movePageDown(This) -> boolean()
Types:
   This = wxGrid()
See external documentation.

movePageUp(This) -> boolean()
Types:
   This = wxGrid()
See external documentation.

registerDataType(This, TypeName, Renderer, Editor) -> ok
Types:
   This = wxGrid()
   TypeName = unicode:chardata()
   Renderer = wxGridCellRenderer:wxGridCellRenderer()
   Editor = wxGridCellEditor:wxGridCellEditor()
See external documentation.

saveEditControlValue(This) -> ok
Types:
   This = wxGrid()
See external documentation.

selectAll(This) -> ok
Types:
   This = wxGrid()
See external documentation.

selectBlock(This, TopLeft, BottomRight) -> ok
Types:
   This = wxGrid()
TopLeft = {R::integer(), C::integer()}
BottomRight = {R::integer(), C::integer()}

Equivalent to selectBlock(This, TopLeft, BottomRight, []).

selectBlock(This, TopLeft, BottomRight, Options::[Option]) -> ok

Types:

This = wxGrid()
TopLeft = {R::integer(), C::integer()}
BottomRight = {R::integer(), C::integer()}
Option = {addToSelected, boolean()}

See external documentation.

selectBlock(This, TopRow, LeftCol, BottomRow, RightCol) -> ok

Types:

This = wxGrid()
TopRow = integer()
LeftCol = integer()
BottomRow = integer()
RightCol = integer()

Equivalent to selectBlock(This, TopRow, LeftCol, BottomRow, RightCol, []).

selectBlock(This, TopRow, LeftCol, BottomRow, RightCol, Options::[Option]) -> ok

Types:

This = wxGrid()
TopRow = integer()
LeftCol = integer()
BottomRow = integer()
RightCol = integer()
Option = {addToSelected, boolean()}

See external documentation.

selectCol(This, Col) -> ok

Types:

This = wxGrid()
Col = integer()

Equivalent to selectCol(This, Col, []).

selectCol(This, Col, Options::[Option]) -> ok

Types:

This = wxGrid()
Col = integer()
Option = {addToSelected, boolean()}
see external documentation.

selectRow(This, Row) -> ok
Types:
    This = wxGrid()
    Row = integer()
Equivalent to selectRow(This, Row, []).
setCellBackgroundColour(This, Row, Col, Val) -> ok
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
  Val = wx:wx_colour()
See external documentation.

Also:
setCellBackgroundColour(This, Colour, Row, Col) -> 'ok' when
This::wxGrid(), Colour::wx:wx_colour(), Row::integer(), Col::integer().

setCellEditor(This, Row, Col, Editor) -> ok
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
  Editor = wxGridCellEditor:wxGridCellEditor()
See external documentation.

setCellFont(This, Row, Col, Val) -> ok
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
  Val = wxFont:wxFont()
See external documentation.

setCellRenderer(This, Row, Col, Renderer) -> ok
Types:
  This = wxGrid()
  Row = integer()
  Col = integer()
  Renderer = wxGridCellRenderer:wxGridCellRenderer()
See external documentation.

setCellTextColour(This, Col) -> ok
Types:
  This = wxGrid()
  Col = wx:wx_colour()
See external documentation.

setCellTextColour(This, Row, Col, Val) -> ok
Types:
This = wxGrid()
Row = integer()
Col = integer()
Val = wx:wx_colour()

See external documentation.
Also:
setCellTextColour(This, Val, Row, Col) -> 'ok'
This::wxGrid(), Val::wx:wx_colour(), Row::integer(), Col::integer().

setCellValue(This, Coords, S) -> ok
Types:
   This = wxGrid()
   Coords = {R::integer(), C::integer()}
   S = unicode:chardata()

See external documentation.

setCellValue(This, Row, Col, S) -> ok
Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
   S = unicode:chardata()

See external documentation.
Also:
setCellValue(This, Val, Row, Col) -> 'ok'
This::wxGrid(), Val::unicode:chardata(), Row::integer(), Col::integer().

setColAttr(This, Col, Attr) -> ok
Types:
   This = wxGrid()
   Col = integer()
   Attr = wxGridCellAttr:wxGridCellAttr()

See external documentation.

setColFormatBool(This, Col) -> ok
Types:
   This = wxGrid()
   Col = integer()

See external documentation.

setColFormatNumber(This, Col) -> ok
Types:
   This = wxGrid()
   Col = integer()
See external documentation.

setColFormatFloat(This, Col) -> ok
Types:
   This = wxGrid()
   Col = integer()
Equivalent to setColFormatFloat(This, Col, [])..

setColFormatFloat(This, Col, Options::([Option])) -> ok
Types:
   This = wxGrid()
   Col = integer()
   Option = {width, integer()} | {precision, integer()}
See external documentation.

setColFormatCustom(This, Col, TypeName) -> ok
Types:
   This = wxGrid()
   Col = integer()
   TypeName = unicode:chardata()
See external documentation.

setColLabelAlignment(This, Horiz, Vert) -> ok
Types:
   This = wxGrid()
   Horiz = integer()
   Vert = integer()
See external documentation.

setColLabelSize(This, Height) -> ok
Types:
   This = wxGrid()
   Height = integer()
See external documentation.

setColLabelValue(This, Col, Val) -> ok
Types:
   This = wxGrid()
   Col = integer()
   Val = unicode:chardata()
See external documentation.
wxGrid

setColMinimalWidth(This, Col, Width) -> ok
Types:
   This = wxGrid()
   Col = integer()
   Width = integer()
See external documentation.

setColMinimalAcceptableWidth(This, Width) -> ok
Types:
   This = wxGrid()
   Width = integer()
See external documentation.

setColSize(This, Col, Width) -> ok
Types:
   This = wxGrid()
   Col = integer()
   Width = integer()
See external documentation.

setDefaultCellAlignment(This, Horiz, Vert) -> ok
Types:
   This = wxGrid()
   Horiz = integer()
   Vert = integer()
See external documentation.

setDefaultCellBackgroundColour(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wx:wx_colour()
See external documentation.

setDefaultCellFont(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wxFont:wxFont()
See external documentation.

setDefaultCellTextColour(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wx:wx_colour()
setDefaultEditor(This, Editor) -> ok
Types:
    This = wxGrid()
    Editor = wxGridCellEditor:wxGridCellEditor()
See external documentation.

setDefaultRenderer(This, Renderer) -> ok
Types:
    This = wxGrid()
    Renderer = wxGridCellRenderer:wxGridCellRenderer()
See external documentation.

setDefaultColSize(This, Width) -> ok
Types:
    This = wxGrid()
    Width = integer()
Equivalent to setDefaultColSize(This, Width, []).

setDefaultColSize(This, Width, Options::[Option]) -> ok
Types:
    This = wxGrid()
    Width = integer()
    Option = {resizeExistingCols, boolean()}
See external documentation.

setDefaultRowSize(This, Height) -> ok
Types:
    This = wxGrid()
    Height = integer()
Equivalent to setDefaultRowSize(This, Height, []).

setDefaultRowSize(This, Height, Options::[Option]) -> ok
Types:
    This = wxGrid()
    Height = integer()
    Option = {resizeExistingRows, boolean()}
See external documentation.

setGridCursor(This, Row, Col) -> ok
Types:
    This = wxGrid()
Row = integer()
Col = integer()
See external documentation.

setGridLineColour(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wx:wx_colour()
See external documentation.

setLabelBackgroundColour(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wx:wx_colour()
See external documentation.

setLabelFont(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wxFont:wxFont()
See external documentation.

setLabelTextColour(This, Val) -> ok
Types:
   This = wxGrid()
   Val = wx:wx_colour()
See external documentation.

setMargins(This, ExtraWidth, ExtraHeight) -> ok
Types:
   This = wxGrid()
   ExtraWidth = integer()
   ExtraHeight = integer()
See external documentation.

setReadOnly(This, Row, Col) -> ok
Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
Equivalent to setReadOnly(This, Row, Col, []).
setReadOnly(This, Row, Col, Options::[Option]) -> ok

Types:
   This = wxGrid()
   Row = integer()
   Col = integer()
   Option = {isReadOnly, boolean()}

See external documentation.

setRowAttr(This, Row, Attr) -> ok

Types:
   This = wxGrid()
   Row = integer()
   Attr = wxGridCellAttr:wxGridCellAttr()

See external documentation.

setRowLabelAlignment(This, Horiz, Vert) -> ok

Types:
   This = wxGrid()
   Horiz = integer()
   Vert = integer()

See external documentation.

setRowLabelSize(This, Width) -> ok

Types:
   This = wxGrid()
   Width = integer()

See external documentation.

setRowLabelValue(This, Row, Val) -> ok

Types:
   This = wxGrid()
   Row = integer()
   Val = unicode:chardata()

See external documentation.

setRowMinimalHeight(This, Row, Width) -> ok

Types:
   This = wxGrid()
   Row = integer()
   Width = integer()

See external documentation.
wxGrid

setRowMinimalAcceptableHeight(This, Width) -> ok
Types:
    This = wxGrid()
    Width = integer()
See external documentation.

setRowSize(This, Row, Height) -> ok
Types:
    This = wxGrid()
    Row = integer()
    Height = integer()
See external documentation.

setScrollLineX(This, X) -> ok
Types:
    This = wxGrid()
    X = integer()
See external documentation.

setScrollLineY(This, Y) -> ok
Types:
    This = wxGrid()
    Y = integer()
See external documentation.

setSelectionBackground(This, C) -> ok
Types:
    This = wxGrid()
    C = wx:wx_colour()
See external documentation.

setSelectionForeground(This, C) -> ok
Types:
    This = wxGrid()
    C = wx:wx_colour()
See external documentation.

setSelectionMode(This, Selmode) -> ok
Types:
    This = wxGrid()
    Selmode = wx:wx_enum()
See external documentation.
Selmode = ?wxGrid_wxGridSelectCells | ?wxGrid_wxGridSelectRows | ?wxGrid_wxGridSelectColumns
showCellEditControl(This) -> ok
Types:
   This = wxGrid()
See external documentation.

xToCol(This, X) -> integer()
Types:
   This = wxGrid()
   X = integer()
Equivalent to xToCol(This, X, []).

xToCol(This, X, Options::[Option]) -> integer()
Types:
   This = wxGrid()
   X = integer()
   Option = {clipToMinMax, boolean()}
See external documentation.

xToEdgeOfCol(This, X) -> integer()
Types:
   This = wxGrid()
   X = integer()
See external documentation.

yToEdgeOfRow(This, Y) -> integer()
Types:
   This = wxGrid()
   Y = integer()
See external documentation.

yToRow(This, Y) -> integer()
Types:
   This = wxGrid()
   Y = integer()
See external documentation.

destroy(This::wxGrid()) -> ok
Destroys this object, do not use object again
wxGridBagSizer

Erlang module

See external documentation: wxGridBagSizer.

This class is derived (and can use functions) from:
wxFlexGridSizer
wxGridSizer
wxSizer

DATA TYPES

wxGridBagSizer()

An object reference, The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGridBagSizer()
Equivalente to new([]).

new(Options::[Option]) -> wxGridBagSizer()

Types:
Option = {vgap, integer()} | {hgap, integer()}

See external documentation.

add(This, Item) -> wxSizerItem:wxSizerItem()

Types:
This = wxGridBagSizer()
Item = wxSizerItem:wxSizerItem() | wxGBSizerItem:wxGBSizerItem()

See external documentation.

add(This, Width, Height) -> wxSizerItem:wxSizerItem()

Types:
This = wxGridBagSizer()
Width = integer()
Height = integer()

See external documentation.

Also:
add(This, Window, Pos) -> wxSizerItem:wxSizerItem() when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), Pos::{R::integer(), C::integer()};
(This, Window, [Option]) -> wxSizerItem:wxSizerItem() when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(),
Option :: [{proportion', integer()} | {flag', integer()}}
add(This, Width, Height, Pos) -> wxSizerItem:wxSizerItem()

Types:

- **This** = wxGridBagSizer()
- **Width** = integer()
- **Height** = integer()
- **Pos** = {R::integer(), C::integer()}

See external documentation.

Also:

add(This, Width, Height, [Option]) -> wxSizerItem:wxSizerItem() when

This::wxGridBagSizer(), Width::integer(), Height::integer(),
Option :: {'proportion', integer()}
| {'flag', integer()}
| {'border', integer()}
| {'userData', wx:wx_object()};
(This, Window, Pos, [Option]) -> wxSizerItem:wxSizerItem() when

This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), Pos::{R::integer(), C::integer()},
Option :: {'span', {RS::integer(), CS::integer()}}
| {'flag', integer()}
| {'border', integer()}
| {'userData', wx:wx_object()};

add(This, Width, Height, Pos, Options::[Option]) -> wxSizerItem:wxSizerItem()

Types:

- **This** = wxGridBagSizer()
- **Width** = integer()
- **Height** = integer()
- **Pos** = {R::integer(), C::integer()}
- **Option** = {span, {RS::integer(), CS::integer()}} | {flag, integer()} | {border, integer()} | {userData, wx:wx_object()}

See external documentation.

calcMin(This) -> {W::integer(), H::integer()}

Types:

- **This** = wxGridBagSizer()

See external documentation.

checkForIntersection(This, Item) -> boolean()

Types:

- **This** = wxGridBagSizer()
- **Item** = wxGBSizerItem:wxGBSizerItem()

Equivalent to checkForIntersection(This, Item, []).
checkForIntersection(This, Pos, Span) -> boolean()
Types:
   This = wxGridBagSizer()
   Pos = {R::integer(), C::integer()}
   Span = {RS::integer(), CS::integer()}
See external documentation.
Also:
checkForIntersection(This, Item, [Option]) -> boolean() when
This::wxGridBagSizer(), Item::wxGBSizerItem:wxGBSizerItem(),
Option :: ['excludeItem', wxGBSizerItem:wxGBSizerItem()].

checkForIntersection(This, Pos, Span, Options::[Option]) -> boolean()
Types:
   This = wxGridBagSizer()
   Pos = {R::integer(), C::integer()}
   Span = {RS::integer(), CS::integer()}
   Option = {excludeItem, wxGBSizerItem:wxGBSizerItem()}
See external documentation.

findItem(This, Window) -> wxGBSizerItem:wxGBSizerItem()
Types:
   This = wxGridBagSizer()
   Window = wxWindow:wxWindow() | wxSizer:wxSizer()
See external documentation.

findItemAtPoint(This, Pt) -> wxGBSizerItem:wxGBSizerItem()
Types:
   This = wxGridBagSizer()
   Pt = {X::integer(), Y::integer()}
See external documentation.

findItemAtPosition(This, Pos) -> wxGBSizerItem:wxGBSizerItem()
Types:
   This = wxGridBagSizer()
   Pos = {R::integer(), C::integer()}
See external documentation.

findItemWithData(This, UserData) -> wxGBSizerItem:wxGBSizerItem()
Types:
   This = wxGridBagSizer()
   UserData = wx:wx_object()
See external documentation.
getCellSize(This, Row, Col) -> {W::integer(), H::integer()}
Types:
   This = wxGridBagSizer()
   Row = integer()
   Col = integer()
See external documentation.

getCellSize(This) -> {W::integer(), H::integer()}
Types:
   This = wxGridBagSizer()
See external documentation.

getItemPosition(This, Index) -> {R::integer(), C::integer()}
Types:
   This = wxGridBagSizer()
   Index = integer()
See external documentation.
Also:
getItemPosition(This, Window) -> {R::integer(), C::integer()} when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer().

getItemSpan(This, Index) -> {RS::integer(), CS::integer()}
Types:
   This = wxGridBagSizer()
   Index = integer()
See external documentation.
Also:
getItemSpan(This, Window) -> {RS::integer(), CS::integer()} when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer().

setEmptyCellSize(This, Sz) -> ok
Types:
   This = wxGridBagSizer()
   Sz = {W::integer(), H::integer()}
See external documentation.

setItemPosition(This, Index, Pos) -> boolean()
Types:
   This = wxGridBagSizer()
   Index = integer()
   Pos = {R::integer(), C::integer()}
See external documentation.
Also:
setItemPosition(This, Window, Pos) -> boolean() when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer().
**wxGridBagSizer**

This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), Pos::{R::integer(), C::integer()}.

setItemSpan(This, Index, Span) -> boolean()

Types:

This = wxGridBagSizer()
Index = integer()
Span = {RS::integer(), CS::integer()}

See external documentation.

Also:

setItemSpan(This, Window, Span) -> boolean() when
This::wxGridBagSizer(), Window::wxWindow:wxWindow() | wxSizer:wxSizer(), Span::{RS::integer(), CS::integer()}.

destroy(This::wxGridBagSizer()) -> ok

Destroys this object, do not use object again
wxGridCellAttr

Erlang module

See external documentation: wxGridCellAttr.

DATA TYPES

wxGridCellAttr()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

setTextColour(This, ColText) -> ok
Types:
  This = wxGridCellAttr()
  ColText = wx:wx_colour()
See external documentation.

setBackgroundColour(This, ColBack) -> ok
Types:
  This = wxGridCellAttr()
  ColBack = wx:wx_colour()
See external documentation.

setFont(This, Font) -> ok
Types:
  This = wxGridCellAttr()
  Font = wxFont:wxFont()
See external documentation.

setAlignment(This, HAlign, VAlign) -> ok
Types:
  This = wxGridCellAttr()
  HAlign = integer()
  VAlign = integer()
See external documentation.

setReadOnly(This) -> ok
Types:
  This = wxGridCellAttr()
Equivalent to setReadOnly(This, []).
setReadOnly(This, Options::[Option]) -> ok
Types:
    This = wxGridCellAttr()
    Option = {isReadOnly, boolean()}
See external documentation.

setRenderer(This, Renderer) -> ok
Types:
    This = wxGridCellAttr()
    Renderer = wxGridCellRenderer:wxGridCellRenderer()
See external documentation.

setEditor(This, Editor) -> ok
Types:
    This = wxGridCellAttr()
    Editor = wxGridCellEditor:wxGridCellEditor()
See external documentation.

hasTextColour(This) -> boolean()
Types:
    This = wxGridCellAttr()
See external documentation.

hasBackgroundColour(This) -> boolean()
Types:
    This = wxGridCellAttr()
See external documentation.

hasFont(This) -> boolean()
Types:
    This = wxGridCellAttr()
See external documentation.

hasAlignment(This) -> boolean()
Types:
    This = wxGridCellAttr()
See external documentation.

hasRenderer(This) -> boolean()
Types:
    This = wxGridCellAttr()
See external documentation.
hasEditor(This) -> boolean()
Types:
   This = wxGridCellAttr()
See external documentation.

ggetTextColour(This) -> wx:wx_colour4()
Types:
   This = wxGridCellAttr()
See external documentation.

ggetBackgroundColour(This) -> wx:wx_colour4()
Types:
   This = wxGridCellAttr()
See external documentation.

g.getFont(This) -> wxFont:wxFont()
Types:
   This = wxGridCellAttr()
See external documentation.

ggetAlignment(This) -> {HAlign::integer(), VAlign::integer()}
Types:
   This = wxGridCellAttr()
See external documentation.

ggetRenderer(This, Grid, Row, Col) -> wxGridCellRenderer:wxGridCellRenderer()
Types:
   This = wxGridCellAttr()
   Grid = wxGrid:wxGrid()
   Row = integer()
   Col = integer()
See external documentation.

ggetEditor(This, Grid, Row, Col) -> wxGridCellEditor:wxGridCellEditor()
Types:
   This = wxGridCellAttr()
   Grid = wxGrid:wxGrid()
   Row = integer()
   Col = integer()
See external documentation.

isReadOnly(This) -> boolean()
Types:
wxGridCellAttr

This = wxGridCellAttr()
See external documentation.

setDefAttr(This, DefAttr) -> ok
Types:
  This = wxGridCellAttr()
  DefAttr = wxGridCellAttr()
See external documentation.
wxGridCellBoolEditor

Erlang module

See external documentation: wxGridCellBoolEditor.
This class is derived (and can use functions) from:
wxGridCellEditor

**DATA TYPES**

wxGridCellBoolEditor()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxGridCellBoolEditor()
See external documentation.

isTrueValue(Value) -> boolean()
Types:
   Value = unicode:chardata()
See external documentation.

useStringValues() -> ok
Equivalent to useStringValues([]).

useStringValues(Options::[Option]) -> ok
Types:
   Option = {valueTrue, unicode:chardata()} | {valueFalse, unicode:chardata()}
See external documentation.

destroy(This::wxGridCellBoolEditor()) -> ok
Destroys this object, do not use object again
wxGridCellBoolRenderer

Erlang module

See external documentation: wxGridCellBoolRenderer.
This class is derived (and can use functions) from:
wxGridCellRenderer

DATA TYPES
wxGridCellBoolRenderer()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGridCellBoolRenderer()
See external documentation.

destroy(This::wxGridCellBoolRenderer()) -> ok
Destroys this object, do not use object again
wxGridCellChoiceEditor

Erlang module

See external documentation: wxGridCellChoiceEditor.

This class is derived (and can use functions) from:
wxGridCellEditor

DATA TYPES
wxGridCellChoiceEditor()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparsion stored on disc or distributed for use on other nodes.

Exports

new(Choices) -> wxGridCellChoiceEditor()

Types:

    Choices = [unicode:chardata()]

Equivalent to new(Choices, []).

new(Choices, Options::[Option]) -> wxGridCellChoiceEditor()

Types:

    Choices = [unicode:chardata()]
    Option = {allowOthers, boolean()}

See external documentation.

setParameters(This, Params) -> ok

Types:

    This = wxGridCellChoiceEditor()
    Params = unicode:chardata()

See external documentation.

destroy(This::wxGridCellChoiceEditor()) -> ok

Destroys this object, do not use object again.
wxGridCellEditor

Erlang module

See external documentation: wxGridCellEditor.

DATA TYPES

wxGridCellEditor()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

create(This, Parent, Id, EvtHandler) -> ok
Types:

  This = wxGridCellEditor()
  Parent = wxWindow:wxWindow()
  Id = integer()
  EvtHandler = wxEvtHandler:wxEvtHandler()

See external documentation.

isCreated(This) -> boolean()
Types:

  This = wxGridCellEditor()

See external documentation.

setSize(This, Rect) -> ok
Types:

  This = wxGridCellEditor()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}

See external documentation.

show(This, Show) -> ok
Types:

  This = wxGridCellEditor()
  Show = boolean()

Equivalent to show(This, Show, []).

show(This, Show, Options::[Option]) -> ok
Types:

  This = wxGridCellEditor()
  Show = boolean()
  Option = {attr, wxGridCellAttr:wxGridCellAttr()}
See external documentation.

paintBackground(This, RectCell, Attr) -> ok
Types:
   This = wxGridCellEditor()
   RectCell = {X::integer(), Y::integer(), W::integer(), H::integer()}
   Attr = wxGridCellAttr:wxGridCellAttr()

This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

beginEdit(This, Row, Col, Grid) -> ok
Types:
   This = wxGridCellEditor()
   Row = integer()
   Col = integer()
   Grid = wxGrid:wxGrid()

See external documentation.

detailEdit(This, Row, Col, Grid) -> boolean()
Types:
   This = wxGridCellEditor()
   Row = integer()
   Col = integer()
   Grid = wxGrid:wxGrid()

This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

reset(This) -> ok
Types:
   This = wxGridCellEditor()

See external documentation.

startingKey(This, Event) -> ok
Types:
   This = wxGridCellEditor()
   Event = wxKeyEvent:wxKeyEvent()

See external documentation.

startingClick(This) -> ok
Types:
   This = wxGridCellEditor()

See external documentation.
handleReturn(This, Event) -> ok

Types:
   This = wxGridCellEditor()
   Event = wxKeyEvent:wxKeyEvent()

See external documentation.
wxGridCellFloatEditor

Erlang module

See external documentation: `wxGridCellFloatEditor`.

This class is derived (and can use functions) from:

`wxGridCellEditor`

**DATA TYPES**

`wxGridCellFloatEditor()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxGridCellFloatEditor()`

Equivalent to `new([])`.

`new(Options::[Option]) -> wxGridCellFloatEditor()`

*Types:*

```
Option = {width, integer()} | {precision, integer()}
```

See external documentation.

`setParameters(This, Params) -> ok`

*Types:*

```
This = wxGridCellFloatEditor()

Params = unicode:chardata()
```

See external documentation.

`destroy(This::wxGridCellFloatEditor()) -> ok`

Destroys this object, do not use object again.
wxGridCellFloatRenderer

Erlang module

See external documentation: wxGridCellFloatRenderer.

This class is derived (and can use functions) from:
wxGridCellStringRenderer
wxGridCellRenderer

DATA TYPES

wxGridCellFloatRenderer()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGridCellFloatRenderer()
Equivalent to new([]).

new(Options::[Option]) -> wxGridCellFloatRenderer()
Types:
  Option = {width, integer()} | {precision, integer()}
See external documentation.

getPrecision(This) -> integer()
Types:
  This = wxGridCellFloatRenderer()
See external documentation.

getWidth(This) -> integer()
Types:
  This = wxGridCellFloatRenderer()
See external documentation.

setParameters(This, Params) -> ok
Types:
  This = wxGridCellFloatRenderer()
  Params = unicode:chardata()
See external documentation.

setPrecision(This, Precision) -> ok
Types:
  This = wxGridCellFloatRenderer()
Precision = integer()
See external documentation.

setWidth(This, Width) -> ok
Types:
   This = wxGridCellFloatRenderer()
   Width = integer()
See external documentation.

destroy(This::wxGridCellFloatRenderer()) -> ok
Destroys this object, do not use object again
wxGridCellNumberEditor

Erlang module

See external documentation: `wxGridCellNumberEditor`.

This class is derived (and can use functions) from:
`wxGridCellTextEditor`
`wxGridCellEditor`

**DATA TYPES**

`wxGridCellNumberEditor()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxGridCellNumberEditor()`
Equivalent to `new([]).`

`new(Options::[Option]) -> wxGridCellNumberEditor()`
Types:
```
  Option = {min, integer()} | {max, integer()}
```
See external documentation.

`getValue(This) -> unicode:charlist()`
Types:
```
  This = wxGridCellNumberEditor()
```
See external documentation.

`setParameters(This, Params) -> ok`
Types:
```
  This = wxGridCellNumberEditor()
  Params = unicode:chardata()
```
See external documentation.

`destroy(This::wxGridCellNumberEditor()) -> ok`
Destroys this object, do not use object again
wxGridCellNumberRenderer

Erlang module

See external documentation: wxGridCellNumberRenderer.

This class is derived (and can use functions) from:
wxGridCellStringRenderer
wxGridCellRenderer

DATA TYPES
wxGridCellNumberRenderer()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxGridCellNumberRenderer()
See external documentation.

destroy(This::wxGridCellNumberRenderer()) -> ok
Destroys this object, do not use object again
wxGridCellRenderer

Erlang module

See external documentation: wxGridCellRenderer.

DATA TYPES
wxGridCellRenderer()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

draw(This, Grid, Attr, Dc, Rect, Row, Col, IsSelected) -> ok
Types:
  This = wxGridCellRenderer()
  Grid = wxGrid:wxGrid()
  Attr = wxGridCellAttr:wxGridCellAttr()
  Dc = wxDC:wxDC()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
  Row = integer()
  Col = integer()
  IsSelected = boolean()

See external documentation.

getBestSize(This, Grid, Attr, Dc, Row, Col) -> {W::integer(), H::integer()}
Types:
  This = wxGridCellRenderer()
  Grid = wxGrid:wxGrid()
  Attr = wxGridCellAttr:wxGridCellAttr()
  Dc = wxDC:wxDC()
  Row = integer()
  Col = integer()

See external documentation.
wxGridCellStringRenderer

Erlang module

See external documentation: `wxGridCellStringRenderer`.
This class is derived (and can use functions) from:
`wxGridCellRenderer`

**DATA TYPES**

`wxGridCellStringRenderer()`

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxGridCellStringRenderer()`
See external documentation.

`destroy(This::wxGridCellStringRenderer()) -> ok`
Destroys this object, do not use object again
**wxGridCellTextEditor**

Erlang module

See external documentation: `wxGridCellTextEditor`.

This class is derived (and can use functions) from:

`wxGridCellEditor`

**DATA TYPES**

`wxGridCellTextEditor()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxGridCellTextEditor()`

See external documentation.

`setParameters(This, Params) -> ok`  
Types:

- `This = wxGridCellTextEditor()`  
- `Params = unicode:chardata()`

See external documentation.

`destroy(This::wxGridCellTextEditor()) -> ok`  
Destroys this object, do not use object again
wxGridEvent

Erlang module

See external documentation: wxGridEvent.

Use wxEvtHandler:connect/3 with EventType:

grid_cell_left_click, grid_cell_right_click, grid_cell_left_dclick, grid_cell_right_dclick, grid_label_left_click, grid_label_right_click, grid_label_left_dclick, grid_label_right_dclick, grid_row_size, grid_col_size, grid_range_select, grid_cell_change, grid_select_cell, grid_editor_shown, grid_editor_hidden, grid_editor_created, grid_cell_begin_drag

See also the message variant #wxGrid/ event record type.

This class is derived (and can use functions) from:
wxNotifyEvent
wxCommandEvent
wxEvent

DATA TYPES

wxGridEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

altDown(This) -> boolean()
Types:
This = wxGridEvent()
See external documentation.

controlDown(This) -> boolean()
Types:
This = wxGridEvent()
See external documentation.

getCol(This) -> integer()
Types:
This = wxGridEvent()
See external documentation.

getPosition(This) -> {X::integer(), Y::integer()}
Types:
This = wxGridEvent()
See external documentation.
getRow(This) -> integer()
Types:
   This = wxGridEvent()
See external documentation.

metaDown(This) -> boolean()
Types:
   This = wxGridEvent()
See external documentation.

selecting(This) -> boolean()
Types:
   This = wxGridEvent()
See external documentation.

shiftDown(This) -> boolean()
Types:
   This = wxGridEvent()
See external documentation.
wxGridSizer

Erlang module

See external documentation: wxGridSizer.

This class is derived (and can use functions) from:
wxSizer

DATA TYPES

wxGridSizer()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Cols) -> wxGridSizer()
Types:
   Cols = integer()
Equivalent to new(Cols, []).

new(Cols, Options::[Option]) -> wxGridSizer()
Types:
   Cols = integer()
   Option = {vgap, integer()} | {hgap, integer()}
See external documentation.

new(Rows, Cols, Vgap, Hgap) -> wxGridSizer()
Types:
   Rows = integer()
   Cols = integer()
   Vgap = integer()
   Hgap = integer()
See external documentation.

getCols(This) -> integer()
Types:
   This = wxGridSizer()
See external documentation.

getHGap(This) -> integer()
Types:
   This = wxGridSizer()
See external documentation.
wxGridSizer

getRows(This) -> integer()
Types:
    This = wxGridSizer()
See external documentation.

goingVGap(This) -> integer()
Types:
    This = wxGridSizer()
See external documentation.

setCols(This, Cols) -> ok
Types:
    This = wxGridSizer()
    Cols = integer()
See external documentation.

setHGap(This, Gap) -> ok
Types:
    This = wxGridSizer()
    Gap = integer()
See external documentation.

setRows(This, Rows) -> ok
Types:
    This = wxGridSizer()
    Rows = integer()
See external documentation.

setVGap(This, Gap) -> ok
Types:
    This = wxGridSizer()
    Gap = integer()
See external documentation.

destroy(This::wxGridSizer()) -> ok
Destroys this object, do not use object again
wxHelpEvent

Erlang module

See external documentation: wxHelpEvent.
Use wxEvtHandler:connect/3 with EventType:

help, detailed_help

See also the message variant #wxHelp/ event record type.
This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxHelpEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getOrigin(This) -> wx:wx_enum()

Types:

    This = wxHelpEvent() 

See external documentation.

getAddress(This) -> {X::integer(), Y::integer()}

Types:

    This = wxHelpEvent() 

See external documentation.

setPosition(This, Pos) -> ok

Types:

    This = wxHelpEvent()
    Pos = {X::integer(), Y::integer()}

See external documentation.
wxHtmlEasyPrinting

Erlang module

See external documentation: wxHtmlEasyPrinting.

**DATA TYPES**

wxHtmlEasyPrinting()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxHtmlEasyPrinting()

Equivalent to new([]).

new(Options::[Option]) -> wxHtmlEasyPrinting()

Types:

```
Option = {name, unicode:chardata()} | {parentWindow, wxWindow:wxWindow()}
```

See external documentation.

getPrintData(This) -> wxPrintData:wxPrintData()

Types:

```
This = wxHtmlEasyPrinting()
```

See external documentation.

getPageSetupData(This) -> wxPageSetupDialogData:wxPageSetupDialogData()

Types:

```
This = wxHtmlEasyPrinting()
```

See external documentation.

previewFile(This, Htmlfile) -> boolean()

Types:

```
This = wxHtmlEasyPrinting()
Htmlfile = unicode:chardata()
```

See external documentation.

previewText(This, Htmltext) -> boolean()

Types:

```
This = wxHtmlEasyPrinting()
Htmltext = unicode:chardata()
```

Equivalent to previewText(This, Htmltext, []).
previewText(This, Htmltext, Options::[Option]) -> boolean()
Types:
   This = wxHtmlEasyPrinting()
   Htmltext = unicode:chardata()
   Option = (basepath, unicode:chardata())
See external documentation.

printFile(This, Htmlfile) -> boolean()
Types:
   This = wxHtmlEasyPrinting()
   Htmlfile = unicode:chardata()
See external documentation.

printText(This, Htmltext) -> boolean()
Types:
   This = wxHtmlEasyPrinting()
   Htmltext = unicode:chardata()
Equivalent to printText(This, Htmltext, []).

printText(This, Htmltext, Options::[Option]) -> boolean()
Types:
   This = wxHtmlEasyPrinting()
   Htmltext = unicode:chardata()
   Option = (basepath, unicode:chardata())
See external documentation.

pageSetup(This) -> ok
Types:
   This = wxHtmlEasyPrinting()
See external documentation.

setFonts(This, Normal_face, Fixed_face) -> ok
Types:
   This = wxHtmlEasyPrinting()
   Normal_face = unicode:chardata()
   Fixed_face = unicode:chardata()
Equivalent to setFonts(This, Normal_face, Fixed_face, []).

setFonts(This, Normal_face, Fixed_face, Options::[Option]) -> ok
Types:
   This = wxHtmlEasyPrinting()
   Normal_face = unicode:chardata()
   Fixed_face = unicode:chardata()
wxHtmlEasyPrinting

Option = {sizes, [integer()]} See external documentation.

setHeader(This, Header) -> ok
Types:
  This = wxHtmlEasyPrinting()
  Header = unicode:chardata()
Equivalent to setHeader(This, Header, []).

setHeader(This, Header, Options::[Option]) -> ok
Types:
  This = wxHtmlEasyPrinting()
  Header = unicode:chardata()
  Option = {pg, integer()}
See external documentation.

setFooter(This, Footer) -> ok
Types:
  This = wxHtmlEasyPrinting()
  Footer = unicode:chardata()
Equivalent to setFooter(This, Footer, []).

setFooter(This, Footer, Options::[Option]) -> ok
Types:
  This = wxHtmlEasyPrinting()
  Footer = unicode:chardata()
  Option = {pg, integer()}
See external documentation.

destroy(This::wxHtmlEasyPrinting()) -> ok
Destroys this object, do not use object again
wxHtmlLinkEvent

Erlang module

See external documentation: wxHtmlLinkEvent.

Use wxEvtHandler:connect/3 with EventType:

    command_html_link_clicked

See also the message variant #wxHtmlLink{} event record type.

This class is derived (and can use functions) from:
wxCommandEvent
wxEvent

DATA TYPES

wxHtmlLinkEvent()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

getLinkInfo(This) -> wx:wxHtmlLinkInfo()

Types:

    This = wxHtmlLinkEvent()

See external documentation.
wxHtmlWindow

Erlang module

See external documentation: *wxHtmlWindow*.

This class is derived (and can use functions) from:

*wxScrolledWindow*

*wxPanel*

*wxWindow*

*wxEvtHandler*

**DATA TYPES**

*wxHtmlWindow()*

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxHtmlWindow()

See external documentation.

new(Parent) -> wxHtmlWindow()

Types:

```
Parent = wxWindow:wxWindow()
```

Equivalent to `new(Parent, [])`.

new(Parent, Options::[Option]) -> wxHtmlWindow()

Types:

```
Parent = wxWindow:wxWindow()
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, 
{W::integer(), H::integer()}} | {style, integer()}
```

See external documentation.

appendToPage(This, Source) -> boolean()

Types:

```
This = wxHtmlWindow()
Source = unicode:chardata()
```

See external documentation.

getOpenedAnchor(This) -> unicode:charlist()

Types:

```
This = wxHtmlWindow()
```

See external documentation.
getOpenedPage(This) -> unicode:charlist()
Types:
  This = wxHtmlWindow()
See external documentation.

getOpenedPageTitle(This) -> unicode:charlist()
Types:
  This = wxHtmlWindow()
See external documentation.

getRelatedFrame(This) -> wxFrame:wxFrame()
Types:
  This = wxHtmlWindow()
See external documentation.

historyBack(This) -> boolean()
Types:
  This = wxHtmlWindow()
See external documentation.

historyCanBack(This) -> boolean()
Types:
  This = wxHtmlWindow()
See external documentation.

historyCanForward(This) -> boolean()
Types:
  This = wxHtmlWindow()
See external documentation.

historyClear(This) -> ok
Types:
  This = wxHtmlWindow()
See external documentation.

historyForward(This) -> boolean()
Types:
  This = wxHtmlWindow()
See external documentation.

loadFile(This, Filename) -> boolean()
Types:
  This = wxHtmlWindow()
Filename = unicode:chardata()
See external documentation.

loadPage(This, Location) -> boolean()
Types:
   This = wxHtmlWindow()
   Location = unicode:chardata()
See external documentation.

selectAll(This) -> ok
Types:
   This = wxHtmlWindow()
See external documentation.

selectionToText(This) -> unicode:charlist()
Types:
   This = wxHtmlWindow()
See external documentation.

selectLine(This, Pos) -> ok
Types:
   This = wxHtmlWindow()
   Pos = {X::integer(), Y::integer()}
See external documentation.

selectWord(This, Pos) -> ok
Types:
   This = wxHtmlWindow()
   Pos = {X::integer(), Y::integer()}
See external documentation.

setBorders(This, B) -> ok
Types:
   This = wxHtmlWindow()
   B = integer()
See external documentation.

setFonts(This, Normal_face, Fixed_face) -> ok
Types:
   This = wxHtmlWindow()
   Normal_face = unicode:chardata()
   Fixed_face = unicode:chardata()
Equivalent to setFonts(This, Normal_face, Fixed_face, []).
setFonts(This, Normal_face, Fixed_face, Options::{Option}) -> ok
Types:
   This = wxHtmlWindow()
   Normal_face = unicode:chardata()
   Fixed_face = unicode:chardata()
   Option = {sizes, integer()}
See external documentation.

setPage(This, Source) -> boolean()
Types:
   This = wxHtmlWindow()
   Source = unicode:chardata()
See external documentation.

setRelatedFrame(This, Frame, Format) -> ok
Types:
   This = wxHtmlWindow()
   Frame = wxFrame:wxFrame()
   Format = unicode:chardata()
See external documentation.

setRelatedStatusBar(This, Bar) -> ok
Types:
   This = wxHtmlWindow()
   Bar = integer()
See external documentation.

toText(This) -> unicode:charlist()
Types:
   This = wxHtmlWindow()
See external documentation.

destroy(This::wxHtmlWindow()) -> ok
Destroys this object, do not use object again
wxIcon

Erlang module

See external documentation: wxIcon.

This class is derived (and can use functions) from:
wxBitmap

**DATA TYPES**

wxIcon()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxIcon()

See [external documentation](#).

new(Filename) -> wxIcon()

Types:

```erlang
Filename = unicode:chardata()
```

See [external documentation](#).

Also:

new(Loc) -> wxIcon() when
Loc::wx:wx_object().

Type = ?wxBITMAP_TYPE_INVALID | ?wxBITMAP_TYPE_BMP | ?wxBITMAP_TYPE_BMP_RESOURCE |
| ?wxBITMAP_TYPE_TYPE_RESOURCE | ?wxBITMAP_TYPE_ICO | ?wxBITMAP_TYPE_ICO_RESOURCE |
| ?wxBITMAP_TYPE_CUR | ?wxBITMAP_TYPE_CUR_RESOURCE | ?wxBITMAP_TYPE_XBM | ?
wxBITMAP_TYPE_XBM_DATA | ?wxBITMAP_TYPE_XPM | ?wxBITMAP_TYPE_XPM_DATA | ?
wxBITMAP_TYPE_TIF | ?wxBITMAP_TYPE_TIF_RESOURCE | ?wxBITMAP_TYPE_GIF | ?
wxBITMAP_TYPE_GIF_RESOURCE | ?wxBITMAP_TYPE_PNG | ?wxBITMAP_TYPE_PNG_RESOURCE |
| ?wxBITMAP_TYPE_JPEG | ?wxBITMAP_TYPE_JPEG_RESOURCE | ?wxBITMAP_TYPE_PNM | ?
xBITMAP_TYPE_PNM_RESOURCE | ?wxBITMAP_TYPE_PCX | ?wxBITMAP_TYPE_PCX_RESOURCE |
| ?wxBITMAP_TYPE_PICT | ?wxBITMAP_TYPE_PICT_RESOURCE | ?wxBITMAP_TYPE_ICON | ?
xBITMAP_TYPE_ICON_RESOURCE | ?wxBITMAP_TYPE_ANI | ?wxBITMAP_TYPE_IFF | ?
xBITMAP_TYPE_TGA | ?wxBITMAP_TYPE_MACCURSOR | ?
xBITMAP_TYPE_MACCURSORRESOURCE | ?wxBITMAP_TYPE_ANY
```

new(Filename, Options::[Option]) -> wxIcon()

Types:

```erlang
Filename = unicode:chardata()
```

Option = {type, wx:wx_enum()} | {desiredWidth, integer()} |
| {desiredHeight, integer()}

See [external documentation](#).

copyFromBitmap(This, Bmp) -> ok

Types:

This = wxIcon()
Bmp = wxBitmap:wxBitmap()

See external documentation.

destroy(This::wxIcon()) -> ok

Destroys this object, do not use object again
wxIconBundle

Erlang module

See external documentation: wxIconBundle.

DATA TYPES

wxIconBundle()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxIconBundle()
See external documentation.

call(new(Ic)) -> wxIconBundle()
Types:
   Ic = wxIconBundle() | wxIcon:wxIcon()
See external documentation.

call(new(File, Type)) -> wxIconBundle()
Types:
   File = unicode:chardata()
   Type = integer()
See external documentation.

call(addIcon(This, Icon)) -> ok
Types:
   This = wxIconBundle()
   Icon = wxIcon:wxIcon()
See external documentation.

call(addIcon(This, File, Type)) -> ok
Types:
   This = wxIconBundle()
   File = unicode:chardata()
   Type = integer()
See external documentation.

call(getIcon(This)) -> wxIcon:wxIcon()
Types:
   This = wxIconBundle()
Equivalent to \texttt{getIcon(This, [])}.

\texttt{getIcon(This, Options::[Option]) -> wxIcon:wxIcon()}

Types:

\begin{itemize}
  \item \texttt{This = \texttt{wxIconBundle()}}
  \item \texttt{Option = \{size, integer()\}}
\end{itemize}

See \texttt{external documentation}.

Also:

\texttt{getIcon(This, Size) -> wxIcon:wxIcon() when This::wxIconBundle(), Size::\{W::integer(), H::integer()\}.}

\texttt{destroy(This::wxIconBundle()) -> ok}

Destroys this object, do not use object again
wxIconizeEvent

Erlang module

See external documentation: wxIconizeEvent.

Use wxEvtHandler:connect/3 with EventType:

    iconize

See also the message variant #wxIconize{} event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxIconizeEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

    iconized(This) -> boolean()

Types:

    This = wxIconizeEvent()

See external documentation.
wxIdleEvent

Erlang module

See external documentation: wxIdleEvent.

Use wxEvtHandler:connect/3 with EventType:

idle

See also the message variant #wxIdle{} event record type.

This class is derived (and can use functions) from:

wxEvent

DATA TYPES

wxIdleEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

canSend(Win) -> boolean()

Types:

Win = wxWindow:wxWindow()

This function is deprecated: not available in wxWidgets-2.9 and later

See external documentation.

getMode() -> wx:wx_enum()

See external documentation.

Res = ?wxIDLE_PROCESS_ALL | ?wxIDLE_PROCESS_SPECIFIED

requestMore(This) -> ok

Types:

This = wxIdleEvent()

Equivalent to requestMore(This, []).

requestMore(This, Options::[Option]) -> ok

Types:

This = wxIdleEvent()

Option = {needMore, boolean()}

See external documentation.

moreRequested(This) -> boolean()

Types:

This = wxIdleEvent()
wxIdleEvent

See external documentation.

setMode(Mode) -> ok
Types:

    Mode = wx:wx_enum()

See external documentation.
Mode = ?wxIDLE_PROCESS_ALL | ?wxIDLE_PROCESS_SPECIFIED
wxImage

Erlang module

See external documentation: wxImage.
All (default) image handlers are initialized.

DATA TYPES

wxImage()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxImage()
See external documentation.

new(Name) -> wxImage()
Types:

Name = unicode:chardata()

Equivalent to new(Name, []).

new(Width, Height) -> wxImage()
Types:

Width = integer()
Height = integer()

See external documentation.
Also:
new(Name, [Option]) -> wxImage() when
Name::unicode:chardata(),
Option :: ['type', integer()]
| ['index', integer()].

new(Width, Height, Data) -> wxImage()
Types:

Width = integer()
Height = integer()
Data = binary()

See external documentation.
Also:
new(Width, Height, [Option]) -> wxImage() when
Width::integer(), Height::integer(),
Option :: ['clear', boolean()];
(Name, Mimetype, [Option]) -> wxImage() when
new(Width, Height, Data, Alpha) -> wxImage()

Types:
  Width = integer()
  Height = integer()
  Data = binary()
  Alpha = binary()

See external documentation.

Also:
new(Width, Height, Data, [Option]) -> wxImage() when Width::integer(), Height::integer(), Data::binary(), Option :: [‘static_data’, boolean()].

new(Width, Height, Data, Alpha, Options::[Option]) -> wxImage()

Types:
  Width = integer()
  Height = integer()
  Data = binary()
  Alpha = binary()
  Option = [‘static_data’, boolean()]

See external documentation.

blur(This, Radius) -> wxImage()

Types:
  This = wxImage()
  Radius = integer()

See external documentation.

blurHorizontal(This, Radius) -> wxImage()

Types:
  This = wxImage()
  Radius = integer()

See external documentation.

blurVertical(This, Radius) -> wxImage()

Types:
  This = wxImage()
  Radius = integer()

See external documentation.

convertAlphaToMask(This) -> boolean()

Types:
This = wxImage()
Equivalent to convertAlphaToMask(This, []).

convertAlphaToMask(This, Options::[Option]) -> boolean()
Types:
  This = wxImage()
  Option = {threshold, integer()}
See external documentation.

convertToGreyscale(This) -> wxImage()
Types:
  This = wxImage()
Equivalent to convertToGreyscale(This, []).

convertToGreyscale(This, Options::[Option]) -> wxImage()
Types:
  This = wxImage()
  Option = {lr, number()} | {lg, number()} | {lb, number()}
See external documentation.

cvtColorMono(This, R, G, B) -> wxImage()
Types:
  This = wxImage()
  R = integer()
  G = integer()
  B = integer()
See external documentation.

copy(This) -> wxImage()
Types:
  This = wxImage()
See external documentation.

create(This, Width, Height) -> boolean()
Types:
  This = wxImage()
  Width = integer()
  Height = integer()
Equivalent to create(This, Width, Height, []).

create(This, Width, Height, Data) -> boolean()
Types:
  This = wxImage()
wxImage

\[
\begin{align*}
\text{Width} &= \text{integer()} \\
\text{Height} &= \text{integer()} \\
\text{Data} &= \text{binary()}
\end{align*}
\]

See external documentation.
Also:
create(This, Width, Height, [Option]) -> boolean() when
This::wxImage(), Width::integer(), Height::integer(),
Option :: ['clear', boolean()].

create(This, Width, Height, Data, Alpha) -> boolean()

Types:
\[
\begin{align*}
\text{This} &= \text{wxImage()} \\
\text{Width} &= \text{integer()} \\
\text{Height} &= \text{integer()} \\
\text{Data} &= \text{binary()} \\
\text{Alpha} &= \text{binary()}
\end{align*}
\]

See external documentation.
Also:
create(This, Width, Height, Data, [Option]) -> boolean() when
This::wxImage(), Width::integer(), Height::integer(), Data::binary(),
Option :: ['static_data', boolean()].

create(This, Width, Height, Data, Alpha, Options::[Option]) -> boolean()

Types:
\[
\begin{align*}
\text{This} &= \text{wxImage()} \\
\text{Width} &= \text{integer()} \\
\text{Height} &= \text{integer()} \\
\text{Data} &= \text{binary()} \\
\text{Alpha} &= \text{binary()} \\
\text{Option} &= \{\text{static_data}, \text{boolean()}\}
\end{align*}
\]

See external documentation.

Destroy(This) -> ok

Types:
\[
\begin{align*}
\text{This} &= \text{wxImage()}
\end{align*}
\]

See external documentation.

findFirstUnusedColour(This) -> Result

Types:
\[
\begin{align*}
\text{Result} &= \{\text{Res::boolean()}, \text{R::integer()}, \text{G::integer()}, \text{B::integer()}\} \\
\text{This} &= \text{wxImage()}
\end{align*}
\]

Equivalent to findFirstUnusedColour(This, []).
findFirstUnusedColour(This, Options::[Option]) -> Result
Types:
   Result = {Res::boolean(), R::integer(), G::integer(), B::integer()}
   This = wxImage()
   Option = {startR, integer()} | {startG, integer()} | {startB, integer()}
See external documentation.

getImageExtWildcard() -> unicode:charlist()
See external documentation.

gtAlpha(This) -> binary()
Types:
   This = wxImage()
See external documentation.

gtAlpha(This, X, Y) -> integer()
Types:
   This = wxImage()
   X = integer()
   Y = integer()
See external documentation.

gtBlue(This, X, Y) -> integer()
Types:
   This = wxImage()
   X = integer()
   Y = integer()
See external documentation.

data(This) -> binary()
Types:
   This = wxImage()
See external documentation.

data(This, X, Y) -> integer()
Types:
   This = wxImage()
   X = integer()
   Y = integer()
See external documentation.

tGreen(This, X, Y) -> integer()
Types:
   This = wxImage()
   X = integer()
   Y = integer()
See external documentation.

tImageCount(Name) -> integer()
Types:
Name = unicode:chardata()
Equivalent to getImageCount(Name, []).

g getImageCount(Name, Options::[Option]) -> integer()
Types:
  Name = unicode:chardata()
  Option = {type, wx:wx_enum()}
See external documentation.
Type = ?wxBITMAP_TYPE_INVALID | ?wxBITMAP_TYPE_BMP | ?wxBITMAP_TYPE_BMP_RESOURCE
  | ?wxBITMAP_TYPE_RESOURCE | ?wxBITMAP_TYPE_ICO | ?wxBITMAP_TYPE_ICO_RESOURCE

g getHeight(This) -> integer()
Types:
  This = wxImage()
See external documentation.

g getMaskBlue(This) -> integer()
Types:
  This = wxImage()
See external documentation.

g getMaskGreen(This) -> integer()
Types:
  This = wxImage()
See external documentation.

g getMaskRed(This) -> integer()
Types:
  This = wxImage()
See external documentation.

g getOrFindMaskColour(This) -> Result
Types:
  Result = {Res::boolean(), R::integer(), G::integer(), B::integer()}
  This = wxImage()
See external documentation.

getPalette(This) -> wxPalette:wxPalette()
Types:
  This = wxImage()
See external documentation.

getRed(This, X, Y) -> integer()
Types:
  This = wxImage()
  X = integer()
  Y = integer()
See external documentation.

getSubImage(This, Rect) -> wxImage()
Types:
  This = wxImage()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
See external documentation.

getWidth(This) -> integer()
Types:
  This = wxImage()
See external documentation.

hasAlpha(This) -> boolean()
Types:
  This = wxImage()
See external documentation.

hasMask(This) -> boolean()
Types:
  This = wxImage()
See external documentation.

getOption(This, Name) -> unicode:charlist()
Types:
  This = wxImage()
  Name = unicode:chardata()
See external documentation.

getOptionInt(This, Name) -> integer()
Types:
wxImage

This = wxImage()
Name = unicode:chardata()
See external documentation.

hasOption(This, Name) -> boolean()
Types:
  This = wxImage()
  Name = unicode:chardata()
See external documentation.

initAlpha(This) -> ok
Types:
  This = wxImage()
See external documentation.

initStandardHandlers() -> ok
See external documentation.

isTransparent(This, X, Y) -> boolean()
Types:
  This = wxImage()
  X = integer()
  Y = integer()
Equivalent to isTransparent(This, X, Y, []).

isTransparent(This, X, Y, Options::[Option]) -> boolean()
Types:
  This = wxImage()
  X = integer()
  Y = integer()
  Option = {threshold, integer()}
See external documentation.

loadFile(This, Name) -> boolean()
Types:
  This = wxImage()
  Name = unicode:chardata()
Equivalent to loadFile(This, Name, []).

loadFile(This, Name, Options::[Option]) -> boolean()
Types:
  This = wxImage()
  Name = unicode:chardata()
Option = {type, integer()} | {index, integer()}
See external documentation.

loadFile(This, Name, Mimetype, Options::[Option]) -> boolean()
Types:

    This = wxImage()
    Name = unicode:chardata()
    Mimetype = unicode:chardata()
    Option = {index, integer()}
See external documentation.

ok(This) -> boolean()
Types:

    This = wxImage()
See external documentation.

removeHandler(Name) -> boolean()
Types:

    Name = unicode:chardata()
See external documentation.

mirror(This) -> wxImage()
Types:

    This = wxImage()
Equivalent to mirror(This, []).

mirror(This, Options::[Option]) -> wxImage()
Types:

    This = wxImage()
    Option = {horizontally, boolean()}
See external documentation.

replace(This, R1, G1, B1, R2, G2, B2) -> ok
Types:

    This = wxImage()
    R1 = integer()
    G1 = integer()
    B1 = integer()
    R2 = integer()
    G2 = integer()
    B2 = integer()
See external documentation.
rescale(This, Width, Height) -> wxImage()
Types:
   This = wxImage()
   Width = integer()
   Height = integer()
Equivalent to rescale(This, Width, Height, []).

rescale(This, Width, Height, Options::[Option]) -> wxImage()
Types:
   This = wxImage()
   Width = integer()
   Height = integer()
   Option = {quality, wx:wx_enum()}
See external documentation.
   Quality = integer

resize(This, Size, Pos) -> wxImage()
Types:
   This = wxImage()
   Size = {W::integer(), H::integer()}
   Pos = {X::integer(), Y::integer()}
Equivalent to resize(This, Size, Pos, []).

resize(This, Size, Pos, Options::[Option]) -> wxImage()
Types:
   This = wxImage()
   Size = {W::integer(), H::integer()}
   Pos = {X::integer(), Y::integer()}
   Option = {r, integer()} | {g, integer()} | {b, integer()}
See external documentation.

rotate(This, Angle, Centre_of_rotation) -> wxImage()
Types:
   This = wxImage()
   Angle = number()
   Centre_of_rotation = {X::integer(), Y::integer()}
Equivalent to rotate(This, Angle, Centre_of_rotation, []).

rotate(This, Angle, Centre_of_rotation, Options::[Option]) -> wxImage()
Types:
   This = wxImage()
   Angle = number()
   Centre_of_rotation = {X::integer(), Y::integer()}
Option = \{interpolating, boolean\} | \{offset_after_rotation, 
\{X::integer(), Y::integer\}\}

See external documentation.

rotateHue(This, Angle) -> ok
Types:
  This = wxImage()
  Angle = number()

See external documentation.

rotate90(This) -> wxImage()
Types:
  This = wxImage()
Equivalent to rotate90(This, []).

rotate90(This, Options::[Option]) -> wxImage()
Types:
  This = wxImage()
  Option = \{clockwise, boolean\}\}

See external documentation.

saveFile(This, Name) -> boolean()
Types:
  This = wxImage()
  Name = unicode:chardata()

See external documentation.

saveFile(This, Name, Type) -> boolean()
Types:
  This = wxImage()
  Name = unicode:chardata()
  Type = integer()

See external documentation.

Also:
saveFile(This, Name, Mimetype) -> boolean() when
This::wxImage(), Name::unicode:chardata(), Mimetype::unicode:chardata().

scale(This, Width, Height) -> wxImage()
Types:
  This = wxImage()
  Width = integer()
  Height = integer()
Equivalent to scale(This, Width, Height, []).
wxImage

scale(This, Width, Height, Options::[Option]) -> wxImage()
Types:
   This = wxImage()
   Width = integer()
   Height = integer()
   Option = \{quality, wx:wx_enum()\}
See external documentation.
Quality = integer

size(This, Size, Pos) -> wxImage()
Types:
   This = wxImage()
   Size = \{W::integer(), H::integer()\}
   Pos = \{X::integer(), Y::integer()\}
Equivalent to size(This, Size, Pos, []).

size(This, Size, Pos, Options::[Option]) -> wxImage()
Types:
   This = wxImage()
   Size = \{W::integer(), H::integer()\}
   Pos = \{X::integer(), Y::integer()\}
   Option = \{r, integer()\} | \{g, integer()\} | \{b, integer()\}
See external documentation.

setAlpha(This, Alpha) -> ok
Types:
   This = wxImage()
   Alpha = binary()
Equivalent to setAlpha(This, Alpha, []).

setAlpha(This, Alpha, Options::[Option]) -> ok
Types:
   This = wxImage()
   Alpha = binary()
   Option = \{static_data, boolean()\}
See external documentation.

setAlpha(This, X, Y, Alpha) -> ok
Types:
   This = wxImage()
   X = integer()
   Y = integer()
   Alpha = integer()
See external documentation.

**setData(This, Data) -> ok**

Types:

This = wxImage()
Data = binary()

Equivalent to `setData(This, Data, [])`.

**setData(This, Data, Options::[Option]) -> ok**

Types:

This = wxImage()
Data = binary()
Option = {static_data, boolean()}

See external documentation.

**setData(This, Data, New_width, New_height) -> ok**

Types:

This = wxImage()
Data = binary()
New_width = integer()
New_height = integer()

Equivalent to `setData(This, Data, New_width, New_height, [])`.

**setData(This, Data, New_width, New_height, Options::[Option]) -> ok**

Types:

This = wxImage()
Data = binary()
New_width = integer()
New_height = integer()
Option = {static_data, boolean()}

See external documentation.

**setMask(This) -> ok**

Types:

This = wxImage()

Equivalent to `setMask(This, [])`.

**setMask(This, Options::[Option]) -> ok**

Types:

This = wxImage()
Option = {mask, boolean()}

See external documentation.
wxImage

setMaskColour(This, R, G, B) -> ok
Types:
  This = wxImage()
  R = integer()
  G = integer()
  B = integer()
See external documentation.

setMaskFromImage(This, Mask, Mr, Mg, Mb) -> boolean()
Types:
  This = wxImage()
  Mask = wxImage()
  Mr = integer()
  Mg = integer()
  Mb = integer()
See external documentation.

setOption(This, Name, Value) -> ok
Types:
  This = wxImage()
  Name = unicode:chardata()
  Value = integer()
See external documentation.
Also:
setOption(This, Name, Value) -> 'ok' when
This::wxImage(), Name::unicode:chardata(), Value::unicode:chardata().

setPalette(This, Palette) -> ok
Types:
  This = wxImage()
  Palette = wxPalette:wxPalette()
See external documentation.

setRGB(This, Rect, R, G, B) -> ok
Types:
  This = wxImage()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
  R = integer()
  G = integer()
  B = integer()
See external documentation.
setRGB(This, X, Y, R, G, B) -> ok

Types:
   This = wxImage()
   X = integer()
   Y = integer()
   R = integer()
   G = integer()
   B = integer()

See external documentation.

destroy(This::wxImage()) -> ok

Destroys this object, do not use object again
wxImageList

Erlang module

See external documentation: \texttt{wxImageList}.

**DATA TYPES**

\texttt{wxImageList()}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

\texttt{new()} \rightarrow \texttt{wxImageList()}

See external documentation.

\texttt{new(Width, Height)} \rightarrow \texttt{wxImageList()}

Types:

\begin{verbatim}
  Width = integer()
  Height = integer()
\end{verbatim}

Equivalent to \texttt{new(Width, Height, [])}.

\texttt{new(Width, Height, Options::[Option])} \rightarrow \texttt{wxImageList()}

Types:

\begin{verbatim}
  Width = integer()
  Height = integer()
  Option = \{mask, boolean()\} \mid \{initialCount, integer()\}
\end{verbatim}

See external documentation.

\texttt{add(This, Bitmap)} \rightarrow \texttt{integer()}

Types:

\begin{verbatim}
  This = \texttt{wxImageList()}
  Bitmap = \texttt{wxBitmap:wxBitmap()}
\end{verbatim}

See external documentation.

\texttt{add(This, Bitmap, Mask)} \rightarrow \texttt{integer()}

Types:

\begin{verbatim}
  This = \texttt{wxImageList()}
  Bitmap = \texttt{wxBitmap:wxBitmap()}
  Mask = \texttt{wxBitmap:wxBitmap()}
\end{verbatim}

See external documentation.

Also:

\texttt{add(This, Bitmap, MaskColour)} \rightarrow \texttt{integer()} when
create(This, Width, Height) -> boolean()
Types:
   This = wxImageList()
   Width = integer()
   Height = integer()
Equivalent to create(This, Width, Height, []).

create(This, Width, Height, Options::[Option]) -> boolean()
Types:
   This = wxImageList()
   Width = integer()
   Height = integer()
   Option = {mask, boolean()} | {initialCount, integer()}
See external documentation.

draw(This, Index, Dc, X, Y) -> boolean()
Types:
   This = wxImageList()
   Index = integer()
   Dc = wxDC:wxDC()
   X = integer()
   Y = integer()
Equivalent to draw(This, Index, Dc, X, Y, []).

draw(This, Index, Dc, X, Y, Options::[Option]) -> boolean()
Types:
   This = wxImageList()
   Index = integer()
   Dc = wxDC:wxDC()
   X = integer()
   Y = integer()
   Option = {flags, integer()} | {solidBackground, boolean()}
See external documentation.

getBitmap(This, Index) -> wxBitmap:wxBitmap()
Types:
   This = wxImageList()
   Index = integer()
See external documentation.
wxImageList

getIcon(This, Index) -> wxIcon:wxIcon()
Types:
    This = wxImageList()
    Index = integer()
See external documentation.

getCode(This) -> integer()
Types:
    This = wxImageList()
See external documentation.

getSize(This, Index) -> Result
Types:
    Result = {Res::boolean(), Width::integer(), Height::integer()}
    This = wxImageList()
    Index = integer()
See external documentation.

remove(This, Index) -> boolean()
Types:
    This = wxImageList()
    Index = integer()
See external documentation.

replaceAll(This) -> boolean()
Types:
    This = wxImageList()
See external documentation.

replace(This, Index, Bitmap) -> boolean()
Types:
    This = wxImageList()
    Index = integer()
    Bitmap = wxBitmap:wxBitmap()
See external documentation.

replace(This, Index, Bitmap, Mask) -> boolean()
Types:
    This = wxImageList()
    Index = integer()
    Bitmap = wxBitmap:wxBitmap()
    Mask = wxBitmap:wxBitmap()
See external documentation.
destroy(This::wxImageList()) -> ok
Destroys this object, do not use object again
wxInitDialogEvent

Erlang module

See external documentation: `wxInitDialogEvent`.

Use `wxEvtHandler:connect/3` with EventType:

```
init_dialog
```

See also the message variant `#wxInitDialog/1` event record type.

This class is derived (and can use functions) from:

```
wxEvent
```

**DATA TYPES**

`wxInitDialogEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxJoystickEvent

Erlang module

See external documentation: **wxJoystickEvent**.
Use `wxEvtHandler:connect/3` with EventType:

```
    joy_button_down, joy_button_up, joy_move, joy_zmove
```

See also the message variant `#wxJoystick` event record type.

This class is derived (and can use functions) from: `wxEvent`

**DATA TYPES**

wxJoystickEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`buttonDown(This) -> boolean()`

Types:

```
    This = wxJoystickEvent()
```

Equivalent to `buttonDown(This, [])`.

`buttonDown(This, Options::{Option}) -> boolean()`

Types:

```
    This = wxJoystickEvent()
    Option = {but, integer()}
```

See [external documentation](#).

`buttonIsDown(This) -> boolean()`

Types:

```
    This = wxJoystickEvent()
```

Equivalent to `buttonIsDown(This, [])`.

`buttonIsDown(This, Options::{Option}) -> boolean()`

Types:

```
    This = wxJoystickEvent()
    Option = {but, integer()}
```

See [external documentation](#).

`buttonUp(This) -> boolean()`

Types:
wxJoystickEvent

This = \textit{wxJoystickEvent()}
Equivalent to \textit{buttonUp(This, []).}

\texttt{buttonUp(This, Options::[Option]) \rightarrow boolean()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
    \item Option = \{but, integer()\}
\end{itemize}
See external documentation.

\texttt{getButtonChange(This) \rightarrow integer()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{getButtonState(This) \rightarrow integer()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{getJoystick(This) \rightarrow integer()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{getPosition(This) \rightarrow \{X::integer(), Y::integer()\}}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{getZPosition(This) \rightarrow integer()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{isButton(This) \rightarrow boolean()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.

\texttt{isMove(This) \rightarrow boolean()}
Types:
\begin{itemize}
    \item This = \textit{wxJoystickEvent()}
\end{itemize}
See external documentation.
isZMove(This) -> boolean()

Types:
   This = wxJoystickEvent()

See external documentation.
wxKeyEvent

Erlang module

See external documentation: \texttt{wxKeyEvent}.

Use \texttt{wxEvtHandler:connect/3} with EventType:

\begin{verbatim}
  char, char_hook, key_down, key_up
\end{verbatim}

See also the message variant \#\texttt{wxKey/} event record type.

This class is derived (and can use functions) from:

\texttt{wxEvent}

\textbf{DATA TYPES}

\texttt{wxKeyEvent()}

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

\textbf{Exports}

\begin{verbatim}
altDown(This) -&gt; boolean()
\end{verbatim}

Types:

\begin{verbatim}
This = wxKeyEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
cmdDown(This) -&gt; boolean()
\end{verbatim}

Types:

\begin{verbatim}
This = wxKeyEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
controlDown(This) -&gt; boolean()
\end{verbatim}

Types:

\begin{verbatim}
This = wxKeyEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
getKeyCode(This) -&gt; integer()
\end{verbatim}

Types:

\begin{verbatim}
This = wxKeyEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
getModifiers(This) -&gt; integer()
\end{verbatim}

Types:

\begin{verbatim}
This = wxKeyEvent()
\end{verbatim}

See external documentation.
GetPosition(This) -> {X::integer(), Y::integer()}
Types:
  This = wxKeyEvent()
See external documentation.

getRawKeyCode(This) -> integer()
Types:
  This = wxKeyEvent()
See external documentation.

getRawKeyFlags(This) -> integer()
Types:
  This = wxKeyEvent()
See external documentation.

getUnicodeKey(This) -> integer()
Types:
  This = wxKeyEvent()
See external documentation.

getX(This) -> integer()
Types:
  This = wxKeyEvent()
See external documentation.

getY(This) -> integer()
Types:
  This = wxKeyEvent()
See external documentation.

hasModifiers(This) -> boolean()
Types:
  This = wxKeyEvent()
See external documentation.

metaDown(This) -> boolean()
Types:
  This = wxKeyEvent()
See external documentation.

shiftDown(This) -> boolean()
Types:
  This = wxKeyEvent()
wxKeyEvent

See external documentation.
wxLayoutAlgorithm

Erlang module

See external documentation: **wxLayoutAlgorithm**.

### DATA TYPES

**wxLayoutAlgorithm()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**new() -> wxLayoutAlgorithm()**

See external documentation.

**layoutFrame(This, Frame) -> boolean()**

Types:

- **This** = **wxLayoutAlgorithm()**
- **Frame** = **wxFrame:wxFrame()**

Equivalent to `layoutFrame(This, Frame, [])`.

**layoutFrame(This, Frame, Options::[Option]) -> boolean()**

Types:

- **This** = **wxLayoutAlgorithm()**
- **Frame** = **wxFrame:wxFrame()**
- **Option** = `{mainWindow, wxWindow:wxWindow()}`

See external documentation.

**layoutMDIFrame(This, Frame) -> boolean()**

Types:

- **This** = **wxLayoutAlgorithm()**
- **Frame** = **wxMDIParentFrame:wxMDIParentFrame()**

Equivalent to `layoutMDIFrame(This, Frame, [])`.

**layoutMDIFrame(This, Frame, Options::[Option]) -> boolean()**

Types:

- **This** = **wxLayoutAlgorithm()**
- **Frame** = **wxMDIParentFrame:wxMDIParentFrame()**
- **Option** = `{rect, X::integer(), Y::integer(), W::integer(), H::integer()}`

See external documentation.
wxLayoutAlgorithm

layoutWindow(This, Frame) -> boolean()
Types:
  This = wxLayoutAlgorithm()
  Frame = wxWindow:wxWindow()
Equivalent to layoutWindow(This, Frame, []).

layoutWindow(This, Frame, Options::[Option]) -> boolean()
Types:
  This = wxLayoutAlgorithm()
  Frame = wxWindow:wxWindow()
  Option = {mainWindow, wxWindow:wxWindow()}
See external documentation.

destroy(This::wxLayoutAlgorithm()) -> ok
Destroys this object, do not use object again
**wxListBox**

Erlang module

See external documentation: *wxListBox*.

This class is derived (and can use functions) from:
*wxControlWithItems*
*wxControl*
*wxWindow*
*wxEvtHandler*

**DATA TYPES**

wxListBox()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxListBox()

See external documentation.

new(Parent, Id) -> wxListBox()

Types:

Parent = *wxWindow:*wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxListBox()

Types:

Parent = *wxWindow:*wxWindow()
Id = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {choices, [unicode:chardata()]} | {style, integer()} | {validator, *wx:*wx_object()}

See external documentation.

create(This, Parent, Id, Pos, Size, Choices) -> boolean()

Types:

This = *wxListbox()
Parent = *wxWindow:*wxWindow()
Id = integer()
Pos = {X::integer(), Y::integer()}
Size = {W::integer(), H::integer()}
Choices = [unicode:chardata()]
wxListBox

Equivalent to `create(This, Parent, Id, Pos, Size, Choices, []).`

```erlang
create(This, Parent, Id, Pos, Size, Choices, Options::[Option]) -> boolean()
```

Types:
- `This` = `wxListBox()`
- `Parent` = `wxWindow::wxWindow()`
- `Id` = `integer()`
- `Pos` = `{X::integer(), Y::integer()}`
- `Size` = `{W::integer(), H::integer()}`
- `Choices` = `[unicode:chardata()]`
- `Option` = `{style, integer()} | {validator, wx:wx_object()}`

See external documentation.

```erlang
deselect(This, N) -> ok
```

Types:
- `This` = `wxListBox()`
- `N` = `integer()`

See external documentation.

```erlang
getSelections(This) -> Result
```

Types:
- `Result` = `{Res::integer(), ASelections::[integer()]}`
- `This` = `wxListBox()`

See external documentation.

```erlang
insertItems(This, Items, Pos) -> ok
```

Types:
- `This` = `wxListBox()`
- `Items` = `[unicode:chardata()]`
- `Pos` = `integer()`

See external documentation.

```erlang
isSelected(This, N) -> boolean()
```

Types:
- `This` = `wxListBox()`
- `N` = `integer()`

See external documentation.

```erlang
set(This, Items) -> ok
```

Types:
- `This` = `wxListBox()`
- `Items` = `[unicode:chardata()]`

See external documentation.
hitTest(This, Point) -> integer()
Types:
  This = wxListBox()
  Point = {X::integer(), Y::integer()}
See external documentation.

setFirstItem(This, N) -> ok
Types:
  This = wxListBox()
  N = integer()
See external documentation.
Also:
setFirstItem(This, S) -> 'ok' when
This::wxListBox(), S::unicode:chardata().

destroy(This::wxListBox()) -> ok
Destroys this object, do not use object again
**wxListCtrl**

Erlang module

See external documentation: **wxListCtrl**.

This class is derived (and can use functions) from:

*wxControl*

*wxWindow*

*wxEvtHandler*

**DATA TYPES**

**wxListCtrl()**

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

**Exports**

**new()** -> **wxListCtrl()**

See external documentation.

**new(Parent) -> wxListCtrl()**

Types:

Parent = **wxWindow::wxWindow()**

**new(Parent, Options::[Option])** -> **wxListCtrl()**

Types:

Parent = **wxWindow::wxWindow()**

Option = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, **wx::wx_object()**} | {onGetItemText, function()} | {onGetItemAttr, function()} | {onGetItemColumnImage, function()}

Creates a listctrl with optional callback functions:

OnGetItemText = (This, Item, Column) -> unicode:charlist() OnGetItemAttr = (This, Item) -> wxListItemAttr:wxListItemAttr() OnGetItemColumnImage = (This, Item, Column) -> integer()

See external documentation.

**arrange(This) -> boolean()**

Types:

This = **wxListCtrl()**

Equivalent to **arrange(This, [])**.

**arrange(This, Options::[Option])** -> **boolean()**

Types:

This = **wxListCtrl()**
Option = {flag, integer()}
See external documentation.

assignImageList(This, ImageList, Which) -> ok
Types:
  This = wxListCtrl()
  ImageList = wxImageList:wxImageList()
  Which = integer()
See external documentation.

clearAll(This) -> ok
Types:
  This = wxListCtrl()
See external documentation.

create(This, Parent) -> wxListCtrl()
Types:
  This = wxWindow:wxWindow()
  Parent = wxWindow:wxWindow()
Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> wxListCtrl()
Types:
  This = wxWindow:wxWindow()
  Parent = wxWindow:wxWindow()
  Option = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()} | {onGetItemText, function()} | {onGetItemAttr, function()} | {onGetItemColumnImage, function()}
See external documentation.

deleteAllItems(This) -> boolean()
Types:
  This = wxListCtrl()
See external documentation.

deleteColumn(This, Col) -> boolean()
Types:
  This = wxListCtrl()
  Col = integer()
See external documentation.
wxListCtrl

deleteItem(This, Item) -> boolean()
Types:
   This = wxListCtrl()
   Item = integer()
See external documentation.

editLabel(This, Item) -> wxTextCtrl:wxTextCtrl()
Types:
   This = wxListCtrl()
   Item = integer()
See external documentation.

ensureVisible(This, Item) -> boolean()
Types:
   This = wxListCtrl()
   Item = integer()
See external documentation.

findItem(This, Start, Str) -> integer()
Types:
   This = wxListCtrl()
   Start = integer()
   Str = unicode:chardata()
Equivalent to findItem(This, Start, Str, []).

findItem(This, Start, Str, Options::[Option]) -> integer()
Types:
   This = wxListCtrl()
   Start = integer()
   Str = unicode:chardata()
   Option = {partial, boolean()}
See external documentation.
Also:
findItem(This, Start, Pt, Direction) -> integer() when
This::wxListCtrl(), Start::integer(), Pt::[X::integer(), Y::integer()], Direction::integer().

columnGet(This, Col, Item) -> boolean()
Types:
   This = wxListCtrl()
   Col = integer()
   Item = wxListItem:wxListItem()
See external documentation.
getColumnCount(This) -> integer()
Types:
  This = wxListCtrl()
See external documentation.

columnWidth(This, Col) -> integer()
Types:
  This = wxListCtrl()
  Col = integer()
See external documentation.

countPerPage(This) -> integer()
Types:
  This = wxListCtrl()
See external documentation.

geditControl(This) -> wxTextCtrl:wxTextCtrl()
Types:
  This = wxListCtrl()
See external documentation.

gimageList(This, Which) -> wxImageList:wxImageList()
Types:
  This = wxListCtrl()
  Which = integer()
See external documentation.

getItem(This, Info) -> boolean()
Types:
  This = wxListCtrl()
  Info = wxListItem:wxListItem()
See external documentation.

getItemBackgroundColour(This, Item) -> wx:wx_colour4()
Types:
  This = wxListCtrl()
  Item = integer()
See external documentation.

getItemCount(This) -> integer()
Types:
  This = wxListCtrl()
See external documentation.
wxListCtrl

getItemData(This, Item) -> integer()
Types:
   This = wxListCtrl()
   Item = integer()
See external documentation.

ggetItemFont(This, Item) -> wxFont:wxFont()
Types:
   This = wxListCtrl()
   Item = integer()
See external documentation.

ggetItemPosition(This, Item) -> Result
Types:
   Result = {Res::boolean(), Pos::{X::integer(), Y::integer()}}
   This = wxListCtrl()
   Item = integer()
See external documentation.

ggetItemRect(This, Item) -> Result
Types:
   Result = {Res::boolean(), Rect::{X::integer(), Y::integer(), W::integer(),
   H::integer()}}
   This = wxListCtrl()
   Item = integer()
Equivalent to getItemRect(This, Item, []).

ggetItemRect(This, Item, Options::[Option]) -> Result
Types:
   Result = {Res::boolean(), Rect::{X::integer(), Y::integer(), W::integer(),
   H::integer()}}
   This = wxListCtrl()
   Item = integer()
   Option = {code, integer()}
See external documentation.

ggetItemSpacing(This) -> {W::integer(), H::integer()}
Types:
   This = wxListCtrl()
See external documentation.

ggetItemState(This, Item, StateMask) -> integer()
Types:

This = wxListCtrl()
Item = integer()
StateMask = integer()

See external documentation.

getItemText(This, Item) -> unicode:charlist()
Types:
    This = wxListCtrl()
    Item = integer()

See external documentation.

getItemTextColour(This, Item) -> wx:wx_colour4()
Types:
    This = wxListCtrl()
    Item = integer()

See external documentation.

getNextItem(This, Item) -> integer()
Types:
    This = wxListCtrl()
    Item = integer()

Equivalent to getNextItem(This, Item, []).

getNextItem(This, Item, Options::[Option]) -> integer()
Types:
    This = wxListCtrl()
    Item = integer()
    Option = {geometry, integer()} | {state, integer()}

See external documentation.

getSelectedItemCount(This) -> integer()
Types:
    This = wxListCtrl()

See external documentation.

getTextColour(This) -> wx:wx_colour4()
Types:
    This = wxListCtrl()

See external documentation.

getTopItem(This) -> integer()
Types:
    This = wxListCtrl()
See external documentation.

getViewRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
  This = wxListCtrl()
See external documentation.

hitTest(This, Point) -> Result
Types:
  Result = {Res::integer(), Flags::integer(), PSubItem::integer()}
  This = wxListCtrl()
  Point = {X::integer(), Y::integer()}
See external documentation.

insertColumn(This, Col, Heading) -> integer()
Types:
  This = wxListCtrl()
  Col = integer()
  Heading = unicode:chardata()
See external documentation.
Also:
insertColumn(This, Col, Info) -> integer() when
This::wxListCtrl(), Col::integer(), Info::wxListItem:wxListItem().
insertColumn(This, Col, Heading, Options::[Option]) -> integer()
Types:
  This = wxListCtrl()
  Col = integer()
  Heading = unicode:chardata()
  Option = {format, integer()} | {width, integer()}
See external documentation.

insertItem(This, Info) -> integer()
Types:
  This = wxListCtrl()
  Info = wxListItem:wxListItem()
See external documentation.

insertItem(This, Index, ImageIndex) -> integer()
Types:
  This = wxListCtrl()
  Index = integer()
  ImageIndex = integer()
insertItem(This, Index, Label) -> integer() when This::wxListCtrl(), Index::integer(), Label::unicode:chardata().

insertItem(This, Index, Label, ImageIndex) -> integer()
Types:
  This = wxListCtrl()
  Index = integer()
  Label = unicode:chardata()
  ImageIndex = integer()
See external documentation.

refreshItem(This, Item) -> ok
Types:
  This = wxListCtrl()
  Item = integer()
See external documentation.

refreshItems(This, ItemFrom, ItemTo) -> ok
Types:
  This = wxListCtrl()
  ItemFrom = integer()
  ItemTo = integer()
See external documentation.

scrollList(This, Dx, Dy) -> boolean()
Types:
  This = wxListCtrl()
  Dx = integer()
  Dy = integer()
See external documentation.

setBackgroundColour(This, Colour) -> boolean()
Types:
  This = wxListCtrl()
  Colour = wx:wx_colour()
See external documentation.

setColumn(This, Col, Item) -> boolean()
Types:
  This = wxListCtrl()
  Col = integer()
  Item = wxListItem:wxListItem()
wxListCtrl

See external documentation.

```erlang
setColumnWidth(This, Col, Width) -> boolean()
Types:
    This = wxListCtrl()
    Col = integer()
    Width = integer()

See external documentation.
```

```erlang
setImageList(This, ImageList, Which) -> ok
Types:
    This = wxListCtrl()
    ImageList = wxImageList:wxImageList()
    Which = integer()

See external documentation.
```

```erlang
setItem(This, Info) -> boolean()
Types:
    This = wxListCtrl()
    Info = wxListItem:wxListItem()

See external documentation.
```

```erlang
setItem(This, Index, Col, Label) -> integer()
Types:
    This = wxListCtrl()
    Index = integer()
    Col = integer()
    Label = unicode:chardata()

Equivalent to `setItem(This, Index, Col, Label, []).`
```

```erlang
setItem(This, Index, Col, Label, Options::[Option]) -> integer()
Types:
    This = wxListCtrl()
    Index = integer()
    Col = integer()
    Label = unicode:chardata()
    Option = {imageId, integer()}

See external documentation.
```

```erlang
setItemBackgroundColour(This, Item, Col) -> ok
Types:
    This = wxListCtrl()
    Item = integer()
```

340 | Ericsson AB. All Rights Reserved.: wxErlang
Col = \texttt{wx:wx\_colour()}

See external documentation.

\texttt{setItemCount(This, Count) -> ok}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Count = \texttt{integer()}
\end{verbatim}

See external documentation.

\texttt{setItemData(This, Item, Data) -> boolean()}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Item = \texttt{integer()}
  Data = \texttt{integer()}
\end{verbatim}

See external documentation.

\texttt{setItemFont(This, Item, F) -> ok}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Item = \texttt{integer()}
  F = \texttt{wxFont:wxFont()}
\end{verbatim}

See external documentation.

\texttt{setItemImage(This, Item, Image) -> boolean()}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Item = \texttt{integer()}
  Image = \texttt{integer()}
\end{verbatim}

Equivalent to \texttt{setItemImage(This, Item, Image, []).}

\texttt{setItemImage(This, Item, Image, Options::[Option]) -> boolean()}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Item = \texttt{integer()}
  Image = \texttt{integer()}
  Option = \{selImage, \texttt{integer()}\}
\end{verbatim}

See external documentation.

\texttt{setItemColumnImage(This, Item, Column, Image) -> boolean()}

Types:
\begin{verbatim}
  This = \texttt{wxListCtrl()}
  Item = \texttt{integer()}
  Column = \texttt{integer()}
\end{verbatim}
wxListCtrl

Image = integer()
See external documentation.

setItemPosition(This, Item, Pos) -> boolean()
Types:
   This = wxListCtrl()
   Item = integer()
   Pos = {X::integer(), Y::integer()}
See external documentation.

setItemState(This, Item, State, StateMask) -> boolean()
Types:
   This = wxListCtrl()
   Item = integer()
   State = integer()
   StateMask = integer()
See external documentation.

setItemText(This, Item, Str) -> ok
Types:
   This = wxListCtrl()
   Item = integer()
   Str = unicode:chardata()
See external documentation.

setItemTextColour(This, Item, Col) -> ok
Types:
   This = wxListCtrl()
   Item = integer()
   Col = wx:wx_colour()
See external documentation.

setSingleStyle(This, Style) -> ok
Types:
   This = wxListCtrl()
   Style = integer()
Equivalent to setSingleStyle(This, Style, []).

setSingleStyle(This, Style, Options::[Option]) -> ok
Types:
   This = wxListCtrl()
   Style = integer()
   Option = {add, boolean()}

342 | Ericsson AB. All Rights Reserved.: wxErlang
See external documentation.

```erl
setTextColour(This, Col) -> ok
Types:
   This = wxListCtrl()
   Col = wx:wx_colour()
See external documentation.
```

```erl
setWindowStyleFlag(This, Style) -> ok
Types:
   This = wxListCtrl()
   Style = integer()
See external documentation.
```

```erl
sortItems(This::wxListCtrl(), SortCallBack::function()) -> boolean()
Sort the items in the list control
SortCallBack(Item1,Item2) -> integer()
SortCallBack receives the client data associated with two items to compare, and should return 0 if the items are equal, a negative value if the first item is less than the second one and a positive value if the first item is greater than the second one.
NOTE: The callback may not call other (wx) processes.
```

```erl
destroy(This::wxListCtrl()) -> ok
Destroys this object, do not use object again
```
wxListEvent

Erlang module

See external documentation: wxListEvent.

Use wxEvtHandler:connect/3 with EventType:

command_list_begin_drag, command_list_begin_rdrag,
command_list_end_label_edit, command_list_delete_item,
command_list_key_down, command_list_insert_item,
command_list_col_right_click, command_list_col_begin_drag,
command_list_col_end_drag, command_list_item_selected,
command_list_item_right_click, command_list_item_middle_click,
command_list_item_focused, command_list_cache_hint

See also the message variant #wxList() event record type.

This class is derived (and can use functions) from:
wxNotifyEvent
wxCommandEvent
wxEvent

DATA TYPES

wxListEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

getcheCacheFrom(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getcheCacheTo(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getKeyCode(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getIndex(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getColumn(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getPoint(This) -> {X::integer(), Y::integer()}
Types:
   This = wxListEvent()
See external documentation.

getLabel(This) -> unicode:charlist()
Types:
   This = wxListEvent()
See external documentation.

getText(This) -> unicode:charlist()
Types:
   This = wxListEvent()
See external documentation.

getImage(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

data(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getMask(This) -> integer()
Types:
   This = wxListEvent()
See external documentation.

getItem(This) -> wxListItem:wxListItem()
Types:
   This = wxListEvent()
See external documentation.
isEditCancelled(This) -> boolean()

Types:
   This = wxListEvent()

See external documentation.
wxListItem

Erlang module

See external documentation: wxListItem.

DATA TYPES

wxListItem()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

new() -> wxListItem()
See external documentation.

new(Item) -> wxListItem()
Types:
  Item = wxListItem()
See external documentation.

clear(This) -> ok
Types:
  This = wxListItem()
See external documentation.

getAlign(This) -> wx:wx_enum()
Types:
  This = wxListItem()
See external documentation.


getBackgroundColour(This) -> wx:wx_colour4()
Types:
  This = wxListItem()
See external documentation.

column(This) -> integer()
Types:
  This = wxListItem()
See external documentation.
getFont(This) -> wxFont:wxFont()
Types:
   This = wxListItem()
See external documentation.

getId(This) -> integer()
Types:
   This = wxListItem()
See external documentation.

getImage(This) -> integer()
Types:
   This = wxListItem()
See external documentation.

getMask(This) -> integer()
Types:
   This = wxListItem()
See external documentation.

getState(This) -> integer()
Types:
   This = wxListItem()
See external documentation.

getText(This) -> unicode:charlist()
Types:
   This = wxListItem()
See external documentation.

getTextColour(This) -> wx:wx_colour4()
Types:
   This = wxListItem()
See external documentation.

getWidth(This) -> integer()
Types:
   This = wxListItem()
See external documentation.

setAlign(This, Align) -> ok
Types:
   This = wxListItem()
wxListItem

Align = \texttt{wx:wx\_enum()}

See external documentation.

setBackgroundColour(This, ColBack) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item ColBack = \texttt{wx:wx\_colour()}
\end{itemize}

See external documentation.

setColumn(This, Col) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item Col = \texttt{integer()}
\end{itemize}

See external documentation.

setFont(This, Font) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item Font = \texttt{wxFont:wxFont()}
\end{itemize}

See external documentation.

setId(This, Id) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item Id = \texttt{integer()}
\end{itemize}

See external documentation.

setImage(This, Image) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item Image = \texttt{integer()}
\end{itemize}

See external documentation.

setMask(This, Mask) -> ok
Types:
\begin{itemize}
\item This = \texttt{wxListItem()}
\item Mask = \texttt{integer()}
\end{itemize}

See external documentation.

setState(This, State) -> ok
Types:
wxListItem

This = wxListItem()
State = integer()
See external documentation.

setStateMask(This, StateMask) -> ok
Types:
   This = wxListItem()
   StateMask = integer()
See external documentation.

setText(This, Text) -> ok
Types:
   This = wxListItem()
   Text = unicode:chardata()
See external documentation.

setTextColour(This, ColText) -> ok
Types:
   This = wxListItem()
   ColText = wx:wx_colour()
See external documentation.

setWidth(This, Width) -> ok
Types:
   This = wxListItem()
   Width = integer()
See external documentation.

destroy(This::wxListItem()) -> ok
Destroys this object, do not use object again
wxListItemAttr

Erlang module

See external documentation: wxListItemAttr.

DATA TYPES

wxListItemAttr()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxListItemAttr()
See external documentation.

new(ColText, ColBack, Font) -> wxListItemAttr()
Types:
  ColText = wx:wx_colour()
  ColBack = wx:wx_colour()
  Font = wxFont:wxFont()
See external documentation.

getBackgroundColour(This) -> wx:wx_colour4()
Types:
  This = wxListItemAttr()
See external documentation.

gETCHfont(This) -> wxFont:wxFont()
Types:
  This = wxListItemAttr()
See external documentation.

getTextColour(This) -> wx:wx_colour4()
Types:
  This = wxListItemAttr()
See external documentation.

hasBackgroundColor(This) -> boolean()
Types:
  This = wxListItemAttr()
See external documentation.
wxListItemAttr

hasFont(This) -> boolean()
Types:
    This = wxListItemAttr()
See external documentation.

hasTextColour(This) -> boolean()
Types:
    This = wxListItemAttr()
See external documentation.

setBackgroundColor(This, ColBack) -> ok
Types:
    This = wxListItemAttr()
    ColBack = wx:wx_colour()
See external documentation.

setFont(This, Font) -> ok
Types:
    This = wxListItemAttr()
    Font = wxFont:wxFont()
See external documentation.

setTextColour(This, ColText) -> ok
Types:
    This = wxListItemAttr()
    ColText = wx:wx_colour()
See external documentation.

destroy(This::wxListItemAttr()) -> ok
Destroys this object, do not use object again
wxListView

Erlang module

See external documentation: `wxListView`.
This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

`wxListView()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

clearColumnImage(This, Col) -> ok
Types:
    `This = wxListView()`
    `Col = integer()`
See external documentation.

focus(This, Index) -> ok
Types:
    `This = wxListView()`
    `Index = integer()`
See external documentation.

getFirstSelected(This) -> integer()
Types:
    `This = wxListView()`
See external documentation.

getFocusedItem(This) -> integer()
Types:
    `This = wxListView()`
See external documentation.

getNextSelected(This, Item) -> integer()
Types:
    `This = wxListView()`
    `Item = integer()`
See external documentation.

isSelected(This, Index) -> boolean()
Types:
   This = wxListView()
   Index = integer()
See external documentation.

select(This, N) -> ok
Types:
   This = wxListView()
   N = integer()
Equivalent to select(This, N, []).

select(This, N, Options::[Option]) -> ok
Types:
   This = wxListView()
   N = integer()
   Option = {on, boolean()}
See external documentation.

setColumnImage(This, Col, Image) -> ok
Types:
   This = wxListView()
   Col = integer()
   Image = integer()
See external documentation.
wxListbook

Erlang module

See external documentation: wxListbook.
This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxListbook()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparsion stored on disc or distributed for use on other nodes.

Exports

new() -> wxListbook()
See external documentation.
	new(Parent, Id) -> wxListbook()
Types:

Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxListbook()
Types:

Parent = wxWindow:wxWindow()
Id = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(),
H::integer()}} | {style, integer()}

See external documentation.

addPage(This, Page, Text) -> boolean()
Types:

This = wxListbook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()

Equivalent to addPage(This, Page, Text, []).

addPage(This, Page, Text, Options::[Option]) -> boolean()
Types:
This = wxListbook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean()} | {imageId, integer()}

See external documentation.

advanceSelection(This) -> ok
Types:
  This = wxListbook()
Equivalent to advanceSelection(This, []).

advanceSelection(This, Options::[Option]) -> ok
Types:
  This = wxListbook()
  Option = {forward, boolean()}
See external documentation.

assignImageList(This, ImageList) -> ok
Types:
  This = wxListbook()
  ImageList = wxImageList:wxImageList()
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
  This = wxListbook()
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
  This = wxListbook()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

deleteAllPages(This) -> boolean()
Types:
  This = wxListbook()
See external documentation.
deletePage(This, N) -> boolean()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

removePage(This, N) -> boolean()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

getCurrentPage(This) -> wxWindow:wxWindow()
Types:
   This = wxListbook()
See external documentation.

getImageList(This) -> wxImageList:wxImageList()
Types:
   This = wxListbook()
See external documentation.

getPage(This, N) -> wxWindow:wxWindow()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

getPageCount(This) -> integer()
Types:
   This = wxListbook()
See external documentation.

getPageImage(This, N) -> integer()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

getPageText(This, N) -> unicode:charlist()
Types:
   This = wxListbook()
   N = integer()
wxListbook

See external documentation.

getSelection(This) -> integer()
Types:
  This = wxListbook()
See external documentation.

hitTest(This, Pt) -> Result
Types:
  Result = {Res::integer(), Flags::integer()}
  This = wxListbook()
  Pt = {X::integer(), Y::integer()}
See external documentation.

insertPage(This, N, Page, Text) -> boolean()
Types:
  This = wxListbook()
  N = integer()
  Page = wxWindow:wxWindow()
  Text = unicode:chardata()
Equivalent to insertPage(This, N, Page, Text, []).

insertPage(This, N, Page, Text, Options::[Option]) -> boolean()
Types:
  This = wxListbook()
  N = integer()
  Page = wxWindow:wxWindow()
  Text = unicode:chardata()
  Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

setImageList(This, ImageList) -> ok
Types:
  This = wxListbook()
  ImageList = wxImageList:wxImageList()
See external documentation.

setPageSize(This, Size) -> ok
Types:
  This = wxListbook()
  Size = {W::integer(), H::integer()}
See external documentation.
setPageImage(This, N, ImageId) -> boolean()
Types:
   This = wxListbook()
   N = integer()
   ImageId = integer()
See external documentation.

setPageText(This, N, StrText) -> boolean()
Types:
   This = wxListbook()
   N = integer()
   StrText = unicode:chardata()
See external documentation.

setSelection(This, N) -> integer()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

changeSelection(This, N) -> integer()
Types:
   This = wxListbook()
   N = integer()
See external documentation.

destroy(This::wxListbook()) -> ok
Destroys this object, do not use object again
wxLocale

Erlang module

See external documentation: wxLocale.

DATA TYPES

wxLocale()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxLocale()
See external documentation.

new(Language) -> wxLocale()
Types:
  Language = integer()
Equivalent to new(Language, []).

new(Language, Options::[Option]) -> wxLocale()
Types:
  Language = integer()
  Option = {flags, integer()}
See external documentation.

init(This) -> boolean()
Types:
  This = wxLocale()
Equivalent to init(This, []).

init(This, Options::[Option]) -> boolean()
Types:
  This = wxLocale()
  Option = {language, integer()} | {flags, integer()}
See external documentation.

addCatalog(This, SzDomain) -> boolean()
Types:
  This = wxLocale()
  SzDomain = unicode:chardata()
See external documentation.
addCatalog(This, SzDomain, MsgIdLanguage, MsgIdCharset) -> boolean()

Types:

This = wxLocale()

SzDomain = unicode:chardata()

MsgIdLanguage = wx:wx_enum()

MsgIdCharset = unicode:chardata()

See external documentation.

addCatalogLookupPathPrefix(Prefix) -> ok

Types:

Prefix = unicode:chardata()
getCanonicalName(This) -> unicode:charlist()
Types:
   This = wxLocale()
See external documentation.

getLanguage(This) -> integer()
Types:
   This = wxLocale()
See external documentation.

getLanguageName(Lang) -> unicode:charlist()
Types:
   Lang = integer()
See external documentation.

getLocale(This) -> unicode:charlist()
Types:
   This = wxLocale()
See external documentation.

getName(This) -> unicode:charlist()
Types:
   This = wxLocale()
See external documentation.

getString(This, SzOrigString) -> unicode:charlist()
Types:
   This = wxLocale()
   SzOrigString = unicode:chardata()
Equivalent to getString(This, SzOrigString, []).

getString(This, SzOrigString, Options::[Option]) -> unicode:charlist()
Types:
   This = wxLocale()
   SzOrigString = unicode:chardata()
   Option = {szDomain, unicode:chardata()}
See external documentation.

getString(This, SzOrigString, SzOrigString2, N) -> unicode:charlist()
Types:
   This = wxLocale()
   SzOrigString = unicode:chardata()
   SzOrigString2 = unicode:chardata()
**wxLocale**

\[
N = \text{integer()}
\]
Equivalent to `getString(This, SzOrigString, SzOrigString2, N, []).`

`getString(This, SzOrigString, SzOrigString2, N, Options::[Option]) -> unicode:charlist()`

Types:
- `This = wxLocale()`
- `SzOrigString = unicode:chardata()`
- `SzOrigString2 = unicode:chardata()`
- `N = integer()`
- `Option = {szDomain, unicode:chardata()}`

See external documentation.

`getHeaderValue(This, SzHeader) -> unicode:charlist()`

Types:
- `This = wxLocale()`
- `SzHeader = unicode:chardata()`

Equivalent to `getHeaderValue(This, SzHeader, [])`.

`getHeaderValue(This, SzHeader, Options::[Option]) -> unicode:charlist()`

Types:
- `This = wxLocale()`
- `SzHeader = unicode:chardata()`
- `Option = {szDomain, unicode:chardata()}`

See external documentation.

`getSysName(This) -> unicode:charlist()`

Types:
- `This = wxLocale()`

See external documentation.

`getSystemEncoding() -> wx:wx_enum()`

See external documentation.

wxLocale

```
getSystemEncodingName() -> unicode:charlist()

See external documentation.

getSystemLanguage() -> integer()

See external documentation.

isLoaded(This, SzDomain) -> boolean()
Types:
  This = wxLocale()
  SzDomain = unicode:chardata()

See external documentation.

isOk(This) -> boolean()
Types:
  This = wxLocale()

See external documentation.

destroy(This::wxLocale()) -> ok
Destroys this object, do not use object again
```
wxLogNull

Erlang module

See external documentation: wxLogNull.

DATA TYPES

wxLogNull()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxLogNull()
See external documentation.

destroy(This::wxLogNull()) -> ok
Destroy this object, do not use object again
wxMDIChildFrame

Erlang module

See external documentation: wxMDIChildFrame.
This class is derived (and can use functions) from:
wxFrame
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES
wxMDIChildFrame()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxMDIChildFrame()
See external documentation.

new(Parent, Id, Title) -> wxMDIChildFrame()
Types:
    Parent = wxMDIParentFrame:wxMDIParentFrame()
    Id = integer()
    Title = unicode:chardata()
Equivalent to new(Parent, Id, Title, []).

new(Parent, Id, Title, Options::[Option]) -> wxMDIChildFrame()
Types:
    Parent = wxMDIParentFrame:wxMDIParentFrame()
    Id = integer()
    Title = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} \ {size, {W::integer(), H::integer()}} \ {style, integer()}
See external documentation.

activate(This) -> ok
Types:
    This = wxMDIChildFrame()
See external documentation.
wxMDIChildFrame

create(This, Parent, Id, Title) -> boolean()
Types:
  This = wxMDIChildFrame()
  Parent = wxMDIParentFrame:wxMDIParentFrame()
  Id = integer()
  Title = unicode:chardata()
Equivalent to create(This, Parent, Id, Title, []).

create(This, Parent, Id, Title, Options::[Option]) -> boolean()
Types:
  This = wxMDIChildFrame()
  Parent = wxMDIParentFrame:wxMDIParentFrame()
  Id = integer()
  Title = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

maximize(This) -> ok
Types:
  This = wxMDIChildFrame()
Equivalent to maximize(This, []).

maximize(This, Options::[Option]) -> ok
Types:
  This = wxMDIChildFrame()
  Option = {maximize, boolean()}
See external documentation.

restore(This) -> ok
Types:
  This = wxMDIChildFrame()
See external documentation.

destroy(This::wxMDIChildFrame()) -> ok
Destroys this object, do not use object again
wxMDIClientWindow

Erlang module

See external documentation: wxMDIClientWindow.

This class is derived (and can use functions) from:
- wxWindow
- wxEvtHandler

**DATA TYPES**

wxMDIClientWindow()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxMDIClientWindow()

See external documentation.

new(Parent) -> wxMDIClientWindow()

Types:
- Parent = wxMDIParentFrame:wxMDIParentFrame()

This function is deprecated: not available in wxWidgets-2.9 and later

Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxMDIClientWindow()

Types:
- Parent = wxMDIParentFrame:wxMDIParentFrame()
- Option = {style, integer()}

This function is deprecated: not available in wxWidgets-2.9 and later

See external documentation.

createClient(This, Parent) -> boolean()

Types:
- This = wxMDIClientWindow()
- Parent = wxMDIParentFrame:wxMDIParentFrame()

Equivalent to createClient(This, Parent, []).

createClient(This, Parent, Options::[Option]) -> boolean()

Types:
- This = wxMDIClientWindow()
- Parent = wxMDIParentFrame:wxMDIParentFrame()
- Option = {style, integer()}
See external documentation.

```erlang
destroy(This::wxMDIClientWindow()) -> ok
```

Destroys this object, do not use object again
wxMDIParentFrame

Erlang module

See external documentation: wxMDIParentFrame.

This class is derived (and can use functions) from:
wxFrame
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxMDIParentFrame()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxMDIParentFrame()
See external documentation.

new(Parent, Id, Title) -> wxMDIParentFrame()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
Equivalent to new(Parent, Id, Title, []).

new(Parent, Id, Title, Options::[Option]) -> wxMDIParentFrame()
Types:
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

activateNext(This) -> ok
Types:
  This = wxMDIParentFrame()
See external documentation.
activatePrevious(This) -> ok
Types:
    This = wxMDIParentFrame()
See external documentation.

arrangeIcons(This) -> ok
Types:
    This = wxMDIParentFrame()
See external documentation.

cascade(This) -> ok
Types:
    This = wxMDIParentFrame()
See external documentation.

create(This, Parent, Id, Title) -> boolean()
Types:
    This = wxMDIParentFrame()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Title = unicode:chardata()
Equivalent to create(This, Parent, Id, Title, []).

create(This, Parent, Id, Title, Options::[Option]) -> boolean()
Types:
    This = wxMDIParentFrame()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Title = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

getActiveChild(This) -> wxMDIChildFrame:wxMDIChildFrame()
Types:
    This = wxMDIParentFrame()
See external documentation.

getClientWindow(This) -> wxMDIClientWindow:wxMDIClientWindow()
Types:
    This = wxMDIParentFrame()
See external documentation.
tile(This) -> ok
Types:
  This = wxMDIParentFrame()
Equivalent to tile(This, []).

tile(This, Options::[Option]) -> ok
Types:
  This = wxMDIParentFrame()
  Option = {orient, wx:wx_enum()}
See external documentation.
Orient = ?wxHORIZONTAL | ?wxVERTICAL | ?wxBOTH

destroy(This::wxMDIParentFrame()) -> ok
Destroys this object, do not use object again
wxMask

Erlang module

See external documentation: wxMask.

DATA TYPES

wxMask()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxMask()
See external documentation.

new(Bitmap) -> wxMask()
Types:
  Bitmap = wxBitmap:wxBitmap()
See external documentation.

new(Bitmap, PaletteIndex) -> wxMask()
Types:
  Bitmap = wxBitmap:wxBitmap()
  PaletteIndex = integer()
See external documentation.
Also:
new(Bitmap, Colour) -> wxMask() when
Bitmap::wxBitmap:wxBitmap(), Colour::wx:wx_colour().

create(This, Bitmap) -> boolean()
Types:
  This = wxMask()
  Bitmap = wxBitmap:wxBitmap()
See external documentation.

create(This, Bitmap, PaletteIndex) -> boolean()
Types:
  This = wxMask()
  Bitmap = wxBitmap:wxBitmap()
  PaletteIndex = integer()
See external documentation.
Also:
create(This, Bitmap, Colour) -> boolean() when This::wxMask(), Bitmap::wxBitmap:wxBitmap(), Colour::wx:wx_colour().

destroy(This::wxMask()) -> ok
Destroys this object, do not use object again
wxMaximizeEvent

Erlang module

See external documentation: wxMaximizeEvent.
Use wxEvtHandler:connect/3 with EventType:

maximize

See also the message variant #wxMaximize{} event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxMaximizeEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxMemoryDC
Erlang module

See external documentation: wxMemoryDC.
This class is derived (and can use functions) from:
wxDC

**DATA TYPES**

wxMemoryDC()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxMemoryDC()
See external documentation.

new(Dc) -> wxMemoryDC()
Types:

```
Dc = wxDC:wxDC() | wxBitmap:wxBitmap()
```
See external documentation.

selectObject(This, Bmp) -> ok
Types:

```
This = wxMemoryDC()
Bmp = wxBitmap:wxBitmap()
```
See external documentation.

selectObjectAsSource(This, Bmp) -> ok
Types:

```
This = wxMemoryDC()
Bmp = wxBitmap:wxBitmap()
```
See external documentation.

destroy(This::wxMemoryDC()) -> ok
Destroys this object, do not use object again
wxMenu

Erlang module

See external documentation: wxMenu.

This class is derived (and can use functions) from:
wxEvtHandler

**DATA TYPES**

wxMenu()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxMenu()

Equivalent to new([]).

new(Options::[Option]) -> wxMenu()

Types:

- Option = {style, integer()}

See external documentation.

new(Title, Options::[Option]) -> wxMenu()

Types:

- Title = unicode:chardata()
- Option = {style, integer()}

See external documentation.

append(This, Item) -> wxMenuItem:wxMenuItem()

Types:

- This = wxMenu()
- Item = wxMenuItem:wxMenuItem()

See external documentation.

append(This, Itemid, Text) -> wxMenuItem:wxMenuItem()

Types:

- This = wxMenu()
- Itemid = integer()
- Text = unicode:chardata()

Equivalent to append(This, Itemid, Text, []).
append(This, Itemid, Text, Submenu) -> wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Submenu = wxMenu()

See external documentation.

Also:
append(This, Itemid, Text, [Option]) -> wxMenuItem:wxMenuItem() when
This::wxMenu(), Itemid::integer(), Text::unicode:chardata(),
Option :: [{'help', unicode:chardata()}
| [{'kind', wx:wx_enum()}].


append(This, Itemid, Text, Help, IsCheckable) -> ok

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Help = unicode:chardata()
IsCheckable = boolean()

See external documentation.

Also:
append(This, Itemid, Text, Submenu, [Option]) -> wxMenuItem:wxMenuItem() when
This::wxMenu(), Itemid::integer(), Text::unicode:chardata(), Submenu::wxMenu(),
Option :: [{'help', unicode:chardata()}.

appendCheckItem(This, Itemid, Text) -> wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()

Equivalent to appendCheckItem(This, Itemid, Text, []).

appendCheckItem(This, Itemid, Text, Options::[Option]) ->
wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Option = {help, unicode:chardata()}

See external documentation.
appendRadioItem(This, Itemid, Text) -> wxMenuItem:wxMenuItem()
Types:
   This = wxMenu()
   Itemid = integer()
   Text = unicode:chardata()

Equivalent to appendRadioItem(This, Itemid, Text, []).

appendRadioItem(This, Itemid, Text, Options::[Option]) ->
wxMenuItem:wxMenuItem()
Types:
   This = wxMenu()
   Itemid = integer()
   Text = unicode:chardata()
   Option = {help, unicode:chardata()}

See external documentation.

appendSeparator(This) -> wxMenuItem:wxMenuItem()
Types:
   This = wxMenu()

See external documentation.

break(This) -> ok
Types:
   This = wxMenu()

See external documentation.

check(This, Itemid, Check) -> ok
Types:
   This = wxMenu()
   Itemid = integer()
   Check = boolean()

See external documentation.

delete(This, Itemid) -> boolean()
Types:
   This = wxMenu()
   Itemid = integer()

See external documentation.

Also:
delete(This, Item) -> boolean() when
This::wxMenu(), Item::wxMenuItem:wxMenuItem().

Destroy(This, Itemid) -> boolean()
Types:
This = wxMenu()
Itemid = integer()

See external documentation.
Also:
Destroy(This, Item) -> boolean() when
This::wxMenu(), Item::wxMenuItem:wxMenuItem().

enable(This, Itemid, Enable) -> ok
Types:
   This = wxMenu()
   Itemid = integer()
   Enable = boolean()

See external documentation.

findItem(This, Itemid) -> wxMenuItem:wxMenuItem()
Types:
   This = wxMenu()
   Itemid = integer()

See external documentation.
Also:
findItem(This, Item) -> integer() when
This::wxMenu(), Item::unicode:chardata().

findItemByPosition(This, Position) -> wxMenuItem:wxMenuItem()
Types:
   This = wxMenu()
   Position = integer()

See external documentation.

getHelpString(This, Itemid) -> unicode:charlist()
Types:
   This = wxMenu()
   Itemid = integer()

See external documentation.

getLabel(This, Itemid) -> unicode:charlist()
Types:
   This = wxMenu()
   Itemid = integer()

See external documentation.

getMenuItemCount(This) -> integer()
Types:
   This = wxMenu()
wxMenu

See external documentation.

call \( \text{getMenuItems}(This) \rightarrow [\text{wxMenuItem}:\text{wxMenuItem}()] \)

Types:
\( \text{This} = \text{wxMenu}() \)

See external documentation.

call \( \text{getTitle}(This) \rightarrow \text{unicode:charlist()} \)

Types:
\( \text{This} = \text{wxMenu}() \)

See external documentation.

call \( \text{insert}(This, Pos, Itemid) \rightarrow \text{wxMenuItem}:\text{wxMenuItem}() \)

Types:
\( \text{This} = \text{wxMenu}() \)
\( \text{Pos} = \text{integer}() \)
\( \text{Itemid} = \text{integer}() \)

See external documentation.

Also:
\( \text{insert}(This, Pos, Item) \rightarrow \text{wxMenuItem}:\text{wxMenuItem}() \) when
\( \text{This::wxMenu}(), \text{Pos::integer}(), \text{Item::wxMenuItem:wxMenuItem}() \).

\( \text{Kind} = \text{?wxITEM_SEPARATOR} | \text{?wxITEM_NORMAL} | \text{?wxITEM_CHECK} | \text{?wxITEM_RADIO} | ?\text{wxITEM_MAX} \)

\( \text{insert}(This, Pos, Itemid, Options::[Option]) \rightarrow \text{wxMenuItem}:\text{wxMenuItem}() \)

Types:
\( \text{This} = \text{wxMenu}() \)
\( \text{Pos} = \text{integer}() \)
\( \text{Itemid} = \text{integer}() \)
\( \text{Option} = \{\text{text}, \text{unicode:chardata}()\} | \{\text{help}, \text{unicode:chardata}()\} | \{\text{kind}, \text{wx:wx_enum}()\} \)

See external documentation.

\( \text{Kind} = \text{?wxITEM_SEPARATOR} | \text{?wxITEM_NORMAL} | \text{?wxITEM_CHECK} | \text{?wxITEM_RADIO} | ?\text{wxITEM_MAX} \)

\( \text{insert}(This, Pos, Itemid, Text, Submenu) \rightarrow \text{wxMenuItem}:\text{wxMenuItem}() \)

Types:
\( \text{This} = \text{wxMenu}() \)
\( \text{Pos} = \text{integer}() \)
\( \text{Itemid} = \text{integer}() \)
\( \text{Text} = \text{unicode:chardata}() \)
\( \text{Submenu} = \text{wxMenu}() \)

Equivalent to \( \text{insert}(This, Pos, Itemid, Text, Submenu, []) \).

382 | Ericsson AB. All Rights Reserved.: wxErlang
insert(This, Pos, Itemid, Text, Help, IsCheckable) -> ok
Types:
  This = wxMenu()
  Pos = integer()
  Itemid = integer()
  Text = unicode:chardata()
  Help = unicode:chardata()
  IsCheckable = boolean()
See external documentation.
Also:
insert(This, Pos, Itemid, Text, Submenu, [Option]) -> wxMenuItem:wxMenuItem() when
This::wxMenu(), Pos::integer(), Itemid::integer(), Text::unicode:chardata(), Submenu::wxMenu(),
Option :: ['help', unicode:chardata()].

insertCheckItem(This, Pos, Itemid, Text) -> wxMenuItem:wxMenuItem()
Types:
  This = wxMenu()
  Pos = integer()
  Itemid = integer()
  Text = unicode:chardata()
Equivalent to insertCheckItem(This, Pos, Itemid, Text, []).

insertCheckItem(This, Pos, Itemid, Text, Options::[Option]) ->
wxMenuItem:wxMenuItem()
Types:
  This = wxMenu()
  Pos = integer()
  Itemid = integer()
  Text = unicode:chardata()
  Option = {help, unicode:chardata()}
See external documentation.

insertRadioItem(This, Pos, Itemid, Text) -> wxMenuItem:wxMenuItem()
Types:
  This = wxMenu()
  Pos = integer()
  Itemid = integer()
  Text = unicode:chardata()
Equivalent to insertRadioItem(This, Pos, Itemid, Text, []).

insertRadioItem(This, Pos, Itemid, Text, Options::[Option]) ->
wxMenuItem:wxMenuItem()
Types:
  This = wxMenu()
wxMenu

Pos = integer()
Itemid = integer()
Text = unicode:chardata()
Option = {help, unicode:chardata()}

See external documentation.

insertSeparator(This, Pos) -> wxMenuItem:wxMenuItem()
Types:
    This = wxMenu()
    Pos = integer()

See external documentation.

isChecked(This, Itemid) -> boolean()
Types:
    This = wxMenu()
    Itemid = integer()

See external documentation.

isEnabled(This, Itemid) -> boolean()
Types:
    This = wxMenu()
    Itemid = integer()

See external documentation.

prepend(This, Itemid) -> wxMenuItem:wxMenuItem()
Types:
    This = wxMenu()
    Itemid = integer()

See external documentation.

Also:
prefpend(This, Item) -> wxMenuItem:wxMenuItem() when
This::wxMenu(), Item::wxMenuItem:wxMenuItem().

prepend(This, Itemid, Options::[Option]) -> wxMenuItem:wxMenuItem()
Types:
    This = wxMenu()
    Itemid = integer()
    Option = {text, unicode:chardata()} | {help, unicode:chardata()} | {kind, wx:wx_enum()}

See external documentation.

prepend(This, Itemid, Text, Submenu) -> wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Submenu = wxMenu()

Equivalent to prepend(This, Itemid, Text, Submenu, []).

prepend(This, Itemid, Text, Help, IsCheckable) -> ok

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Help = unicode:chardata()
IsCheckable = boolean()

See external documentation.

Also:
prepend(This, Itemid, Text, Submenu, [Option]) -> wxMenuItem:wxMenuItem() when
This::wxMenu(), Itemid::integer(), Text::unicode:chardata(), Submenu::wxMenu(),
Option :: {'help', unicode:chardata()}. 

prependCheckItem(This, Itemid, Text) -> wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()

Equivalent to prependCheckItem(This, Itemid, Text, []).

prependCheckItem(This, Itemid, Text, Options::[Option]) ->
wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()
Option = {help, unicode:chardata()}

See external documentation.

prependRadioItem(This, Itemid, Text) -> wxMenuItem:wxMenuItem()

Types:

This = wxMenu()
Itemid = integer()
Text = unicode:chardata()

Equivalent to prependRadioItem(This, Itemid, Text, []).
prependRadioItem(This, Itemid, Text, Options::[Option]) ->
    wxMenuItem:wxMenuItem()

Types:
    This = wxMenu()
    Itemid = integer()
    Text = unicode:chardata()
    Option = {help, unicode:chardata()}

See external documentation.

prependSeparator(This) -> wxMenuItem:wxMenuItem()

Types:
    This = wxMenu()

See external documentation.

remove(This, Itemid) -> wxMenuItem:wxMenuItem()

Types:
    This = wxMenu()
    Itemid = integer()

See external documentation.
Also:
remove(This, Item) -> wxMenuItem:wxMenuItem() when
This::wxMenu(). Item::wxMenuItem:wxMenuItem().

setHelpString(This, Itemid, HelpString) -> ok

Types:
    This = wxMenu()
    Itemid = integer()
    HelpString = unicode:chardata()

See external documentation.

setLabel(This, Itemid, Label) -> ok

Types:
    This = wxMenu()
    Itemid = integer()
    Label = unicode:chardata()

See external documentation.

setTitle(This, Title) -> ok

Types:
    This = wxMenu()
    Title = unicode:chardata()

See external documentation.
destroy(This::wxMenu()) -> ok

Destroys this object, do not use object again
wxMenuBar

Erlang module

See external documentation: wxMenuBar.

This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES

wxMenuBar()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxMenuBar()
See external documentation.

new(Style) -> wxMenuBar()
Types:
    Style = integer()
See external documentation.

append(This, Menu, Title) -> boolean()
Types:
    This = wxMenuBar()
    Menu = wxMenu:wxMenu()
    Title = unicode:chardata()
See external documentation.

check(This, Itemid, Check) -> ok
Types:
    This = wxMenuBar()
    Itemid = integer()
    Check = boolean()
See external documentation.

enable(This) -> boolean()
Types:
    This = wxMenuBar()
Equivalent to enable(This, []).
enable(This, Options::[Option]) -> boolean()
Types:
    This = wxMenuBar()
    Option = {enable, boolean()}
See external documentation.

enable(This, Itemid, Enable) -> ok
Types:
    This = wxMenuBar()
    Itemid = integer()
    Enable = boolean()
See external documentation.

enableTop(This, Pos, Flag) -> ok
Types:
    This = wxMenuBar()
    Pos = integer()
    Flag = boolean()
See external documentation.

findMenu(This, Title) -> integer()
Types:
    This = wxMenuBar()
    Title = unicode:chardata()
See external documentation.

findMenuItem(This, MenuString, ItemString) -> integer()
Types:
    This = wxMenuBar()
    MenuString = unicode:chardata()
    ItemString = unicode:chardata()
See external documentation.

findItem(This, Id) -> wxMenuItem:wxMenuItem()
Types:
    This = wxMenuBar()
    Id = integer()
See external documentation.

getHelpString(This, Itemid) -> unicode:charlist()
Types:
    This = wxMenuBar()
    Itemid = integer()
wxMenuBar

See external documentation.

getLabel(This) -> unicode:charlist()
Types:
   This = wxMenuBar()
See external documentation.

getLabel(This, Itemid) -> unicode:charlist()
Types:
   This = wxMenuBar()
   Itemid = integer()
See external documentation.

getLabelTop(This, Pos) -> unicode:charlist()
Types:
   This = wxMenuBar()
   Pos = integer()
See external documentation.

getMenu(This, Pos) -> wxMenu:wxMenu()
Types:
   This = wxMenuBar()
   Pos = integer()
See external documentation.

getMenuCount(This) -> integer()
Types:
   This = wxMenuBar()
See external documentation.

insert(This, Pos, Menu, Title) -> boolean()
Types:
   This = wxMenuBar()
   Pos = integer()
   Menu = wxMenu:wxMenu()
   Title = unicode:chardata()
See external documentation.

isChecked(This, Itemid) -> boolean()
Types:
   This = wxMenuBar()
   Itemid = integer()
See external documentation.
setAutoWindowMenu(Enable) -> ok
Types:
   Enable = boolean()
See external documentation.

getAutoWindowMenu() -> boolean()
See external documentation.

oSXGetAppleMenu(This) -> wxMenu:wxMenu()
Types:
   This = wxMenuBar()
See external documentation.

isEnabled(This) -> boolean()
Types:
   This = wxMenuBar()
See external documentation.

isEnabled(This, Itemid) -> boolean()
Types:
   This = wxMenuBar()
   Itemid = integer()
See external documentation.

remove(This, Pos) -> wxMenu:wxMenu()
Types:
   This = wxMenuBar()
   Pos = integer()
See external documentation.

replace(This, Pos, Menu, Title) -> wxMenu:wxMenu()
Types:
   This = wxMenuBar()
   Pos = integer()
   Menu = wxMenu:wxMenu()
   Title = unicode:chardata()
See external documentation.

setHelpString(This, Itemid, HelpString) -> ok
Types:
   This = wxMenuBar()
   Itemid = integer()
   HelpString = unicode:chardata()
wxMenuBar

See external documentation.

setLabel(This, S) -> ok
Types:
   This = wxMenuBar()
   S = unicode:chardata()
See external documentation.

setLabel(This, Itemid, Label) -> ok
Types:
   This = wxMenuBar()
   Itemid = integer()
   Label = unicode:chardata()
See external documentation.

setLabelTop(This, Pos, Label) -> ok
Types:
   This = wxMenuBar()
   Pos = integer()
   Label = unicode:chardata()
See external documentation.

destroy(This::wxMenuBar()) -> ok
Destroys this object, do not use object again
wxMenuEvent

Erlang module

See external documentation: wxMenuEvent.
Use \texttt{wxEvtHandler:connect/3} with EventType:
\begin{verbatim}
    menu_open, menu_close, menu_highlight
\end{verbatim}
See also the message variant \texttt{"wxMenu/} event record type.
This class is derived (and can use functions) from: wxEvent

\section*{DATA TYPES}

\begin{verbatim}
xwMenuEvent()
\end{verbatim}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

\section*{Exports}

\begin{verbatim}
getMenu(This) -> wxMenu:wxMenu()
\end{verbatim}

\textbf{Types:}
\begin{verbatim}
    This = wxMenuEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
getMenuId(This) -> integer()
\end{verbatim}

\textbf{Types:}
\begin{verbatim}
    This = wxMenuEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
isPopup(This) -> boolean()
\end{verbatim}

\textbf{Types:}
\begin{verbatim}
    This = wxMenuEvent()
\end{verbatim}

See external documentation.
wxMenuItem

Erlang module

See external documentation: `wxMenuItem`.

DATA TYPES

`wxMenuItem()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

`new() -> wxMenuItem()`
Equivalent to `new([])`.

`new(Options::[Option]) -> wxMenuItem()`
Types:
- `Option = {parentMenu, wxMenu:wxMenu()} | {id, integer()} | {text, unicode:chardata()} | {help, unicode:chardata()} | {kind, wx:wx_enum()} | {subMenu, wxMenu:wxMenu()}`
See external documentation.


`check(This) -> ok`  
Types:
- `This = wxMenuItem()`  
Equivalent to `check(This, [])`.

`check(This, Options::[Option]) -> ok`  
Types:
- `This = wxMenuItem()`  
- `Option = {check, boolean()}`  
See external documentation.

`enable(This) -> ok`  
Types:
- `This = wxMenuItem()`  
Equivalent to `enable(This, [])`.

`enable(This, Options::[Option]) -> ok`  
Types:
This = wxMenuItem()
Option = {enable, boolean()}
See external documentation.

getBitmap(This) -> wxBitmap:wxBitmap()
Types:
  This = wxMenuItem()
See external documentation.

getHelp(This) -> unicode:charlist()
Types:
  This = wxMenuItem()
See external documentation.

getId(This) -> integer()
Types:
  This = wxMenuItem()
See external documentation.

getKind(This) -> wx:wx_enum()
Types:
  This = wxMenuItem()
See external documentation.


getLabel(This) -> unicode:charlist()
Types:
  This = wxMenuItem()
See external documentation.

g.getLabelFromText(Text) -> unicode:charlist()
Types:
  Text = unicode:chardata()
See external documentation.

g.getMenu(This) -> wxMenu:wxMenu()
Types:
  This = wxMenuItem()
See external documentation.

g.getText(This) -> unicode:charlist()
Types:
  This = wxMenuItem()
wxMenuItem

See external documentation.

getSubMenu(This) -> wxMenu:wxMenu()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

isCheckable(This) -> boolean()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

isChecked(This) -> boolean()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

isEnabled(This) -> boolean()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

isSeparator(This) -> boolean()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

isSubMenu(This) -> boolean()
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
See external documentation.

setBitmap(This, Bitmap) -> ok
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
  Bitmap = \textcolor{blue}{wxBitmap:wxBitmap()}
See external documentation.

setHelp(This, Str) -> ok
Types:
  \textcolor{blue}{This} = \textcolor{blue}{wxMenuItem()}
  Str = \textcolor{blue}{unicode:chardata()}
See external documentation.
setMenu(This, Menu) -> ok
Types:
   This = wxMenuItem()
   Menu = wxMenu:wxMenu()
See external documentation.

setSubMenu(This, Menu) -> ok
Types:
   This = wxMenuItem()
   Menu = wxMenu:wxMenu()
See external documentation.

setText(This, Str) -> ok
Types:
   This = wxMenuItem()
   Str = unicode:chardata()
See external documentation.

destroy(This::wxMenuItem()) -> ok
Destroys this object, do not use object again
wxMessageDialog

Erlang module

See external documentation: wxMessageDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxMessageDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Parent, Message) -> wxMessageDialog()

Types:

    Parent = wxWindow:wxWindow()
    Message = unicode:chardata()

Equivalent to new(Parent, Message, []).

new(Parent, Message, Options::[Option]) -> wxMessageDialog()

Types:

    Parent = wxWindow:wxWindow()
    Message = unicode:chardata()
    Option = {caption, unicode:chardata()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

destroy(This::wxMessageDialog()) -> ok

Destroys this object, do not use object again
wxMiniFrame

Erlang module

See external documentation: **wxMiniFrame**.

This class is derived (and can use functions) from:

- `wxFrame`
- `wxTopLevelWindow`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

`wxMiniFrame()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxMiniFrame()`

See external documentation.

`new(Parent, Id, Title) -> wxMiniFrame()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Title = unicode:chardata()`

Equivalent to `new(Parent, Id, Title, [])`.

`new(Parent, Id, Title, Options::[Option]) -> wxMiniFrame()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Title = unicode:chardata()`
- `Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

`create(This, Parent, Id, Title) -> boolean()`

Types:

- `This = wxMiniFrame()`
- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Title = unicode:chardata()`

Ericsson AB. All Rights Reserved.: wxErlang | 399
wxMiniFrame

Equivalent to `create(This, Parent, Id, Title, []).`

`create(This, Parent, Id, Title, Options::[Option])` -> boolean()

Types:

- `This = wxMiniFrame()`
- `Parent = wxWindow:wxWindow()`
- `Id = integer()`
- `Title = unicode:chardata()`
- `Option = {pos, {X::integer(), Y::integer()}}` | `{size, {W::integer(), H::integer()}}` | `{style, integer()}`

See external documentation.

`destroy(This::wxMiniFrame())` -> ok

Destroys this object, do not use object again
wxMirrorDC

Erlang module

See external documentation: wxMirrorDC.
This class is derived (and can use functions) from:
wxDC

DATA TYPES
wxMirrorDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Dc, Mirror) -> wxMirrorDC()
Types:
   Dc = wxDC:wxDC()
   Mirror = boolean()
See external documentation.

destroy(This::wxMirrorDC()) -> ok
Destroys this object, do not use object again
wxMouseCaptureChangedEvent

Erlang module

See external documentation: `wxMouseCaptureChangedEvent`.

Use `wxEvtHandler:connect/3` with EventType:

    mouse_capture_changed

See also the message variant `#wxMouseCaptureChanged{}` event record type.

This class is derived (and can use functions) from:

`wxEvent`

DATA TYPES

`wxMouseCaptureChangedEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`getCapturedWindow(This) -> wXWindow:WXWindow()`

Types:

    This = wxMouseCaptureChangedEvent()

See external documentation.
wxMouseCaptureLostEvent

Erlang module

See external documentation: wxMouseCaptureLostEvent.
Use wxEvtHandler:connect/3 with EventType:

mouse_capture_lost

See also the message variant #wxMouseCaptureLost[] event record type.
This class is derived (and can use functions) from:
wxEvt

DATA TYPES
wxMouseCaptureLostEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxMouseEvent

Erlang module

See external documentation: `wxMouseEvent`.

Use `wxEvtHandler:connect/3` with EventType:

- `left_down`, `left_up`, `middle_down`, `middle_up`, `right_down`, `right_up`, `motion`, `enter_window`, `leave_window`, `left_dclick`, `middle_dclick`, `right_dclick`, `mousewheel`

See also the message variant `#wxMouse{}` event record type.

This class is derived (and can use functions) from:

`wxEvent`

**DATA TYPES**

`wxMouseEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`altDown(This) -> boolean()`

Types:

```
This = wxMouseEvent()
```

See external documentation.

`button(This, But) -> boolean()`

Types:

```
This = wxMouseEvent()
But = integer()
```

See external documentation.

`buttonDClick(This) -> boolean()`

Types:

```
This = wxMouseEvent()
```

Equivalent to `buttonDClick(This, [])`.

`buttonDClick(This, Options::[Option]) -> boolean()`

Types:

```
This = wxMouseEvent()
Option = {but, integer()}
```

See external documentation.
buttonDown(This) -> boolean()
Types:
  This = wxMouseEvent()
Equivalent to buttonDown(This, []).

buttonDown(This, Options::*[Option]) -> boolean()
Types:
  This = wxMouseEvent()
  Option = {but, integer()}
See external documentation.

buttonUp(This) -> boolean()
Types:
  This = wxMouseEvent()
Equivalent to buttonUp(This, []).

buttonUp(This, Options::*[Option]) -> boolean()
Types:
  This = wxMouseEvent()
  Option = {but, integer()}
See external documentation.

cmdDown(This) -> boolean()
Types:
  This = wxMouseEvent()
See external documentation.

controlDown(This) -> boolean()
Types:
  This = wxMouseEvent()
See external documentation.

dragging(This) -> boolean()
Types:
  This = wxMouseEvent()
See external documentation.

entering(This) -> boolean()
Types:
  This = wxMouseEvent()
See external documentation.
**wxMouseEvent**

getButton(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

getPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxMouseEvent()
See external documentation.

getLogicalPosition(This, Dc) -> {X::integer(), Y::integer()}
Types:
   This = wxMouseEvent()
   Dc = wxDC:wxDC()
See external documentation.

getLinesPerAction(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

getWheelRotation(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

getWheelDelta(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

getX(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

getY(This) -> integer()
Types:
   This = wxMouseEvent()
See external documentation.

isButton(This) -> boolean()
Types:
This = wxMouseEvent()
See external documentation.

isPageScroll(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

leaving(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

leftDClick(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

leftDown(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

leftIsDown(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

leftUp(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

metaDown(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.

middleDClick(This) -> boolean()
Types:
    This = wxMouseEvent()
See external documentation.
wxMouseEvent

middleDown(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

middleIsDown(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

middleUp(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

moving(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

rightDClick(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

rightDown(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

rightIsDown(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

rightUp(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

shiftDown(This) -> boolean()
Types:
   This = wxMouseEvent()
See external documentation.

getWheelAxis(This) -> wx:wx_enum()

Types:
   This = wxMouseEvent()

See external documentation.
Res = ?wxMOUSE_WHEEL_VERTICAL | ?wxMOUSE_WHEEL_HORIZONTAL
**wxMoveEvent**

Erlang module

See external documentation: [wxMoveEvent](#).

Use `wxEvtHandler:connect/3` with EventType:

```
    move
```

See also the message variant `#wxMove{}` event record type.

This class is derived (and can use functions) from:

wxEvent

### DATA TYPES

wxMoveEvent()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

### Exports

```
getPosition(This) -> {X::integer(), Y::integer()}
```

Types:

```
    This = wxMoveEvent()
```

See external documentation.
wxMultiChoiceDialog

Erlang module

See external documentation: wxMultiChoiceDialog.
This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxMultiChoiceDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxMultiChoiceDialog()
See external documentation.

new(Parent, Message, Caption, Choices) -> wxMultiChoiceDialog()
Types:
  Parent = wxWindow:wxWindow()
  Message = unicode:chardata()
  Caption = unicode:chardata()
  Choices = [unicode:chardata()]
Equivalent to new(Parent, Message, Caption, Choices, []).

new(Parent, Message, Caption, Choices, Options::[Option]) ->
wxMultiChoiceDialog()
Types:
  Parent = wxWindow:wxWindow()
  Message = unicode:chardata()
  Caption = unicode:chardata()
  Choices = [unicode:chardata()]
  Option = {style, integer()} | {pos, {X::integer(), Y::integer()}}
See external documentation.

getSelections(This) -> [integer()]
Types:
  This = wxMultiChoiceDialog()
See external documentation.
wxMultiChoiceDialog

setSelections(This, Selections) -> ok

Types:
    This = wxMultiChoiceDialog()
    Selections = [integer()]

See external documentation.

destroy(This::wxMultiChoiceDialog()) -> ok

Destroys this object, do not use object again
wxNavigationKeyEvent

Erlang module

See external documentation: wxNavigationKeyEvent.

Use wxEvtHandler:connect/3 with EventType:

  navigation_key

See also the message variant #wxNavigationKey{} event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxNavigationKeyEvent()

  An object reference. The representation is internal and can be changed without notice. It can't be used for
  comparison stored on disc or distributed for use on other nodes.

Exports

getDirection(This) -> boolean()
Types:
  This = wxNavigationKeyEvent()
See external documentation.

setDirection(This, BForward) -> ok
Types:
  This = wxNavigationKeyEvent()
  BForward = boolean()
See external documentation.

isWindowChange(This) -> boolean()
Types:
  This = wxNavigationKeyEvent()
See external documentation.

setWindowChange(This, BIs) -> ok
Types:
  This = wxNavigationKeyEvent()
  BIs = boolean()
See external documentation.

isFromTab(This) -> boolean()
Types:
This = wxNavigationKeyEvent()
See external documentation.

setFromTab(This, BIs) -> ok
Types:
   This = wxNavigationKeyEvent()
   BIs = boolean()
See external documentation.

getCurrentFocus(This) -> wxWindow:wxWindow()
Types:
   This = wxNavigationKeyEvent()
See external documentation.

setCurrentFocus(This, Win) -> ok
Types:
   This = wxNavigationKeyEvent()
   Win = wxWindow:wxWindow()
See external documentation.
wxNotebook

Erlang module

See external documentation: wxNotebook.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxNotebook()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxNotebook()

See external documentation.

ew(Parent, Winid) -> wxNotebook()

Types:
  Parent = wxWindow:wxWindow()
  Winid = integer()

Equivalent to new(Parent, Winid, []).

new(Parent, Winid, Options::[Option]) -> wxNotebook()

Types:
  Parent = wxWindow:wxWindow()
  Winid = integer()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

addPage(This, Page, Text) -> boolean()

Types:
  This  = wxNotebook()
  Page  = wxWindow:wxWindow()
  Text  = unicode:chardata()

Equivalent to addPage(This, Page, Text, []).

addPage(This, Page, Text, Options::[Option]) -> boolean()

Types:
This = wxNotebook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean()} | {imageId, integer()}

See external documentation.

advanceSelection(This) -> ok
Types:
  This = wxNotebook()
Equivalent to advanceSelection(This, []).

advanceSelection(This, Options::[Option]) -> ok
Types:
  This = wxNotebook()
  Option = {forward, boolean()}
See external documentation.

assignImageList(This, ImageList) -> ok
Types:
  This = wxNotebook()
  ImageList = wxImageList:wxImageList()
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
  This = wxNotebook()
  Parent = wxWindow:wxWindow()
  Id = integer()
Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
  This = wxNotebook()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

deleteAllPages(This) -> boolean()
Types:
  This = wxNotebook()
See external documentation.
deletePage(This, NPage) -> boolean()
Types:
   This = wxNotebook()
   NPage = integer()
See external documentation.

removePage(This, NPage) -> boolean()
Types:
   This = wxNotebook()
   NPage = integer()
See external documentation.

getCurrentPage(This) -> wxWindow:wxWindow()
Types:
   This = wxNotebook()
See external documentation.

getImageList(This) -> wxImageList:wxImageList()
Types:
   This = wxNotebook()
See external documentation.

getPage(This, N) -> wxWindow:wxWindow()
Types:
   This = wxNotebook()
   N = integer()
See external documentation.

getPageCount(This) -> integer()
Types:
   This = wxNotebook()
See external documentation.

g getPageImage(This, NPage) -> integer()
Types:
   This = wxNotebook()
   NPage = integer()
See external documentation.

g getPageText(This, NPage) -> unicode:charlist()
Types:
   This = wxNotebook()
   NPage = integer()
wxNotebook

See external documentation.

getRowCount(This) -> integer()
Types:
   This = wxNotebook()

See external documentation.

getSelection(This) -> integer()
Types:
   This = wxNotebook()

See external documentation.

getThemeBackgroundColour(This) -> wx:wx_colour4()
Types:
   This = wxNotebook()

See external documentation.

hitTest(This, Pt) -> Result
Types:
   Result = {Res::integer(), Flags::integer()}
   This = wxNotebook()
   Pt = {X::integer(), Y::integer()}

See external documentation.

insertPage(This, Position, Win, StrText) -> boolean()
Types:
   This = wxNotebook()
   Position = integer()
   Win = wxWindow:wxWindow()
   StrText = unicode:chardata()

Equivalent to insertPage(This, Position, Win, StrText, []).

insertPage(This, Position, Win, StrText, Options::[Option]) -> boolean()
Types:
   This = wxNotebook()
   Position = integer()
   Win = wxWindow:wxWindow()
   StrText = unicode:chardata()
   Option = {bSelect, boolean()} | {imageId, integer()}

See external documentation.

setImageList(This, ImageList) -> ok
Types:
This = wxNotebook()
ImageList = wxImageList:wxImageList()

See external documentation.

setPadding(This, Padding) -> ok
Types:
   This = wxNotebook()
   Padding = {W::integer(), H::integer()}

See external documentation.

setPageSize(This, Size) -> ok
Types:
   This = wxNotebook()
   Size = {W::integer(), H::integer()}

See external documentation.

setPageImage(This, NPage, NImage) -> boolean()
Types:
   This = wxNotebook()
   NPage = integer()
   NImage = integer()

See external documentation.

setPageText(This, NPage, StrText) -> boolean()
Types:
   This = wxNotebook()
   NPage = integer()
   StrText = unicode:chardata()

See external documentation.

setSelection(This, NPage) -> integer()
Types:
   This = wxNotebook()
   NPage = integer()

See external documentation.

changeSelection(This, NPage) -> integer()
Types:
   This = wxNotebook()
   NPage = integer()

See external documentation.
wxNotebook

destroy(This::wxNotebook()) -> ok

Destroys this object, do not use object again
wxNotebookEvent

Erlang module

See external documentation: wxNotebookEvent.

Use wxEvtHandler:connect/3 with EventType:
   command_notebook_page_changed, command_notebook_page_changing

See also the message variant #wxNotebookEvent event record type.

This class is derived (and can use functions) from:
   wxNotifyEvent
   wxCommandEvent
   wxEvent

DATA TYPES

wxNotebookEvent()

   An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getOldSelection(This) -> integer()

Types:
   This = wxNotebookEvent()

See external documentation.

getSelection(This) -> integer()

Types:
   This = wxNotebookEvent()

See external documentation.

setOldSelection(This, NOldSel) -> ok

Types:
   This = wxNotebookEvent()
   NOldSel = integer()

See external documentation.

setSelection(This, NSel) -> ok

Types:
   This = wxNotebookEvent()
   NSel = integer()

See external documentation.
wxNotifyEvent

Erlang module

See external documentation: wxNotifyEvent.

This class is derived (and can use functions) from:
wxCommandEvent
wxEvent

DATA TYPES

wxNotifyEvent()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

allow(This) -> ok
Types:
  This = wxNotifyEvent()
See external documentation.

isAllowed(This) -> boolean()
Types:
  This = wxNotifyEvent()
See external documentation.

veto(This) -> ok
Types:
  This = wxNotifyEvent()
See external documentation.
wxOverlay
Erlang module

See external documentation: wxOverlay.

DATA TYPES
wxOverlay()
An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

call new() -> wxOverlay()
See external documentation.

call reset(This) -> ok
Types:
    This = wxOverlay()
See external documentation.

call destroy(This::wxOverlay()) -> ok
Destroys this object, do not use object again
wxPageSetupDialog

Erlang module

See external documentation: wxPageSetupDialog.

DATA TYPES

wxPageSetupDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Parent) -> wxPageSetupDialog()
Types:

   Parent = wxWindow:wxWindow()

Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxPageSetupDialog()
Types:

   Parent = wxWindow:wxWindow()
   Option = [data, wxPageSetupDialogData:wxPageSetupDialogData()]

See external documentation.

getPageSetupData(This) -> wxPageSetupDialogData:wxPageSetupDialogData()
Types:

   This = wxPageSetupDialog()

See external documentation.

showModal(This) -> integer()
Types:

   This = wxPageSetupDialog()

See external documentation.

destroy(This::wxPageSetupDialog()) -> ok
Destroys this object, do not use object again
wxPageSetupDialogData

Erlang module

See external documentation: wxPageSetupDialogData.

DATA TYPES

wxPageSetupDialogData()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPageSetupDialogData()
See external documentation.

new(PrintData) -> wxPageSetupDialogData()
Types:
  \texttt{PrintData} = \texttt{wxPrintData:wxPrintData()} | \texttt{wxPageSetupDialogData()}
See external documentation.

enableHelp(This, Flag) -> ok
Types:
  \texttt{This} = \texttt{wxPageSetupDialogData()}
  \texttt{Flag} = boolean()
See external documentation.

enableMargins(This, Flag) -> ok
Types:
  \texttt{This} = \texttt{wxPageSetupDialogData()}
  \texttt{Flag} = boolean()
See external documentation.

enableOrientation(This, Flag) -> ok
Types:
  \texttt{This} = \texttt{wxPageSetupDialogData()}
  \texttt{Flag} = boolean()
See external documentation.

enablePaper(This, Flag) -> ok
Types:
  \texttt{This} = \texttt{wxPageSetupDialogData()}
  \texttt{Flag} = boolean()
See external documentation.

enablePrinter(This, Flag) -> ok
Types:
  This = wxPageSetupDialogData()
  Flag = boolean()
See external documentation.

getDefaultMinMargins(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getEnableMargins(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getEnableOrientation(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getEnablePaper(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getEnablePrinter(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getEnableHelp(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.

getDefaultInfo(This) -> boolean()
Types:
  This = wxPageSetupDialogData()
See external documentation.
getMarginTopLeft(This) -> {X::integer(), Y::integer()}
Types:
  This = wxPageSetupDialogData()
See external documentation.

getMarginBottomRight(This) -> {X::integer(), Y::integer()}
Types:
  This = wxPageSetupDialogData()
See external documentation.

getMinMarginTopLeft(This) -> {X::integer(), Y::integer()}
Types:
  This = wxPageSetupDialogData()
See external documentation.

getMinMarginBottomRight(This) -> {X::integer(), Y::integer()}
Types:
  This = wxPageSetupDialogData()
See external documentation.

g PaperId(This) -> wx:wx_enum()
Types:
  This = wxPageSetupDialogData()
See external documentation.
  ?wxPAPER_ENV_DL |
  ?wxPAPER_LEGAL_EXTRA | ?wxPAPER_TABLOID_EXTRA | ?wxPAPER_A4_EXTRA |
  ?wxPAPER_LETTER_TRANSVERSE | ?wxPAPER_A4_TRANSVERSE |
  ?wxPAPER_LETTER_EXTRA_TRANSVERSE | ?wxPAPER_A5_TRANSVERSE |
  ?wxPAPER_A4_TRANSVERSE | ?wxPAPER_A6 | ?wxPAPER_JENV_KAKU2 |
  ?wxPAPER_B4 | ?wxPAPER_B5 |
  ?wxPAPER_B4_JIS_ROTATED | ?wxPAPER_B5_JIS_ROTATED |
  ?wxPAPER_JAPANESE_POSTCARD_ROTATED | ?wxPAPER_DBL_JAPANESE_POSTCARD_ROTATED | ?wxPAPER_A4_ROTATED |
  ?wxPAPER_A5_ROTATED | ?wxPAPER_B4_JIS_ROTATED |
  ?wxPAPER_A5_JIS_ROTATED | ?wxPAPER_B5_JIS_ROTATED |
wxPageSetupDialogData

g PaperSize(This) -> {W::integer(), H::integer()}
Types:
   This = wxPageSetupDialogData()
See external documentation.

g PrintData(This) -> wxPrintData:wxPrintData()
Types:
   This = wxPageSetupDialogData()
See external documentation.

isOk(This) -> boolean()
Types:
   This = wxPageSetupDialogData()
See external documentation.

setDefaultInfo(This, Flag) -> ok
Types:
   This = wxPageSetupDialogData()
   Flag = boolean()
See external documentation.

setDefaultMinMargins(This, Flag) -> ok
Types:
   This = wxPageSetupDialogData()
   Flag = boolean()
See external documentation.

setMarginTopLeft(This, Pt) -> ok
Types:
   This = wxPageSetupDialogData()
   Pt = {X::integer(), Y::integer()}
See external documentation.
setMarginBottomRight(This, Pt) -> ok
Types:
  This = wxPageSetupDialogData()
  Pt = (X::integer(), Y::integer())
See external documentation.

setMinMarginTopLeft(This, Pt) -> ok
Types:
  This = wxPageSetupDialogData()
  Pt = (X::integer(), Y::integer())
See external documentation.

setMinMarginBottomRight(This, Pt) -> ok
Types:
  This = wxPageSetupDialogData()
  Pt = (X::integer(), Y::integer())
See external documentation.

setPaperId(This, Id) -> ok
Types:
  This = wxPageSetupDialogData()
  Id = wx:wx_enum()
See external documentation.

| ?wxPAPER_A4_TRANSVERSE | ?wxPAPER_LETTER_EXTRA_TRANSVERSE | ?wxPAPER_A4_TRANSVERSE |
| ?wxPAPER_B4 | ?wxPAPER_B5 | ?wxPAPER_A4_TRANSVERSE |
| ?wxPAPER_B5 | ?wxPAPER_A5 | ?wxPAPER_A5_TRANSVERSE |
| ?wxPAPER_B5_TRANSVERSE | ?wxPAPER_B6 | ?wxPAPER_B6_TRANSVERSE |
| ?wxPAPER_A2 | ?wxPAPER_A3_TRANSVERSE | ?wxPAPER_A3_EXTRA | ?wxPAPER_A3_EXTRA_TRANSVERSE |
| ?wxPAPER_A6 | ?wxPAPER_JENV_KAKU2 | ?wxPAPER_JENV_KAKU3 |
| ?wxPAPER_JENV_CHOU3 | ?wxPAPER_JENV_CHOU4 | ?wxPAPER_JENV_KAKU2 |
| ?wxPAPER_JENV_KAKU3 | ?wxPAPER_JENV_CHOU3 | ?wxPAPER_JENV_CHOU4 |
| ?wxPAPER_LETTER_ROTATED | ?wxPAPER_A3_ROTATED | ?wxPAPER_A4_ROTATED |
| ?wxPAPER_B4_JIS_ROTATED | ?wxPAPER_B5_JIS_ROTATED | ?wxPAPER_JENV_KAKU3_ROTATED |
| ?wxPAPER_JENV_KAKU3_ROTATED | ?wxPAPER_JENV_CHOU3_ROTATED | ?wxPAPER_JENV_CHOU4_ROTATED | ?wxPAPER_B6_JIS
setPaperSize(This, Id) -> ok

Types:

This = wxPageSetupDialogData()
Id = wx:wx_enum()

See external documentation.
Also:

setPageSize(This, Sz) -> 'ok' when
This::wxPageSetupDialogData(), Sz::{W::integer(), H::integer()}.

setPrintData(This, PrintData) -> ok
Types:
  This = wxPageSetupDialogData()
  PrintData = wxPrintData:wxPrintData()
See external documentation.

destroy(This::wxPageSetupDialogData()) -> ok
Destroys this object, do not use object again
wxPaintDC

Erlang module

See external documentation: wxPaintDC.
This class is derived (and can use functions) from:
wxWindowDC
wxDC

DATA TYPES

wxPaintDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPaintDC()

This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

ew(Win) -> wxPaintDC()
Types:
    Win = wxWindow:wxWindow()
See external documentation.

destroy(This::wxPaintDC()) -> ok
Destroys this object, do not use object again
wxPaintEvent

Erlang module

See external documentation: wxPaintEvent.
Use wxEvtHandler:connect/3 with EventType:

paint

See also the message variant #wxPaint/ event record type.
This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxPaintEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxPalette

Erlang module

See external documentation: wxPalette.

DATA TYPES

wxPalette()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPalette()

See external documentation.

new(Red, Green, Blue) -> wxPalette()

Types:

Red = binary()
Green = binary()
Blue = binary()

See external documentation.

create(This, Red, Green, Blue) -> boolean()

Types:

This = wxPalette()
Red = binary()
Green = binary()
Blue = binary()

See external documentation.

getColoursCount(This) -> integer()

Types:

This = wxPalette()

See external documentation.

getPixel(This, Red, Green, Blue) -> integer()

Types:

This = wxPalette()
Red = integer()
Green = integer()
Blue = integer()

See external documentation.
getRGB(This, Pixel) -> Result
Types:
   Result = {Res::boolean(), Red::integer(), Green::integer(),
             Blue::integer()}
   This = wxPalette()
   Pixel = integer()
See external documentation.

isOk(This) -> boolean()
Types:
   This = wxPalette()
See external documentation.

destroy(This::wxPalette()) -> ok
Destroys this object, do not use object again
wxPaletteChangedEvent

Erlang module

See external documentation: wxPaletteChangedEvent.

Use wxEvtHandler:connect/3 with EventType:

    palette_changed

See also the message variant #wxPaletteChanged/ event record type.

This class is derived (and can use functions) from:

wxEvent

DATA TYPES

wxPaletteChangedEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

setChangedWindow(This, Win) -> ok

Types:

    This = wxPaletteChangedEvent()
    Win = wxWindow:wxWindow()

See external documentation.

getChangedWindow(This) -> wxWindow:wxWindow()

Types:

    This = wxPaletteChangedEvent()

See external documentation.
wxPanel

Erlang module

See external documentation: wxPanel.

This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES

wxPanel()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPanel()
See external documentation.

new(Parent) -> wxPanel()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxPanel()
Types:
    Parent = wxWindow:wxWindow()
    Option = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

new(Parent, X, Y, Width, Height) -> wxPanel()
Types:
    Parent = wxWindow:wxWindow()
    X = integer()
    Y = integer()
    Width = integer()
    Height = integer()
Equivalent to new(Parent, X, Y, Width, Height, []).

new(Parent, X, Y, Width, Height, Options::[Option]) -> wxPanel()
Types:
    Parent = wxWindow:wxWindow()
wxPanel

\[\begin{align*}
X & = \text{integer()} \\
Y & = \text{integer()} \\
\text{Width} & = \text{integer()} \\
\text{Height} & = \text{integer()} \\
\text{Option} & = \{\text{style, integer()}\}
\end{align*}\]

See external documentation.

\text{initDialog(This)} \rightarrow \text{ok}

Types:
\[\text{This} = \text{wxPanel()}\]

See external documentation.

\text{setFocusIgnoringChildren(This)} \rightarrow \text{ok}

Types:
\[\text{This} = \text{wxPanel()}\]

See external documentation.

\text{destroy(This::wxPanel())} \rightarrow \text{ok}

Destroys this object, do not use object again
wxPasswordEntryDialog

Erlang module

See external documentation: **wxPasswordEntryDialog**.

This class is derived (and can use functions) from:

*wxTextEntryDialog*
*wxDialog*
*wxTopLevelWindow*
*wxWindow*
*wxEvtHandler*

**DATA TYPES**

*wxPasswordEntryDialog()*

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

*new*(Parent, Message) -> wxPasswordEntryDialog()

Types:

Parent = *wxWindow:*wxWindow()
Message = unicode:chardata()

Equivalent to *new*(Parent, Message, []).

*new*(Parent, Message, Options::[Option]) -> wxPasswordEntryDialog()

Types:

Parent = *wxWindow:*wxWindow()
Message = unicode:chardata()
Option = {caption, unicode:chardata()} | {value, unicode:chardata()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

*destroy*(This::wxPasswordEntryDialog()) -> ok

Destroys this object, do not use object again.
wxPen

Erlang module

See external documentation: wxPen.

DATA TYPES

wxPen()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPen()
See external documentation.

ew(Colour) -> wxPen()
Types:
  Colour = wx:wx_colour()
Equivalent to new(Colour, []).

ew(Colour, Options::[Option]) -> wxPen()
Types:
  Colour = wx:wx_colour()
  Option = {width, integer()} | {style, integer()};
See external documentation.

getcap(This) -> integer()
Types:
  This = wxPen()
See external documentation.

getcollection(This) -> wx:wx_colour4()
Types:
  This = wxPen()
See external documentation.

getcap İl(This) -> integer()
Types:
  This = wxPen()
See external documentation.
getStyle(This) -> integer()
Types:
    This = wxPen()
See external documentation.

getWidth(This) -> integer()
Types:
    This = wxPen()
See external documentation.

isOk(This) -> boolean()
Types:
    This = wxPen()
See external documentation.

setCap(This, CapStyle) -> ok
Types:
    This = wxPen()
    CapStyle = wx:wx_enum()
See external documentation.

setColour(This, Colour) -> ok
Types:
    This = wxPen()
    Colour = wx:wx_colour()
See external documentation.

setColour(This, Red, Green, Blue) -> ok
Types:
    This = wxPen()
    Red = integer()
    Green = integer()
    Blue = integer()
See external documentation.

setJoin(This, JoinStyle) -> ok
Types:
    This = wxPen()
    JoinStyle = wx:wx_enum()
See external documentation.

JoinStyle = integer
wxPen

```erlang
setStyle(This, Style) -> ok
Types:
  This = wxPen()
  Style = integer()
See external documentation.
```

```erlang
setWidth(This, Width) -> ok
Types:
  This = wxPen()
  Width = integer()
See external documentation.
```

```erlang
destroy(This::wxPen()) -> ok
Destroys this object, do not use object again
```
wxPickerBase

Erlang module

See external documentation: wxPickerBase.
This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES
wxPickerBase()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

setInternalMargin(This, Newmargin) -&gt; ok
Types:
  This = wxPickerBase()
  Newmargin = integer()
See external documentation.

getInternalMargin(This) -&gt; integer()
Types:
  This = wxPickerBase()
See external documentation.

setTextCtrlProportion(This, Prop) -&gt; ok
Types:
  This = wxPickerBase()
  Prop = integer()
See external documentation.

setPickerCtrlProportion(This, Prop) -&gt; ok
Types:
  This = wxPickerBase()
  Prop = integer()
See external documentation.

getTextCtrlProportion(This) -&gt; integer()
Types:
  This = wxPickerBase()
wxPickerBase

See external documentation.

getPickerCtrlProportion(This) -> integer()
Types:
    This = wxPickerBase()
See external documentation.

hasTextCtrl(This) -> boolean()
Types:
    This = wxPickerBase()
See external documentation.

getTextCtrl(This) -> wxTextCtrl:wxTextCtrl()
Types:
    This = wxPickerBase()
See external documentation.

isTextCtrlGrowable(This) -> boolean()
Types:
    This = wxPickerBase()
See external documentation.

setPickerCtrlGrowable(This) -> ok
Types:
    This = wxPickerBase()
Equivalent to setPickerCtrlGrowable(This, []).

setPickerCtrlGrowable(This, Options::[Option]) -> ok
Types:
    This = wxPickerBase()
    Option = {grow, boolean()}
See external documentation.

setTextCtrlGrowable(This) -> ok
Types:
    This = wxPickerBase()
Equivalent to setTextCtrlGrowable(This, []).

setTextCtrlGrowable(This, Options::[Option]) -> ok
Types:
    This = wxPickerBase()
    Option = {grow, boolean()}
See external documentation.
isPickerCtrlGrowable(This) -> boolean()

Types:

    This = wxPickerBase()

See external documentation.
wxPopupTransientWindow

Erlang module

See external documentation: `wxPopupTransientWindow`.

This class is derived (and can use functions) from:
- `wxPopupWindow`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

`wxPopupTransientWindow()`

- An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxPopupTransientWindow()`

See external documentation.

`new(Parent) -> wxPopupTransientWindow()`

Types:

- `Parent` = `wxWindow:wxWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent, Options::{Option}) -> wxPopupTransientWindow()`

Types:

- `Parent` = `wxWindow:wxWindow()`
- `Option` = `{style, integer()}`

See external documentation.

`popup(This) -> ok`

Types:

- `This` = `wxPopupTransientWindow()`

Equivalent to `popup(This, [])`.

`popup(This, Options::{Option}) -> ok`

Types:

- `This` = `wxPopupTransientWindow()`
- `Option` = `{focus, wxWindow:wxWindow()}`

See external documentation.
dismiss(This) -> ok

Types:

This = wxPopupTransientWindow()

See external documentation.

destroy(This::wxPopupTransientWindow()) -> ok

Destroys this object, do not use object again
wxPopupWindow

Erlang module

See external documentation: `wxPopupWindow`.

This class is derived (and can use functions) from:
`wxWindow`
`wxEvtHandler`

DATA TYPES

`wxPopupWindow()`

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

`new() -> wxPopupWindow()`
See external documentation.

`new(Parent) -> wxPopupWindow()`
Types:
`Parent = wxWindow:wxWindow()`
Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxPopupWindow()`
Types:
`Parent = wxWindow:wxWindow()`
`Option = {flags, integer()}`
See external documentation.

`create(This, Parent) -> boolean()`
Types:
`This = wxPopupWindow()`
`Parent = wxWindow:wxWindow()`
Equivalent to `create(This, Parent, [])`.

`create(This, Parent, Options::[Option]) -> boolean()`
Types:
`This = wxPopupWindow()`
`Parent = wxWindow:wxWindow()`
`Option = {flags, integer()}`
See external documentation.
position(This, PtOrigin, Size) -> ok
Types:
  This = wxPopupWindow()
  PtOrigin = {X::integer(), Y::integer()}
  Size = {W::integer(), H::integer()}

See external documentation.

destroy(This::wxPopupWindow()) -> ok
Destroys this object, do not use object again
wxPostScriptDC

Erlang module

See external documentation: *wxPostScriptDC*.

This class is derived (and can use functions) from: *wxDC*

**DATA TYPES**

*wxPostScriptDC*( )

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new( ) -> *wxPostScriptDC*( )

See external documentation.

new(PrintData) -> *wxPostScriptDC*( )

Types:

PrintData = *wxPrintData*: *wxPrintData*( )

See external documentation.

setResolution(Ppi) -> ok

Types:

Ppi = integer()

This function is deprecated: not available in wxWidgets-2.9 and later

See external documentation.

getResolution() -> integer()

This function is deprecated: not available in wxWidgets-2.9 and later

See external documentation.

destroy(This::*wxPostScriptDC*( )) -> ok

Destroys this object, do not use object again
wxPreviewCanvas

Erlang module

See external documentation: wxPreviewCanvas.

This class is derived (and can use functions) from:
wxScrolledWindow
wxPanel
wxWindow
wxEvtHandler

DATA TYPES

wxPreviewCanvas()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.
wxPreviewControlBar

Erlang module

See external documentation: \texttt{wxPreviewControlBar}.

This class is derived (and can use functions) from:
\texttt{wxPanel}
\texttt{wxWindow}
\texttt{wxEvtHandler}

**DATA TYPES**

\texttt{wxPreviewControlBar()}

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

\texttt{new(Preview, Buttons, Parent) -> wxPreviewControlBar()}

Types:
\begin{Verbatim}
  \begin{verbatim}
  Preview = \texttt{wxPrintPreview:wxPrintPreview()}
  Buttons = \texttt{integer()}
  Parent = \texttt{wxWindow:wxWindow()}
  \end{verbatim}
\end{Verbatim}

Equivalent to \texttt{new(Preview, Buttons, Parent, []).}

\texttt{new(Preview, Buttons, Parent, Options::[Option]) -> wxPreviewControlBar()}

Types:
\begin{Verbatim}
  \begin{verbatim}
  Preview = \texttt{wxPrintPreview:wxPrintPreview()}
  Buttons = \texttt{integer()}
  Parent = \texttt{wxWindow:wxWindow()}
  Option = \{\texttt{pos, \{X::integer(), Y::integer()\}} | \{\texttt{size, \{W::integer(), H::integer()\}} | \{\texttt{style, integer()}}\}
  \end{verbatim}
\end{Verbatim}

See external documentation.

\texttt{createButtons(This) -> ok}

Types:
\begin{Verbatim}
  \begin{verbatim}
  This = \texttt{wxPreviewControlBar()}
  \end{verbatim}
\end{Verbatim}

See external documentation.

\texttt{getPrintPreview(This) -> wxPrintPreview:wxPrintPreview()}

Types:
\begin{Verbatim}
  \begin{verbatim}
  This = \texttt{wxPreviewControlBar()}
  \end{verbatim}
\end{Verbatim}

See external documentation.
getZoomControl(This) -> integer()
Types:
   This = wxPreviewControlBar()
See external documentation.

setZoomControl(This, Zoom) -> ok
Types:
   This = wxPreviewControlBar()
   Zoom = integer()
See external documentation.

destroy(This::wxPreviewControlBar()) -> ok
Destroys this object, do not use object again
wxPreviewFrame

Erlang module

See external documentation: wxPreviewFrame.

This class is derived (and can use functions) from:

wxFrame
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxPreviewFrame()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Preview, Parent) -> wxPreviewFrame()

Types:
  Preview = wxPrintPreview:wxPrintPreview()
  Parent = wxWindow:wxWindow()

Equivalent to new(Preview, Parent, []).

new(Preview, Parent, Options::[Option]) -> wxPreviewFrame()

Types:
  Preview = wxPrintPreview:wxPrintPreview()
  Parent = wxWindow:wxWindow()
  Option = {title, unicode:chardata()} | {pos, {X::integer(), Y::integer()}}
           | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

createControlBar(This) -> ok

Types:
  This = wxPreviewFrame()

See external documentation.

createCanvas(This) -> ok

Types:
  This = wxPreviewFrame()

See external documentation.
initialize(This) -> ok
Types:

  This = wxPreviewFrame()

See external documentation.

onCloseWindow(This, Event) -> ok
Types:

  This = wxPreviewFrame()
  Event = wxCloseEvent:wxCloseEvent()

See external documentation.

destroy(This::wxPreviewFrame()) -> ok
Destroys this object, do not use object again
wxPrintData

Erlang module

See external documentation: wxPrintData.

DATA TYPES

wxPrintData()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPrintData()
See external documentation.

new(PrintData) -> wxPrintData()
Types:
    PrintData = wxPrintData()
See external documentation.

collate(This) -> boolean()
Types:
    This = wxPrintData()
See external documentation.

getBin(This) -> wx:wx_enum()
Types:
    This = wxPrintData()

getColour(This) -> boolean()
Types:
    This = wxPrintData()
See external documentation.

duplex(This) -> wx:wx_enum()
Types:
    This = wxPrintData()
See external documentation.
Res = ?wxDUPLEX_SIMPLEX | ?wxDUPLEX_HORIZONTAL | ?wxDUPLEX_VERTICAL

getNoCopies(This) -> integer()
Types:
   This = wxPrintData()
See external documentation.

gerOrientation(This) -> integer()
Types:
   This = wxPrintData()
See external documentation.

getPaperId(This) -> wx:wx_enum()
Types:
   This = wxPrintData()
See external documentation.
wxPrintData

wxPAPER_PENV_6_ROTATED | ?wxPAPER_PENV_7_ROTATED | ?wxPAPER_PENV_8_ROTATED | ?wxPAPER_PENV_9_ROTATED | ?wxPAPER_PENV_10_ROTATED

getPrinterName(This) -> unicode:charlist()
Types:
   This = wxPrintData()
See external documentation.

getQuality(This) -> integer()
Types:
   This = wxPrintData()
See external documentation.

isOk(This) -> boolean()
Types:
   This = wxPrintData()
See external documentation.

setBin(This, Bin) -> ok
Types:
   This = wxPrintData()
   Bin = wx:wx_enum()
See external documentation.
   Bin = ?wxPRINTBIN_DEFAULT | ?wxPRINTBIN_ONLYONE | ?wxPRINTBIN_LOWER

setCollate(This, Flag) -> ok
Types:
   This = wxPrintData()
   Flag = boolean()
See external documentation.

setColour(This, Colour) -> ok
Types:
   This = wxPrintData()
   Colour = boolean()
See external documentation.

setDuplex(This, Duplex) -> ok
Types:
   This = wxPrintData()
Duplex = \textit{wx:wx_enum()}

See external documentation.

Duplex = ?wxDUPLEX_SIMPLEX | ?wxDUPLEX_HORIZONTAL | ?wxDUPLEX_VERTICAL

setNoCopies(This, V) -> ok

Types:

\begin{verbatim}
This = wxPrintData()
V = integer()
\end{verbatim}

See external documentation.

setOrientation(This, Orient) -> ok

Types:

\begin{verbatim}
This = wxPrintData()
Orient = integer()
\end{verbatim}

See external documentation.

setPaperId(This, SizeId) -> ok

Types:

\begin{verbatim}
This = wxPrintData()
SizeId = \textit{wx:wx_enum()}
\end{verbatim}

See external documentation.

\begin{verbatim}
?wxPAPER_A5 | ?wxPAPER_B4 | ?wxPAPER_B5 | ?wxPAPER_FOLIO |
?wxPAPER_FANFOLD_STD_GERMAN | ?wxPAPER_FANFOLD_LGL_GERMAN |
?wxPAPER_ENV_INVITE | ?wxPAPER_LETTER_EXTRA | ?wxPAPER_LEGAL_EXTRA |
?wxPAPER_LETTER_EXTRA_TRANSVERSE | ?wxPAPER_A_PLUS | ?wxPAPER_B_PLUS |
?wxPAPER_A4_PLUS | ?wxPAPER_A5_TRANSVERSE | ?wxPAPER_B5_TRANSVERSE |
?wxPAPER_A3_EXTRA | ?wxPAPER_A5_EXTRA | ?wxPAPER_B5_EXTRA |
?wxPAPER_A2 | ?wxPAPER_A3_TRANSVERSE | ?wxPAPER_A3_EXTRA_TRANSVERSE | ?wxPAPER_A6 |
?wxPAPER_LETTER_ROTATED | ?wxPAPER_A3_ROTATED | ?wxPAPER_A4_ROTATED |
?wxPAPER_A5_ROTATED | ?wxPAPER_B4_JIS_ROTATED | ?wxPAPER_B5_JIS_ROTATED |
?wxPAPER_JAPANESE_POSTCARD_ROTATED | ?wxPAPER_JENV_KAKU2_ROTATED | ?wxPAPER_JENV_KAKU3_ROTATED |
?wxPAPER_JENV_CHOU3_ROTATED | ?wxPAPER_JENV_CHOU4_ROTATED | ?wxPAPER_B6_JIS |
?wxPAPER_B6_JIS_ROTATED | ?wxPAPER_B6_ENV_12X11 | ?wxPAPER_JENV_YOU4 |
?wxPAPER_PENV_1 | ?wxPAPER_PENV_2 | ?wxPAPER_PENV_3 | ?wxPAPER_PENV_4 |
\end{verbatim}
setPrinterName(This, Name) -> ok
Types:
   This = wxPrintData()
   Name = unicode:chardata()
See external documentation.

setQuality(This, Quality) -> ok
Types:
   This = wxPrintData()
   Quality = integer()
See external documentation.

destroy(This::wxPrintData()) -> ok
Destroys this object, do not use object again
wxPrintDialog
Erlang module

See external documentation: wxPrintDialog.
This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES
wxPrintDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for
comparsion stored on disc or distributed for use on other nodes.

Exports

new(Parent) -> wxPrintDialog()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxPrintDialog()
Types:
    Parent = wxWindow:wxWindow()
    Option = {data, wxPrintDialogData:wxPrintDialogData()}
See external documentation.
Also:
new(Parent, Data) -> wxPrintDialog() when
Parent::wxWindow:wxWindow(), Data::wxPrintData:wxPrintData().

getPrintDialogData(This) -> wxPrintDialogData:wxPrintDialogData()
Types:
    This = wxPrintDialog()
See external documentation.

getPrintDC(This) -> wxDC:wxDC()
Types:
    This = wxPrintDialog()
See external documentation.

destroy(This::wxPrintDialog()) -> ok
Destroys this object, do not use object again
wxPrintDialogData

Erlang module

See external documentation: wxPrintDialogData.

DATA TYPES

wxPrintDialogData()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxPrintDialogData()
See external documentation.

new(DialogData) -> wxPrintDialogData()
Types:
   DialogData = wxPrintDialogData() | wxPrintData:wxPrintData()
See external documentation.

enableHelp(This, Flag) -> ok
Types:
   This = wxPrintDialogData()
   Flag = boolean()
See external documentation.

enablePageNumbers(This, Flag) -> ok
Types:
   This = wxPrintDialogData()
   Flag = boolean()
See external documentation.

enablePrintToFile(This, Flag) -> ok
Types:
   This = wxPrintDialogData()
   Flag = boolean()
See external documentation.

enableSelection(This, Flag) -> ok
Types:
   This = wxPrintDialogData()
   Flag = boolean()
See external documentation.

getAllPages(This) -> boolean()
Types:
    This = wxPrintDialogData()
See external documentation.

getcollate(This) -> boolean()
Types:
    This = wxPrintDialogData()
See external documentation.

getFromPage(This) -> integer()
Types:
    This = wxPrintDialogData()
See external documentation.

getMaxPage(This) -> integer()
Types:
    This = wxPrintDialogData()
See external documentation.

getMinPage(This) -> integer()
Types:
    This = wxPrintDialogData()
See external documentation.

getNoCopies(This) -> integer()
Types:
    This = wxPrintDialogData()
See external documentation.

getPrintData(This) -> wxPrintData:wxPrintData()
Types:
    This = wxPrintDialogData()
See external documentation.

getPrintToFile(This) -> boolean()
Types:
    This = wxPrintDialogData()
See external documentation.
wxPrintDialogData

getSelection(This) -> boolean()
Types:
    This = wxPrintDialogData()
See external documentation.

getoPage(This) -> integer()
Types:
    This = wxPrintDialogData()
See external documentation.

isOk(This) -> boolean()
Types:
    This = wxPrintDialogData()
See external documentation.

setCollate(This, Flag) -> ok
Types:
    This = wxPrintDialogData()
    Flag = boolean()
See external documentation.

setFromPage(This, V) -> ok
Types:
    This = wxPrintDialogData()
    V = integer()
See external documentation.

setMaxPage(This, V) -> ok
Types:
    This = wxPrintDialogData()
    V = integer()
See external documentation.

setMinPage(This, V) -> ok
Types:
    This = wxPrintDialogData()
    V = integer()
See external documentation.

setNoCopies(This, V) -> ok
Types:
    This = wxPrintDialogData()
    V = integer()
See external documentation.

setPrintData(This, PrintData) -> ok
Types:
  This = wxPrintDialogData()
  PrintData = wxPrintData:wxPrintData()
See external documentation.

setPrintToFile(This, Flag) -> ok
Types:
  This = wxPrintDialogData()
  Flag = boolean()
See external documentation.

setSelection(This, Flag) -> ok
Types:
  This = wxPrintDialogData()
  Flag = boolean()
See external documentation.

setToPage(This, V) -> ok
Types:
  This = wxPrintDialogData()
  V = integer()
See external documentation.

destroy(This::wxPrintDialogData()) -> ok
Destroy this object, do not use object again
wxPrintPreview

Erlang module

See external documentation: wxPrintPreview.

**DATA TYPES**

wxPrintPreview()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new(Printout) -> wxPrintPreview()

Types:

- **Printout** = wxPrintout:wxPrintout()

Equivalent to new(Printout, []).

new(Printout, Options::[Option]) -> wxPrintPreview()

Types:

- **Printout** = wxPrintout:wxPrintout()
- **Option** = [printoutForPrinting, wxPrintout:wxPrintout()] | [data, wxPrintDialogData:wxPrintDialogData()]

See external documentation.

new(Printout, PrintoutForPrinting, Data) -> wxPrintPreview()

Types:

- **Printout** = wxPrintout:wxPrintout()
- **PrintoutForPrinting** = wxPrintout:wxPrintout()
- **Data** = wxPrintData:wxPrintData()

See external documentation.

getCanvas(This) -> wxPreviewCanvas:wxPreviewCanvas()

Types:

- **This** = wxPrintPreview()

See external documentation.

gGetCurrentPage(This) -> integer()

Types:

- **This** = wxPrintPreview()

See external documentation.
getFrame(This) -> wxFrame:wxFrame()
Types:
   This = wxPrintPreview()
See external documentation.

getMaxPage(This) -> integer()
Types:
   This = wxPrintPreview()
See external documentation.

getMinPage(This) -> integer()
Types:
   This = wxPrintPreview()
See external documentation.

getPrintout(This) -> wxPrintout:wxPrintout()
Types:
   This = wxPrintPreview()
See external documentation.

getPrintoutForPrinting(This) -> wxPrintout:wxPrintout()
Types:
   This = wxPrintPreview()
See external documentation.

isOk(This) -> boolean()
Types:
   This = wxPrintPreview()
See external documentation.

paintPage(This, Canvas, Dc) -> boolean()
Types:
   This = wxPrintPreview()
   Canvas = wxPreviewCanvas:wxPreviewCanvas()
   Dc = wxDC:wxDC()
See external documentation.

print(This, Interactive) -> boolean()
Types:
   This = wxPrintPreview()
   Interactive = boolean()
See external documentation.
renderPage(This, PageNum) -> boolean()
Types:
    This = wxPrintPreview()
    PageNum = integer()
See external documentation.

setCanvas(This, Canvas) -> ok
Types:
    This = wxPrintPreview()
    Canvas = wxPreviewCanvas:wxPreviewCanvas()
See external documentation.

setCurrentPage(This, PageNum) -> boolean()
Types:
    This = wxPrintPreview()
    PageNum = integer()
See external documentation.

setFrame(This, Frame) -> ok
Types:
    This = wxPrintPreview()
    Frame = wxFrame:wxFrame()
See external documentation.

setPrintout(This, Printout) -> ok
Types:
    This = wxPrintPreview()
    Printout = wxPrintout:wxPrintout()
See external documentation.

setZoom(This, Percent) -> ok
Types:
    This = wxPrintPreview()
    Percent = integer()
See external documentation.

destroy(This::wxPrintPreview()) -> ok
Destroy this object, do not use object again
wxPrinter

Erlang module

See external documentation: wxPrinter.

**DATA TYPES**

wxPrinter()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxPrinter()

Equivalent to `new([])`.

new(Options::[Option]) -> wxPrinter()

Types:

  Option = {data, wxPrintDialogData:wxPrintDialogData()}

See external documentation.

createAbortWindow(This, Parent, Printout) -> wxWindow:wxWindow()

Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()
  Printout = wxPrintout:wxPrintout()

See external documentation.

getAbort(This) -> boolean()

Types:

  This = wxPrinter()

See external documentation.

getLastError() -> wx:wx_enum()

See external documentation.

Res = ?wxPRINTER_NO_ERROR | ?wxPRINTER_CANCELLED | ?wxPRINTER_ERROR

getPrintDialogData(This) -> wxPrintDialogData:wxPrintDialogData()

Types:

  This = wxPrinter()

See external documentation.
print(This, Parent, Printout) -> boolean()
Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()
  Printout = wxPrintout:wxPrintout()

Equivalent to print(This, Parent, Printout, []).

print(This, Parent, Printout, Options::[Option]) -> boolean()
Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()
  Printout = wxPrintout:wxPrintout()
  Option = {prompt, boolean()}

See external documentation.

printDialog(This, Parent) -> wxDC:wxDC()
Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()

See external documentation.

reportError(This, Parent, Printout, Message) -> ok
Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()
  Printout = wxPrintout:wxPrintout()
  Message = unicode:chardata()

See external documentation.

setup(This, Parent) -> boolean()
Types:

  This = wxPrinter()
  Parent = wxWindow:wxWindow()

See external documentation.

destroy(This::wxPrinter()) -> ok
Destroys this object, do not use object again
wxPrintout

Erlang module

See external documentation: wxPrintout.

DATA TYPES

wxPrintout()

An object reference. The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new(Title::string(), OnPrintPage::function()) -> wxPrintout:wxPrintout()
@equiv new(Title, OnPrintPage, [])

new(Title::string(), OnPrintPage::function(), Opts::[Option]) ->
wxPrintout:wxPrintout()

Types:
    Option = {onPreparePrinting, OnPreparePrinting::function()}
            | {onBeginPrinting, OnBeginPrinting::function()}
            | {onEndPrinting, OnEndPrinting::function()}
            | {onBeginDocument,
              OnBeginDocument::function()}
            | {onEndDocument, OnEndDocument::function()}
            | {hasPage, HasPage::function()}
            | {getPageInfo, GetPageInfo::function()}

Creates a wxPrintout object with a callback fun and optionally other callback funs.

OnPrintPage(This,Page) -> boolean()
OnPreparePrinting(This) -> term()
OnBeginPrinting(This) -> term()
OnEndPrinting(This) -> term()
OnBeginDocument(This,StartPage,EndPage) -> boolean()
OnEndDocument(This) -> term()
HasPage(This,Page) -> boolean()
getPageInfo(This) -> {MinPage::integer(), MaxPage::integer(),
                      PageFrom::integer(), PageTo::integer()}

The This argument is the wxPrintout object reference to this object
NOTE: The callbacks may not call other processes.

getDC(This) -> wxDC:wxDC()

Types:
    This = wxPrintout()
See external documentation.

getPageSizeMM(This) -> {W::integer(), H::integer()}
Types:
   This = wxPrintout()
See external documentation.

getPageSizePixels(This) -> {W::integer(), H::integer()}
Types:
   This = wxPrintout()
See external documentation.

generateRectPixels(This) -> {X::integer(), Y::integer(), W::integer(),
H::integer()}
Types:
   This = wxPrintout()
See external documentation.

generatePPIPrinter(This) -> {X::integer(), Y::integer()}
Types:
   This = wxPrintout()
See external documentation.

generatePPIScreen(This) -> {X::integer(), Y::integer()}
Types:
   This = wxPrintout()
See external documentation.

generateTitle(This) -> unicode:charlist()
Types:
   This = wxPrintout()
See external documentation.

isPreview(This) -> boolean()
Types:
   This = wxPrintout()
See external documentation.

fitThisSizeToPaper(This, ImageSize) -> ok
Types:
   This = wxPrintout()
   ImageSize = {W::integer(), H::integer()}
See external documentation.
fitThisSizeToPage(This, ImageSize) -> ok
Types:
   This = wxPrintout()
   ImageSize = {W::integer(), H::integer()}
See external documentation.

fitThisSizeToPageMargins(This, ImageSize, PageSetupData) -> ok
Types:
   This = wxPrintout()
   ImageSize = {W::integer(), H::integer()}
   PageSetupData = wxPageSetupDialogData:wxPageSetupDialogData()
See external documentation.

mapScreenSizeToPaper(This) -> ok
Types:
   This = wxPrintout()
See external documentation.

mapScreenSizeToPage(This) -> ok
Types:
   This = wxPrintout()
See external documentation.

mapScreenSizeToPageMargins(This, PageSetupData) -> ok
Types:
   This = wxPrintout()
   PageSetupData = wxPageSetupDialogData:wxPageSetupDialogData()
See external documentation.

mapScreenSizeToDevice(This) -> ok
Types:
   This = wxPrintout()
See external documentation.

getLogicalPaperRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
   This = wxPrintout()
See external documentation.

getLogicalPageRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
This = wxPrintout()
See external documentation.

getLogicalPageMarginsRect(This, PageSetupData) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
    This = wxPrintout()
    PageSetupData = wxPageSetupDialogData:wxPageSetupDialogData()
See external documentation.

setLogicalOrigin(This, X, Y) -> ok
Types:
    This = wxPrintout()
    X = integer()
    Y = integer()
See external documentation.

offsetLogicalOrigin(This, Xoff, Yoff) -> ok
Types:
    This = wxPrintout()
    Xoff = integer()
    Yoff = integer()
See external documentation.

destroy(This::wxPrintout()) -> ok
Destroys this object, do not use object again
wxProgressDialog

Erlang module

See external documentation: `wxProgressDialog`.

This class is derived (and can use functions) from:
- `wxDialog`
- `wxTopLevelWindow`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

wxProgressDialog()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new(Title, Message) -> wxProgressDialog()

Types:
- `Title = unicode:chardata()`
- `Message = unicode:chardata()`

Equivalent to `new(Title, Message, [])`.

new(Title, Message, Options::[Option]) -> wxProgressDialog()

Types:
- `Title = unicode:chardata()`
- `Message = unicode:chardata()`
- `Option = {maximum, integer()} | {parent, wxWindow:wxWindow()} | {style, integer()}`

See external documentation.

resume(This) -> ok

Types:
- `This = wxProgressDialog()`

See external documentation.

update(This) -> ok

Types:
- `This = wxProgressDialog()`

See external documentation.
wxProgressDialog

update(This, Value) -> boolean()
Types:
  This = wxProgressDialog()
  Value = integer()
Equivalent to update(This, Value, []).

update(This, Value, Options::[Option]) -> boolean()
Types:
  This = wxProgressDialog()
  Value = integer()
  Option = {newmsg, unicode:chardata()}
See external documentation.

destroy(This::wxProgressDialog()) -> ok
Destroys this object, do not use object again
wxQueryNewPaletteEvent

Erlang module

Use wxEvtHandler:connect/3 with EventType:

query_new_palette

See also the message variant #wxQueryNewPalette{} event record type.
This class is derived (and can use functions) from: wxEvent

DATA TYPES

wxQueryNewPaletteEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

Exports

setPaletteRealized(This, Realized) -> ok
Types:

This = wxQueryNewPaletteEvent()

Realized = boolean()
See external documentation.

getPaletteRealized(This) -> boolean()
Types:

This = wxQueryNewPaletteEvent()
See external documentation.
wxRadioBox

Erlang module

See external documentation: `wxRadioBox`.

This class is derived (and can use functions) from:

* `wxControl`
* `wxWindow`
* `wxEvtHandler`

**DATA TYPES**

* `wxRadioBox()`

  An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

* `new(Parent, Id, Title, Pos, Size, Choices) -> wxRadioBox()`
  Types:
  ```
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
  Pos = {X::integer(), Y::integer()}
  Size = {W::integer(), H::integer()}
  Choices = [unicode:chardata()]
  
  Equivalent to `new(Parent, Id, Title, Pos, Size, Choices, [])`.
  ```

* `new(Parent, Id, Title, Pos, Size, Choices, Options::[Option]) -> wxRadioBox()`
  Types:
  ```
  Parent = wxWindow:wxWindow()
  Id = integer()
  Title = unicode:chardata()
  Pos = {X::integer(), Y::integer()}
  Size = {W::integer(), H::integer()}
  Choices = [unicode:chardata()]
  
  Option = {majorDim, integer()} | {style, integer()} | {val, wx:wx_object()}
  ```
  
  See external documentation.

* `create(This, Parent, Id, Title, Pos, Size, Choices) -> boolean()`
  Types:
  ```
  This = wxRadioBox()
  Parent = wxWindow:wxWindow()
  ```
Id = integer()
Title = unicode:chardata()
Pos = {X::integer(), Y::integer()}
Size = {W::integer(), H::integer()}
Choices = [unicode:chardata()]

Equivalent to create(This, Parent, Id, Title, Pos, Size, Choices, []).

create(This, Parent, Id, Title, Pos, Size, Choices, Options::[Option]) -> boolean()

Types:
   This = wxRadioBox()
   Parent = wxWindow:wxWindow()
   Id = integer()
   Title = unicode:chardata()
   Pos = {X::integer(), Y::integer()}
   Size = {W::integer(), H::integer()}
   Choices = [unicode:chardata()]
   Option = {majorDim, integer()} | {style, integer()} | {val, wx:wx_object()}

See external documentation.

enable(This) -> boolean()

Types:
   This = wxRadioBox()

Equivalent to enable(This, []).

enable(This, N) -> boolean()

Types:
   This = wxRadioBox()
   N = integer()

See external documentation.

Also:
enable(This, [Option]) -> boolean() when
This::wxRadioBox(),
Option :: ['enable', boolean()].

enable(This, N, Options::[Option]) -> boolean()

Types:
   This = wxRadioBox()
   N = integer()
   Option = {enable, boolean()}

See external documentation.
getSelection(This) -> integer()
Types:
   This = wxRadioBox()
See external documentation.

getString(This, N) -> unicode:charlist()
Types:
   This = wxRadioBox()
   N = integer()
See external documentation.

setSelection(This, N) -> ok
Types:
   This = wxRadioBox()
   N = integer()
See external documentation.

show(This) -> boolean()
Types:
   This = wxRadioBox()
Equivalent to show(This, []).

show(This, N) -> boolean()
Types:
   This = wxRadioBox()
   N = integer()
See external documentation.
Also:
show(This, [Option]) -> boolean() when
This::wxRadioBox(),
Option :: {'show', boolean()}.

show(This, N, Options::[Option]) -> boolean()
Types:
   This = wxRadioBox()
   N = integer()
   Option = {show, boolean()}
See external documentation.

getColumnCount(This) -> integer()
Types:
   This = wxRadioBox()
See external documentation.
getItemHelpText(This, N) -> unicode:charlist()
Types:
    This = wxRadioBox()
    N = integer()
See external documentation.

ggetItemToolTip(This, Item) -> wxToolTip:wxToolTip()
Types:
    This = wxRadioBox()
    Item = integer()
See external documentation.

ggetItemFromPoint(This, Pt) -> integer()
Types:
    This = wxRadioBox()
    Pt = {X::integer(), Y::integer()}
See external documentation.

ggetRowCount(This) -> integer()
Types:
    This = wxRadioBox()
See external documentation.

isItemEnabled(This, N) -> boolean()
Types:
    This = wxRadioBox()
    N = integer()
See external documentation.

isItemShown(This, N) -> boolean()
Types:
    This = wxRadioBox()
    N = integer()
See external documentation.

setItemHelpText(This, N, HelpText) -> ok
Types:
    This = wxRadioBox()
    N = integer()
    HelpText = unicode:chardata()
See external documentation.
setItemToolTip(This, Item, Text) -> ok

Types:

  This = wxRadioBox()
  Item = integer()
  Text = unicode:chardata()

See external documentation.

destroy(This::wxRadioBox()) -> ok

Destroys this object, do not use object again
wxRadioButton

Erlang module

See external documentation: wxRadioButton.
This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES
wxRadioButton()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparsion stored on disc or distributed for use on other nodes.

Exports

new() -> wxRadioButton()
See external documentation.

new(Parent, Id, Label) -> wxRadioButton()
Types:
    Parent = wxWindow:wxWindow()
    Id = integer()
    Label = unicode:chardata()
Equivalent to new(Parent, Id, Label, []).

new(Parent, Id, Label, Options::[Option]) -> wxRadioButton()
Types:
    Parent = wxWindow:wxWindow()
    Id = integer()
    Label = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(),
    H::integer()}} | {style, integer()} | {validator, wx:wx_object()}
See external documentation.

create(This, Parent, Id, Label) -> boolean()
Types:
    This = wxRadioButton()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Label = unicode:chardata()
Equivalent to create(This, Parent, Id, Label, []).
create(This, Parent, Id, Label, Options::[Option]) -> boolean()

Types:
  This = wxRadioButton()
  Parent = wxWindow:wxWindow()
  Id = integer()
  Label = unicode:chardata()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getValue(This) -> boolean()

Types:
  This = wxRadioButton()

See external documentation.

setValue(This, Val) -> ok

Types:
  This = wxRadioButton()
  Val = boolean()

See external documentation.

destroy(This::wxRadioButton()) -> ok

Destroys this object, do not use object again
wxRegion

Erlang module

See external documentation: wxRegion.

DATA TYPES

wxRegion()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxRegion()
See external documentation.

new(Bmp) -> wxRegion()
Types:
   Bmp = wxBitmap:wxBitmap()
See external documentation.
Also:
new(Rect) -> wxRegion() when
Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.

new(TopLeft, BottomRight) -> wxRegion()
Types:
   TopLeft = {X::integer(), Y::integer()}
   BottomRight = {X::integer(), Y::integer()}
See external documentation.

new(X, Y, W, H) -> wxRegion()
Types:
   X = integer()
   Y = integer()
   W = integer()
   H = integer()
See external documentation.

clear(This) -> ok
Types:
   This = wxRegion()
See external documentation.
contains(This, Pt) -> wx:wx_enum()
Types:
  This = wxRegion()
  Pt = {X::integer(), Y::integer()}
See external documentation.
Also:
contains(This, Rect) -> wx:wx_enum() when
This::wxRegion(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.  
Res = ?wxOutRegion | ?wxPartRegion | ?wxInRegion

contains(This, X, Y) -> wx:wx_enum()
Types:
  This = wxRegion()
  X = integer()
  Y = integer()
See external documentation.
Res = ?wxOutRegion | ?wxPartRegion | ?wxInRegion

contains(This, X, Y, W, H) -> wx:wx_enum()
Types:
  This = wxRegion()
  X = integer()
  Y = integer()
  W = integer()
  H = integer()
See external documentation.
Res = ?wxOutRegion | ?wxPartRegion | ?wxInRegion

convertToBitmap(This) -> wxBitmap:wxBitmap()
Types:
  This = wxRegion()
See external documentation.

getBox(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
  This = wxRegion()
See external documentation.

intersect(This, Region) -> boolean()
Types:
  This = wxRegion()
  Region = wxRegion()
See external documentation.
Also:
intersect(This, Rect) -> boolean() when
This::wxRegion(), Rect::[X::integer(), Y::integer(), W::integer(), H::integer()].

intersect(This, X, Y, W, H) -> boolean()
Types:
  This = wxRegion()
  X = integer()
  Y = integer()
  W = integer()
  H = integer()
See external documentation.

isEmpty(This) -> boolean()
Types:
  This = wxRegion()
See external documentation.

subtract(This, Region) -> boolean()
Types:
  This = wxRegion()
  Region = wxRegion()
See external documentation.
Also:
subtract(This, Rect) -> boolean() when
This::wxRegion(), Rect::[X::integer(), Y::integer(), W::integer(), H::integer()].

subtract(This, X, Y, W, H) -> boolean()
Types:
  This = wxRegion()
  X = integer()
  Y = integer()
  W = integer()
  H = integer()
See external documentation.

offset(This, Pt) -> boolean()
Types:
  This = wxRegion()
  Pt = {X::integer(), Y::integer()}
See external documentation.

offset(This, X, Y) -> boolean()
Types:
  This = wxRegion()
X = integer()
Y = integer()

See external documentation.

union(This, Region) -> boolean()
Types:
   This = wxRegion()
   Region = wxRegion() | wxBitmap:wxBitmap()

See external documentation.
Also:
union(This, Rect) -> boolean() when
This::wxRegion(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.

union(This, Bmp, Transp) -> boolean()
Types:
   This = wxRegion()
   Bmp = wxBitmap:wxBitmap()
   Transp = wx:wx_colour()

Equivalent to union(This, Bmp, Transp, []).

union(This, Bmp, Transp, Options::[Option]) -> boolean()
Types:
   This = wxRegion()
   Bmp = wxBitmap:wxBitmap()
   Transp = wx:wx_colour()
   Option = {tolerance, integer()}

See external documentation.

union(This, X, Y, W, H) -> boolean()
Types:
   This = wxRegion()
   X = integer()
   Y = integer()
   W = integer()
   H = integer()

See external documentation.

Xor(This, Region) -> boolean()
Types:
   This = wxRegion()
   Region = wxRegion()

See external documentation.
Also:
'Xor'(This, Rect) -> boolean() when
This::wxRegion(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}. 

Xor(This, X, Y, W, H) -> boolean()
Types:
  This = wxRegion()
  X = integer()
  Y = integer()
  W = integer()
  H = integer()
See external documentation.

destroy(This::wxRegion()) -> ok
Destroys this object, do not use object again
wxSashEvent

Erlang module

See external documentation: wxSashEvent.

Use wxEvtHandler:connect/3 with EventType:

  sash_dragged

See also the message variant #wxSash/ event record type.

This class is derived (and can use functions) from:
wxCommandEvent
wxEvent

DATA TYPES

wxSashEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getEdge(This) -> wx:wx_enum()

Types:

  This = wxSashEvent()

See external documentation.


getDragRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}

Types:

  This = wxSashEvent()

See external documentation.

getDragStatus(This) -> wx:wx_enum()

Types:

  This = wxSashEvent()

See external documentation.

Res = ?wxSASH_STATUS_OK | ?wxSASH_STATUS_OUT_OF_RANGE
**wxSashLayoutWindow**

Erlang module

See external documentation: `wxSashLayoutWindow`.

This class is derived (and can use functions) from:
- `wxSashWindow`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

`wxSashLayoutWindow()`

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxSashLayoutWindow()`

See external documentation.

`new(Parent) -> wxSashLayoutWindow()`

Types:

- `Parent = wxWindow:wxWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxSashLayoutWindow()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

`create(This, Parent) -> boolean()`

Types:

- `This = wxSashLayoutWindow()`
- `Parent = wxWindow:wxWindow()`

Equivalent to `create(This, Parent, [])`.

`create(This, Parent, Options::[Option]) -> boolean()`

Types:

- `This = wxSashLayoutWindow()`
- `Parent = wxWindow:wxWindow()`
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

getAlignment(This) -> wx:wx_enum()
Types:
   This = wxSashLayoutWindow()

See external documentation.

getAddression(This, Align) -> ok
Types:
   This = wxSashLayoutWindow()
   Align = wx:wx_enum()

See external documentation.

setDefaultSize(This, Size) -> ok
Types:
   This = wxSashLayoutWindow()
   Size = {W::integer(), H::integer()}

See external documentation.

setOrientation(This, Orient) -> ok
Types:
   This = wxSashLayoutWindow()
   Orient = wx:wx_enum()

See external documentation.
Orient = ?wxLAYOUT_HORIZONTAL | ?wxLAYOUT_VERTICAL

destroy(This::wxSashLayoutWindow()) -> ok
Destroys this object, do not use object again
wxSashWindow
Erlang module

See external documentation: wxSashWindow.
This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES
wxSashWindow()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxSashWindow()
See external documentation.

ew(Parent) -> wxSashWindow()
Types:
Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxSashWindow()
Types:
Parent = wxWindow:wxWindow()
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

getSashVisible(This, Edge) -> boolean()
Types:
This = wxSashWindow()
Edge = wx:wx_enum()
See external documentation.

getMaximumSizeX(This) -> integer()
Types:
This = wxSashWindow()
See external documentation.
getMaximumSizeY(This) -> integer()
Types:
    This = wxSashWindow()
See external documentation.

getMinimumSizeX(This) -> integer()
Types:
    This = wxSashWindow()
See external documentation.

getMinimumSizeY(This) -> integer()
Types:
    This = wxSashWindow()
See external documentation.

setMaximumSizeX(This, Max) -> ok
Types:
    This = wxSashWindow()
    Max = integer()
See external documentation.

setMaximumSizeY(This, Max) -> ok
Types:
    This = wxSashWindow()
    Max = integer()
See external documentation.

setMinimumSizeX(This, Min) -> ok
Types:
    This = wxSashWindow()
    Min = integer()
See external documentation.

setMinimumSizeY(This, Min) -> ok
Types:
    This = wxSashWindow()
    Min = integer()
See external documentation.

setSashVisible(This, Edge, Sash) -> ok
Types:
    This = wxSashWindow()
    Edge = wx:wx_enum()
Sash = boolean()
See external documentation.

destroy(This::wxSashWindow()) -> ok
Destroys this object, do not use object again
wxScreenDC

Erlang module

See external documentation: wxScreenDC.
This class is derived (and can use functions) from:
wxDC

DATA TYPES

wxScreenDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxScreenDC()
See external documentation.

destroy(This::wxScreenDC()) -> ok
Destroys this object, do not use object again
wxScrollBar

Erlang module

See external documentation: `wxScrollBar`.

This class is derived (and can use functions) from:

wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxScrollBar()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxScrollBar()

See external documentation.

new(Parent, Id) -> wxScrollBar()

Types:

Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxScrollBar()

Types:

Parent = wxWindow:wxWindow()
Id = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id) -> boolean()

Types:

This = wxScrollBar()
Parent = wxWindow:wxWindow()
Id = integer()

Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()

Types:
**wxScrollBar**

This = wxScrollBar()
Parent = wxWindow:wxWindow()
Id = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getRange(This) -> integer()
Types:
   This = wxScrollBar()
See external documentation.

g getPageSize(This) -> integer()
Types:
   This = wxScrollBar()
See external documentation.

g getThumbPosition(This) -> integer()
Types:
   This = wxScrollBar()
See external documentation.

g getThumbSize(This) -> integer()
Types:
   This = wxScrollBar()
See external documentation.

setThumbPosition(This, ViewStart) -> ok
Types:
   This = wxScrollBar()
   ViewStart = integer()
See external documentation.

setScrollbar(This, Position, ThumbSize, Range, PageSize) -> ok
Types:
   This = wxScrollBar()
   Position = integer()
   ThumbSize = integer()
   Range = integer()
   PageSize = integer()

Equivalent to setScrollbar(This, Position, ThumbSize, Range, PageSize, []).
setScrollbar(This, Position, ThumbSize, Range, PageSize, Options::[Option]) -> ok

Types:
   This = wxScrollBar()
   Position = integer()
   ThumbSize = integer()
   Range = integer()
   PageSize = integer()
   Option = {refresh, boolean()}

See external documentation.

destroy(This::wxScrollBar()) -> ok

DESTROYs this object, do not use object again.
wxScrollEvent

Erlang module

See external documentation: **wxScrollEvent**.

Use `wxEvtHandler:connect/3` with EventType:

- `scroll_top`
- `scroll_bottom`
- `scroll_lineup`
- `scroll_linedown`
- `scroll_pageup`
- `scroll_pagedown`
- `scroll_thumbtrack`
- `scroll_thumbrelease`
- `scroll_changed`

See also the message variant `#wxScroll{ }` event record type.

This class is derived (and can use functions) from:

- `wxCommandEvent`
- `wxEvent`

**DATA TYPES**

**wxScrollEvent()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

**getOrientation(This) -> integer()**

Types:

- **This = wxScrollEvent()**

See external documentation.

**getPosition(This) -> integer()**

Types:

- **This = wxScrollEvent()**

See external documentation.
**wxScrollWinEvent**

Erlang module

See external documentation: `wxScrollWinEvent`.

Use `wxEvtHandler:connect/3` with EventType:
- `scrollwin_top`
- `scrollwin_bottom`
- `scrollwin_lineup`
- `scrollwin_linedown`
- `scrollwin_pageup`
- `scrollwin_pagedown`
- `scrollwin_thumbtrack`
- `scrollwin_thumbrelease`

See also the message variant `#wxScrollWin/[]` event record type.

This class is derived (and can use functions) from:
`wxEvent`

**DATA TYPES**

`wxScrollWinEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`getOrientation(This) -> integer()`

Types:
- `This = wxScrollWinEvent()`

See external documentation.

`getPosition(This) -> integer()`

Types:
- `This = wxScrollWinEvent()`

See external documentation.
wxScrolledWindow

Erlang module

See external documentation: `wxScrolledWindow`.

This class is derived (and can use functions) from:

- `wxPanel`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

`wxScrolledWindow()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxScrolledWindow()`

See external documentation.

`new(Parent) -> wxScrolledWindow()`

Types:

- `Parent = wxWindow:wxWindow()`

Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxScrolledWindow()`

Types:

- `Parent = wxWindow:wxWindow()`
- `Option = {winid, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

`calcScrolledPosition(This, Pt) -> {X::integer(), Y::integer()}`

Types:

- `This = wxScrolledWindow()`
- `Pt = {X::integer(), Y::integer()}

See external documentation.

`calcScrolledPosition(This, X, Y) -> {Xx::integer(), Yy::integer()}`

Types:

- `This = wxScrolledWindow()`
- `X = integer()`
- `Y = integer()`
See external documentation.

calcUnscrolledPosition(This, Pt) -> {X::integer(), Y::integer()}
Types:
   This = wxScrolledWindow()
   Pt = (X::integer(), Y::integer())
See external documentation.

calcUnscrolledPosition(This, X, Y) -> {Xx::integer(), Yy::integer()}
Types:
   This = wxScrolledWindow()
   X = integer()
   Y = integer()
See external documentation.

enableScrolling(This, X_scrolling, Y_scrolling) -> ok
Types:
   This = wxScrolledWindow()
   X_scrolling = boolean()
   Y_scrolling = boolean()
See external documentation.

getScrollPixelsPerUnit(This) -> {PixelsPerUnitX::integer(), PixelsPerUnitY::integer()}
Types:
   This = wxScrolledWindow()
See external documentation.

getViewStart(This) -> {X::integer(), Y::integer()}
Types:
   This = wxScrolledWindow()
See external documentation.

doPrepareDC(This, Dc) -> ok
Types:
   This = wxScrolledWindow()
   Dc = wxDC:wxDC()
See external documentation.

prepareDC(This, Dc) -> ok
Types:
   This = wxScrolledWindow()
   Dc = wxDC:wxDC()
scroll(This, X, Y) -> ok
Types:
   This = wxScrolledWindow()
   X = integer()
   Y = integer()
See external documentation.

setScrollbars(This, PixelsPerUnitX, PixelsPerUnitY, NoUnitsX, NoUnitsY) -> ok
Types:
   This = wxScrolledWindow()
   PixelsPerUnitX = integer()
   PixelsPerUnitY = integer()
   NoUnitsX = integer()
   NoUnitsY = integer()
Equivalent to setScrollbars(This, PixelsPerUnitX, PixelsPerUnitY, NoUnitsX, NoUnitsY, []).

setScrollbars(This, PixelsPerUnitX, PixelsPerUnitY, NoUnitsX, NoUnitsY, Options::[Option]) -> ok
Types:
   This = wxScrolledWindow()
   PixelsPerUnitX = integer()
   PixelsPerUnitY = integer()
   NoUnitsX = integer()
   NoUnitsY = integer()
   Option = {xPos, integer()} | {yPos, integer()} | {noRefresh, boolean()}
See external documentation.

setScrollRate(This, Xstep, Ystep) -> ok
Types:
   This = wxScrolledWindow()
   Xstep = integer()
   Ystep = integer()
See external documentation.

setTargetWindow(This, Target) -> ok
Types:
   This = wxScrolledWindow()
   Target = wxWindow:wxWindow()
See external documentation.
destroy(This::wxScrolledWindow()) -> ok
Destroys this object, do not use object again
wxSetCursorEvent

Erlang module

See external documentation: **wxSetCursorEvent**.

Use `wxEvtHandler:connect/3` with EventType:

```
set_cursor
```

See also the message variant `#wxSetCursor{}` event record type.

This class is derived (and can use functions) from:

```wxEvtHandler:connect/3```

**DATA TYPES**

**wxSetCursorEvent()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.

**Exports**

**getCursor(This) -> wxCursor:wxCursor()**

Types:

```
This = wxSetCursorEvent()
```

See external documentation.

**getX(This) -> integer()**

Types:

```
This = wxSetCursorEvent()
```

See external documentation.

**getY(This) -> integer()**

Types:

```
This = wxSetCursorEvent()
```

See external documentation.

**hasCursor(This) -> boolean()**

Types:

```
This = wxSetCursorEvent()
```

See external documentation.

**setCursor(This, Cursor) -> ok**

Types:

```
This = wxSetCursorEvent()
Cursor = wxCursor:wxCursor()
```
See external documentation.
wxShowEvent

Erlang module

See external documentation: `wxShowEvent`.

Use `wxEvtHandler:connect/3` with EventType:

```
show
```

See also the message variant `#wxShow/1` event record type.

This class is derived (and can use functions) from:

`wxEvent`

**DATA TYPES**

wxShowEvent()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`setShow(This, Show) -> ok`

Types:

```
This = wxShowEvent()
Show = boolean()
```

See external documentation.

`getShow(This) -> boolean()`

Types:

```
This = wxShowEvent()
```

See external documentation.
wxSingleChoiceDialog

Erlang module

See external documentation: wxSingleChoiceDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES

wxSingleChoiceDialog()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxSingleChoiceDialog()

See external documentation.

new(Parent, Message, Caption, Choices) -> wxSingleChoiceDialog()

Types:

Parent = wxWindow:wxWindow()
Message = unicode:chardata()
Caption = unicode:chardata()
Choices = [unicode:chardata()]

Equivalent to new(Parent, Message, Caption, Choices, []).

new(Parent, Message, Caption, Choices, Options::[Option]) ->
wxSingleChoiceDialog()

Types:

Parent = wxWindow:wxWindow()
Message = unicode:chardata()
Caption = unicode:chardata()
Choices = [unicode:chardata()]
Option = {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

getSelection(This) -> integer()

Types:

This = wxSingleChoiceDialog()

See external documentation.
wxSingleChoiceDialog

getStringSelection(This) -> unicode:charlist()
Types:
    This = wxSingleChoiceDialog()
See external documentation.

setSelection(This, Sel) -> ok
Types:
    This = wxSingleChoiceDialog()
    Sel = integer()
See external documentation.

destroy(This::wxSingleChoiceDialog()) -> ok
Destroys this object, do not use object again
wxSizeEvent

Erlang module

See external documentation: wxSizeEvent.

Use wxEvtHandler:connect/3 with EventType:

    size

See also the message variant #wxSize/ event record type.

This class is derived (and can use functions) from:
wxEvent

DATA TYPES

wxSizeEvent()

    An object reference. The representation is internal and can be changed without notice. It can't be used for comparrison stored on disc or distributed for use on other nodes.

Exports

getSize(This) -> {W::integer(), H::integer()}

Types:

    This = wxSizeEvent()

See external documentation.
wxSizer

Erlang module

See external documentation: wxSizer.

DATA TYPES

wxSizer()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

add(This, Window) -> wxSizerItem:wxSizerItem()

Types:

  This = wxSizer()
  Window = wxWindow:wxWindow() | wxSizer()

Equivalent to add(This, Window, []).

add(This, Width, Height) -> wxSizerItem:wxSizerItem()

Types:

  This = wxSizer()
  Width = integer()
  Height = integer()

See external documentation.

Also:
add(This, Window, [Option]) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(),
Option :: [{proportion, integer()}
    | {flag, integer()}
    | {border, integer()}
    | {userData, wx:wx_object()};
(This, Window, Flags) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), Flags::wxSizerFlags:wxSizerFlags().

add(This, Width, Height, Options::[Option]) -> wxSizerItem:wxSizerItem()

Types:

  This = wxSizer()
  Width = integer()
  Height = integer()
  Option = [{proportion, integer()} | {flag, integer()} | {border, integer()}
    | {userData, wx:wx_object()}]

See external documentation.
addSpacer(This, Size) -> wxSizerItem:wxSizerItem()
Types:
    This  = wxSizer()
    Size  = integer()
See external documentation.

addStretchSpacer(This) -> wxSizerItem:wxSizerItem()
Types:
    This  = wxSizer()
Equivalent to addStretchSpacer(This, []).

addStretchSpacer(This, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
    This  = wxSizer()
    Option = {prop, integer()}
See external documentation.

calcMin(This) -> {W::integer(), H::integer()}
Types:
    This  = wxSizer()
See external documentation.

clear(This) -> ok
Types:
    This  = wxSizer()
Equivalent to clear(This, []).

clear(This, Options::[Option]) -> ok
Types:
    This  = wxSizer()
    Option = {delete_windows, boolean()}
See external documentation.

detach(This, Index) -> boolean()
Types:
    This  = wxSizer()
    Index = integer()
See external documentation.
Also:
detach(This, Window) -> boolean() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer().
wxSizer

fit(This, Window) -> {W::integer(), H::integer()}
Types:
   This = wxSizer()
   Window = wxWindow:wxWindow()
See external documentation.

fitInside(This, Window) -> ok
Types:
   This = wxSizer()
   Window = wxWindow:wxWindow()
See external documentation.

getchildren(This) -> [wxSizerItem:wxSizerItem()]
Types:
   This = wxSizer()
See external documentation.

ggetItem(This, Window) -> wxSizerItem:wxSizerItem()
Types:
   This = wxSizer()
   Window = wxWindow:wxWindow() | wxSizer()
See external documentation.
Also:
getItem(This, Index) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Index::integer().

ggetItem(This, Window, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
   This = wxSizer()
   Window = wxWindow:wxWindow() | wxSizer()
   Option = {recursive, boolean()}
See external documentation.

getSize(This) -> {W::integer(), H::integer()}
Types:
   This = wxSizer()
See external documentation.

getPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxSizer()
See external documentation.
getMinSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxSizer()
See external documentation.

hide(This, Window) -> boolean()
Types:
  This = wxSizer()
  Window = wxWindow:wxWindow() | wxSizer()
See external documentation.
Also:
hide(This, Index) -> boolean() when
This::wxSizer(), Index::integer().

hide(This, Window, Options::*[Option]) -> boolean()
Types:
  This = wxSizer()
  Window = wxWindow:wxWindow() | wxSizer()
  Option = {recursive, boolean()}
See external documentation.

insert(This, Index, Item) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Index = integer()
  Item = wxSizerItem:wxSizerItem()
See external documentation.

insert(This, Index, Width, Height) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Index = integer()
  Width = integer()
  Height = integer()
See external documentation.
Also:
insert(This, Index, Window, [Option]) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Index::integer(), Window::wxWindow:wxWindow() | wxSizer(),
Option :: [{'proportion', integer()}]
| [{'flag', integer()}
| [{'border', integer()}
| [{'userData', wx:wx_object()}];
(This, Index, Window, Flags) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Index::integer(), Window::wxWindow:wxWindow() | wxSizer(),
Flags::wxSizerFlags:wxSizerFlags().
insert(This, Index, Width, Height, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
    This = wxSizer()
    Index = integer()
    Width = integer()
    Height = integer()
    Option = {proportion, integer()} | {flag, integer()} | {border, integer()} | {userData, wx:wx_object()}
See external documentation.

insertSpacer(This, Index, Size) -> wxSizerItem:wxSizerItem()
Types:
    This = wxSizer()
    Index = integer()
    Size = integer()
See external documentation.

insertStretchSpacer(This, Index) -> wxSizerItem:wxSizerItem()
Types:
    This = wxSizer()
    Index = integer()
Equivalent to insertStretchSpacer(This, Index, []).

insertStretchSpacer(This, Index, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
    This = wxSizer()
    Index = integer()
    Option = {prop, integer()}
See external documentation.

isShown(This, Index) -> boolean()
Types:
    This = wxSizer()
    Index = integer()
See external documentation.
Also:
isShown(This, Window) -> boolean() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer().

layout(This) -> ok
Types:
    This = wxSizer()
prepend(This, Item) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Item = wxSizerItem:wxSizerItem()
See external documentation.

prepend(This, Width, Height) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Width = integer()
  Height = integer()
See external documentation.
Also:
prepend(This, Window, [Option]) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(),
Option :: [{'proportion', integer()}|
{'flag', integer()}|
{'border', integer()}|
{'userData', wx:wx_object()}]:
(prepend(This, Window, Flags) -> wxSizerItem:wxSizerItem() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), Flags::wxSizerFlags:wxSizerFlags().

prepend(This, Width, Height, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Width = integer()
  Height = integer()
  Option = {proportion, integer()} | {flag, integer()} | {border, integer()} |
  {userData, wx:wx_object()}
See external documentation.

prependSpacer(This, Size) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
  Size = integer()
See external documentation.

prependStretchSpacer(This) -> wxSizerItem:wxSizerItem()
Types:
  This = wxSizer()
Equivalent to prependStretchSpacer(This, []).
prependStretchSpacer(This, Options::[Option]) -> wxSizerItem:wxSizerItem()
Types:
    This = wxSizer()
    Option = {prop, integer()}
See external documentation.

recalcSizes(This) -> ok
Types:
    This = wxSizer()
See external documentation.

remove(This, Index) -> boolean()
Types:
    This = wxSizer()
    Index = integer()
See external documentation.
Also:
remove(This, Sizer) -> boolean() when
This::wxSizer(), Sizer::wxSizer().

replace(This, Oldwin, Newwin) -> boolean()
Types:
    This = wxSizer()
    Oldwin = wxWindow:wxWindow() | wxSizer()
    Newwin = wxWindow:wxWindow() | wxSizer()
See external documentation.
Also:
replace(This, Index, Newitem) -> boolean() when
This::wxSizer(), Index::integer(), Newitem::wxSizerItem:wxSizerItem().

replace(This, Oldwin, Newwin, Options::[Option]) -> boolean()
Types:
    This = wxSizer()
    Oldwin = wxWindow:wxWindow() | wxSizer()
    Newwin = wxWindow:wxWindow() | wxSizer()
    Option = {recursive, boolean()}
See external documentation.

setDimension(This, X, Y, Width, Height) -> ok
Types:
    This = wxSizer()
    X = integer()
    Y = integer()
    Width = integer()
Height = integer()
See external documentation.

setMinSize(This, Size) -> ok
Types:
   This = wxSizer()
   Size = {W::integer(), H::integer()}
See external documentation.

setMinSize(This, Width, Height) -> ok
Types:
   This = wxSizer()
   Width = integer()
   Height = integer()
See external documentation.

setItemMinSize(This, Index, Size) -> boolean()
Types:
   This = wxSizer()
   Index = integer()
   Size = {W::integer(), H::integer()}
See external documentation.
Also:
setItemMinSize(This, Window, Size) -> boolean() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), Size::{W::integer(), H::integer()}.  

setItemMinSize(This, Index, Width, Height) -> boolean()
Types:
   This = wxSizer()
   Index = integer()
   Width = integer()
   Height = integer()
See external documentation.
Also:
setItemMinSize(This, Window, Width, Height) -> boolean() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer(), Width::integer(), Height::integer().  

setSizeHints(This, Window) -> ok
Types:
   This = wxSizer()
   Window = wxWindow:wxWindow()
See external documentation.
wxSizer

setVirtualSizeHints(This, Window) -> ok
Types:
    This = wxSizer()
    Window = wxWindow:wxWindow()
See external documentation.

decide(This, Window) -> boolean()
Types:
    This = wxSizer()
    Window = wxWindow:wxWindow()
See external documentation.

 Also:
decide(This, Window) -> boolean() when
This::wxSizer(), Window::wxWindow:wxWindow() | wxSizer();
(This, Decide) -> 'ok' when
This::wxSizer(), Decide::boolean().

decide(This, Index, Decisions::[Decision]) -> boolean()
Types:
    This = wxSizer()
    Index = integer()
    Decision = {show, boolean()} | {recursive, boolean()}
See external documentation.
 Also:
decide(This, Index, [Decision]) -> boolean() when
This::wxSizer(), Index::integer(),
Decision :: ['show', boolean()] | ['recursive', boolean()].
wxSizerFlags

Erlang module

See external documentation: `wxSizerFlags`.

DATA TYPES

`wxSizerFlags()`

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparsion stored on disc or distributed for use on other nodes.

Exports

```
new() -> wxSizerFlags()
```
Equivalent to `new([])`.

```
new(Options::[Option]) -> wxSizerFlags()
Types:
    Option = {proportion, integer()}
See external documentation.
```

```
align(This, Alignment) -> wxSizerFlags()
Types:
    This = wxSizerFlags()
    Alignment = integer()
See external documentation.
```

```
border(This) -> wxSizerFlags()
```
Equivalent to `border(This, [])`.

```
border(This, Options::[Option]) -> wxSizerFlags()
Types:
    This = wxSizerFlags()
    Option = {direction, integer()}
See external documentation.
```

```
border(This, Direction, BorderInPixels) -> wxSizerFlags()
Types:
    This = wxSizerFlags()
    Direction = integer()
    BorderInPixels = integer()
```
wxSizerFlags

See external documentation.

center(This) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
See external documentation.

centre(This) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
See external documentation.

expand(This) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
See external documentation.

left(This) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
See external documentation.

proportion(This, Proportion) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
   Proportion = integer()
See external documentation.

right(This) -> wxSizerFlags()
Types:
   This = wxSizerFlags()
See external documentation.

destroy(This::wxSizerFlags()) -> ok
Destroys this object, do not use object again
**wxSizerItem**

Erlang module

See external documentation: `wxSizerItem`.

**DATA TYPES**

`wxSizerItem()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxSizerItem()`

See external documentation.

`new(Window, Flags) -> wxSizerItem()`

Types:
```
Window = wxWindow:wxWindow() | wxSizer:wxSizer()
Flags = wxSizerFlags:wxSizerFlags()
```

See external documentation.

`new(Width, Height, Flags) -> wxSizerItem()`

Types:
```
Width = integer()
Height = integer()
Flags = wxSizerFlags:wxSizerFlags()
```

See external documentation.

`new(Window, Proportion, Flag, Border, UserData) -> wxSizerItem()`

Types:
```
Window = wxWindow:wxWindow() | wxSizer:wxSizer()
Proportion = integer()
Flag = integer()
Border = integer()
UserData = wx:wx_object()
```

See external documentation.

`new(Width, Height, Proportion, Flag, Border, UserData) -> wxSizerItem()`

Types:
```
Width = integer()
Height = integer()
Proportion = integer()
```
wxSizerItem

Flag = integer()
Border = integer()
UserData = wx:wx_object()

See external documentation.

calcMin(This) -> {W::integer(), H::integer()}
Types:
  This = wxSizerItem()
See external documentation.

deleteWindows(This) -> ok
Types:
  This = wxSizerItem()
See external documentation.

detachSizer(This) -> ok
Types:
  This = wxSizerItem()
See external documentation.

getBorder(This) -> integer()
Types:
  This = wxSizerItem()
See external documentation.

getFlag(This) -> integer()
Types:
  This = wxSizerItem()
See external documentation.

getMinSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxSizerItem()
See external documentation.

getPosition(This) -> {X::integer(), Y::integer()}
Types:
  This = wxSizerItem()
See external documentation.

getProportion(This) -> integer()
Types:
  This = wxSizerItem()
See external documentation.

getRatio(This) -> number()
Types:
  This = wxSizerItem()
See external documentation.

getRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
  This = wxSizerItem()
See external documentation.

getSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxSizerItem()
See external documentation.

getSizer(This) -> wxSizer:wxSizer()
Types:
  This = wxSizerItem()
See external documentation.

getSpacer(This) -> {W::integer(), H::integer()}
Types:
  This = wxSizerItem()
See external documentation.

getUserData(This) -> wx:wx_object()
Types:
  This = wxSizerItem()
See external documentation.

getWindow(This) -> wxWindow:wxWindow()
Types:
  This = wxSizerItem()
See external documentation.

isSizer(This) -> boolean()
Types:
  This = wxSizerItem()
See external documentation.
isVisible(This) -> boolean()
Types:
   This = wxSizerItem()
See external documentation.

isSpacer(This) -> boolean()
Types:
   This = wxSizerItem()
See external documentation.

isWindow(This) -> boolean()
Types:
   This = wxSizerItem()
See external documentation.

setBorder(This, Border) -> ok
Types:
   This = wxSizerItem()
   Border = integer()
See external documentation.

setDimension(This, Pos, Size) -> ok
Types:
   This = wxSizerItem()
   Pos = {X::integer(), Y::integer()}
   Size = {W::integer(), H::integer()}
See external documentation.

setFlag(This, Flag) -> ok
Types:
   This = wxSizerItem()
   Flag = integer()
See external documentation.

setInitSize(This, X, Y) -> ok
Types:
   This = wxSizerItem()
   X = integer()
   Y = integer()
See external documentation.

setMinSize(This, Size) -> ok
Types:
**wxSizerItem**

This = \textit{wxSizerItem()}
Size = \{W::integer(), H::integer()\}

See external documentation.

setMinSize(This, X, Y) -> ok
Types:
This = \textit{wxSizerItem()}
X = integer()
Y = integer()

See external documentation.

setProportion(This, Proportion) -> ok
Types:
This = \textit{wxSizerItem()}
Proportion = integer()

See external documentation.

setRatio(This, Ratio) -> ok
Types:
This = \textit{wxSizerItem()}
Ratio = number()

See external documentation.

Also:
setRatio(This, Size) -> 'ok' when
This::wxSizerItem(), Size::\{W::integer(), H::integer()\}.

setRatio(This, Width, Height) -> ok
Types:
This = \textit{wxSizerItem()}
Width = integer()
Height = integer()

See external documentation.

setSizer(This, Sizer) -> ok
Types:
This = \textit{wxSizerItem()}
Sizer = \textit{wxSizer::wxSizer()}

See external documentation.

setSpacer(This, Size) -> ok
Types:
This = \textit{wxSizerItem()}
Size = \{W::integer(), H::integer()\}
wxSizerItem

See external documentation.

setSpacer(This, Width, Height) -> ok
Types:
   This = wxSizerItem()
   Width = integer()
   Height = integer()

See external documentation.

setWindow(This, Window) -> ok
Types:
   This = wxSizerItem()
   Window = wxWindow:wxWindow()

See external documentation.

show(This, Show) -> ok
Types:
   This = wxSizerItem()
   Show = boolean()

See external documentation.

destroy(This::wxSizerItem()) -> ok
Destroys this object, do not use object again
wxSlider

Erlang module

See external documentation: wxSlider.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxSlider()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxSlider()

See external documentation.

new(Parent, Id, Value, MinValue, MaxValue) -> wxSlider()

Types:

Parent = wxWindow:wxWindow()
Id = integer()
Value = integer()
MinValue = integer()
MaxValue = integer()

Equivalent to new(Parent, Id, Value, MinValue, MaxValue, []).

new(Parent, Id, Value, MinValue, MaxValue, Options::[Option]) -> wxSlider()

Types:

Parent = wxWindow:wxWindow()
Id = integer()
Value = integer()
MinValue = integer()
MaxValue = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Value, MinValue, MaxValue) -> boolean()

Types:

This = wxSlider()
wxSlider

Parent = wxWindow:wxWindow()
Id = integer()
Value = integer()
MinValue = integer()
MaxValue = integer()

Equivalent to create(This, Parent, Id, Value, MinValue, MaxValue, []).

create(This, Parent, Id, Value, MinValue, MaxValue, Options::[Option]) -> boolean()

Types:
This = wxSlider()
Parent = wxWindow:wxWindow()
Id = integer()
Value = integer()
MinValue = integer()
MaxValue = integer()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getLineSize(This) -> integer()

Types:
This = wxSlider()

See external documentation.

getMax(This) -> integer()

Types:
This = wxSlider()

See external documentation.

getMin(This) -> integer()

Types:
This = wxSlider()

See external documentation.

getPageSize(This) -> integer()

Types:
This = wxSlider()

See external documentation.

getThumbLength(This) -> integer()

Types:
This = wxSlider()
See external documentation.

```erlang
getValue(This) -> integer()
```
Types:
```
This = wxSlider()
```
See external documentation.

```erlang
setLineSize(This, LineSize) -> ok
```
Types:
```
This = wxSlider()
LineSize = integer()
```
See external documentation.

```erlang
setPageSize(This, PageSize) -> ok
```
Types:
```
This = wxSlider()
PageSize = integer()
```
See external documentation.

```erlang
setRange(This, MinValue, MaxValue) -> ok
```
Types:
```
This = wxSlider()
MinValue = integer()
MaxValue = integer()
```
See external documentation.

```erlang
setThumbLength(This, LenPixels) -> ok
```
Types:
```
This = wxSlider()
LenPixels = integer()
```
See external documentation.

```erlang
setValue(This, Value) -> ok
```
Types:
```
This = wxSlider()
Value = integer()
```
See external documentation.

```erlang
destroy(This::wxSlider()) -> ok
```
Destroys this object, do not use object again
wxSpinButton

Erlang module

See external documentation: `wxSpinButton`.

This class is derived (and can use functions) from:

`wxControl`

`wxWindow`

`wxEvtHandler`

**DATA TYPES**

`wxSpinButton()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxSpinButton()`

See external documentation.

`new(Parent) -> wxSpinButton()`

Types:

```
PARENT = wxWindow:wxWindow()
```

Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxSpinButton()`

Types:

```
PARENT = wxWindow:wxWindow()

OPTION = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
```

See external documentation.

`create(This, Parent) -> boolean()`

Types:

```
THIS = wxSpinButton()

PARENT = wxWindow:wxWindow()
```

Equivalent to `create(This, Parent, [])`.

`create(This, Parent, Options::[Option]) -> boolean()`

Types:

```
THIS = wxSpinButton()

PARENT = wxWindow:wxWindow()
```
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, 
{W::integer(), H::integer()}} | {style, integer()}

See external documentation.

getMax(This) -> integer()
Types:
  This = wxSpinButton()
See external documentation.

getMin(This) -> integer()
Types:
  This = wxSpinButton()
See external documentation.

getValue(This) -> integer()
Types:
  This = wxSpinButton()
See external documentation.

setRange(This, MinVal, MaxVal) -> ok
Types:
  This = wxSpinButton()
  MinVal = integer()
  MaxVal = integer()
See external documentation.

setValue(This, Value) -> ok
Types:
  This = wxSpinButton()
  Value = integer()
See external documentation.

destroy(This::wxSpinButton()) -> ok
Destroys this object, do not use object again
wxSpinCtrl

Erlang module

See external documentation: wxSpinCtrl.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxSpinCtrl()

An object reference, The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxSpinCtrl()

See external documentation.

new(Parent) -> wxSpinCtrl()

Types:

   Parent = wxWindow:wxWindow()

Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxSpinCtrl()

Types:

   Parent = wxWindow:wxWindow()
   Option = {id, integer()} | {value, unicode:chardata()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}}
   | {style, integer()} | {min, integer()} | {max, integer()} | {initial, integer()}

See external documentation.

create(This, Parent) -> boolean()

Types:

   This = wxSpinCtrl()
   Parent = wxWindow:wxWindow()

Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()

Types:

   This = wxSpinCtrl()
   Parent = wxWindow:wxWindow()
Option = {id, integer()} | {value, unicode:chardata()} | {pos, X::integer(), Y::integer()} | {size, W::integer(), H::integer()} |
| {style, integer()} | {min, integer()} | {max, integer()} | {initial, integer()}

See external documentation.

setValue(This, Value) -> ok

Types:
   This = wxSpinCtrl()
   Value = integer()

See external documentation.
Also:
setValue(This, Text) -> 'ok' when
This::wxSpinCtrl(), Text::unicode:chardata().

getValue(This) -> integer()

Types:
   This = wxSpinCtrl()

See external documentation.

setRange(This, MinVal, MaxVal) -> ok

Types:
   This = wxSpinCtrl()
   MinVal = integer()
   MaxVal = integer()

See external documentation.

setSelection(This, From, To) -> ok

Types:
   This = wxSpinCtrl()
   From = integer()
   To = integer()

See external documentation.

getMin(This) -> integer()

Types:
   This = wxSpinCtrl()

See external documentation.

getMax(This) -> integer()

Types:
   This = wxSpinCtrl()

See external documentation.
wxSpinCtrl

destroy(This::wxSpinCtrl()) -> ok
Destroys this object, do not use object again
wxSpinEvent

Erlang module

See external documentation: \texttt{wxSpinEvent}.

Use \texttt{wxEvtHandler:connect/3} with EventType:

\begin{verbatim}
    command_spinctrl_updated, spin_up, spin_down, spin
\end{verbatim}

See also the message variant \texttt{#wxSpin/} event record type.

This class is derived (and can use functions) from:

\begin{verbatim}
    wxNotifyEvent
    wxCommandEvent
    wxEvent
\end{verbatim}

\section*{DATA TYPES}

\begin{verbatim}
wxSpinEvent()
\end{verbatim}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

\section*{Exports}

\begin{verbatim}
getPosition(This) -> integer()
\end{verbatim}

Types:

\begin{verbatim}
    This = wxSpinEvent()
\end{verbatim}

See external documentation.

\begin{verbatim}
setPosition(This, Pos) -> ok
\end{verbatim}

Types:

\begin{verbatim}
    This = wxSpinEvent()
    Pos = integer()
\end{verbatim}

See external documentation.
wxSplashScreen

Erlang module

See external documentation: **wxSplashScreen**.

This class is derived (and can use functions) from:

*wxFrame*
*wxTopLevelWindow*
*wxWindow*
*wxEvtHandler*

**DATA TYPES**

*wxSplashScreen()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

*new() -> wxSplashScreen()*

See external documentation.

*new(Bitmap, SplashStyle, Milliseconds, Parent, Id) -> wxSplashScreen()*

Types:

- **Bitmap** = **wxBitmap::wxBitmap()**
- **SplashStyle** = **integer()**
- **Milliseconds** = **integer()**
- **Parent** = **wxWindow::wxWindow()**
- **Id** = **integer()**

Equivalent to *new(Bitmap, SplashStyle, Milliseconds, Parent, Id, [])*.

*new(Bitmap, SplashStyle, Milliseconds, Parent, Id, Options::[Option]) -> wxSplashScreen()*

Types:

- **Bitmap** = **wxBitmap::wxBitmap()**
- **SplashStyle** = **integer()**
- **Milliseconds** = **integer()**
- **Parent** = **wxWindow::wxWindow()**
- **Id** = **integer()**
- **Option** = **{pos, {X::integer(), Y::integer()}}** | **{size, {W::integer(), H::integer()}}** | **{style, integer()}**

See external documentation.

*getSplashStyle(This) -> integer()*

Types:
This = wxSplashScreen()
See external documentation.

getTimeout(This) -> integer()
Types:
    This = wxSplashScreen()
See external documentation.

destroy(This::wxSplashScreen()) -> ok
Destroys this object, do not use object again
wxSplitterEvent

Erlang module

See external documentation: \texttt{wxSplitterEvent}.

Use \texttt{wxEvtHandler:connect/3} with EventType:

\begin{itemize}
\item \texttt{command\_splitter\_sash\_pos\_changed}.
\item \texttt{command\_splitter\_sash\_pos\_changing}.
\item \texttt{command\_splitter\_doubleclicked}.
\item \texttt{command\_splitter\_unsplit}
\end{itemize}

See also the message variant \texttt{#wxSplitter/1} event record type.

This class is derived (and can use functions) from:

\texttt{wxNotifyEvent}

\texttt{wxCommandEvent}

\texttt{wxEvent}

\section*{DATA TYPES}

\texttt{wxSplitterEvent()}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

\section*{Exports}

\begin{itemize}
\item \texttt{getSashPosition(This) -> integer()}
\item \texttt{getX(This) -> integer()}
\item \texttt{getY(This) -> integer()}
\item \texttt{getWindowBeingRemoved(This) -> \texttt{wxWindow:wxWindow()}}
\item \texttt{setSashPosition(This, Pos) -> ok}
\end{itemize}

Types:

\begin{itemize}
\item \texttt{This = \texttt{wxSplitterEvent()}}
\item \texttt{This = \texttt{wxSplitterEvent()}}
\item \texttt{This = \texttt{wxSplitterEvent()}}
\item \texttt{This = \texttt{wxSplitterEvent()}}
\end{itemize}

See external documentation.
This = wxSplitterEvent()
Pos = integer()
See external documentation.
wxSplitterWindow

Erlang module

See external documentation: wxSplitterWindow.
This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES
wxSplitterWindow()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxSplitterWindow()
See external documentation.

new(Parent) -> wxSplitterWindow()
Types:
    Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxSplitterWindow()
Types:
    Parent = wxWindow:wxWindow()
    Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

create(This, Parent) -> boolean()
Types:
    This = wxSplitterWindow()
    Parent = wxWindow:wxWindow()
Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()
Types:
    This = wxSplitterWindow()
    Parent = wxWindow:wxWindow()
    Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

542 | Ericsson AB. All Rights Reserved.: wxErlang
See external documentation.

getMinimumPaneSize(This) -> integer()
Types:
  This = wxSplitterWindow()
See external documentation.

g getSashGravity(This) -> number()
Types:
  This = wxSplitterWindow()
See external documentation.

g getSashPosition(This) -> integer()
Types:
  This = wxSplitterWindow()
See external documentation.

g getSplitMode(This) -> wx:wx_enum()
Types:
  This = wxSplitterWindow()
See external documentation.
Res = ?wxSPLIT_HORIZONTAL | ?wxSPLIT_VERTICAL

g getWindow1(This) -> wxWindow:wxWindow()
Types:
  This = wxSplitterWindow()
See external documentation.

g getWindow2(This) -> wxWindow:wxWindow()
Types:
  This = wxSplitterWindow()
See external documentation.

initialize(This, Window) -> ok
Types:
  This = wxSplitterWindow()
  Window = wxWindow:wxWindow()
See external documentation.

isSplit(This) -> boolean()
Types:
  This = wxSplitterWindow()
See external documentation.
replaceWindow(This, WinOld, WinNew) -> boolean()
Types:
   This = wxSplitterWindow()
   WinOld = wxWindow:wxWindow()
   WinNew = wxWindow:wxWindow()
See external documentation.

setSashGravity(This, Gravity) -> ok
Types:
   This = wxSplitterWindow()
   Gravity = number()
See external documentation.

setSashPosition(This, Position) -> ok
Types:
   This = wxSplitterWindow()
   Position = integer()
Equivalent to setSashPosition(This, Position, []).

setSashPosition(This, Position, Options::[Option]) -> ok
Types:
   This = wxSplitterWindow()
   Position = integer()
   Option = {redraw, boolean()}
See external documentation.

setSashSize(This, Width) -> ok
Types:
   This = wxSplitterWindow()
   Width = integer()
See external documentation.

setMinimumPaneSize(This, Min) -> ok
Types:
   This = wxSplitterWindow()
   Min = integer()
See external documentation.

setSplitMode(This, Mode) -> ok
Types:
   This = wxSplitterWindow()
   Mode = integer()
See external documentation.
splitHorizontally(This, Window1, Window2) -> boolean()
Types:
  This = wxSplitterWindow()
  Window1 = wxWindow:wxWindow()
  Window2 = wxWindow:wxWindow()
Equivalent to splitHorizontally(This, Window1, Window2, []).

splitHorizontally(This, Window1, Window2, Options::[Option]) -> boolean()
Types:
  This = wxSplitterWindow()
  Window1 = wxWindow:wxWindow()
  Window2 = wxWindow:wxWindow()
  Option = {sashPosition, integer()}
See external documentation.

splitVertically(This, Window1, Window2) -> boolean()
Types:
  This = wxSplitterWindow()
  Window1 = wxWindow:wxWindow()
  Window2 = wxWindow:wxWindow()
Equivalent to splitVertically(This, Window1, Window2, []).

splitVertically(This, Window1, Window2, Options::[Option]) -> boolean()
Types:
  This = wxSplitterWindow()
  Window1 = wxWindow:wxWindow()
  Window2 = wxWindow:wxWindow()
  Option = {sashPosition, integer()}
See external documentation.

unsplit(This) -> boolean()
Types:
  This = wxSplitterWindow()
Equivalent to unsplit(This, []).

unsplit(This, Options::[Option]) -> boolean()
Types:
  This = wxSplitterWindow()
  Option = {toRemove, wxWindow:wxWindow()}
See external documentation.

updateSize(This) -> ok
Types:
This = wxSplitterWindow()

See external documentation.

destroy(This::wxSplitterWindow()) -> ok

Destroys this object, do not use object again
wxStaticBitmap

Erlang module

See external documentation: **wxStaticBitmap**.

This class is derived (and can use functions) from:

*wxControl*
*wxWindow*
*wxEvtHandler*

**DATA TYPES**

**wxStaticBitmap()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

**new() -> wxStaticBitmap**

See external documentation.

**new(Parent, Id, Label) -> wxStaticBitmap**

Types:

- **Parent** = *wxWindow:wxWindow*
- **Id** = integer()
- **Label** = *wxBitmap:wxBitmap*

Equivalent to **new(Parent, Id, Label, [])**.

**new(Parent, Id, Label, Options::[Option]) -> wxStaticBitmap**

Types:

- **Parent** = *wxWindow:wxWindow*
- **Id** = integer()
- **Label** = *wxBitmap:wxBitmap*
- **Option** = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

**create(This, Parent, Id, Label) -> boolean**

Types:

- **This** = *wxStaticBitmap*
- **Parent** = *wxWindow:wxWindow*
- **Id** = integer()
- **Label** = *wxBitmap:wxBitmap*

Equivalent to **create(This, Parent, Id, Label, [])**.
create(This, Parent, Id, Label, Options::[Option]) -> boolean()

Types:

This = wxStaticBitmap()
Parent = wxWindow:wxWindow()
Id = integer()
Label = wxBitmap:wxBitmap()
Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

getBitmap(This) -> wxBitmap:wxBitmap()

Types:

This = wxStaticBitmap()

See external documentation.

setBitmap(This, Bitmap) -> ok

Types:

This = wxStaticBitmap()
Bitmap = wxBitmap:wxBitmap()

See external documentation.

destroy(This::wxStaticBitmap()) -> ok

Destroys this object, do not use object again
**wxStaticBox**

Erlang module

See external documentation: **wxStaticBox**.

This class is derived (and can use functions) from:

- **wxControl**
- **wxWindow**
- **wxEvtHandler**

### DATA TYPES

- **wxStaticBox()**

  An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

### Exports

- **new() -> wxStaticBox()**
  See external documentation.

- **new(Parent, Id, Label) -> wxStaticBox()**
  Types:
  
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`
  - `Label = unicode:chardata()`

  Equivalent to `new(Parent, Id, Label, [])`.

- **new(Parent, Id, Label, Options::[Option]) -> wxStaticBox()**
  Types:
  
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`
  - `Label = unicode:chardata()`
  - `Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

  See external documentation.

- **create(This, Parent, Id, Label) -> boolean()**
  Types:
  
  - `This = wxStaticBox()`
  - `Parent = wxWindow:wxWindow()`
  - `Id = integer()`
  - `Label = unicode:chardata()`

  Equivalent to `create(This, Parent, Id, Label, [])`. 
create(This, Parent, Id, Label, Options::[Option]) -> boolean()

Types:

- **This** = `wxStaticBox()`  
- **Parent** = `wxWindow:wxWindow()`  
- **Id** = `integer()`  
- **Label** = `unicode:chardata()`  
- **Option** = `{pos, {X::integer(), Y::integer()}}` | `{size, {W::integer(), H::integer()}}` | `{style, integer()}`

See external documentation.

destroy(This::wxStaticBox()) -> ok

Destroys this object, do not use object again.
wxStaticBoxSizer

Erlang module

See external documentation: *wxStaticBoxSizer*. This class is derived (and can use functions) from: *wxBoxSizer* *wxSizer*

**DATA TYPES**

*wxStaticBoxSizer()*

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new(Orient, Win) -> wxStaticBoxSizer()

Types:

- Orient = integer()
- Win = wxWindow:wxWindow()

See external documentation. Also:

new(Box, Orient) -> wxStaticBoxSizer() when Box::wxStaticBox:wxStaticBox(), Orient::integer().

call(new(Orient, Win, Options:=[Option])) -> wxStaticBoxSizer()

Types:

- Orient = integer()
- Win = wxWindow:wxWindow()
- Option = {label, unicode:chardata()}

See external documentation.

call(getStaticBox(This)) -> wxStaticBox:wxStaticBox()

Types:

- This = wxStaticBoxSizer()

See external documentation.

call(destroy(This::wxStaticBoxSizer())) -> ok

Destroys this object, do not use object again
wxStaticLine

Erlang module

See external documentation: `wxStaticLine`.

This class is derived (and can use functions) from:

- `wxControl`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

`wxStaticLine()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxStaticLine()

See external documentation.

class new(Parent) -> wxStaticLine()

Types:

- `Parent = wxWindow:wxWindow()`

   Equivalent to `new(Parent, [])`.

class new(Parent, Options::[Option]) -> wxStaticLine()

Types:

- `Parent = wxWindow:wxWindow()`
- `Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

class create(This, Parent) -> boolean()

Types:

- `This = wxStaticLine()`
- `Parent = wxWindow:wxWindow()`

Equivalent to `create(This, Parent, [])`.

class create(This, Parent, Options::[Option]) -> boolean()

Types:

- `This = wxStaticLine()`
- `Parent = wxWindow:wxWindow()`
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

isVertical(This) -> boolean()
Types:
    This = wxStaticLine()
See external documentation.

getDefaultSize() -> integer()
See external documentation.

destroy(This::wxStaticLine()) -> ok
Destroy this object, do not use object again
wxStaticText

Erlang module

See external documentation: **wxStaticText**.

This class is derived (and can use functions) from:

- **wxControl**
- **wxWindow**
- **wxEvtHandler**

### DATA TYPES

**wxStaticText()**

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

### Exports

**new() -> wxStaticText()**

See external documentation.

**new(Parent, Id, Label) -> wxStaticText()**

Types:

- `Parent` = `wxWindow:wxWindow()`
- `Id` = `integer()`
- `Label` = `unicode:chardata()`

Equivalent to `new(Parent, Id, Label, [])`.

**new(Parent, Id, Label, Options::[Option]) -> wxStaticText()**

Types:

- `Parent` = `wxWindow:wxWindow()`
- `Id` = `integer()`
- `Label` = `unicode:chardata()`
- `Option` = `{pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}`

See external documentation.

**create(This, Parent, Id, Label) -> boolean()**

Types:

- `This` = `wxStaticText()`
- `Parent` = `wxWindow:wxWindow()`
- `Id` = `integer()`
- `Label` = `unicode:chardata()`

Equivalent to `create(This, Parent, Id, Label, [])`.

554 | Ericsson AB. All Rights Reserved.: wxErlang
create(This, Parent, Id, Label, Options::[Option]) -> boolean()
Types:
    This = wxStaticText()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Label = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

getLabel(This) -> unicode:charlist()
Types:
    This = wxStaticText()
See external documentation.

setLabel(This, Label) -> ok
Types:
    This = wxStaticText()
    Label = unicode:chardata()
See external documentation.

wrap(This, Width) -> ok
Types:
    This = wxStaticText()
    Width = integer()
See external documentation.

destroy(This::wxStaticText()) -> ok
Destroys this object, do not use object again
wxStatusBar

Erlang module

See external documentation: wxStatusBar.
This class is derived (and can use functions) from:
wxWindow
wxEvtHandler

DATA TYPES

wxStatusBar()

An object reference, The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxStatusBar()
See external documentation.

new(Parent) -> wxStatusBar()
Types:
  Parent = wxWindow:wxWindow()
Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxStatusBar()
Types:
  Parent = wxWindow:wxWindow()
  Option = {winid, integer()} | {style, integer()}
See external documentation.

create(This, Parent) -> boolean()
Types:
  This = wxStatusBar()
  Parent = wxWindow:wxWindow()
Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()
Types:
  This = wxStatusBar()
  Parent = wxWindow:wxWindow()
  Option = {winid, integer()} | {style, integer()}
See external documentation.
getFieldRect(This, I) -> Result
Types:
  Result = {Res::boolean(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}}
  This = wxStatusBar()
  I = integer()
See external documentation.

getFieldsCount(This) -> integer()
Types:
  This = wxStatusBar()
See external documentation.

getStatusText(This) -> unicode:charlist()
Types:
  This = wxStatusBar()
Equivalent to getStatusText(This, []).

getStatusText(This, Options::[Option]) -> unicode:charlist()
Types:
  This = wxStatusBar()
  Option = {number, integer()}
See external documentation.

popStatusText(This) -> ok
Types:
  This = wxStatusBar()
Equivalent to popStatusText(This, []).

popStatusText(This, Options::[Option]) -> ok
Types:
  This = wxStatusBar()
  Option = {number, integer()}
See external documentation.

pushStatusText(This, Text) -> ok
Types:
  This = wxStatusBar()
  Text = unicode:chardata()
Equivalent to pushStatusText(This, Text, []).

pushStatusText(This, Text, Options::[Option]) -> ok
Types:
This is \texttt{wxStatusBar()}
\texttt{Text = unicode:chardata()}
\texttt{Option = \{number, integer()\}}

See external documentation.

\texttt{setFieldsCount(This, Number) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Number = integer()}

Equivalent to \texttt{setFieldsCount(This, Number, [])}.

\texttt{setFieldsCount(This, Number, Options::[Option]) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Number = integer()}
\texttt{Option = \{widths, [integer()]\}}

See external documentation.

\texttt{setMinHeight(This, Height) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Height = integer()}

See external documentation.

\texttt{setStatusText(This, Text) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Text = unicode:chardata()}

Equivalent to \texttt{setStatusText(This, Text, [])}.

\texttt{setStatusText(This, Text, Options::[Option]) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Text = unicode:chardata()}
\texttt{Option = \{number, integer()\}}

See external documentation.

\texttt{setStatusWidths(This, Widths_field) -> ok}
Types:
\texttt{This = wxStatusBar()}
\texttt{Widths_field = [integer()]}
setStatusStyles(This, Styles) -> ok
Types:
   This = wxStatusBar()
   Styles = [integer()]
See external documentation.

destroy(This::wxStatusBar()) -> ok
Destroys this object, do not use object again
wxStdDialogButtonSizer

Erlang module

See external documentation: `wxStdDialogButtonSizer`.
This class is derived (and can use functions) from:
`wxBoxSizer`
`wxSizer`

**DATA TYPES**

`wxStdDialogButtonSizer()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxStdDialogButtonSizer()`

See external documentation.

`addButton(This, Button) -> ok`

Types:

```
This = wxStdDialogButtonSizer()
Button = wxButton:wxButton()
```

See external documentation.

`realize(This) -> ok`

Types:

```
This = wxStdDialogButtonSizer()
```

See external documentation.

`setAffirmativeButton(This, Button) -> ok`

Types:

```
This = wxStdDialogButtonSizer()
Button = wxButton:wxButton()
```

See external documentation.

`setCancelButton(This, Button) -> ok`

Types:

```
This = wxStdDialogButtonSizer()
Button = wxButton:wxButton()
```

See external documentation.
setNegativeButton(This, Button) -> ok
Types:
   This = wxStdDialogButtonSizer()
   Button = wxButton::wxButton()
See external documentation.

destroy(This::wxStdDialogButtonSizer()) -> ok
Destroys this object, do not use object again
wxStyledTextCtrl

Erlang module

See external documentation: wxStyledTextCtrl.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxStyledTextCtrl()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxStyledTextCtrl()

See external documentation.

new(Parent) -> wxStyledTextCtrl()

Types:
   Parent = wxWindow:wxWindow()

Equivalent to new(Parent, []).

new(Parent, Options::[Option]) -> wxStyledTextCtrl()

Types:
   Parent = wxWindow:wxWindow()
   Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

create(This, Parent) -> boolean()

Types:
   This = wxStyledTextCtrl()
   Parent = wxWindow:wxWindow()

Equivalent to create(This, Parent, []).

create(This, Parent, Options::[Option]) -> boolean()

Types:
   This = wxStyledTextCtrl()
   Parent = wxWindow:wxWindow()
Option = \{id, integer()\} | \{pos, \{X::integer(), Y::integer()\}\} | \{size, \{W::integer(), H::integer()\}\} | \{style, integer()\}

See external documentation.

addText(This, Text) -> ok
Types:
   This = \texttt{wxStyledTextCtrl()}
   Text = \texttt{unicode:chardata()}

See external documentation.

addStyledText(This, Data) -> ok
Types:
   This = \texttt{wxStyledTextCtrl()}
   Data = \texttt{wx:wx_object()}

See external documentation.

insertText(This, Pos, Text) -> ok
Types:
   This = \texttt{wxStyledTextCtrl()}
   Pos = integer()
   Text = \texttt{unicode:chardata()}

See external documentation.

clearAll(This) -> ok
Types:
   This = \texttt{wxStyledTextCtrl()}

See external documentation.

clearDocumentStyle(This) -> ok
Types:
   This = \texttt{wxStyledTextCtrl()}

See external documentation.

getLength(This) -> integer()
Types:
   This = \texttt{wxStyledTextCtrl()}

See external documentation.

getCharAt(This, Pos) -> integer()
Types:
   This = \texttt{wxStyledTextCtrl()}
   Pos = integer()

See external documentation.
getCurrentPos(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

getAnchor(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

getStyleAt(This, Pos) -> integer()
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.

redo(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setUndoCollection(This, CollectUndo) -> ok
Types:
   This = wxStyledTextCtrl()
   CollectUndo = boolean()
See external documentation.

selectAll(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setSavePoint(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

getStyledText(This, StartPos, EndPos) -> wx:wx_object()
Types:
   This = wxStyledTextCtrl()
   startPos = integer()
   EndPos = integer()
See external documentation.
canRedo(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.

markerLineFromHandle(This, Handle) -> integer()
Types:
    This = wxStyledTextCtrl()
    Handle = integer()
See external documentation.

markerDeleteHandle(This, Handle) -> ok
Types:
    This = wxStyledTextCtrl()
    Handle = integer()
See external documentation.

getUndoCollection(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.

ggetViewWhiteSpace(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setViewWhiteSpace(This, ViewWS) -> ok
Types:
    This = wxStyledTextCtrl()
    ViewWS = integer()
See external documentation.

positionFromPoint(This, Pt) -> integer()
Types:
    This = wxStyledTextCtrl()
    Pt = {X::integer(), Y::integer()}
See external documentation.

positionFromPointClose(This, X, Y) -> integer()
Types:
    This = wxStyledTextCtrl()
    X = integer()
Y = integer()
See external documentation.

gotoLine(This, Line) -> ok
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

gotoPos(This, Pos) -> ok
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.

setAnchor(This, PosAnchor) -> ok
Types:
   This = wxStyledTextCtrl()
   PosAnchor = integer()
See external documentation.

curLine(This) -> Result
Types:
   Result = {Res::unicode:charlist(), LinePos::integer()}
   This = wxStyledTextCtrl()
See external documentation.

getEndStyled(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

convertEOLs(This, EolMode) -> ok
Types:
   This = wxStyledTextCtrl()
   EolMode = integer()
See external documentation.

eolMode(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.
setEOLMode(This, EolMode) -> ok
Types:
   This = wxStyledTextCtrl()
   EolMode = integer()
See external documentation.

startStyling(This, Pos, Mask) -> ok
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
   Mask = integer()
See external documentation.

setStyling(This, Length, Style) -> ok
Types:
   This = wxStyledTextCtrl()
   Length = integer()
   Style = integer()
See external documentation.

getBufferedDraw(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setBufferedDraw(This, Buffered) -> ok
Types:
   This = wxStyledTextCtrl()
   Buffered = boolean()
See external documentation.

setTabWidth(This, TabWidth) -> ok
Types:
   This = wxStyledTextCtrl()
   TabWidth = integer()
See external documentation.

getTabWidth(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.
wxStyledTextCtrl

setCodePage(This, CodePage) -> ok
Types:
   This = wxStyledTextCtrl()
   CodePage = integer()
See external documentation.

markerDefine(This, MarkerNumber, MarkerSymbol) -> ok
Types:
   This = wxStyledTextCtrl()
   MarkerNumber = integer()
   MarkerSymbol = integer()
Equivalent to markerDefine(This, MarkerNumber, MarkerSymbol, []).

markerDefine(This, MarkerNumber, MarkerSymbol, Options::[Option]) -> ok
Types:
   This = wxStyledTextCtrl()
   MarkerNumber = integer()
   MarkerSymbol = integer()
   Option = {foreground, wx:wx_colour()} | {background, wx:wx_colour()}
See external documentation.

markerSetForeground(This, MarkerNumber, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   MarkerNumber = integer()
   Fore = wx:wx_colour()
See external documentation.

markerSetBackground(This, MarkerNumber, Back) -> ok
Types:
   This = wxStyledTextCtrl()
   MarkerNumber = integer()
   Back = wx:wx_colour()
See external documentation.

markerAdd(This, Line, MarkerNumber) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
   MarkerNumber = integer()
See external documentation.
markerDelete(This, Line, MarkerNumber) -> ok
Types:
  This = wxStyledTextCtrl()
  Line = integer()
  MarkerNumber = integer()
See external documentation.

markerDeleteAll(This, MarkerNumber) -> ok
Types:
  This = wxStyledTextCtrl()
  MarkerNumber = integer()
See external documentation.

markerGet(This, Line) -> integer()
Types:
  This = wxStyledTextCtrl()
  Line = integer()
See external documentation.

markerNext(This, LineStart, MarkerMask) -> integer()
Types:
  This = wxStyledTextCtrl()
  LineStart = integer()
  MarkerMask = integer()
See external documentation.

markerPrevious(This, LineStart, MarkerMask) -> integer()
Types:
  This = wxStyledTextCtrl()
  LineStart = integer()
  MarkerMask = integer()
See external documentation.

markerDefineBitmap(This, MarkerNumber, Bmp) -> ok
Types:
  This = wxStyledTextCtrl()
  MarkerNumber = integer()
  Bmp = wxBitmap:wxBitmap()
See external documentation.

markerAddSet(This, Line, Set) -> ok
Types:
  This = wxStyledTextCtrl()
Line = integer()
Set = integer()
See external documentation.

markerSetAlpha(This, MarkerNumber, Alpha) -> ok
Types:
  This = wxStyledTextCtrl()
  MarkerNumber = integer()
  Alpha = integer()
See external documentation.

setMarginType(This, Margin, MarginType) -> ok
Types:
  This = wxStyledTextCtrl()
  Margin = integer()
  MarginType = integer()
See external documentation.

getMarginType(This, Margin) -> integer()
Types:
  This = wxStyledTextCtrl()
  Margin = integer()
See external documentation.

setMarginWidth(This, Margin, PixelWidth) -> ok
Types:
  This = wxStyledTextCtrl()
  Margin = integer()
  PixelWidth = integer()
See external documentation.

getMarginWidth(This, Margin) -> integer()
Types:
  This = wxStyledTextCtrl()
  Margin = integer()
See external documentation.

setMarginMask(This, Margin, Mask) -> ok
Types:
  This = wxStyledTextCtrl()
  Margin = integer()
  Mask = integer()
See external documentation.
getMarginMask(This, Margin) -> integer()
Types:
    This = wxStyledTextCtrl()
    Margin = integer()
See external documentation.

setMarginSensitive(This, Margin, Sensitive) -> ok
Types:
    This = wxStyledTextCtrl()
    Margin = integer()
    Sensitive = boolean()
See external documentation.

getMarginSensitive(This, Margin) -> boolean()
Types:
    This = wxStyledTextCtrl()
    Margin = integer()
See external documentation.

styleClearAll(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

styleSetForeground(This, Style, Fore) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Fore = wx:wx_colour()
See external documentation.

styleSetBackground(This, Style, Back) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Back = wx:wx_colour()
See external documentation.

styleSetBold(This, Style, Bold) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Bold = boolean()
See external documentation.

styleSetItalic(This, Style, Italic) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Italic = boolean()
See external documentation.

styleSetSize(This, Style, SizePoints) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    SizePoints = integer()
See external documentation.

styleSetFaceName(This, Style, FontName) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    FontName = unicode:chardata()
See external documentation.

styleSetEOLFilled(This, Style, Filled) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Filled = boolean()
See external documentation.

styleResetDefault(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

styleSetUnderline(This, Style, Underline) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Underline = boolean()
See external documentation.
styleSetCase(This, Style, CaseForce) -> ok
Types:
   This = wxStyledTextCtrl()
   Style = integer()
   CaseForce = integer()
See external documentation.

styleSetHotSpot(This, Style, Hotspot) -> ok
Types:
   This = wxStyledTextCtrl()
   Style = integer()
   Hotspot = boolean()
See external documentation.

setSelForeground(This, UseSetting, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   UseSetting = boolean()
   Fore = wx:wx_colour()
See external documentation.

setSelBackground(This, UseSetting, Back) -> ok
Types:
   This = wxStyledTextCtrl()
   UseSetting = boolean()
   Back = wx:wx_colour()
See external documentation.

getSelAlpha(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setSelAlpha(This, Alpha) -> ok
Types:
   This = wxStyledTextCtrl()
   Alpha = integer()
See external documentation.

setCaretForeground(This, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   Fore = wx:wx_colour()
See external documentation.

**cmdKeyAssign**(This, Key, Modifiers, Cmd) -> ok

Types:
   This = wxStyledTextCtrl()
   Key = integer()
   Modifiers = integer()
   Cmd = integer()

See external documentation.

**cmdKeyClear**(This, Key, Modifiers) -> ok

Types:
   This = wxStyledTextCtrl()
   Key = integer()
   Modifiers = integer()

See external documentation.

**cmdKeyClearAll**(This) -> ok

Types:
   This = wxStyledTextCtrl()

See external documentation.

**setStyleBytes**(This, Length) -> integer()

Types:
   This = wxStyledTextCtrl()
   Length = integer()

See external documentation.

**styleSetVisible**(This, Style, Visible) -> ok

Types:
   This = wxStyledTextCtrl()
   Style = integer()
   Visible = boolean()

See external documentation.

**getCaretPeriod**(This) -> integer()

Types:
   This = wxStyledTextCtrl()

See external documentation.

**setCaretPeriod**(This, PeriodMilliseconds) -> ok

Types:
   This = wxStyledTextCtrl()
PeriodMilliseconds = integer()
See external documentation.

setWordChars(This, Characters) -> ok
Types:
   This = wxStyledTextCtrl()
   Characters = unicode:chardata()
See external documentation.

beginUndoAction(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

discardUndoAction(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

indicatorGetStyle(This, Indic) -> integer()
Types:
   This = wxStyledTextCtrl()
   Indic = integer()
See external documentation.

indicatorSetStyle(This, Indic, Style) -> ok
Types:
   This = wxStyledTextCtrl()
   Indic = integer()
   Style = integer()
See external documentation.

indicatorSetForeground(This, Indic, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   Indic = integer()
   Fore = wx:wx_colour()
See external documentation.

indicatorGetForeground(This, Indic) -> wx:wx_colour4()
Types:
   This = wxStyledTextCtrl()
   Indic = integer()
See external documentation.

setWhitespaceForeground(This, UseSetting, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   UseSetting = boolean()
   Fore = wx:wx_colour()
See external documentation.

setWhitespaceBackground(This, UseSetting, Back) -> ok
Types:
   This = wxStyledTextCtrl()
   UseSetting = boolean()
   Back = wx:wx_colour()
See external documentation.

getStyleBits(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setLineState(This, Line, State) -> ok
Types:
   This = wxStyledTextCtrl()
   Line = integer()
   State = integer()
See external documentation.

getLineState(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

getMaxLineState(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

g getCaretLineVisible(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.
setCaretLineVisible(This, Show) -> ok
Types:
  This = wxStyledTextCtrl()
  Show = boolean()
See external documentation.

getCaretLineBackground(This) -> wx:wx_colour4()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setCaretLineBackground(This, Back) -> ok
Types:
  This = wxStyledTextCtrl()
  Back = wx:wx_colour()
See external documentation.

autoCompShow(This, LenEntered, ItemList) -> ok
Types:
  This = wxStyledTextCtrl()
  LenEntered = integer()
  ItemList = unicode:chardata()
See external documentation.

autoCompCancel(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

autoCompActive(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

autoCompPosStart(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

autoCompComplete(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.
autoCompStops(This, CharSet) -> ok
Types:
   This = wxStyledTextCtrl()
   CharSet = unicode:chardata()
See external documentation.

autoCompSetSeparator(This, SeparatorCharacter) -> ok
Types:
   This = wxStyledTextCtrl()
   SeparatorCharacter = integer()
See external documentation.

autoCompGetSeparator(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

autoCompSelect(This, Text) -> ok
Types:
   This = wxStyledTextCtrl()
   Text = unicode:chardata()
See external documentation.

autoCompSetCancelAtStart(This, Cancel) -> ok
Types:
   This = wxStyledTextCtrl()
   Cancel = boolean()
See external documentation.

autoCompGetCancelAtStart(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

autoCompSetFillUps(This, CharSet) -> ok
Types:
   This = wxStyledTextCtrl()
   CharSet = unicode:chardata()
See external documentation.

autoCompSetChooseSingle(This, ChooseSingle) -> ok
Types:
   This = wxStyledTextCtrl()
ChooseSingle = boolean()
See external documentation.

autoCompGetChooseSingle(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

autoCompSetIgnoreCase(This, IgnoreCase) -> ok
Types:
   This = wxStyledTextCtrl()
   IgnoreCase = boolean()
See external documentation.

autoCompGetIgnoreCase(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

userListShow(This, ListType, ItemList) -> ok
Types:
   This = wxStyledTextCtrl()
   ListType = integer()
   ItemList = unicode:chardata()
See external documentation.

autoCompSetAutoHide(This, AutoHide) -> ok
Types:
   This = wxStyledTextCtrl()
   AutoHide = boolean()
See external documentation.

autoCompGetAutoHide(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

autoCompSetDropRestOfWord(This, DropRestOfWord) -> ok
Types:
   This = wxStyledTextCtrl()
   DropRestOfWord = boolean()
See external documentation.
autoCompGetDropRestOfWord(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.

registerImage(This, Type, Bmp) -> ok
Types:
    This = wxStyledTextCtrl()
    Type = integer()
    Bmp = wxBitmap:wxBitmap()
See external documentation.

clearRegisteredImages(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

autoCompGetTypeSeparator(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

autoCompSetTypeSeparator(This, SeparatorCharacter) -> ok
Types:
    This = wxStyledTextCtrl()
    SeparatorCharacter = integer()
See external documentation.

autoCompSetMaxWidth(This, CharacterCount) -> ok
Types:
    This = wxStyledTextCtrl()
    CharacterCount = integer()
See external documentation.

autoCompGetMaxWidth(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

autoCompSetMaxHeight(This, RowCount) -> ok
Types:
    This = wxStyledTextCtrl()
    RowCount = integer()
autoCompGetMaxHeight(This) -> integer()
Types:

    This =$\textit{wxStyledTextCtrl}()$

See external documentation.

setIndent(This, IndentSize) -> ok
Types:

    This =$\textit{wxStyledTextCtrl}()$
    IndentSize = integer()

See external documentation.

getIndent(This) -> integer()
Types:

    This =$\textit{wxStyledTextCtrl}()$

See external documentation.

setUseTabs(This, UseTabs) -> ok
Types:

    This =$\textit{wxStyledTextCtrl}()$
    UseTabs = boolean()

See external documentation.

getUseTabs(This) -> boolean()
Types:

    This =$\textit{wxStyledTextCtrl}()$

See external documentation.

setLineIndentation(This, Line, IndentSize) -> ok
Types:

    This =$\textit{wxStyledTextCtrl}()$
    Line = integer()
    IndentSize = integer()

See external documentation.

getLineIndentation(This, Line) -> integer()
Types:

    This =$\textit{wxStyledTextCtrl}()$
    Line = integer()

See external documentation.
getLineIndentPosition(This, Line) -> integer()
Types:
  This = wxStyledTextCtrl()
  Line = integer()
See external documentation.

column(This, Pos) -> integer()
Types:
  This = wxStyledTextCtrl()
  Pos = integer()
See external documentation.

setUseHorizontalScrollBar(This, Show) -> ok
Types:
  This = wxStyledTextCtrl()
  Show = boolean()
See external documentation.

getUseHorizontalScrollBar(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setIndentationGuides(This, Show) -> ok
Types:
  This = wxStyledTextCtrl()
  Show = boolean()
See external documentation.

getIndentationGuides(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setHighlightGuide(This, Column) -> ok
Types:
  This = wxStyledTextCtrl()
  Column = integer()
See external documentation.

getHighlightGuide(This) -> integer()
Types:
  This = wxStyledTextCtrl()
getLineEndPosition(This, Line) -> integer()
Types:
    This = wxStyledTextCtrl()
    Line = integer()
See external documentation.

codePage(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

caretForeground(This) -> wx:wx_colour4()
Types:
    This = wxStyledTextCtrl()
See external documentation.

readOnly(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.

currentPos(This, Pos) -> ok
Types:
    This = wxStyledTextCtrl()
    Pos = integer()
See external documentation.

selectionStart(This, Pos) -> ok
Types:
    This = wxStyledTextCtrl()
    Pos = integer()
See external documentation.

selectionStart(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

selectionEnd(This, Pos) -> ok
Types:
    This = wxStyledTextCtrl()
wxStyledTextCtrl

Pos = integer()
See external documentation.

getSelectionEnd(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setPrintMagnification(This, Magnification) -> ok
Types:
  This = wxStyledTextCtrl()
  Magnification = integer()
See external documentation.

getPrintMagnification(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setPrintColourMode(This, Mode) -> ok
Types:
  This = wxStyledTextCtrl()
  Mode = integer()
See external documentation.

getPrintColourMode(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

findText(This, MinPos, MaxPos, Text) -> integer()
Types:
  This = wxStyledTextCtrl()
  MinPos = integer()
  MaxPos = integer()
  Text = unicode:chardata()
Equivalent to findText(This, MinPos, MaxPos, Text, []).

findText(This, MinPos, MaxPos, Text, Options::[Option]) -> integer()
Types:
  This = wxStyledTextCtrl()
  MinPos = integer()
  MaxPos = integer()
Text = unicode:chardata()
Option = {flags, integer()}

See external documentation.

formatRange(This, DoDraw, StartPos, EndPos, Draw, Target, RenderRect, PageRect) -> integer()
Types:
   This = wXStyledTextCtrl()
   DoDraw = boolean()
   StartPos = integer()
   EndPos = integer()
   Draw = wxDC:wxDC()
   Target = wxDC:wxDC()
   RenderRect = {X::integer(), Y::integer(), W::integer(), H::integer()}
   PageRect = {X::integer(), Y::integer(), W::integer(), H::integer()}

See external documentation.

getFirstVisibleLine(This) -> integer()
Types:
   This = wXStyledTextCtrl()

See external documentation.

getLine(This, Line) -> unicode:charlist()
Types:
   This = wXStyledTextCtrl()
   Line = integer()

See external documentation.

g.getLineCount(This) -> integer()
Types:
   This = wXStyledTextCtrl()

See external documentation.

setMarginLeft(This, PixelWidth) -> ok
Types:
   This = wXStyledTextCtrl()
   PixelWidth = integer()

See external documentation.

getMarginLeft(This) -> integer()
Types:
   This = wXStyledTextCtrl()

See external documentation.
setMarginRight(This, PixelWidth) -> ok
Types:
   This = wxStyledTextCtrl()
   PixelWidth = integer()
See external documentation.

getMarginRight(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

getModify(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setSelection(This, Start, End) -> ok
Types:
   This = wxStyledTextCtrl()
   Start = integer()
   End = integer()
See external documentation.

getSelectedText(This) -> unicode:charlist()
Types:
   This = wxStyledTextCtrl()
See external documentation.

getTextRange(This, StartPos, EndPos) -> unicode:charlist()
Types:
   This = wxStyledTextCtrl()
   StartPos = integer()
   EndPos = integer()
See external documentation.

hideSelection(This, Normal) -> ok
Types:
   This = wxStyledTextCtrl()
   Normal = boolean()
See external documentation.

lineFromPosition(This, Pos) -> integer()
Types:
This = wxStyledTextCtrl()
Pos = integer()
See external documentation.

positionFromLine(This, Line) -> integer()
Types:
  This = wxStyledTextCtrl()  
  Line = integer()
See external documentation.

lineScroll(This, Columns, Lines) -> ok
Types:
  This = wxStyledTextCtrl()  
  Columns = integer()  
  Lines = integer()
See external documentation.

ensureCaretVisible(This) -> ok
Types:
  This = wxStyledTextCtrl()  
See external documentation.

replaceSelection(This, Text) -> ok
Types:
  This = wxStyledTextCtrl()  
  Text = unicode:chardata()
See external documentation.

setReadOnly(This, ReadOnly) -> ok
Types:
  This = wxStyledTextCtrl()  
  ReadOnly = boolean()
See external documentation.

canPaste(This) -> boolean()
Types:
  This = wxStyledTextCtrl()  
See external documentation.

canUndo(This) -> boolean()
Types:
  This = wxStyledTextCtrl()  
See external documentation.
emptyUndoBuffer(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

undo(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

cut(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

copy(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

paste(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

clear(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

setText(This, Text) -> ok
Types:
    This = wxStyledTextCtrl()
    Text = unicode:chardata()
See external documentation.

getText(This) -> unicode:charlist()
Types:
    This = wxStyledTextCtrl()
See external documentation.

getTextLength(This) -> integer()
Types:
This = wxStyledTextCtrl()
See external documentation.

getOvertypes(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setCaretWidth(This, PixelWidth) -> ok
Types:
    This = wxStyledTextCtrl()
    PixelWidth = integer()
See external documentation.

g getCaretWidth(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setTargetStart(This, Pos) -> ok
Types:
    This = wxStyledTextCtrl()
    Pos = integer()
See external documentation.

getTargetStart(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setTargetEnd(This, Pos) -> ok
Types:
    This = wxStyledTextCtrl()
    Pos = integer()
See external documentation.

getTargetEnd(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

replaceTarget(This, Text) -> integer()
Types:
wxStyledTextCtrl

This = wxStyledTextCtrl()
Text = unicode:chardata()
See external documentation.

searchInTarget(This, Text) -> integer()
Types:
   This = wxStyledTextCtrl()
   Text = unicode:chardata()
See external documentation.

setSearchFlags(This, Flags) -> ok
Types:
   This = wxStyledTextCtrl()
   Flags = integer()
See external documentation.

getSearchFlags(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

callTipShow(This, Pos, Definition) -> ok
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
   Definition = unicode:chardata()
See external documentation.

callTipCancel(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

callTipActive(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

callTipPosAtStart(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.
callTipSetHighlight(This, Start, End) -> ok
Types:
   This = wxStyledTextCtrl()
   Start = integer()
   End = integer()
See external documentation.

callTipSetBackground(This, Back) -> ok
Types:
   This = wxStyledTextCtrl()
   Back = wx:wx_colour()
See external documentation.

callTipSetForeground(This, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   Fore = wx:wx_colour()
See external documentation.

callTipSetForegroundHighlight(This, Fore) -> ok
Types:
   This = wxStyledTextCtrl()
   Fore = wx:wx_colour()
See external documentation.

callTipUseStyle(This, TabSize) -> ok
Types:
   This = wxStyledTextCtrl()
   TabSize = integer()
See external documentation.

visibleFromDocLine(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

docLineFromVisible(This, LineDisplay) -> integer()
Types:
   This = wxStyledTextCtrl()
   LineDisplay = integer()
See external documentation.
wxStyledTextCtrl

wrapCount(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

setFoldLevel(This, Line, Level) -> ok
Types:
   This = wxStyledTextCtrl()
   Line = integer()
   Level = integer()
See external documentation.

getFoldLevel(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

getLastChild(This, Line, Level) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
   Level = integer()
See external documentation.

getFoldParent(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

showLines(This, LineStart, LineEnd) -> ok
Types:
   This = wxStyledTextCtrl()
   LineStart = integer()
   LineEnd = integer()
See external documentation.

hideLines(This, LineStart, LineEnd) -> ok
Types:
   This = wxStyledTextCtrl()
   LineStart = integer()
LineEnd = integer()

See external documentation.

getLineVisible(This, Line) -> boolean()
Types:
  This = wxStyledTextCtrl()
  Line = integer()

See external documentation.

setFoldExpanded(This, Line, Expanded) -> ok
Types:
  This = wxStyledTextCtrl()
  Line = integer()
  Expanded = boolean()

See external documentation.

getFoldExpanded(This, Line) -> boolean()
Types:
  This = wxStyledTextCtrl()
  Line = integer()

See external documentation.

toggleFold(This, Line) -> ok
Types:
  This = wxStyledTextCtrl()
  Line = integer()

See external documentation.

ensureVisible(This, Line) -> ok
Types:
  This = wxStyledTextCtrl()
  Line = integer()

See external documentation.

setFoldFlags(This, Flags) -> ok
Types:
  This = wxStyledTextCtrl()
  Flags = integer()

See external documentation.

ensureVisibleEnforcePolicy(This, Line) -> ok
Types:
  This = wxStyledTextCtrl()
Line = integer()
See external documentation.

setTabIndents(This, TabIndents) -> ok
Types:
  This = wxStyledTextCtrl()
  TabIndents = boolean()
See external documentation.

getTabIndents(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setBackSpaceUnIndents(This, BsUnIndents) -> ok
Types:
  This = wxStyledTextCtrl()
  BsUnIndents = boolean()
See external documentation.

getBackSpaceUnIndents(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setMouseDwellTime(This, PeriodMilliseconds) -> ok
Types:
  This = wxStyledTextCtrl()
  PeriodMilliseconds = integer()
See external documentation.

getMouseDwellTime(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

wordStartPosition(This, Pos, OnlyWordCharacters) -> integer()
Types:
  This = wxStyledTextCtrl()
  Pos = integer()
  OnlyWordCharacters = boolean()
See external documentation.
wordEndPosition(This, Pos, OnlyWordCharacters) -> integer()
Types:
  This = wxStyledTextCtrl()
  Pos = integer()
  OnlyWordCharacters = boolean()
See external documentation.

setWrapMode(This, Mode) -> ok
Types:
  This = wxStyledTextCtrl()
  Mode = integer()
See external documentation.

getWrapMode(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setWrapVisualFlags(This, WrapVisualFlags) -> ok
Types:
  This = wxStyledTextCtrl()
  WrapVisualFlags = integer()
See external documentation.

getWrapVisualFlags(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setWrapVisualFlagsLocation(This, WrapVisualFlagsLocation) -> ok
Types:
  This = wxStyledTextCtrl()
  WrapVisualFlagsLocation = integer()
See external documentation.

getWrapVisualFlagsLocation(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setWrapStartIndent(This, Indent) -> ok
Types:
  This = wxStyledTextCtrl()
Indent = integer()
See external documentation.

getWrapStartIndent(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setLayoutCache(This, Mode) -> ok
Types:
    This = wxStyledTextCtrl()
    Mode = integer()
See external documentation.

getLayoutCache(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setScrollWidth(This, PixelWidth) -> ok
Types:
    This = wxStyledTextCtrl()
    PixelWidth = integer()
See external documentation.

getScrollWidth(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

textWidth(This, Style, Text) -> integer()
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Text = unicode:chardata()
See external documentation.

getEndAtLastLine(This) -> boolean()
Types:
    This = wxStyledTextCtrl()
See external documentation.
textHeight(This, Line) -> integer()

Types:
   This = wxStyledTextCtrl()
   Line = integer()

See external documentation.

setUseVerticalScrollBar(This, Show) -> ok

Types:
   This = wxStyledTextCtrl()
   Show = boolean()

See external documentation.

appendText(This, Text) -> ok

Types:
   This = wxStyledTextCtrl()
   Text = unicode:chardata()

See external documentation.

getTwoPhaseDraw(This) -> boolean()

Types:
   This = wxStyledTextCtrl()

See external documentation.

setTwoPhaseDraw(This, TwoPhase) -> ok

Types:
   This = wxStyledTextCtrl()
   TwoPhase = boolean()

See external documentation.

targetFromSelection(This) -> ok

Types:
   This = wxStyledTextCtrl()

See external documentation.

linesJoin(This) -> ok

Types:
   This = wxStyledTextCtrl()

See external documentation.
linesSplit(This, PixelWidth) -> ok
Types:
    This = wxStyledTextCtrl()
    PixelWidth = integer()
See external documentation.

setFoldMarginColour(This, UseSetting, Back) -> ok
Types:
    This = wxStyledTextCtrl()
    UseSetting = boolean()
    Back = wx:wx_colour()
See external documentation.

setFoldMarginHiColour(This, UseSetting, Fore) -> ok
Types:
    This = wxStyledTextCtrl()
    UseSetting = boolean()
    Fore = wx:wx_colour()
See external documentation.

textDown(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

textDownExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

textUp(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

textUpExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

textLeft(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

charLeftExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

charRight(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

charRightExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordLeft(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordLeftExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordRight(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordRightExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

home(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.
wxStyledTextCtrl

homeExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

lineEnd(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

lineEndExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

documentStart(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

documentStartExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

documentEnd(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

documentEndExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

pageUp(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

pageUpExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

**pageDown(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**pageDownExtend(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**editToggleOvertType(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**cancel(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**deleteBack(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**tab(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**backTab(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.

**newLine(This) -> ok**
Types:

\[
\text{This} = \text{wxStyledTextCtrl()}
\]
See external documentation.
formFeed(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

vCHome(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

vCHomeExtend(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

zoomIn(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

zoomOut(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

delWordLeft(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

delWordRight(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineCut(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineDelete(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineTranspose(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineDuplicate(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lowerCase(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

upperCase(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineScrollDown(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

lineScrollUp(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

deleteBackNotLine(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.

homeDisplay(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.
homeDisplayExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeLineEndDisplay(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeLineEndDisplayExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeHomeWrapExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeLineEndWrap(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeLineEndWrapExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeVCHomeWrap(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeVCHomeWrapExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

typeLineCopy(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

moveCaretInsideView(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

lineLength(This, Line) -> integer()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

braceHighlight(This, Pos1, Pos2) -> ok
Types:
   This = wxStyledTextCtrl()
   Pos1 = integer()
   Pos2 = integer()
See external documentation.

braceBadLight(This, Pos) -> ok
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.

braceMatch(This, Pos) -> integer()
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.

ggetViewEOL(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setViewEOL(This, Visible) -> ok
Types:
   This = wxStyledTextCtrl()
   Visible = boolean()
See external documentation.
```erlang
wxStyledTextCtrl

setModEventMask(This, Mask) -> ok
Types:
  This = wxStyledTextCtrl()
  Mask = integer()
See external documentation.

getEdgeColumn(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setEdgeColumn(This, Column) -> ok
Types:
  This = wxStyledTextCtrl()
  Column = integer()
See external documentation.

setEdgeMode(This, Mode) -> ok
Types:
  This = wxStyledTextCtrl()
  Mode = integer()
See external documentation.

getEdgeMode(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

getEdgeColour(This) -> wx:wx_colour4()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setEdgeColour(This, EdgeColour) -> ok
Types:
  This = wxStyledTextCtrl()
  EdgeColour = wx:wx_colour()
See external documentation.

searchAnchor(This) -> ok
Types:
  This = wxStyledTextCtrl()
See external documentation.
```
wxStyledTextCtrl

searchNext(This, Flags, Text) -> integer()
Types:
  This = wxStyledTextCtrl()
  Flags = integer()
  Text = unicode:chardata()
See external documentation.

searchPrev(This, Flags, Text) -> integer()
Types:
  This = wxStyledTextCtrl()
  Flags = integer()
  Text = unicode:chardata()
See external documentation.

linesOnScreen(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

usePopUp(This, AllowPopUp) -> ok
Types:
  This = wxStyledTextCtrl()
  AllowPopUp = boolean()
See external documentation.

selectionIsRectangle(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setZoom(This, Zoom) -> ok
Types:
  This = wxStyledTextCtrl()
  Zoom = integer()
See external documentation.

gerZoom(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

getModEventMask(This) -> integer()
Types:
wxStyledTextCtrl

This = wxStyledTextCtrl()
See external documentation.

setSTCFocus(This, Focus) -> ok
Types:
  This = wxStyledTextCtrl()
  Focus = boolean()
See external documentation.

getSTCFocus(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setStatus(This, StatusCode) -> ok
Types:
  This = wxStyledTextCtrl()
  StatusCode = integer()
See external documentation.

getStatus(This) -> integer()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setMouseDownCaptures(This, Captures) -> ok
Types:
  This = wxStyledTextCtrl()
  Captures = boolean()
See external documentation.

getMouseDownCaptures(This) -> boolean()
Types:
  This = wxStyledTextCtrl()
See external documentation.

setSTCCursor(This, CursorType) -> ok
Types:
  This = wxStyledTextCtrl()
  CursorType = integer()
See external documentation.
getSTCCursor(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setControlCharSymbol(This, Symbol) -> ok
Types:
   This = wxStyledTextCtrl()
   Symbol = integer()
See external documentation.

getControlCharSymbol(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordPartLeft(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordPartLeftExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordPartRight(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordPartRightExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setVisiblePolicy(This, VisiblePolicy, VisibleSlop) -> ok
Types:
   This = wxStyledTextCtrl()
   VisiblePolicy = integer()
   VisibleSlop = integer()
See external documentation.
wxStyledTextCtrl

delLineLeft(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

delLineRight(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

getXOffset(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

chooseCaretX(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

setXCaretPolicy(This, CaretPolicy, CaretSlop) -> ok
Types:
    This = wxStyledTextCtrl()
    CaretPolicy = integer()
    CaretSlop = integer()
See external documentation.

setYCaretPolicy(This, CaretPolicy, CaretSlop) -> ok
Types:
    This = wxStyledTextCtrl()
    CaretPolicy = integer()
    CaretSlop = integer()
See external documentation.

getPrintWrapMode(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

setHotspotActiveForeground(This, UseSetting, Fore) -> ok
Types:
    This = wxStyledTextCtrl()
    UseSetting = boolean()
Fore = wx:wx_colour()
See external documentation.

setHotspotActiveBackground(This, UseSetting, Back) -> ok
Types:
   This = wxStyledTextCtrl()
   UseSetting = boolean()
   Back = wx:wx_colour()
See external documentation.

setHotspotActiveUnderline(This, Underline) -> ok
Types:
   This = wxStyledTextCtrl()
   Underline = boolean()
See external documentation.

setHotspotSingleLine(This, SingleLine) -> ok
Types:
   This = wxStyledTextCtrl()
   SingleLine = boolean()
See external documentation.

paraDownExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

paraUp(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

paraUpExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

positionBefore(This, Pos) -> integer()
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.
wxStyledTextCtrl

positionAfter(This, Pos) -> integer()
Types:
    This = wxStyledTextCtrl()
    Pos = integer()
See external documentation.

copyRange(This, Start, End) -> ok
Types:
    This = wxStyledTextCtrl()
    Start = integer()
    End = integer()
See external documentation.

copyText(This, Length, Text) -> ok
Types:
    This = wxStyledTextCtrl()
    Length = integer()
    Text = unicode:chardata()
See external documentation.

setSelectionMode(This, Mode) -> ok
Types:
    This = wxStyledTextCtrl()
    Mode = integer()
See external documentation.

getSelectionMode(This) -> integer()
Types:
    This = wxStyledTextCtrl()
See external documentation.

lineDownRectExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

lineUpRectExtend(This) -> ok
Types:
    This = wxStyledTextCtrl()
See external documentation.

charLeftRectExtend(This) -> ok
Types:
**wxStyledTextCtrl**

This = wxStyledTextCtrl()
See external documentation.

charRightRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

homeRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

vCHomeRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

lineEndRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

pageUpRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

pageDownRectExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

stutteredPageUp(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

stutteredPageUpExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.
stutteredPageDown(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

stutteredPageDownExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordLeftEnd(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordLeftEndExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordRightEnd(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

wordRightEndExtend(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setWhitespaceChars(This, Characters) -> ok
Types:
   This = wxStyledTextCtrl()
   Characters = unicode:chardata()
See external documentation.

setCharsDefault(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

autoCompGetCurrent(This) -> integer()
Types:
This = wxstyledTextCtrl()
See external documentation.

allocate(This, Bytes) -> ok
Types:
   This = wxstyledTextCtrl()
   Bytes = integer()
See external documentation.

findColumn(This, Line, Column) -> integer()
Types:
   This = wxstyledTextCtrl()
   Line = integer()
   Column = integer()
See external documentation.

getcaretSticky(This) -> boolean()
Types:
   This = wxstyledTextCtrl()
See external documentation.

setCaretSticky(This, UseCaretStickyBehaviour) -> ok
Types:
   This = wxstyledTextCtrl()
   UseCaretStickyBehaviour = boolean()
See external documentation.

toggleCaretSticky(This) -> ok
Types:
   This = wxstyledTextCtrl()
See external documentation.

setPasteConvertEndings(This, Convert) -> ok
Types:
   This = wxstyledTextCtrl()
   Convert = boolean()
See external documentation.

getc PasteConvertEndings(This) -> boolean()
Types:
   This = wxstyledTextCtrl()
See external documentation.
wxStyledTextCtrl

selectionDuplicate(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setCaretLineBackAlpha(This, Alpha) -> ok
Types:
   This = wxStyledTextCtrl()
   Alpha = integer()
See external documentation.

caretLineBackAlpha(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

startRecord(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

stopRecord(This) -> ok
Types:
   This = wxStyledTextCtrl()
See external documentation.

setLexer(This, Lexer) -> ok
Types:
   This = wxStyledTextCtrl()
   Lexer = integer()
See external documentation.

getLexer(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

colourise(This, Start, End) -> ok
Types:
   This = wxStyledTextCtrl()
   Start = integer()
   End = integer()
See external documentation.
setProperty(This, Key, Value) -> ok
Types:
   This = wxStyledTextCtrl()
   Key = unicode:chardata()
   Value = unicode:chardata()
See external documentation.

setKeyWords(This, KeywordSet, KeyWords) -> ok
Types:
   This = wxStyledTextCtrl()
   KeywordSet = integer()
   KeyWords = unicode:chardata()
See external documentation.

setLexerLanguage(This, Language) -> ok
Types:
   This = wxStyledTextCtrl()
   Language = unicode:chardata()
See external documentation.

getProperty(This, Key) -> unicode:charlist()
Types:
   This = wxStyledTextCtrl()
   Key = unicode:chardata()
See external documentation.

getStyleBitsNeeded(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

currentLine(This) -> integer()
Types:
   This = wxStyledTextCtrl()
See external documentation.

styleSetSpec(This, StyleNum, Spec) -> ok
Types:
   This = wxStyledTextCtrl()
   StyleNum = integer()
   Spec = unicode:chardata()
See external documentation.
wxStyledTextCtrl

styleSetFont(This, StyleNum, Font) -> ok
Types:
  This = wxStyledTextCtrl()
  StyleNum = integer()
  Font = wxFont::wxFont()
See external documentation.

styleSetFontAttr(This, StyleNum, Size, FaceName, Bold, Italic, Underline) -> ok
Types:
  This = wxStyledTextCtrl()
  StyleNum = integer()
  Size = integer()
  FaceName = unicode:chardata()
  Bold = boolean()
  Italic = boolean()
  Underline = boolean()
Equivalent to styleSetFontAttr(This, StyleNum, Size, FaceName, Bold, Italic, Underline, []).

styleSetFontAttr(This, StyleNum, Size, FaceName, Bold, Italic, Underline, Options:::[Option]) -> ok
Types:
  This = wxStyledTextCtrl()
  StyleNum = integer()
  Size = integer()
  FaceName = unicode:chardata()
  Bold = boolean()
  Italic = boolean()
  Underline = boolean()
  Option = {encoding, wx::wx_enum()}
See external documentation.
Encoding = ?wxFONTENCODING_SYSTEM | ?wxFONTENCODING_DEFAULT | ?
wxFONTENCODING_ISO8859_1 | ?wxFONTENCODING_ISO8859_2 | ?wxFONTENCODING_ISO8859_3 | ?
wxFONTENCODING_ISO8859_4 | ?wxFONTENCODING_ISO8859_5 | ?wxFONTENCODING_ISO8859_6 | ?
wxFONTENCODING_ISO8859_7 | ?wxFONTENCODING_ISO8859_8 | ?wxFONTENCODING_ISO8859_9 | ?
| ?wxFONTENCODING_ISO8859_MAX | ?wxFONTENCODING_KOI8 | ?wxFONTENCODING_KOI8_U | |
| | ?wxFONTENCODING_ALTERNATIVE | | ?wxFONTENCODING_BULGARIAN | | ?
wxFONTENCODING_CP437 | ?wxFONTENCODING_CP850 | ?wxFONTENCODING_CP852 | |
wxFONTENCODING_CP855 | ?wxFONTENCODING_CP866 | ?wxFONTENCODING_CP874 | |
wxFONTENCODING_CP932 | ?wxFONTENCODING_CP936 | ?wxFONTENCODING_CP949 | |
wxFONTENCODING_CP950 | ?wxFONTENCODING_CP1250 | ?wxFONTENCODING_CP1251 | |
wxFONTENCODING_CP1252 | ?wxFONTENCODING_CP1253 | ?wxFONTENCODING_CP1254 | |
wxFONTENCODING_CP1255 | ?wxFONTENCODING_CP1256 | ?wxFONTENCODING_CP1257 | |
wxFONTENCODING_CP12_MAX | | ?wxFONTENCODING_UTF7 | | ?wxFONTENCODING_UTF8 | |
| ?wxFontEncoding_MACJAPANESE | ?wxFontEncoding_MACCHINESEETRAD | ?
wxFontEncoding_MACKOREAN | ?wxFontEncoding_MACARABIC | ?
wxFontEncoding_MACHEBREW | ?wxFontEncoding_MACGREEK | ?
wxFontEncoding_MACCYRILLIC | ?wxFontEncoding_MACDEVANAGARI | ?
wxFontEncoding_MACGURMUKHI | ?wxFontEncoding_MACGUJARATI | ?
wxFontEncoding_MACORIYA | ?wxFontEncoding_MACBENGALI | ?
wxFontEncoding_MACTAMIL | ?wxFontEncoding_MACTELUGU | ?
wxFontEncoding_MACKANNADA | ?wxFontEncoding_MACMALAYALAM | ?
wxFontEncoding_MACSINHALA SE | ?wxFontEncoding_MACBURMESE | ?
wxFontEncoding_MACKHMER | ?wxFontEncoding_MACTHAI | ?
wxFontEncoding_MACLAOTIAN | ?wxFontEncoding_MACGEORGIAN | ?
wxFontEncoding_MACARMENIAN | ?wxFontEncoding_MACCHINESESIMP | ?
wxFontEncoding_MACTIBETAN | ?wxFontEncoding_MACMONGOLIAN | ?
wxFontEncoding_MACETHIOPIAN | ?wxFontEncoding_MACCARETSK | ?
wxFontEncoding_MACVIETNAMESE | ?wxFontEncoding_MACARABIC | ?
wxFontEncoding_MACSYMBOL | ?wxFontEncoding_MACDINGBATS | ?
wxFontEncoding_MACHEBREW | ?wxFontEncoding_MACALT | ?
wxFontEncoding_MACCELTIC | ?wxFontEncoding_MACCELTIC | ?
wxFontEncoding_MACKEYBOARD | ?wxFontEncoding_MACMAX | ?wxFontEncoding_MACMIN
wxFontEncoding_UNICODE | ?wxFontEncoding_GB2312 | ?wxFontEncoding_BIG5 | ?

styleCharacterSet(This, Style, CharacterSet) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    CharacterSet = integer()
See external documentation.

styleSetFontEncoding(This, Style, Encoding) -> ok
Types:
    This = wxStyledTextCtrl()
    Style = integer()
    Encoding = wx:wx_enum()
See external documentation.

Encoding = ?wxFontEncoding_SYSTEM | ?wxFontEncoding_DEFAULT | ?
wxFontEncoding_ISO8859_1 | ?wxFontEncoding_ISO8859_2 | ?wxFontEncoding_ISO8859_3 | ?
wxFontEncoding_ISO8859_4 | ?wxFontEncoding_ISO8859_5 | ?wxFontEncoding_ISO8859_6 | ?
| ?wxFontEncoding_ALTERNATIVE | ?wxFontEncoding_BULGARIAN | ?
wxFontEncoding_CP437 | ?wxFontEncoding_CP850 | ?wxFontEncoding_CP852 | ?
wxFontEncoding_CP855 | ?wxFontEncoding_CP866 | ?wxFontEncoding_CP874 | ?
wxStyledTextCtrl

**cmdKeyExecute(This, Cmd) -> ok**

Types:
- **This** = `wxStyledTextCtrl`
- **Cmd** = `integer`

See external documentation.

**setMargins(This, Left, Right) -> ok**

Types:
- **This** = `wxStyledTextCtrl`
- **Left** = `integer`
- **Right** = `integer`

See external documentation.

**getSelection(This) -> {StartPos::integer(), EndPos::integer()}**

Types:
- **This** = `wxStyledTextCtrl`

See external documentation.
pointFromPosition(This, Pos) -> {X::integer(), Y::integer()}
Types:
   This = wxStyledTextCtrl()
   Pos = integer()
See external documentation.

scrollToLine(This, Line) -> ok
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

scrollToColumn(This, Column) -> ok
Types:
   This = wxStyledTextCtrl()
   Column = integer()
See external documentation.

setVScrollBar(This, Bar) -> ok
Types:
   This = wxStyledTextCtrl()
   Bar = wxScrollBar:wxScrollBar()
See external documentation.

setHScrollBar(This, Bar) -> ok
Types:
   This = wxStyledTextCtrl()
   Bar = wxScrollBar:wxScrollBar()
See external documentation.

getLastKeydownProcessed(This) -> boolean()
Types:
   This = wxStyledTextCtrl()
See external documentation.

setLastKeydownProcessed(This, Val) -> ok
Types:
   This = wxStyledTextCtrl()
   Val = boolean()
See external documentation.

saveFile(This, Filename) -> boolean()
Types:
wxStyledTextCtrl

This = wxStyledTextCtrl()
Filename = unicode:chardata()

See external documentation.

loadFile(This, Filename) -> boolean()
Types:
  This = wxStyledTextCtrl()
  Filename = unicode:chardata()

See external documentation.

doDragOver(This, X, Y, Def) -> wx:wx_enum()
Types:
  This = wxStyledTextCtrl()
  X = integer()
  Y = integer()
  Def = wx:wx_enum()

See external documentation.


doDropText(This, X, Y, Data) -> boolean()
Types:
  This = wxStyledTextCtrl()
  X = integer()
  Y = integer()
  Data = unicode:chardata()

See external documentation.

getUseAntiAliasing(This) -> boolean()
Types:
  This = wxStyledTextCtrl()

See external documentation.

addTextRaw(This, Text) -> ok
Types:
  This = wxStyledTextCtrl()
  Text = binary()

See external documentation.

insertTextRaw(This, Pos, Text) -> ok
Types:
  This = wxStyledTextCtrl()
  Pos = integer()
Text = binary()
See external documentation.

getCurLineRaw(This) -> Result
Types:
   Result = {Res::binary(), LinePos::integer()}
   This = wxStyledTextCtrl()
See external documentation.

getLineRaw(This, Line) -> binary()
Types:
   This = wxStyledTextCtrl()
   Line = integer()
See external documentation.

getSelectedTextRaw(This) -> binary()
Types:
   This = wxStyledTextCtrl()
See external documentation.

getTextRangeRaw(This, StartPos, EndPos) -> binary()
Types:
   This = wxStyledTextCtrl()
   StartPos = integer()
   EndPos = integer()
See external documentation.

setTextRaw(This, Text) -> ok
Types:
   This = wxStyledTextCtrl()
   Text = binary()
See external documentation.

getTextRaw(This) -> binary()
Types:
   This = wxStyledTextCtrl()
See external documentation.

appendTextRaw(This, Text) -> ok
Types:
   This = wxStyledTextCtrl()
   Text = binary()
See external documentation.
wxStyledTextCtrl

destroy(This::wxStyledTextCtrl()) -> ok
Destroys this object, do not use object again
wxStyledTextEvent

Erlang module

See external documentation: wxStyledTextEvent.

Use wxEvtHandler:connect/3 with EventType:

  stc_change, stc_styleneeded, stc_charadded, stc_savepointreached, stc_savepointleft, 
  stc_romodifyattempt, stc_key, stc_doubleclick, stc_updateui, stc_modified, stc_macrorecord, 
  stc_marginclick, stc_needshown, stc_painted, stc_userlistselection, stc_uridropped, stc_dwellstart, 
  stc_dwellend, stc_start_drag, stc_drag_over, stc_do_drop, stc_zoom, stc_hotspot_click, 
  stc_hotspot_dclick, stc_calltip_click, stc_autocomp_selection

See also the message variant #wxStyledText/1 event record type.

This class is derived (and can use functions) from:
wxCommandEvent
wxEvent

DATA TYPES
wxStyledTextEvent()

  An object reference. The representation is internal and can be changed without notice. It can't be used for 
  comparsion stored on disc or distributed for use on other nodes.

Exports

getPosition(This) -> integer()
Types:
  This = wxStyledTextEvent()
See external documentation.

getKey(This) -> integer()
Types:
  This = wxStyledTextEvent()
See external documentation.

getModifiers(This) -> integer()
Types:
  This = wxStyledTextEvent()
See external documentation.

getModificationType(This) -> integer()
Types:
  This = wxStyledTextEvent()
See external documentation.
wxStyledTextEvent

**getText**

getText(This) -> unicode:charlist()

Types:

This = wxStyledTextEvent()

See external documentation.

**getLength**

getLength(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getLinesAdded**

getLinesAdded(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getLine**

getLine(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getFoldLevelNow**

getFoldLevelNow(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getFoldLevelPrev**

getFoldLevelPrev(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getMargin**

getMargin(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getMessage**

getMessage(This) -> integer()

Types:

This = wxStyledTextEvent()

See external documentation.

**getWParam**

getWParam(This) -> integer()

Types:

This = wxStyledTextEvent()
See external documentation.

getLParam(This) -> integer()
Types:
    This = wxStyledTextEvent()
See external documentation.

getListType(This) -> integer()
Types:
    This = wxStyledTextEvent()
See external documentation.

getX(This) -> integer()
Types:
    This = wxStyledTextEvent()
See external documentation.

getY(This) -> integer()
Types:
    This = wxStyledTextEvent()
See external documentation.

getDragText(This) -> unicode:charlist()
Types:
    This = wxStyledTextEvent()
See external documentation.

getDragAllowMove(This) -> boolean()
Types:
    This = wxStyledTextEvent()
See external documentation.

getDragResult(This) -> wx:wx_enum()
Types:
    This = wxStyledTextEvent()
See external documentation.

getShift(This) -> boolean()
Types:
    This = wxStyledTextEvent()
See external documentation.
wxStyledTextEvent

getControl(This) -> boolean()
Types:
    This = wxStyledTextEvent()
See external documentation.

getAlt(This) -> boolean()
Types:
    This = wxStyledTextEvent()
See external documentation.
wxSysColourChangedEvent

Erlang module

See external documentation: \texttt{wxSysColourChangedEvent}.

Use \texttt{wxEvtHandler:connect/3} with \texttt{EventType}:

\begin{verbatim}
    sys_colour_changed
\end{verbatim}

See also the message variant \texttt{#wxSysColourChanged[]} event record type.

This class is derived (and can use functions) from:

\texttt{wxEvent}

**DATA TYPES**

\texttt{wxSysColourChangedEvent()}

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxSystemOptions

Erlang module

See external documentation: wxSystemOptions.

DATA TYPES
wxSystemOptions()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getOption(Name) -> unicode:charlist()
Types:
   Name = unicode:chardata()
See external documentation.

getOptionInt(Name) -> integer()
Types:
   Name = unicode:chardata()
See external documentation.

hasOption(Name) -> boolean()
Types:
   Name = unicode:chardata()
See external documentation.

isFalse(Name) -> boolean()
Types:
   Name = unicode:chardata()
See external documentation.

setOption(Name, Value) -> ok
Types:
   Name = unicode:chardata()
   Value = integer()
See external documentation.
Also:
setOption(Name, Value) -> 'ok' when
Name::unicode:chardata(), Value::unicode:chardata().
wxSystemSettings

Erlang module

See external documentation: wxSystemSettings.

DATA TYPES

wxSystemSettings()

An object reference. The representation is internal and can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

getColour(Index) -> wx:wx_colour4()

Types:

  _Index = wx:wx_enum()

See external documentation.


getFont(Index) -> wxFont:wxFont()

Types:

  _Index = wx:wx_enum()

See external documentation.


getMetric(Index) -> integer()

Types:

  _Index = wx:wx_enum()

Equivalent to getMetric(Index, []).
getMetric(Index, Options::[Option]) -> integer()

Types:

     Index = wx:wx_enum()
     Option = [win, wxWindow:wxWindow()]

See external documentation.

     | ?wxSYS_NETWORK_PRESENT | ?wxSYS_PENWINDOWS_PRESENT | ?wxSYS_SHOW_SOUNDS | ?
wxSYS_SWAP_BUTTONS

getscreenType() -> wx:wx_enum()

See external documentation.

Res = ?wxSYS_SCREEN_NONE | ?wxSYS_SCREEN_TINY | ?wxSYS_SCREEN_PDA | ?
wxSYS_SCREEN_SMALL | ?wxSYS_SCREEN_DESKTOP
wxTaskBarIcon

Erlang module

See external documentation: wxTaskBarIcon.

This class is derived (and can use functions) from:
wxEvtHandler

**DATA TYPES**

wxTaskBarIcon()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -&gt; wxTaskBarIcon()
See external documentation.

popupMenu(This, Menu) -&gt; boolean()
Types:

This = wxTaskBarIcon()
Menu = wxMenu:wxMenu()

See external documentation.

removeIcon(This) -&gt; boolean()
Types:

This = wxTaskBarIcon()

See external documentation.

setIcon(This, Icon) -&gt; boolean()
Types:

This = wxTaskBarIcon()
Icon = wxIcon:wxIcon()

Equivalent to setIcon(This, Icon, []).

setIcon(This, Icon, Options::[Option]) -&gt; boolean()
Types:

This = wxTaskBarIcon()
Icon = wxIcon:wxIcon()
Option = {tooltip, unicode:chardata()}

See external documentation.
wxTaskBarIcon

\texttt{destroy(This::wxTaskBarIcon())} -> \texttt{ok}

Destroys this object, do not use object again
wxTaskBarIconEvent

Erlang module

See external documentation: wxTaskBarIconEvent.

Use wxEvtHandler:connect/3 with EventType:

   taskbar_move, taskbar_left_down, taskbar_left_up, taskbar_right_down, taskbar_right_up,
   taskbar_left_dclick, taskbar_right_dclick

See also the message variant #wxTaskBarIcon[] event record type.

This class is derived (and can use functions) from:

wxEvent

DATA TYPES

wxTaskBarIconEvent()

   An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxTextAttr

Erlang module

See external documentation: wxTextAttr.

DATA TYPES

wxTextAttr()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxTextAttr()
See external documentation.

ew(ColText) -> wxTextAttr()
Types:
    ColText = wx:wx_colour()
Equivalent to new(ColText, []).

new(ColText, Options::[Option]) -> wxTextAttr()
Types:
    ColText = wx:wx_colour()
    Option = {colBack, wx:wx_colour()} | {font, wxFont:wxFont()} | {alignment, wx:wx_enum()}
See external documentation.

getAlignment(This) -> wx:wx_enum()
Types:
    This = wxTextAttr()
See external documentation.

getBackgroundColour(This) -> wx:wx_colour4()
Types:
    This = wxTextAttr()
See external documentation.
getFont(This) -> wxFont:wxFont()
Types:
  This = wxTextAttr()
See external documentation.

getLeftIndent(This) -> integer()
Types:
  This = wxTextAttr()
See external documentation.

getLeftSubIndent(This) -> integer()
Types:
  This = wxTextAttr()
See external documentation.

getRightIndent(This) -> integer()
Types:
  This = wxTextAttr()
See external documentation.

getTabs(This) -> [integer()]
Types:
  This = wxTextAttr()
See external documentation.

getTextColour(This) -> wx:wx_colour4()
Types:
  This = wxTextAttr()
See external documentation.

hasBackgroundColour(This) -> boolean()
Types:
  This = wxTextAttr()
See external documentation.

hasFont(This) -> boolean()
Types:
  This = wxTextAttr()
See external documentation.

hasTextColour(This) -> boolean()
Types:
  This = wxTextAttr()
See external documentation.

getFlags(This) -> integer()
Types:
    This = wxTextAttr()
See external documentation.

isDefault(This) -> boolean()
Types:
    This = wxTextAttr()
See external documentation.

setAlignment(This, Alignment) -> ok
Types:
    This = wxTextAttr()
    Alignment = wx:wx_enum()
See external documentation.

setBackgroundColour(This, ColBack) -> ok
Types:
    This = wxTextAttr()
    ColBack = wx:wx_colour()
See external documentation.

setFlags(This, Flags) -> ok
Types:
    This = wxTextAttr()
    Flags = integer()
See external documentation.

setFont(This, Font) -> ok
Types:
    This = wxTextAttr()
    Font = wxFont:wxFont()
Equivalent to setFont(This, Font, []).

setFont(This, Font, Options::[Option]) -> ok
Types:
    This = wxTextAttr()
    Font = wxFont:wxFont()
    Option = {flags, integer()}
See external documentation.

```erlang
setLeftIndent(This, Indent) -> ok
Types:
  This = wxTextAttr()
  Indent = integer()
Equivalent to setLeftIndent(This, Indent, []).
```

```erlang
setLeftIndent(This, Indent, Options::[Option]) -> ok
Types:
  This = wxTextAttr()
  Indent = integer()
  Option = {subIndent, integer()}
See external documentation.
```

```erlang
setRightIndent(This, Indent) -> ok
Types:
  This = wxTextAttr()
  Indent = integer()
See external documentation.
```

```erlang
setTabs(This, Tabs) -> ok
Types:
  This = wxTextAttr()
  Tabs = [integer()]
See external documentation.
```

```erlang
setTextColour(This, ColText) -> ok
Types:
  This = wxTextAttr()
  ColText = wx:wx_colour()
See external documentation.
```

```erlang
destroy(This::wxTextAttr()) -> ok
Destroys this object, do not use object again
```
wxTextCtrl

Erlang module

See external documentation: wxTextCtrl.

This class is derived (and can use functions) from:
  - wxControl
  - wxWindow
  - wxEvtHandler

**DATA TYPES**

```
wxTextCtrl()
```

An object reference. The representation is internal and can be changed without notice. It can't be used for comparision stored on disc or distributed for use on other nodes.

**Exports**

```
new() -> wxTextCtrl()
```

See external documentation.

```
new(Parent, Id) -> wxTextCtrl()
```

Types:

```
Parent = wxWindow:wxWindow()
Id = integer()
```

Equivalent to `new(Parent, Id, [])`.

```
new(Parent, Id, Options::[Option]) -> wxTextCtrl()
```

Types:

```
Parent = wxWindow:wxWindow()
Id = integer()
Option = {value, unicode:chardata()} | {pos, {X::integer(), Y::integer()}}
  | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}
```

See external documentation.

```
appendText(This, Text) -> ok
```

Types:

```
This = wxTextCtrl()
Text = unicode:chardata()
```

See external documentation.

```
canCopy(This) -> boolean()
```

Types:

640 | Ericsson AB. All Rights Reserved.: wxErlang
This = wxTextCtrl()
See external documentation.

canCut(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

canPaste(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

canRedo(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

canUndo(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

clear(This) -> ok
Types:
    This = wxTextCtrl()
See external documentation.

copy(This) -> ok
Types:
    This = wxTextCtrl()
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
    This = wxTextCtrl()
    Parent = wxWindow:wxWindow()
    Id = integer()
Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
    This = wxTextCtrl()
wxTextCtrl

Parent = \texttt{wxWindow:wxWindow()}
Id = integer()
Option = \{value, \texttt{unicode:chardata()}) \mid \{pos, \{X::integer(), Y::integer()}\} \mid \{size, \{W::integer(), H::integer\}} \mid \{style, integer\} \mid \{validator, \texttt{wx:wx_object()}}
See external documentation.

cut(This) -> ok
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
See external documentation.

discardEdits(This) -> ok
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
See external documentation.

changeValue(This, Value) -> ok
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
Value = \texttt{unicode:chardata()}
See external documentation.

emulateKeyPress(This, Event) -> boolean()
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
Event = \texttt{wxKeyEvent:wxKeyEvent()}
See external documentation.

defaultStyle(This) -> \texttt{wxTextAttr:wxTextAttr()}
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
See external documentation.

getInsertionPoint(This) -> integer()
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
See external documentation.

getLastPosition(This) -> integer()
Types:
\hspace*{1cm}This = \texttt{wxTextCtrl()}
See external documentation.
getLineLength(This, LineNo) -> integer()
Types:
    This = wxTextCtrl()
    LineNo = integer()
See external documentation.

getLineText(This, LineNo) -> unicode:charlist()
Types:
    This = wxTextCtrl()
    LineNo = integer()
See external documentation.

getNumberOfLines(This) -> integer()
Types:
    This = wxTextCtrl()
See external documentation.

getRange(This, From, To) -> unicode:charlist()
Types:
    This = wxTextCtrl()
    From = integer()
    To = integer()
See external documentation.

getSelection(This) -> {From::integer(), To::integer()}
Types:
    This = wxTextCtrl()
See external documentation.

getStringSelection(This) -> unicode:charlist()
Types:
    This = wxTextCtrl()
See external documentation.

getStyle(This, Position, Style) -> boolean()
Types:
    This = wxTextCtrl()
    Position = integer()
    Style = wxTextAttr:wxTextAttr()
See external documentation.

getValue(This) -> unicode:charlist()
Types:
This = wxTextCtrl()
See external documentation.

isEditable(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

isModified(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

isMultiLine(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

isSingleLine(This) -> boolean()
Types:
    This = wxTextCtrl()
See external documentation.

loadFile(This, File) -> boolean()
Types:
    This = wxTextCtrl()
    File = unicode:chardata()
Equivalent to loadFile(This, File, []).

loadFile(This, File, Options::[Option]) -> boolean()
Types:
    This = wxTextCtrl()
    File = unicode:chardata()
    Option = {fileType, integer()}
See external documentation.

markDirty(This) -> ok
Types:
    This = wxTextCtrl()
See external documentation.

paste(This) -> ok
Types:
This = wxTextCtrl()
See external documentation.

positionToXY(This, Pos) -> Result
Types:
  Result = {Res::boolean(), X::integer(), Y::integer()}
  This = wxTextCtrl()
  Pos = integer()
See external documentation.

redo(This) -> ok
Types:
  This = wxTextCtrl()
See external documentation.

remove(This, From, To) -> ok
Types:
  This = wxTextCtrl()
  From = integer()
  To = integer()
See external documentation.

replace(This, From, To, Value) -> ok
Types:
  This = wxTextCtrl()
  From = integer()
  To = integer()
  Value = unicode:chardata()
See external documentation.

saveFile(This) -> boolean()
Types:
  This = wxTextCtrl()
Equivalent to saveFile(This, []).

saveFile(This, Options:::[Option]) -> boolean()
Types:
  This = wxTextCtrl()
  Option = {file, unicode:chardata()} | {fileType, integer()}
See external documentation.

setDefaultStyle(This, Style) -> boolean()
Types:
wxTextCtrl

This = wxTextCtrl()
Style = wxTextAttr:wxTextAttr()
See external documentation.

setEditable(This, Editable) -> ok
Types:
This = wxTextCtrl()
Editable = boolean()
See external documentation.

setInsertionPoint(This, Pos) -> ok
Types:
This = wxTextCtrl()
Pos = integer()
See external documentation.

setInsertionPointEnd(This) -> ok
Types:
This = wxTextCtrl()
See external documentation.

setMaxLength(This, Len) -> ok
Types:
This = wxTextCtrl()
Len = integer()
See external documentation.

setSelection(This, From, To) -> ok
Types:
This = wxTextCtrl()
From = integer()
To = integer()
See external documentation.

setStyle(This, Start, End, Style) -> boolean()
Types:
This = wxTextCtrl()
Start = integer()
End = integer()
Style = wxTextAttr:wxTextAttr()
See external documentation.
setValue(This, Value) -> ok

Types:

This = wxTextCtrl()
Value = unicode:chardata()

See external documentation.

showPosition(This, Pos) -> ok

Types:

This = wxTextCtrl()
Pos = integer()

See external documentation.

undo(This) -> ok

Types:

This = wxTextCtrl()

See external documentation.

writeText(This, Text) -> ok

Types:

This = wxTextCtrl()
Text = unicode:chardata()

See external documentation.

xYToPosition(This, X, Y) -> integer()

Types:

This = wxTextCtrl()
X = integer()
Y = integer()

See external documentation.

destroy(This::wxTextCtrl()) -> ok

Destroys this object, do not use object again.
**wxTextDataObject**

Erlang module

See external documentation: *wxTextDataObject*.

This class is derived (and can use functions) from: *wxDataObject*

**DATA TYPES**

**wxTextDataObject()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

new() -> wxTextDataObject()

Equivalent to new([]).

new(Options::[Option]) -> wxTextDataObject()

Types:

\[
\text{Option} = \{\text{text}, \text{unicode}:	ext{chardata}()\}
\]

See external documentation.

getTextLength(This) -> integer()

Types:

\[
\text{This} = \text{wxTextDataObject()}
\]

See external documentation.

getText(This) -> unicode:charlist()

Types:

\[
\text{This} = \text{wxTextDataObject()}
\]

See external documentation.

setText(This, Text) -> ok

Types:

\[
\text{This} = \text{wxTextDataObject()}
\]

\[
\text{Text} = \text{unicode}:	ext{chardata()}
\]

See external documentation.

destroy(This::wxTextDataObject()) -> ok

Destroys this object, do not use object again
wxTextEntryDialog

Erlang module

See external documentation: wxTextEntryDialog.

This class is derived (and can use functions) from:
wxDialog
wxTopLevelWindow
wxWindow
wxEvtHandler

DATA TYPES
wxTextEntryDialog()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new(Parent, Message) -> wxTextEntryDialog()
Types:
  Parent = wxWindow:wxWindow()
  Message = unicode:chardata()
Equivalent to new(Parent, Message, []).

new(Parent, Message, Options::[Option]) -> wxTextEntryDialog()
Types:
  Parent = wxWindow:wxWindow()
  Message = unicode:chardata()
  Option = {caption, unicode:chardata()} | {value, unicode:chardata()} | {style, integer()} | {pos, {X::integer(), Y::integer()}}

See external documentation.

getValue(This) -> unicode:charlist()
Types:
  This = wxTextEntryDialog()

See external documentation.

setValue(This, Val) -> ok
Types:
  This = wxTextEntryDialog()
  Val = unicode:chardata()

See external documentation.
wxTextEntryDialog

`destroy(This::wxTextEntryDialog()) -> ok`

Destroys this object, do not use object again
wxToggleButton

Erlang module

See external documentation: wxToggleButton.

This class is derived (and can use functions) from:
wxControl
wxWindow
wxEvtHandler

DATA TYPES

wxToggleButton()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxToggleButton()

See external documentation.

ew(Parent, Id, Label) -> wxToggleButton()

Types:
Parent = wxWindow:wxWindow()
Id = integer()
Label = unicode:chardata()

Equivalent to new(Parent, Id, Label, []).

new(Parent, Id, Label, Options::[Option]) -> wxToggleButton()

Types:
Parent = wxWindow:wxWindow()
Id = integer()
Label = unicode:chardata()

Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer() | {validator, wx:wx_object()}

See external documentation.

create(This, Parent, Id, Label) -> boolean()

Types:
This = wxToggleButton()
Parent = wxWindow:wxWindow()
Id = integer()
Label = unicode:chardata()

Equivalent to create(This, Parent, Id, Label, []).
wxToggleButton

create(This, Parent, Id, Label, Options::[Option]) -> boolean()

Types:
    This = wxToggleButton()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Label = unicode:chardata()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}

See external documentation.

getValue(This) -> boolean()

Types:
    This = wxToggleButton()

See external documentation.

setValue(This, State) -> ok

Types:
    This = wxToggleButton()
    State = boolean()

See external documentation.

destroy(This::wxToggleButton()) -> ok

Destroys this object, do not use object again
wxToolBar

Erlang module

See external documentation: `wxToolBar`.
This class is derived (and can use functions) from:
`wxControl`
`wxWindow`
`wxEvtHandler`

**DATA TYPES**

*wxToolBar()*

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`addControl(This, Control) -> wx:wx_object()`

Types:

`This = wxToolBar()`  
`Control = wxControl:wxControl()`

See external documentation.

`addSeparator(This) -> wx:wx_object()`

Types:

`This = wxToolBar()`

See external documentation.

`addTool(This, Tool) -> wx:wx_object()`

Types:

`This = wxToolBar()`  
`Tool = wx:wx_object()`

See external documentation.

`addTool(This, Toolid, Bitmap) -> wx:wx_object()`

Types:

`This = wxToolBar()`  
`Toolid = integer()`  
`Bitmap = wxBitmap:wxBitmap()`

Equivalent to `addTool(This, Toolid, Bitmap, [])`.

`addTool(This, Toolid, Label, Bitmap) -> wx:wx_object()`

Types:
This = wxToolBar()
Toolid = integer()
Label = unicode:chardata()
Bitmap = wxBitmap:wxBitmap()

See external documentation.

Also:
addTool(This, Toolid, Bitmap, BmpDisabled) -> wx:wx_object() when
This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(), BmpDisabled::wxBitmap:wxBitmap();
(This, Toolid, Bitmap, [Option]) -> wx:wx_object() when
This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(),
Option :: ['shortHelpString', unicode:chardata()]
| ['longHelpString', unicode:chardata()].


addTool(This, Toolid, Label, Bitmap, BmpDisabled) -> wx:wx_object()

Types:

This = wxToolBar()
Toolid = integer()
Label = unicode:chardata()
Bitmap = wxBitmap:wxBitmap()
BmpDisabled = wxBitmap:wxBitmap()

See external documentation.

Also:
addTool(This, Toolid, Label, Bitmap, [Option]) -> wx:wx_object() when
This::wxToolBar(), Toolid::integer(), Label::unicode:chardata(), Bitmap::wxBitmap:wxBitmap(),
Option :: ['shortHelp', unicode:chardata()]
| ['kind', wx:wx_enum()];
(This, Toolid, Bitmap, BmpDisabled, [Option]) -> wx:wx_object() when
This::wxToolBar(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(), BmpDisabled::wxBitmap:wxBitmap(),
Option :: ['toggle', boolean()]
| ['clientData', wx:wx_object()]
| ['shortHelpString', unicode:chardata()]
| ['longHelpString', unicode:chardata()].


addTool(This, Toolid, Bitmap, BmpDisabled, Toggle, XPos) -> wx:wx_object()

Types:

This = wxToolBar()
Toolid = integer()
Bitmap = wxBitmap:wxBitmap()
BmpDisabled = wxBitmap:wxBitmap()
Toggle = boolean()
XPos = integer()

See external documentation.
Also:
addTool(This, Toolid, Label, Bitmap, BmpDisabled, [Option]) -> wx:wx_object() when
This::wxToolBar(),   Toolid::integer(),   Label::unicode:chardata(),   Bitmap::wxBitmap:wxBitmap(),
BmpDisabled::wxBitmap:wxBitmap(),   Option :: ['kind', wx:wx_enum()]
| ['shortHelp', unicode:chardata()]
| ['longHelp', unicode:chardata()]
| ['data', wx:wx_object()].

addTool(This, Toolid, Bitmap, BmpDisabled, Toggle, XPos, Options::[Option]) -> wx:wx_object()
Types:

  This = wxToolBar()
  Toolid = integer()
  Bitmap = wxBitmap:wxBitmap()
  BmpDisabled = wxBitmap:wxBitmap()
  Toggle = boolean()
  XPos = integer()
  Option = {yPos, integer()} | {clientData, wx:wx_object()} | {shortHelp, unicode:chardata()} | {longHelp, unicode:chardata()}

See external documentation.

addCheckTool(This, Toolid, Label, Bitmap) -> wx:wx_object()
Types:

  This = wxToolBar()
  Toolid = integer()
  Label = unicode:chardata()
  Bitmap = wxBitmap:wxBitmap()

Equivalent to addCheckTool(This, Toolid, Label, Bitmap, []).

addCheckTool(This, Toolid, Label, Bitmap, Options::[Option]) -> wx:wx_object()
Types:

  This = wxToolBar()
  Toolid = integer()
  Label = unicode:chardata()
  Bitmap = wxBitmap:wxBitmap()
  Option = {bmpDisabled, wxBitmap:wxBitmap()} | {shortHelp, unicode:chardata()} | {longHelp, unicode:chardata()} | {data, wx:wx_object()}

See external documentation.
addRadioTool(This, Toolid, Label, Bitmap) -> wx:wx_object()

Types:
  This = wxToolBar()
  Toolid = integer()
  Label = unicode:chardata()
  Bitmap = wxBitmap:wxBitmap()

Equivalent to addRadioTool(This, Toolid, Label, Bitmap, []).

addRadioTool(This, Toolid, Label, Bitmap, Options::[Option]) ->
wx:wx_object()

Types:
  This = wxToolBar()
  Toolid = integer()
  Label = unicode:chardata()
  Bitmap = wxBitmap:wxBitmap()
  Option = {bmpDisabled, wxBitmap:wxBitmap()} || {shortHelp, unicode:chardata()} || {longHelp, unicode:chardata()} || {data, wx:wx_object()}

See external documentation.

addStretchableSpace(This) -> wx:wx_object()

Types:
  This = wxToolBar()

See external documentation.

insertStretchableSpace(This, Pos) -> wx:wx_object()

Types:
  This = wxToolBar()
  Pos = integer()

See external documentation.

deleteTool(This, Toolid) -> boolean()

Types:
  This = wxToolBar()
  Toolid = integer()

See external documentation.

deleteToolByPos(This, Pos) -> boolean()

Types:
  This = wxToolBar()
  Pos = integer()

See external documentation.
enableTool(This, Toolid, Enable) -> ok
Types:
  This = wxToolBar()
  Toolid = integer()
  Enable = boolean()
See external documentation.

findById(This, Toolid) -> wx:wx_object()
Types:
  This = wxToolBar()
  Toolid = integer()
See external documentation.

findControl(This, Toolid) -> wxControl:wxControl()
Types:
  This = wxToolBar()
  Toolid = integer()
See external documentation.

findToolForPosition(This, X, Y) -> wx:wx_object()
Types:
  This = wxToolBar()
  X = integer()
  Y = integer()
See external documentation.

getToolSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxToolBar()
See external documentation.

getToolBitmapSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxToolBar()
See external documentation.

getMargins(This) -> {W::integer(), H::integer()}
Types:
  This = wxToolBar()
See external documentation.

getToolEnabled(This, Toolid) -> boolean()
Types:
wxToolBar

This = wxToolBar()
Toolid = integer()

See external documentation.

getToolLongHelp(This, Toolid) -> unicode:charlist()
Types:
    This = wxToolBar()
    Toolid = integer()

See external documentation.

getToolPacking(This) -> integer()
Types:
    This = wxToolBar()

See external documentation.

getToolPos(This, Id) -> integer()
Types:
    This = wxToolBar()
    Id = integer()

See external documentation.

getToolSeparation(This) -> integer()
Types:
    This = wxToolBar()

See external documentation.

getToolShortHelp(This, Toolid) -> unicode:charlist()
Types:
    This = wxToolBar()
    Toolid = integer()

See external documentation.

getToolState(This, Toolid) -> boolean()
Types:
    This = wxToolBar()
    Toolid = integer()

See external documentation.

insertControl(This, Pos, Control) -> wx:wx_object()
Types:
    This = wxToolBar()
    Pos = integer()
    Control = wxControl:wxControl()
See external documentation.

`insertSeparator(This, Pos) -> wx:wx_object()`

Types:

- `This = wxToolBar()`  
- `Pos = integer()`

See external documentation.

`insertTool(This, Pos, Tool) -> wx:wx_object()`

Types:

- `This = wxToolBar()`  
- `Pos = integer()`  
- `Tool = wx:wx_object()`

See external documentation.

`insertTool(This, Pos, Toolid, Bitmap) -> wx:wx_object()`

Types:

- `This = wxToolBar()`  
- `Pos = integer()`  
- `Toolid = integer()`  
- `Bitmap = wxBitmap:wxBitmap()`

Equivalent to `insertTool(This, Pos, Toolid, Bitmap, [])`.

`insertTool(This, Pos, Toolid, Label, Bitmap) -> wx:wx_object()`

Types:

- `This = wxToolBar()`  
- `Pos = integer()`  
- `Toolid = integer()`  
- `Label = unicode:chardata()`  
- `Bitmap = wxBitmap:wxBitmap()`

See external documentation.

Also:

`insertTool(This, Pos, Toolid, Bitmap, [Option]) -> wx:wx_object()`

Types:

- `This::wxToolBar(), Pos::integer(), Toolid::integer(), Bitmap::wxBitmap:wxBitmap(), Option :: {'bmpDisabled', wxBitmap:wxBitmap()}, {'toggle', boolean()}, {'clientData', wx:wx_object()}, {'shortHelp', unicode:chardata()}, {'longHelp', unicode:chardata}.


`insertTool(This, Pos, Toolid, Label, Bitmap, Options::*[Option]) -> wx:wx_object()`

Types:
wxToolBar

This = wxToolBar()
Pos = integer()
Toolid = integer()
Label = unicode:chardata()
Bitmap = wxBitmap:wxBitmap()
Option = {bmpDisabled, wxBitmap:wxBitmap()} | {kind, wx:wx_enum()}
| {shortHelp, unicode:chardata()} | {longHelp, unicode:chardata()} | {clientData, wx:wx_object()}
See external documentation.

realize(This) -> boolean()
Types:
   This = wxToolBar()
See external documentation.

removeTool(This, Toolid) -> wx:wx_object()
Types:
   This = wxToolBar()
   Toolid = integer()
See external documentation.

setMargins(This, X, Y) -> ok
Types:
   This = wxToolBar()
   X = integer()
   Y = integer()
See external documentation.

setToolBitmapSize(This, Size) -> ok
Types:
   This = wxToolBar()
   Size = {W::integer(), H::integer()}
See external documentation.

setToolLongHelp(This, Toolid, HelpString) -> ok
Types:
   This = wxToolBar()
   Toolid = integer()
   HelpString = unicode:chardata()
See external documentation.
setToolPacking(This, Packing) -> ok
Types:
   This = wxToolBar()
   Packing = integer()
See external documentation.

setToolShortHelp(This, Id, HelpString) -> ok
Types:
   This = wxToolBar()
   Id = integer()
   HelpString = unicode:chardata()
See external documentation.

setToolSeparation(This, Separation) -> ok
Types:
   This = wxToolBar()
   Separation = integer()
See external documentation.

toggleTool(This, ToolId, Toggle) -> ok
Types:
   This = wxToolBar()
   ToolId = integer()
   Toggle = boolean()
See external documentation.
wxToolTip

Erlang module

See external documentation: wxToolTip.

DATA TYPES

wxToolTip()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

enable(Flag) -> ok
Types:
   Flag = boolean()
See external documentation.

setDelay(Msecs) -> ok
Types:
   Msecs = integer()
See external documentation.

new(Tip) -> wxToolTip()
Types:
   Tip = unicode:chardata()
See external documentation.

setTip(This, Tip) -> ok
Types:
   This = wxToolTip()
   Tip = unicode:chardata()
See external documentation.

getTip(This) -> unicode:charlist()
Types:
   This = wxToolTip()
See external documentation.

getWindow(This) -> wxWindow:wxWindow()
Types:
   This = wxToolTip()
See external documentation.
destroy(This::wxToolTip()) -> ok

Destroys this object, do not use object again
wxToolbook

Erlang module

See external documentation: wxToolbook.

This class is derived (and can use functions) from:
    wxControl
    wxWindow
    wxEvtHandler

DATA TYPES

wxToolbook()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxToolbook()

See external documentation.

new(Parent, Id) -> wxToolbook()

Types:
    Parent = wxWindow:wxWindow()
    Id = integer()

Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxToolbook()

Types:
    Parent = wxWindow:wxWindow()
    Id = integer()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}

See external documentation.

addPage(This, Page, Text) -> boolean()

Types:
    This = wxToolbook()
    Page = wxWindow:wxWindow()
    Text = unicode:chardata()

Equivalent to addPage(This, Page, Text, []).

addPage(This, Page, Text, Options::[Option]) -> boolean()

Types:

This = wxToolbook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean() | {imageId, integer()}
See external documentation.  

advanceSelection(This) -> ok
Types:
    This = wxToolbook()
Equivalent to advanceSelection(This, []).  

advanceSelection(This, Options::[Option]) -> ok
Types:
    This = wxToolbook()
    Option = {forward, boolean()}
See external documentation.  

assignImageList(This, ImageList) -> ok
Types:
    This = wxToolbook()
    ImageList = wxImageList:wxImageList()
See external documentation.  

create(This, Parent, Id) -> boolean()
Types:
    This = wxToolbook()
    Parent = wxWindow:wxWindow()
    Id = integer()
Equivalent to create(This, Parent, Id, []).  

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
    This = wxToolbook()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.  

deleteAllPages(This) -> boolean()
Types:
    This = wxToolbook()
See external documentation.
deletePage(This, N) -> boolean()
Types:
   This = wxToolbook()
   N = integer()
See external documentation.
removePage(This, N) -> boolean()
Types:
   This = wxToolbook()
   N = integer()
See external documentation.
getCurrentPage(This) -> wxWindow:wxWindow()
Types:
   This = wxToolbook()
See external documentation.
getImageList(This) -> wxImageList:wxImageList()
Types:
   This = wxToolbook()
See external documentation.
getPage(This, N) -> wxWindow:wxWindow()
Types:
   This = wxToolbook()
   N = integer()
See external documentation.
getPageCount(This) -> integer()
Types:
   This = wxToolbook()
See external documentation.
getPageImage(This, N) -> integer()
Types:
   This = wxToolbook()
   N = integer()
See external documentation.
getPageText(This, N) -> unicode:charlist()
Types:
   This = wxToolbook()
   N = integer()
getSelection(This) -> integer()
Types:
   This = wxToolbook()
See external documentation.

hitTest(This, Pt) -> Result
Types:
   Result = {Res::integer(), Flags::integer()}
   This = wxToolbook()
   Pt = {X::integer(), Y::integer()}
See external documentation.

insertPage(This, N, Page, Text) -> boolean()
Types:
   This = wxToolbook()
   N = integer()
   Page = wxWindow::wxWindow()
   Text = unicode::chardata()
Equivalent to insertPage(This, N, Page, Text, []).

insertPage(This, N, Page, Text, Options::[Option]) -> boolean()
Types:
   This = wxToolbook()
   N = integer()
   Page = wxWindow::wxWindow()
   Text = unicode::chardata()
   Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

setImageList(This, ImageList) -> ok
Types:
   This = wxToolbook()
   ImageList = wxImageList::wxImageList()
See external documentation.

setPageSize(This, Size) -> ok
Types:
   This = wxToolbook()
   Size = {W::integer(), H::integer()}
See external documentation.
setPageImage(This, N, ImageId) -> boolean()
Types:
    This = wxToolbook()
    N = integer()
    ImageId = integer()
See external documentation.

setPageText(This, N, StrText) -> boolean()
Types:
    This = wxToolbook()
    N = integer()
    StrText = unicode:chardata()
See external documentation.

setSelection(This, N) -> integer()
Types:
    This = wxToolbook()
    N = integer()
See external documentation.

changeSelection(This, N) -> integer()
Types:
    This = wxToolbook()
    N = integer()
See external documentation.

destroy(This::wxToolbook()) -> ok
Destroys this object, do not use object again
wxTopLevelWindow

Erlang module

See external documentation: wxTopLevelWindow.

This class is derived (and can use functions) from:
wxWindow  
wxEvtHandler

DATA TYPES
wxTopLevelWindow()

An object reference. The representation is internal and can be changed without notice. It can't be used for
comparison stored on disc or distributed for use on other nodes.

Exports

getIcon(This) -> wxIcon:wxIcon()
Types:
    This = wxTopLevelWindow()
See external documentation.

getIcons(This) -> wxIconBundle:wxIconBundle()
Types:
    This = wxTopLevelWindow()
See external documentation.

getTitle(This) -> unicode:charlist()
Types:
    This = wxTopLevelWindow()
See external documentation.

isActive(This) -> boolean()
Types:
    This = wxTopLevelWindow()
See external documentation.

iconize(This) -> ok
Types:
    This = wxTopLevelWindow()
Equivalent to iconize(This, []).

iconize(This, Options::[Option]) -> ok
Types:
This = wxTopLevelWindow()
Option = {iconize, boolean()}

See external documentation.

isFullScreen(This) -> boolean()
Types:
  This = wxTopLevelWindow()
See external documentation.

isIconized(This) -> boolean()
Types:
  This = wxTopLevelWindow()
See external documentation.

isMaximized(This) -> boolean()
Types:
  This = wxTopLevelWindow()
See external documentation.

maximize(This) -> ok
Types:
  This = wxTopLevelWindow()
Equivalent to maximize(This, []).

maximize(This, Options::[Option]) -> ok
Types:
  This = wxTopLevelWindow()
  Option = {maximize, boolean()}
See external documentation.

requestUserAttention(This) -> ok
Types:
  This = wxTopLevelWindow()
Equivalent to requestUserAttention(This, []).

requestUserAttention(This, Options::[Option]) -> ok
Types:
  This = wxTopLevelWindow()
  Option = {flags, integer()}
See external documentation.

setIcon(This, Icon) -> ok
Types:
This = wxTopLevelWindow()
Icon = wxIcon:wxIcon()

See external documentation.

setIcon(This, Icons) -> ok
Types:
  This = wxTopLevelWindow()
  Icons = wxIconBundle:wxIconBundle()

See external documentation.

centerOnScreen(This) -> ok
Types:
  This = wxTopLevelWindow()

Equivalent to centerOnScreen(This, []).

centerOnScreen(This, Options::[Option]) -> ok
Types:
  This = wxTopLevelWindow()
  Option = (dir, integer())

See external documentation.

centreOnScreen(This) -> ok
Types:
  This = wxTopLevelWindow()

Equivalent to centreOnScreen(This, []).

centreOnScreen(This, Options::[Option]) -> ok
Types:
  This = wxTopLevelWindow()
  Option = (dir, integer())

See external documentation.

setShape(This, Region) -> boolean()
Types:
  This = wxTopLevelWindow()
  Region = wxRegion:wxRegion()

See external documentation.

setTitle(This, Title) -> ok
Types:
  This = wxTopLevelWindow()
  Title = unicode:chardata()

See external documentation.
showFullScreen(This, Show) -> boolean()
Types:
    This = wxTopLevelWindow()
    Show = boolean()
Equivalent to showFullScreen(This, Show, []).

showFullScreen(This, Show, Options::[Option]) -> boolean()
Types:
    This = wxTopLevelWindow()
    Show = boolean()
    Option = {style, integer()}
See external documentation.
**wxTreeCtrl**

Erlang module

See external documentation: `wxTreeCtrl`.

Note: The representation of `treeItemId()` have changed from the original class implementation to be an semi-opaque type. Equality between `TreeItemId`'s can be tested and zero means that the `TreeItem` is invalid.

**DATA TYPES**

`wxTreeCtrl()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`new() -> wxTreeCtrl()`

See external documentation.

`new(Parent) -> wxTreeCtrl()`

Types:

```
Parent = wxWindow:wxWindow()
```

Equivalent to `new(Parent, [])`.

`new(Parent, Options::[Option]) -> wxTreeCtrl()`

Types:

```
Parent = wxWindow:wxWindow()
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()} | {validator, wx:wx_object()}
```

See external documentation.

`addRoot(This, Text) -> integer()`

Types:

```
This = wxTreeCtrl()
Text = unicode:chardata()
```

Equivalent to `addRoot(This, Text, [])`.

`addRoot(This, Text, Options::[Option]) -> integer()`

Types:

```
This = wxTreeCtrl()
Text = unicode:chardata()
Option = {image, integer()} | {selectedImage, integer()} | {data, term()}
```

See external documentation.
wxTreeCtrl

appendItem(This, Parent, Text) -> integer()
Types:
    This = wxTreeCtrl()
    Parent = integer()
    Text = unicode:chardata()
Equivalent to appendItem(This, Parent, Text, []).

appendItem(This, Parent, Text, Options::[Option]) -> integer()
Types:
    This = wxTreeCtrl()
    Parent = integer()
    Text = unicode:chardata()
    Option = {image, integer()} | {selectedImage, integer()} | {data, term()}
See external documentation.

assignImageList(This, ImageList) -> ok
Types:
    This = wxTreeCtrl()
    ImageList = wxImageList:wxImageList()
See external documentation.

assignStateImageList(This, ImageList) -> ok
Types:
    This = wxTreeCtrl()
    ImageList = wxImageList:wxImageList()
See external documentation.

collapse(This, Item) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
See external documentation.

collapseAndReset(This, Item) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
See external documentation.

create(This, Parent) -> boolean()
Types:
    This = wxTreeCtrl()
    Parent = wxWindow:wxWindow()
Equivalent to `create(This, Parent, [])`.

```erlang
create(This, Parent, Options::[Option]) -> boolean()
```

Types:
```
This = wxTreeCtrl()
Parent = wxWindow:wxWindow()
Option = {id, integer()} | {pos, {X::integer(), Y::integer()}} |
         {size, {W::integer(), H::integer()}} | {style, integer()} | {validator,
         wx:wx_object()}
```

See external documentation.

```erlang
delete(This, Item) -> ok
```

Types:
```
This = wxTreeCtrl()
Item = integer()
```

See external documentation.

```erlang
deleteAllItems(This) -> ok
```

Types:
```
This = wxTreeCtrl()
```

See external documentation.

```erlang
deleteChildren(This, Item) -> ok
```

Types:
```
This = wxTreeCtrl()
Item = integer()
```

See external documentation.

```erlang
editLabel(This, Item) -> wxTextCtrl:wxTextCtrl()
```

Types:
```
This = wxTreeCtrl()
Item = integer()
```

See external documentation.

```erlang
ensureVisible(This, Item) -> ok
```

Types:
```
This = wxTreeCtrl()
Item = integer()
```

See external documentation.

```erlang
expand(This, Item) -> ok
```

Types:
```
This = wxTreeCtrl()
```
Item = integer()
See external documentation.

getBoundingRect(This, Item) -> Result
Types:
    Result = {Result::boolean(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}}
    This = wxTreeCtrl()
    Item = integer()
Equivalent to getBoundingRect(This, Item, []).

getBoundingRect(This, Item, Options::[Option]) -> Result
Types:
    Result = {Result::boolean(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}}
    This = wxTreeCtrl()
    Item = integer()
    Option = {textOnly, boolean()}
See external documentation.

gETCHildrenCount(This, Item) -> integer()
Types:
    This = wxTreeCtrl()
    Item = integer()
Equivalent to getCount(This).

gETCHildrenCount(This, Item, Options::[Option]) -> integer()
Types:
    This = wxTreeCtrl()
    Item = integer()
    Option = {recursively, boolean()}
See external documentation.

gETCount(This) -> integer()
Types:
    This = wxTreeCtrl()
See external documentation.

gETEditControl(This) -> wxTextCtrl:wxTextCtrl()
Types:
    This = wxTreeCtrl()
See external documentation.
getFirstChild(This, Item) -> Result
Types:
  Result = {Res::integer(), Cookie::integer()}
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

getNextChild(This, Item, Cookie) -> Result
Types:
  Result = {Res::integer(), Cookie::integer()}
  This = wxTreeCtrl()
  Item = integer()
  Cookie = integer()
See external documentation.

getFirstVisibleItem(This) -> integer()
Types:
  This = wxTreeCtrl()
See external documentation.

getImageList(This) -> wxImageList:wxImageList()
Types:
  This = wxTreeCtrl()
See external documentation.

getIndent(This) -> integer()
Types:
  This = wxTreeCtrl()
See external documentation.

getItemBackgroundColour(This, Item) -> wx:wx_colour4()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

ggetItemData(This, Item) -> term()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.
get_ItemFont(This, Item) -> wxFont:wxFont()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

cgetItemImage(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

cgetItemImage(This, Item, Options::[Option]) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
  Option = {which, wx:wx_enum()}
See external documentation.
Which = ?wxTreeItemIcon_Normal | ?wxTreeItemIcon_Selected | ?wxTreeItemIcon_Expanded | ?wxTreeItemIcon_SelectedExpanded | ?wxTreeItemIcon_Max

cgetItemText(This, Item) -> unicode:charlist()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

cgetItemTextColour(This, Item) -> wx:wx_colour4()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

cgetLastChild(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

cgetNextSibling(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

getNextVisible(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

ggetItemParent(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

getPrevSibling(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

getPrevVisible(This, Item) -> integer()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

getRootItem(This) -> integer()
Types:
  This = wxTreeCtrl()
See external documentation.

getSelection(This) -> integer()
Types:
  This = wxTreeCtrl()
See external documentation.

gGetSelections(This) -> Result
Types:
  Result = {Res::integer(), Val::[integer()]}  
  This = wxTreeCtrl()
See external documentation.
**wxTreeCtrl**

getStateImageList(This) -> wxImageList:wxImageList()

Types:

\[
\text{This} = \text{wxTreeCtrl()} \\
\]

See external documentation.

hitTest(This, Point) -> Result

Types:

\[
\text{Result} = \{\text{Res}::\text{integer()}, \text{Flags}::\text{integer()}\} \\
\text{This} = \text{wxTreeCtrl()} \\
\text{Point} = \{\text{X}::\text{integer()}, \text{Y}::\text{integer()}\} \\
\]

See external documentation.

insertItem(This, Parent, Pos, Text) -> integer()

Types:

\[
\text{This} = \text{wxTreeCtrl()} \\
\text{Parent} = \text{integer()} \\
\text{Pos} = \text{integer()} \\
\text{Text} = \text{unicode:chardata()} \\
\]

Equivalent to insertItem(This, Parent, Pos, Text, []).

insertItem(This, Parent, Pos, Text, Options::[Option]) -> integer()

Types:

\[
\text{This} = \text{wxTreeCtrl()} \\
\text{Parent} = \text{integer()} \\
\text{Pos} = \text{integer()} \\
\text{Text} = \text{unicode:chardata()} \\
\text{Option} = \{\text{image}, \text{integer()}\} \mid \{\text{selImage}, \text{integer()}\} \mid \{\text{data}, \text{term()}\} \\
\]

See external documentation.

isBold(This, Item) -> boolean()

Types:

\[
\text{This} = \text{wxTreeCtrl()} \\
\text{Item} = \text{integer()} \\
\]

See external documentation.

isExpanded(This, Item) -> boolean()

Types:

\[
\text{This} = \text{wxTreeCtrl()} \\
\text{Item} = \text{integer()} \\
\]

See external documentation.

isSelected(This, Item) -> boolean()

Types:
isVisible(This, Item) -> boolean()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

currentItemHasChildren(This, Item) -> boolean()
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

isTreeItemIdOk(Id) -> boolean()
Types:
  Id = integer()
See external documentation.

currentParentItemHasChildrenThis, Parent, Text) -> integer()
Types:
  This = wxTreeCtrl()
  Parent = integer()
  Text = unicode:chardata()
Equivalent to prependItem(This, Parent, Text, []).

prependItem(This, Parent, Text, Options::*[Option]) -> integer()
Types:
  This = wxTreeCtrl()
  Parent = integer()
  Text = unicode:chardata()
  Option = {image, integer()} | {selectedImage, integer()} | {data, term()}
See external documentation.

scrollTo(This, Item) -> ok
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.
wxTreeCtrl

selectItem(This, Item) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
See external documentation.

selectItem(This, Item, Options::[Option]) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
    Option = {select, boolean()}
See external documentation.

setIndent(This, Indent) -> ok
Types:
    This = wxTreeCtrl()
    Indent = integer()
See external documentation.

setImageList(This, ImageList) -> ok
Types:
    This = wxTreeCtrl()
    ImageList = wxImageList:wxImageList()
See external documentation.

setItemBackgroundColour(This, Item, Col) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
    Col = wx:wx_colour()
See external documentation.

setItemBold(This, Item) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
Equivalent to setItemBold(This, Item, []).

setItemBold(This, Item, Options::[Option]) -> ok
Types:
    This = wxTreeCtrl()
    Item = integer()
    Option = {bold, boolean()}

See external documentation.

setItemData(This, Item, Data) -> ok

Types:

This = wxTreeCtrl()
Item = integer()
Data = term()

See external documentation.

setItemDropHighlight(This, Item) -> ok

Types:

This = wxTreeCtrl()
Item = integer()

Equivalent to setItemDropHighlight(This, Item, []).

setItemDropHighlight(This, Item, Options::[Option]) -> ok

Types:

This = wxTreeCtrl()
Item = integer()
Option = {highlight, boolean()}

See external documentation.

setItemFont(This, Item, Font) -> ok

Types:

This = wxTreeCtrl()
Item = integer()
Font = wxFont:wxFont()

See external documentation.

setItemHasChildren(This, Item) -> ok

Types:

This = wxTreeCtrl()
Item = integer()

Equivalent to setItemHasChildren(This, Item, []).

setItemHasChildren(This, Item, Options::[Option]) -> ok

Types:

This = wxTreeCtrl()
Item = integer()
Option = {has, boolean()}

See external documentation.
wxTreeCtrl

setItemImage(This, Item, Image) -> ok
Types:
   This = wxTreeCtrl()
   Item = integer()
   Image = integer()
See external documentation.

setItemImage(This, Item, Image, Options::[Option]) -> ok
Types:
   This = wxTreeCtrl()
   Item = integer()
   Image = integer()
   Option = {which, wx:wx_enum()}
See external documentation.
Which = ?wxTreeItemIcon_Normal | ?wxTreeItemIcon_Selected | ?wxTreeItemIcon_Expanded | ?wxTreeItemIcon_SelectedExpanded | ?wxTreeItemIcon_Max

setItemText(This, Item, Text) -> ok
Types:
   This = wxTreeCtrl()
   Item = integer()
   Text = unicode:chardata()
See external documentation.

setItemTextColour(This, Item, Col) -> ok
Types:
   This = wxTreeCtrl()
   Item = integer()
   Col = wx:wx_colour()
See external documentation.

setStateImageList(This, ImageList) -> ok
Types:
   This = wxTreeCtrl()
   ImageList = wxImageList:wxImageList()
See external documentation.

setWindowStyle(This, Styles) -> ok
Types:
   This = wxTreeCtrl()
   Styles = integer()
See external documentation.
sortChildren(This, Item) -> ok
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

toggle(This, Item) -> ok
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

toggleItemSelection(This, Item) -> ok
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

unselect(This) -> ok
Types:
  This = wxTreeCtrl()
See external documentation.

unselectAll(This) -> ok
Types:
  This = wxTreeCtrl()
See external documentation.

unselectItem(This, Item) -> ok
Types:
  This = wxTreeCtrl()
  Item = integer()
See external documentation.

destroy(This::wxTreeCtrl()) -> ok
Destroys this object, do not use object again
wxTreeEvent

Erlang module

See external documentation: **wxTreeEvent**.

Use **wxEvtHandler:connect/3** with EventType:

```
command_tree_begin_drag, command_tree_begin_rdrag, command_tree_begin_label_edit,
command_tree_end_label_edit, command_tree_delete_item, command_tree_get_info,
command_tree_set_info, command_tree_item_expanded, command_tree_item_expanding,
command_tree_item_collapsed, command_tree_item_collapsing, command_tree_sel_changed,
command_tree_sel_changing, command_tree_key_down, command_tree_item_activated,
command_tree_item_right_click, command_tree_item_middle_click, command_tree_end_drag,
command_tree_state_image_click, command_tree_item_gettooltip, command_tree_item_menu
```

See also the message variant #wxTree{} event record type.

This class is derived (and can use functions) from:

**wxNotifyEvent**

**wxCommandEvent**

**wxEvent**

**DATA TYPES**

**wxTreeEvent()**

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

**getKeyCode(This) -> integer()**

Types:

```
This = wxTreeEvent()
```

See external documentation.

**getItem(This) -> integer()**

Types:

```
This = wxTreeEvent()
```

See external documentation.

**getKeyEvent(This) -> wxKeyEvent:wxKeyEvent()**

Types:

```
This = wxTreeEvent()
```

See external documentation.

**getLabel(This) -> unicode:charlist()**

Types:

```
This = wxTreeEvent()
```
See external documentation.

getOldItem(This) -> integer()
Types:
   
   \texttt{This = wxTreeEvent()}
See external documentation.

getPoint(This) -> \{X::integer(), Y::integer()\}
Types:
   
   \texttt{This = wxTreeEvent()}
See external documentation.

isEditCancelled(This) -> boolean()
Types:
   
   \texttt{This = wxTreeEvent()}
See external documentation.

setToolTip(This, ToolTip) -> ok
Types:
   
   \texttt{This = wxTreeEvent()}
   
   \texttt{ToolTip = unicode:chardata()}
See external documentation.
wxTreebook

Erlang module

See external documentation: **wxTreebook**.

This class is derived (and can use functions) from:

- `wxControl`
- `wxWindow`
- `wxEvtHandler`

**DATA TYPES**

**wxTreebook()**

An object reference, The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

- `new() -> wxTreebook()`  
  See external documentation.

- `new(Parent, Id) -> wxTreebook()`  
  Types:
  ```erlang
  Parent = wxWindow:wxWindow()
  Id = integer()
  ```
  Equivalent to `new(Parent, Id, [])`.

- `new(Parent, Id, Options::[Option]) -> wxTreebook()`  
  Types:
  ```erlang
  Parent = wxWindow:wxWindow()
  Id = integer()
  Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
  ```
  See external documentation.

- `addPage(This, Page, Text) -> boolean()`  
  Types:
  ```erlang
  This = wxTreebook()
  Page = wxWindow:wxWindow()
  Text = unicode:chardata()
  ```
  Equivalent to `addPage(This, Page, Text, [])`.

- `addPage(This, Page, Text, Options::[Option]) -> boolean()`  
  Types:
  ```erlang
  ```
This = wxTreebook()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

advanceSelection(This) -> ok
Types:
    This = wxTreebook()
Equivalent to advanceSelection(This, []).

advanceSelection(This, Options::[Option]) -> ok
Types:
    This = wxTreebook()
    Option = {forward, boolean()}
See external documentation.

assignImageList(This, ImageList) -> ok
Types:
    This = wxTreebook()
    ImageList = wxImageList:wxImageList()
See external documentation.

create(This, Parent, Id) -> boolean()
Types:
    This = wxTreebook()
    Parent = wxWindow:wxWindow()
    Id = integer()
Equivalent to create(This, Parent, Id, []).

create(This, Parent, Id, Options::[Option]) -> boolean()
Types:
    This = wxTreebook()
    Parent = wxWindow:wxWindow()
    Id = integer()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(), H::integer()}} | {style, integer()}
See external documentation.

deleteAllPages(This) -> boolean()
Types:
    This = wxTreebook()
See external documentation.
deletePage(This, Pos) -> boolean()
Types:
    This = wxTreebook()
    Pos = integer()
See external documentation.

removePage(This, N) -> boolean()
Types:
    This = wxTreebook()
    N = integer()
See external documentation.

gGetCurrentPage(This) -> wxWindow:wxWindow()
Types:
    This = wxTreebook()
See external documentation.

getImageList(This) -> wxImageList:wxImageList()
Types:
    This = wxTreebook()
See external documentation.

getPage(This, N) -> wxWindow:wxWindow()
Types:
    This = wxTreebook()
    N = integer()
See external documentation.

getPageCount(This) -> integer()
Types:
    This = wxTreebook()
See external documentation.

getPageImage(This, N) -> integer()
Types:
    This = wxTreebook()
    N = integer()
See external documentation.

getPageText(This, N) -> unicode:charlist()
Types:
    This = wxTreebook()
    N = integer()
See external documentation.

getSelection(This) -> integer()
Types:
   This = wxTreebook()

See external documentation.

expandNode(This, Pos) -> boolean()
Types:
   This = wxTreebook()
   Pos = integer()
Equivalent to expandNode(This, Pos, []).

expandNode(This, Pos, Options::[Option]) -> boolean()
Types:
   This = wxTreebook()
   Pos = integer()
   Option = {expand, boolean()}
See external documentation.

isNodeExpanded(This, Pos) -> boolean()
Types:
   This = wxTreebook()
   Pos = integer()
See external documentation.

hitTest(This, Pt) -> Result
Types:
   Result = {Res::integer(), Flags::integer()}
   This = wxTreebook()
   Pt = {X::integer(), Y::integer()}
See external documentation.

insertPage(This, Pos, Page, Text) -> boolean()
Types:
   This = wxTreebook()
   Pos = integer()
   Page = wxWindow:wxWindow()
   Text = unicode:chardata()
Equivalent to insertPage(This, Pos, Page, Text, []).

insertPage(This, Pos, Page, Text, Options::[Option]) -> boolean()
Types:
This = wxTreebook()
Pos = integer()
Page = wxWindow:wxWindow()
Text = unicode:chardata()
Option = {bSelect, boolean()} | {imageId, integer()}

See external documentation.

insertSubPage(This, Pos, Page, Text) -> boolean()
Types:
  This = wxTreebook()
  Pos = integer()
  Page = wxWindow:wxWindow()
  Text = unicode:chardata()
Equivalent to insertSubPage(This, Pos, Page, Text, []).

insertSubPage(This, Pos, Page, Text, Options::[Option]) -> boolean()
Types:
  This = wxTreebook()
  Pos = integer()
  Page = wxWindow:wxWindow()
  Text = unicode:chardata()
  Option = {bSelect, boolean()} | {imageId, integer()}
See external documentation.

setImageList(This, ImageList) -> ok
Types:
  This = wxTreebook()
  ImageList = wxImageList:wxImageList()
See external documentation.

setPageSize(This, Size) -> ok
Types:
  This = wxTreebook()
  Size = {W::integer(), H::integer()}
See external documentation.

setPageImage(This, N, ImageId) -> boolean()
Types:
  This = wxTreebook()
  N = integer()
  ImageId = integer()
See external documentation.
setPageText(This, N, StrText) -> boolean()
Types:
   This = wxTreebook()
   N = integer()
   StrText = unicode:chardata()
See external documentation.

setSelection(This, N) -> integer()
Types:
   This = wxTreebook()
   N = integer()
See external documentation.

changeSelection(This, N) -> integer()
Types:
   This = wxTreebook()
   N = integer()
See external documentation.

destroy(This::wxTreebook()) -> ok
Destroys this object, do not use object again
wxUpdateUIEvent

Erlang module

See external documentation: `wxUpdateUIEvent`.

Use `wxEvtHandler:connect/3` with EventType:

    update_ui

See also the message variant `#wxUpdateUI{}` event record type.

This class is derived (and can use functions) from:

    wxCommandEvent
    wxEvent

**DATA TYPES**

`wxUpdateUIEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

**Exports**

`canUpdate(Win) -> boolean()`

Types:

    Win = `wxWindow:wxWindow()`

See [external documentation](#).

`check(This, Check) -> ok`

Types:

    This = `wxUpdateUIEvent()`
    Check = `boolean()`

See [external documentation](#).

`enable(This, Enable) -> ok`

Types:

    This = `wxUpdateUIEvent()`
    Enable = `boolean()`

See [external documentation](#).

`show(This, Show) -> ok`

Types:

    This = `wxUpdateUIEvent()`
    Show = `boolean()`

See [external documentation](#).
setChecked(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getEnabled(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getShown(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getSetChecked(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getSetEnabled(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getSetShown(This) -> boolean()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

setText(This) -> unicode:charlist()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getText(This) -> unicode:charlist()
Types:
  \texttt{This} = \texttt{wxUpdateUIEvent()}
See external documentation.

getMode() -> \texttt{wx:wx_enum()}
See external documentation.
Res = \texttt {?wxUPDATE_UI_PROCESS_ALL | ?wxUPDATE_UI_PROCESS_SPECIFIED}
wxUpdateUIEvent

getUpdateInterval() -> integer()
See external documentation.

resetUpdateTime() -> ok
See external documentation.

setMode(Mode) -> ok
Types:
  Mode = wx:wx_enum()
See external documentation.
Mode = ?wxUPDATE_UI_PROCESS_ALL | ?wxUPDATE_UI_PROCESS_SPECIFIED

setText(This, Text) -> ok
Types:
  This = wxUpdateUIEvent()
  Text = unicode:chardata()
See external documentation.

setUpdateInterval(UpdateInterval) -> ok
Types:
  UpdateInterval = integer()
See external documentation.
wxWindow

Erlang module

See external documentation: wxWindow.
This class is derived (and can use functions) from:
wxEvtHandler

DATA TYPES
wxWindow()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxWindow()
See external documentation.

cacheBestSize(This, Size) -> ok
Types:
    This = wxWindow()
    Size = {W::integer(), H::integer()}
See external documentation.

captureMouse(This) -> ok
Types:
    This = wxWindow()
See external documentation.

class derives from:
wxEvtHandler

Exports

new() -> wxWindow()
new(Parent, Id) -> wxWindow()
new(Parent, Id, Options::[Option]) -> wxWindow()
cacheBestSize(This, Size) -> ok
captureMouse(This) -> ok

DATA TYPES
wxWindow()

wxWindow()

An object reference. The representation is internal and can be changed without notice. It can’t be used for
comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxWindow()
See external documentation.

new(Parent, Id) -> wxWindow()
Types:
    Parent = wxWindow()
    Id = integer()
Equivalent to new(Parent, Id, []).

new(Parent, Id, Options::[Option]) -> wxWindow()
Types:
    Parent = wxWindow()
    Id = integer()
    Option = {pos, {X::integer(), Y::integer()}} | {size, {W::integer(),
H::integer()}} | {style, integer()}
See external documentation.

cacheBestSize(This, Size) -> ok
Types:
    This = wxWindow()
    Size = {W::integer(), H::integer()}
See external documentation.

captureMouse(This) -> ok
Types:
    This = wxWindow()
See external documentation.
center(This) -> ok
Types:
   This = wxWindow()
Equivalent to center(This, []).

center(This, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Option = {dir, integer()}
See external documentation.

centerOnParent(This) -> ok
Types:
   This = wxWindow()
Equivalent to centerOnParent(This, []).

centerOnParent(This, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Option = {dir, integer()}
See external documentation.

centre(This) -> ok
Types:
   This = wxWindow()
Equivalent to centre(This, []).

centre(This, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Option = {dir, integer()}
See external documentation.

centreOnParent(This) -> ok
Types:
   This = wxWindow()
Equivalent to centreOnParent(This, []).

centreOnParent(This, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Option = {dir, integer()}
See external documentation.
clearBackground(This) -> ok
Types:
    This = wxWindow()
See external documentation.

clientToScreen(This, Pt) -> {X::integer(), Y::integer()}
Types:
    This = wxWindow()
    Pt = {X::integer(), Y::integer()}
See external documentation.

clientToScreen(This, X, Y) -> {X::integer(), Y::integer()}
Types:
    This = wxWindow()
    X = integer()
    Y = integer()
See external documentation.

close(This) -> boolean()
Types:
    This = wxWindow()
Equivalent to close(This, []).

close(This, Options::[Option]) -> boolean()
Types:
    This = wxWindow()
    Option = {force, boolean()}
See external documentation.

convertDialogToPixels(This, Sz) -> {W::integer(), H::integer()}
Types:
    This = wxWindow()
    Sz = {W::integer(), H::integer()}
See external documentation.

convertPixelsToDialog(This, Sz) -> {W::integer(), H::integer()}
Types:
    This = wxWindow()
    Sz = {W::integer(), H::integer()}
See external documentation.

Destroy(This) -> boolean()
Types:
This = \texttt{wxWindow()}

See external documentation.

destroyChildren(This) -&gt; boolean()

Types:
   \texttt{This = \texttt{wxWindow()}}

See external documentation.

disable(This) -&gt; boolean()

Types:
   \texttt{This = \texttt{wxWindow()}}

See external documentation.

dragAcceptFiles(This, Accept) -&gt; ok

Types:
   \texttt{This = \texttt{wxWindow()}}
   \texttt{Accept = boolean()}

See external documentation.

enable(This) -&gt; boolean()

Types:
   \texttt{This = \texttt{wxWindow()}}

Equivalent to \texttt{enable(This, [])}.

enable(This, Options::[Option]) -&gt; boolean()

Types:
   \texttt{This = \texttt{wxWindow()}}
   \texttt{Option = \{enable, boolean()\}}

See external documentation.

findFocus() -&gt; \texttt{wxWindow()}

See external documentation.

findWindow(This, Winid) -&gt; \texttt{wxWindow()}

Types:
   \texttt{This = \texttt{wxWindow()}}
   \texttt{Winid = integer()}

See external documentation.

Also:
findWindow(This, Name) -&gt; \texttt{wxWindow()} when
\texttt{This::wxWindow(), Name::unicode:chardata()}. 
findWindowById(Winid) -> wxWindow()
Types:
    Winid = integer()
Equivalent to findWindowById(Winid, []).

findWindowById(Winid, Options::[Option]) -> wxWindow()
Types:
    Winid = integer()
    Option = {parent, wxWindow()}
See external documentation.

findWindowByName(Name) -> wxWindow()
Types:
    Name = unicode:chardata()
Equivalent to findWindowByName(Name, []).

findWindowByName(Name, Options::[Option]) -> wxWindow()
Types:
    Name = unicode:chardata()
    Option = {parent, wxWindow()}
See external documentation.

findWindowByLabel(Label) -> wxWindow()
Types:
    Label = unicode:chardata()
Equivalent to findWindowByLabel(Label, []).

findWindowByLabel(Label, Options::[Option]) -> wxWindow()
Types:
    Label = unicode:chardata()
    Option = {parent, wxWindow()}
See external documentation.

fit(This) -> ok
Types:
    This = wxWindow()
See external documentation.

fitInside(This) -> ok
Types:
    This = wxWindow()
See external documentation.
freeze(This) -> ok
Types:
  This = wxWindow()
See external documentation.

getAcceleratorTable(This) -> wxAcceleratorTable:wxAcceleratorTable()
Types:
  This = wxWindow()
See external documentation.

getBackgroundColour(This) -> wx:wx_colour4()
Types:
  This = wxWindow()
See external documentation.

getBackgroundStyle(This) -> wx:wx_enum()
Types:
  This = wxWindow()
See external documentation.

Res = ?wxBG_STYLE_SYSTEM | ?wxBG_STYLE_COLOUR | ?wxBG_STYLE_CUSTOM

giveBestSize(This) -> {W::integer(), H::integer()}
Types:
  This = wxWindow()
See external documentation.

getCaret(This) -> wxCaret:wxCaret()
Types:
  This = wxWindow()
See external documentation.

getCapture() -> wxWindow()
See external documentation.

getCharHeight(This) -> integer()
Types:
  This = wxWindow()
See external documentation.

getCharWidth(This) -> integer()
Types:
  This = wxWindow()
See external documentation.
getChildren(This) -> [wxWindow()]
Types:
  T = wxWindow()
See external documentation.

clientsize(This) -> {W::integer(), H::integer()}
Types:
  T = wxWindow()
See external documentation.

clientsizer(This) -> wxSizer:wxSizer()
Types:
  T = wxWindow()
See external documentation.

cursor(This) -> wxCursor:wxCursor()
Types:
  T = wxWindow()
See external documentation.

dropTarget(This) -> wx::wx_object()
Types:
  T = wxWindow()
See external documentation.

eventHandler(This) -> wxEvtHandler:wxEvtHandler()
Types:
  T = wxWindow()
See external documentation.

extraStyle(This) -> integer()
Types:
  T = wxWindow()
See external documentation.

font(This) -> wxFont:wxFont()
Types:
  T = wxWindow()
See external documentation.

gtForegroundColour(This) -> wx:wx_colour4()
Types:
  T = wxWindow()
See external documentation.

getGrandParent(This) -> wxWindow()
Types:

This = wxWindow()
See external documentation.

getHandle(This) -> integer()
Types:

This = wxWindow()
See external documentation.

getHelpText(This) -> unicode:charlist()
Types:

This = wxWindow()
See external documentation.

getId(This) -> integer()
Types:

This = wxWindow()
See external documentation.

getLabel(This) -> unicode:charlist()
Types:

This = wxWindow()
See external documentation.

getMaxSize(This) -> {W::integer(), H::integer()}
Types:

This = wxWindow()
See external documentation.

getMinSize(This) -> {W::integer(), H::integer()}
Types:

This = wxWindow()
See external documentation.

getName(This) -> unicode:charlist()
Types:

This = wxWindow()
See external documentation.
getParent(This) -> wxWindow()
Types:
   This = wxWindow()
See external documentation.

getPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxWindow()
See external documentation.

getRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
   This = wxWindow()
See external documentation.

getScreenPosition(This) -> {X::integer(), Y::integer()}
Types:
   This = wxWindow()
See external documentation.

getScreenRect(This) -> {X::integer(), Y::integer(), W::integer(), H::integer()}
Types:
   This = wxWindow()
See external documentation.

getScrollPos(This, Orient) -> integer()
Types:
   This = wxWindow()
   Orient = integer()
See external documentation.

getScrollRange(This, Orient) -> integer()
Types:
   This = wxWindow()
   Orient = integer()
See external documentation.

getScrollThumb(This, Orient) -> integer()
Types:
   This = wxWindow()
   Orient = integer()
See external documentation.
**wxWindow**

**getSize(This) -> {W::integer(), H::integer()}
Types:**

\[
This = \text{wxWindow()}
\]

See external documentation.

**getSizer(This) -> wxSizer:wxSizer()
Types:**

\[
This = \text{wxWindow()}
\]

See external documentation.

**getTextExtent(This, String) -> Result
Types:**

\[
Result = \{X::integer(), Y::integer(), \text{Descent::integer()}, \text{ExternalLeading::integer()}
This = \text{wxWindow()}
String = \text{unicode:chardata()}
\]

Equivalent to `getTextExtent(This, String, [])`.

**getTextExtent(This, String, Options::[Option]) -> Result
Types:**

\[
Result = \{X::integer(), Y::integer(), \text{Descent::integer()}, \text{ExternalLeading::integer()}
This = \text{wxWindow()}
String = \text{unicode:chardata()}
Option = \{\text{theFont}, \text{wxFont:wxFont()}
\]

See external documentation.

**getToolTip(This) -> wxToolTip:wxToolTip()
Types:**

\[
This = \text{wxWindow()}
\]

See external documentation.

**getUpdateRegion(This) -> wxRegion:wxRegion()
Types:**

\[
This = \text{wxWindow()}
\]

See external documentation.

**getVirtualSize(This) -> {W::integer(), H::integer()}
Types:**

\[
This = \text{wxWindow()}
\]

See external documentation.
getWindowStyleFlag(This) -> integer()
Types:
   This = wxWindow()
See external documentation.

getWindowVariant(This) -> wx:wx_enum()
Types:
   This = wxWindow()
See external documentation.

hasCapture(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

hasScrollbar(This, Orient) -> boolean()
Types:
   This = wxWindow()
   Orient = integer()
See external documentation.

hasTransparentColorBackground(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

hide(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

inheritAttributes(This) -> ok
Types:
   This = wxWindow()
See external documentation.

initDialog(This) -> ok
Types:
   This = wxWindow()
See external documentation.
invalidateBestSize(This) -> ok
Types:
   This = wxWindow()
See external documentation.

isEnabled(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

isExposed(This, Pt) -> boolean()
Types:
   This = wxWindow()
   Pt = {X::integer(), Y::integer()}
See external documentation.
Also:
isExposed(This, Rect) -> boolean() when
This::wxWindow(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.  

isExposed(This, X, Y) -> boolean()
Types:
   This = wxWindow()
   X = integer()
   Y = integer()
See external documentation.

isExposed(This, X, Y, W, H) -> boolean()
Types:
   This = wxWindow()
   X = integer()
   Y = integer()
   W = integer()
   H = integer()
See external documentation.

isRetained(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

isShown(This) -> boolean()
Types:
   This = wxWindow()
See external documentation.

isVisible(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

isShownOnScreen(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

layout(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

lineDown(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

lineUp(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

lower(This) -> ok
Types:
  This = wxWindow()
See external documentation.

makeModal(This) -> ok
Types:
  This = wxWindow()
Equivalent to makeModal(This, []).

makeModal(This, Options::[Option]) -> ok
Types:
  This = wxWindow()
  Option = {modal, boolean()}
See external documentation.
move(This, Pt) -> ok
Types:
  This = wxWindow()
  Pt = {X::integer(), Y::integer()}
Equivalent to move(This, Pt, []).

move(This, X, Y) -> ok
Types:
  This = wxWindow()
  X = integer()
  Y = integer()
See external documentation.
Also:
move(This, Pt, [Option]) -> 'ok' when
This::wxWindow(), Pt::{X::integer(), Y::integer()},
Option :: ['flags', integer()].

move(This, X, Y, Options::[Option]) -> ok
Types:
  This = wxWindow()
  X = integer()
  Y = integer()
  Option = {flags, integer()}
See external documentation.

moveAfterInTabOrder(This, Win) -> ok
Types:
  This = wxWindow()
  Win = wxWindow()
See external documentation.

moveBeforeInTabOrder(This, Win) -> ok
Types:
  This = wxWindow()
  Win = wxWindow()
See external documentation.

navigate(This) -> boolean()
Types:
  This = wxWindow()
Equivalent to navigate(This, []).

navigate(This, Options::[Option]) -> boolean()
Types:
This = wxWindow()
Option = {flags, integer()}

See external documentation.

pageDown(This) -> boolean()
Types:
  This = wxWindow()

See external documentation.

pageUp(This) -> boolean()
Types:
  This = wxWindow()

See external documentation.

popEventHandler(This) -> wxEvtHandler:wxEvtHandler()
Types:
  This = wxWindow()

Equivalent to popEventHandler(This, []).

popEventHandler(This, Options::[Option]) -> wxEvtHandler:wxEvtHandler()
Types:
  This = wxWindow()
  Option = {deleteHandler, boolean()}

See external documentation.

popupMenu(This, Menu) -> boolean()
Types:
  This = wxWindow()
  Menu = wxMenu:wxMenu()

Equivalent to popupMenu(This, Menu, []).

popupMenu(This, Menu, Options::[Option]) -> boolean()
Types:
  This = wxWindow()
  Menu = wxMenu:wxMenu()
  Option = {pos, {X::integer(), Y::integer()}}

See external documentation.

popupMenu(This, Menu, X, Y) -> boolean()
Types:
  This = wxWindow()
  Menu = wxMenu:wxMenu()
  X = integer()
Y = integer()
See external documentation.

raise(This) -> ok
Types:
  This = wxWindow()
See external documentation.

refresh(This) -> ok
Types:
  This = wxWindow()
Equivalent to refresh(This, []).

refresh(This, Options::[Option]) -> ok
Types:
  This = wxWindow()
  Option = {eraseBackground, boolean()} | {rect, {X::integer(), Y::integer(), W::integer(), H::integer()}}
See external documentation.

refreshRect(This, Rect) -> ok
Types:
  This = wxWindow()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
Equivalent to refreshRect(This, Rect, []).

refreshRect(This, Rect, Options::[Option]) -> ok
Types:
  This = wxWindow()
  Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
  Option = {eraseBackground, boolean()}
See external documentation.

releaseMouse(This) -> ok
Types:
  This = wxWindow()
See external documentation.

removeChild(This, Child) -> ok
Types:
  This = wxWindow()
  Child = wxWindow()
See external documentation.
reparent(This, NewParent) -> boolean()
Types:
  This = wxWindow()
  NewParent = wxWindow()
See external documentation.

screenToClient(This) -> {X::integer(), Y::integer()}
Types:
  This = wxWindow()
See external documentation.

screenToClient(This, Pt) -> {X::integer(), Y::integer()}
Types:
  This = wxWindow()
  Pt = {X::integer(), Y::integer()}
See external documentation.

scrollLines(This, Lines) -> boolean()
Types:
  This = wxWindow()
  Lines = integer()
See external documentation.

scrollPages(This, Pages) -> boolean()
Types:
  This = wxWindow()
  Pages = integer()
See external documentation.

scrollWindow(This, Dx, Dy) -> ok
Types:
  This = wxWindow()
  Dx = integer()
  Dy = integer()
Equivalent to scrollWindow(This, Dx, Dy, []).

scrollWindow(This, Dx, Dy, Options::[Option]) -> ok
Types:
  This = wxWindow()
  Dx = integer()
  Dy = integer()
  Option = {rect, {X::integer(), Y::integer(), W::integer(), H::integer()}}
See external documentation.
setAcceleratorTable(This, Accel) -> ok
Types:
  This = wxWindow()
  Accel = wxAcceleratorTable:wxAcceleratorTable()
See external documentation.

setAutoLayout(This, AutoLayout) -> ok
Types:
  This = wxWindow()
  AutoLayout = boolean()
See external documentation.

setBackgroundColour(This, Colour) -> boolean()
Types:
  This = wxWindow()
  Colour = wx:wx_colour()
See external documentation.

setBackgroundStyle(This, Style) -> boolean()
Types:
  This = wxWindow()
  Style = wx:wx_enum()
See external documentation.
Style = ?wxBG_STYLE_SYSTEM | ?wxBG_STYLE_COLOUR | ?wxBG_STYLE_CUSTOM

setCaret(This, Caret) -> ok
Types:
  This = wxWindow()
  Caret = wxCaret:wxCaret()
See external documentation.

setClientSize(This, Size) -> ok
Types:
  This = wxWindow()
  Size = {W::integer(), H::integer()}
See external documentation.
Also:
setClientSize(This, Rect) -> 'ok' when
This::wxWindow(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()}.  

setClientSize(This, Width, Height) -> ok
Types:
  This = wxWindow()
  Width = integer()
Height = integer()
See external documentation.

setContainingSizer(This, Sizer) -> ok
Types:
    This = wxWindow()
    Sizer = wxSizer:wxSizer()
See external documentation.

setCursor(This, Cursor) -> boolean()
Types:
    This = wxWindow()
    Cursor = wxCursor:wxCursor()
See external documentation.

setMaxSize(This, MaxSize) -> ok
Types:
    This = wxWindow()
    MaxSize = {W::integer(), H::integer()}
See external documentation.

setMinSize(This, MinSize) -> ok
Types:
    This = wxWindow()
    MinSize = {W::integer(), H::integer()}
See external documentation.

setOwnBackgroundColour(This, Colour) -> ok
Types:
    This = wxWindow()
    Colour = wx:wx_colour()
See external documentation.

setOwnFont(This, Font) -> ok
Types:
    This = wxWindow()
    Font = wxFont:wxFont()
See external documentation.

setOwnForegroundColour(This, Colour) -> ok
Types:
    This = wxWindow()
    Colour = wx:wx_colour()
See external documentation.

setDropTarget(This, DropTarget) -> ok  
Types:  
  This = wxWindow()  
  DropTarget = wx:wx_object()  
See external documentation.

setExtraStyle(This, ExStyle) -> ok  
Types:  
  This = wxWindow()  
  ExStyle = integer()  
See external documentation.

setFocus(This) -> ok  
Types:  
  This = wxWindow()  
See external documentation.

setFocusFromKbd(This) -> ok  
Types:  
  This = wxWindow()  
See external documentation.

setFont(This, Font) -> boolean()  
Types:  
  This = wxWindow()  
  Font = wxFont:wxFont()  
See external documentation.

setForegroundColour(This, Colour) -> boolean()  
Types:  
  This = wxWindow()  
  Colour = wx:wx_colour()  
See external documentation.

setHelpText(This, Text) -> ok  
Types:  
  This = wxWindow()  
  Text = unicode:chardata()  
See external documentation.
setId(This, Winid) -> ok
Types:
   This = wxWindow()
   Winid = integer()
See external documentation.

setLabel(This, Label) -> ok
Types:
   This = wxWindow()
   Label = unicode:chardata()
See external documentation.

setName(This, Name) -> ok
Types:
   This = wxWindow()
   Name = unicode:chardata()
See external documentation.

setPalette(This, Pal) -> ok
Types:
   This = wxWindow()
   Pal = wxPalette:wxPalette()
See external documentation.

setScrollbar(This, Orient, Pos, ThumbVisible, Range) -> ok
Types:
   This = wxWindow()
   Orient = integer()
   Pos = integer()
   ThumbVisible = integer()
   Range = integer()
Equivalent to setScrollbar(This, Orient, Pos, ThumbVisible, Range, []).

setScrollbar(This, Orient, Pos, ThumbVisible, Range, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Orient = integer()
   Pos = integer()
   ThumbVisible = integer()
   Range = integer()
   Option = {refresh, boolean()}
See external documentation.
setScrollPos(This, Orient, Pos) -> ok
Types:
   This = wxWindow()
   Orient = integer()
   Pos = integer()
Equivalent to setScrollPos(This, Orient, Pos, []).

setScrollPos(This, Orient, Pos, Options::[Option]) -> ok
Types:
   This = wxWindow()
   Orient = integer()
   Pos = integer()
   Option = {refresh, boolean()}
See external documentation.

setSize(This, Rect) -> ok
Types:
   This = wxWindow()
   Rect = {X::integer(), Y::integer(), W::integer(), H::integer()}
See external documentation.
Also:
setSize(This, Size) -> 'ok' when
This::wxWindow(), Size::{W::integer(), H::integer()}.  

setSize(This, Width, Height) -> ok
Types:
   This = wxWindow()
   Width = integer()
   Height = integer()
See external documentation.
Also:
setSize(This, Rect, [Option]) -> 'ok' when
This::wxWindow(), Rect::{X::integer(), Y::integer(), W::integer(), H::integer()},
Option :: ['sizeFlags', integer()].

setSize(This, X, Y, Width, Height) -> ok
Types:
   This = wxWindow()
   X = integer()
   Y = integer()
   Width = integer()
   Height = integer()
Equivalent to setSize(This, X, Y, Width, Height, []).
setSize(This, X, Y, Width, Height, Options::[Option]) -> ok

Types:
   This = wxWindow()
   X = integer()
   Y = integer()
   Width = integer()
   Height = integer()
   Option = {sizeFlags, integer()}

See external documentation.

setSizeHints(This, MinSize) -> ok

Types:
   This = wxWindow()
   MinSize = {W::integer(), H::integer()}

Equivalent to setSizeHints(This, MinSize, []).

setSizeHints(This, MinW, MinH) -> ok

Types:
   This = wxWindow()
   MinW = integer()
   MinH = integer()

See external documentation.

Also:
setSizeHints(This, MinSize, [Option]) -> 'ok' when
This::wxWindow(), MinSize::[W::integer(), H::integer()],
Option :: [{'maxSize', [W::integer(), H::integer()]}
    | [{'incSize', [W::integer(), H::integer()]}].

setSizeHints(This, MinW, MinH, Options::[Option]) -> ok

Types:
   This = wxWindow()
   MinW = integer()
   MinH = integer()
   Option = {maxW, integer()} | {maxH, integer()} | {incW, integer()} |
           {incH, integer()}

See external documentation.

setSizeSizer(This, Sizer) -> ok

Types:
   This = wxWindow()
   Sizer = wxSizer:wxSizer()

Equivalent to setSizeSizer(This, Sizer, []).
setSizer(This, Sizer, Options:=[Option]) -> ok
Types:
   This = wxWindow()
   Sizer = wxSizer:wxSizer()
   Option = [deleteOld, boolean()]
See external documentation.

setSizerAndFit(This, Sizer) -> ok
Types:
   This = wxWindow()
   Sizer = wxSizer:wxSizer()
Equivalent to setSizerAndFit(This, Sizer, []).

setSizerAndFit(This, Sizer, Options:=[Option]) -> ok
Types:
   This = wxWindow()
   Sizer = wxSizer:wxSizer()
   Option = [deleteOld, boolean()]
See external documentation.

setThemeEnabled(This, EnableTheme) -> ok
Types:
   This = wxWindow()
   EnableTheme = boolean()
See external documentation.

setToolTip(This, Tip) -> ok
Types:
   This = wxWindow()
   Tip = unicode:chardata()
See external documentation.
Also:
setToolTip(This, Tip) -> 'ok' when
This::wxWindow(), Tip::wxToolTip:wxToolTip().

setVirtualSize(This, Size) -> ok
Types:
   This = wxWindow()
   Size = [W::integer(), H::integer()]
See external documentation.

setVirtualSize(This, X, Y) -> ok
Types:
This = \texttt{wxWindow()}
X = \texttt{integer()}
Y = \texttt{integer()}

See external documentation.

\texttt{setVirtualSizeHints(This, MinSize)} -> ok

Types:
\begin{verbatim}
This = \texttt{wxWindow()}
MinSize = \{W::integer(), H::integer()\}
\end{verbatim}

Equivalent to \texttt{setVirtualSizeHints(This, MinSize, [])}.

\texttt{setVirtualSizeHints(This, MinW, MinH)} -> ok

Types:
\begin{verbatim}
This = \texttt{wxWindow()}
MinW = \texttt{integer()}
MinH = \texttt{integer()}
\end{verbatim}

See external documentation.

Also:
\texttt{setVirtualSizeHints(This, MinSize, [Option])} -> 'ok' when
This::\texttt{wxWindow()}, MinSize::\{W::integer(), H::integer()\},
Option :: \{'maxSize', \{W::integer(), H::integer()\}\}.

\texttt{setVirtualSizeHints(This, MinW, MinH, Options::[Option])} -> ok

Types:
\begin{verbatim}
This = \texttt{wxWindow()}
MinW = \texttt{integer()}
MinH = \texttt{integer()}
Option = \{'maxW', \texttt{integer()}\} | \{'maxH', \texttt{integer()}\}
\end{verbatim}

See external documentation.

\texttt{setWindowStyle(This, Style)} -> ok

Types:
\begin{verbatim}
This = \texttt{wxWindow()}
Style = \texttt{integer()}
\end{verbatim}

See external documentation.

\texttt{setWindowStyleFlag(This, Style)} -> ok

Types:
\begin{verbatim}
This = \texttt{wxWindow()}
Style = \texttt{integer()}
\end{verbatim}

See external documentation.

\texttt{setWindowVariant(This, Variant)} -> ok

Types:
This = wxWindow()
Variant = wx:wx_enum()

See external documentation.


shouldInheritColours(This) -> boolean()

Types:
    This = wxWindow()

See external documentation.

show(This) -> boolean()

Types:
    This = wxWindow()

Equivalent to show(This, []).

show(This, Options::[Option]) -> boolean()

Types:
    This = wxWindow()
    Option = {show, boolean()}

See external documentation.

thaw(This) -> ok

Types:
    This = wxWindow()

See external documentation.

transferDataFromWindow(This) -> boolean()

Types:
    This = wxWindow()

See external documentation.

transferDataToWindow(This) -> boolean()

Types:
    This = wxWindow()

See external documentation.

update(This) -> ok

Types:
    This = wxWindow()

See external documentation.
updateWindowUI(This) -> ok
Types:
  This = wxWindow()
Equivalent to updateWindowUI(This, []).

updateWindowUI(This, Options::[Option]) -> ok
Types:
  This = wxWindow()
  Option = {flags, integer()}
See external documentation.

validate(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

warpPointer(This, X, Y) -> ok
Types:
  This = wxWindow()
  X = integer()
  Y = integer()
See external documentation.

setTransparent(This, Alpha) -> boolean()
Types:
  This = wxWindow()
  Alpha = integer()
See external documentation.

canSetTransparent(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

isDoubleBuffered(This) -> boolean()
Types:
  This = wxWindow()
See external documentation.

setDoubleBuffered(This, On) -> ok
Types:
  This = wxWindow()
  On = boolean()
See external documentation.

g getContentScaleFactor(This) -> number()
Types:
   This = wxWindow()
See external documentation.

g getDPI(This) -> {W::integer(), H::integer()}
Types:
   This = wxWindow()
See external documentation.

g fromDIP(This, Sz) -> {W::integer(), H::integer()}
Types:
   This = wxWindow()
   Sz = {W::integer(), H::integer()}
See external documentation.

g toDIP(This, Sz) -> {W::integer(), H::integer()}
Types:
   This = wxWindow()
   Sz = {W::integer(), H::integer()}
See external documentation.

d destroy(This::wxWindow()) -> ok
Destroys this object, do not use object again
wxWindowCreateEvent

Erlang module

See external documentation: `wxWindowCreateEvent`.

Use `wxEvtHandler:connect/3` with EventType:

```
create
```

See also the message variant `#wxWindowCreate{}` event record type.

This class is derived (and can use functions) from:

```
wxCommandEvent
wxEvent
```

**DATA TYPES**

`wxWindowCreateEvent()`

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.
wxWindowDC

Erlang module

See external documentation: wxWindowDC.

This class is derived (and can use functions) from:
wxDC

DATA TYPES
wxWindowDC()

An object reference. The representation is internal and can be changed without notice. It can't be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxWindowDC()

This function is deprecated: not available in wxWidgets-2.9 and later
See external documentation.

ew(Win) -> wxWindowDC()

Types:

Win = wxWindow:wxWindow()

See external documentation.

destroy(This::wxWindowDC()) -> ok

Destroys this object, do not use object again
wxWindowDestroyEvent

Erlang module

See external documentation: `wxWindowDestroyEvent`.

Use `wxEvtHandler:connect/3` with EventType:

    destroy

See also the message variant `#wxWindowDestroy[]` event record type.

This class is derived (and can use functions) from:

    wxCommandEvent
    wxEvent

**DATA TYPES**

`wxWindowDestroyEvent()`

    An object reference. The representation is internal and can be changed without notice. It can't be used for comparsion stored on disc or distributed for use on other nodes.
wxXmlResource

Erlang module

See external documentation: wxXmlResource.

DATA TYPES

wxXmlResource()

An object reference. The representation is internal and can be changed without notice. It can’t be used for comparison stored on disc or distributed for use on other nodes.

Exports

new() -> wxXmlResource()
Equivalent to new([]).

new(Options::[Option]) -> wxXmlResource()
Types:
  Option = {flags, integer()} | {domain, unicode:chardata()}
See external documentation.

new(Filemask, Options::[Option]) -> wxXmlResource()
Types:
  Filemask = unicode:chardata()
  Option = {flags, integer()} | {domain, unicode:chardata()}
See external documentation.

attachUnknownControl(This, Name, Control) -> boolean()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
  Control = wxWindow:wxWindow()
Equivalent to attachUnknownControl(This, Name, Control, []).

attachUnknownControl(This, Name, Control, Options::[Option]) -> boolean()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
  Control = wxWindow:wxWindow()
  Option = {parent, wxWindow:wxWindow()}
See external documentation.
clearHandlers(This) -> ok
Types:
   This = wxXmlResource()
See external documentation.

compareVersion(This, Major, Minor, Release, Revision) -> integer()
Types:
   This = wxXmlResource()
   Major = integer()
   Minor = integer()
   Release = integer()
   Revision = integer()
See external documentation.

get() -> wxXmlResource()
See external documentation.

getFlags(This) -> integer()
Types:
   This = wxXmlResource()
See external documentation.

getVersion(This) -> integer()
Types:
   This = wxXmlResource()
See external documentation.

getXRCID(Str_id) -> integer()
Types:
   Str_id = [unicode:chardata()]
Equivalent to getXRCID(Str_id, []).

getXRCID(Str_id, Options::[Option]) -> integer()
Types:
   Str_id = [unicode:chardata()]
   Option = {value_if_not_found, integer()}
See external documentation.

initAllHandlers(This) -> ok
Types:
   This = wxXmlResource()
See external documentation.
wxXmlResource

load(This, Filemask) -> boolean()
Types:
  This = wxXmlResource()
  Filemask = unicode:chardata()
See external documentation.

loadBitmap(This, Name) -> wxBitmap:wxBitmap()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
See external documentation.

loadDialog(This, Parent, Name) -> wxDialog:wxDialog()
Types:
  This = wxXmlResource()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadDialog(This, Dlg, Parent, Name) -> boolean()
Types:
  This = wxXmlResource()
  Dlg = wxDialog:wxDialog()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadFrame(This, Parent, Name) -> wxFrame:wxFrame()
Types:
  This = wxXmlResource()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadFrame(This, Frame, Parent, Name) -> boolean()
Types:
  This = wxXmlResource()
  Frame = wxFrame:wxFrame()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.
loadIcon(This, Name) -> wxIcon:wxIcon()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
See external documentation.

loadMenu(This, Name) -> wxMenu:wxMenu()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
See external documentation.

loadMenuBar(This, Name) -> wxMenuBar:wxMenuBar()
Types:
  This = wxXmlResource()
  Name = unicode:chardata()
See external documentation.

loadMenuBar(This, Parent, Name) -> wxMenuBar:wxMenuBar()
Types:
  This = wxXmlResource()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadPanel(This, Parent, Name) -> wxPanel:wxPanel()
Types:
  This = wxXmlResource()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadPanel(This, Panel, Parent, Name) -> boolean()
Types:
  This = wxXmlResource()
  Panel = wxPanel:wxPanel()
  Parent = wxWindow:wxWindow()
  Name = unicode:chardata()
See external documentation.

loadToolBar(This, Parent, Name) -> wxToolBar:wxToolBar()
Types:
  This = wxXmlResource()
wxXmlResource

\[ \text{Parent} = \text{wxWindow}:\text{wxWindow}() \]
\[ \text{Name} = \text{unicode}:\text{chardata}() \]

See external documentation.

set(Res) -> wxXmlResource()
Types:
\[ \text{Res} = \text{wxXmlResource}() \]
See external documentation.

setFlags(This, Flags) -> ok
Types:
\[ \text{This} = \text{wxXmlResource}() \]
\[ \text{Flags} = \text{integer}() \]
See external documentation.

unload(This, Filename) -> boolean()
Types:
\[ \text{This} = \text{wxXmlResource}() \]
\[ \text{Filename} = \text{unicode}:\text{chardata}() \]
See external documentation.

xrcctrl(Window, Name, Type) -> wx:wx_object()
Types:
\[ \text{Window} = \text{wxWindow}:\text{wxWindow}() \]
\[ \text{Name} = \text{string}() \]
\[ \text{Type} = \text{atom}() \]
Looks up a control with Name in a window created with XML resources. You can use it to set/get values from controls. The object is type casted to Type. Example:
\[ \text{Xrc} = \text{wxXmlResource}:\text{get}(), \]
\[ \text{Dlg} = \text{wxDialog}:\text{new}(), \]
\[ \text{true} = \text{wxXmlResource}:\text{loadDialog}(<	ext{Xrc}, \text{Dlg}, \text{Frame}, \text{"controls_dialog"}>, \]
\[ \text{LCT} = \text{xrcctrl}(\text{Dlg}, \text{"controls_listctrl"}, \text{wxListCtrl}), \]
\[ \text{wxListCtrl}:\text{insertColumn}(<\text{LCT}, 0, \text{"Name"}, [\{\text{width}, 200\}]), \]

destroy(This::wxXmlResource()) -> ok
Destroys this object, do not use object again
wx_misc

Erlang module

See external documentation: Misc.

Exports

getKeyState(Key) -> boolean()

Types:

\[ \text{Key} = \text{wx:wx_enum()} \]

See external documentation.

getMousePosition() -> {X::integer(), Y::integer()}

See external documentation.

getMouseState() -> wx:wx_wxMouseState()

See external documentation.

setDetectableAutoRepeat(Flag) -> boolean()

Types:

\[ \text{Flag} = \text{boolean()} \]
bell() -> ok
See external documentation.

findMenuItemId(Frame, MenuString, ItemString) -> integer()
Types:
  Frame = wxFrame:wxFrame()
  MenuString = unicode:chardata()
  ItemString = unicode:chardata()
See external documentation.

genericFindWindowAtPoint(Pt) -> wxWindow:wxWindow()
Types:
  Pt = {X::integer(), Y::integer()}
See external documentation.

findWindowAtPoint(Pt) -> wxWindow:wxWindow()
Types:
  Pt = {X::integer(), Y::integer()}
See external documentation.

beginBusyCursor() -> ok
Equivalent to beginBusyCursor([]).

beginBusyCursor(Options::[Option]) -> ok
Types:
  Option = {cursor, wxCursor:wxCursor()}
See external documentation.

disableBusyCursor() -> ok
See external documentation.

isBusy() -> boolean()
See external documentation.

shutdown(WFlags) -> boolean()
Types:
  WFlags = wx:wx_enum()
See external documentation.
WFlags = ?wxSHUTDOWN_POWEROFF | ?wxSHUTDOWN_REBOOT
shell() -> boolean()
Equivalent to \textit{shell([])}.

\textbf{shell(Options::[Option])} -> boolean()

Types:
\[
\text{Option} = \{\text{command}, \text{unicode:chardata()})\}
\]
See \textit{external documentation}.

\textbf{launchDefaultBrowser(Url)} -> boolean()

Types:
\[
\text{Url} = \text{unicode:chardata()}
\]
Equivalent to \textit{launchDefaultBrowser(Url, [])}.

\textbf{launchDefaultBrowser(Url, Options::[Option])} -> boolean()

Types:
\[
\text{Url} = \text{unicode:chardata()}
\quad \text{Option} = \{\text{flags}, \text{integer()})\}
\]
See \textit{external documentation}.

\textbf{getEmailAddress()} -> unicode:charlist()
See \textit{external documentation}.

\textbf{getUserId()} -> unicode:charlist()
See \textit{external documentation}.

\textbf{getHomeDir()} -> unicode:charlist()
See \textit{external documentation}.

\textbf{newId()} -> integer()
See \textit{external documentation}.

\textbf{registerId(Id)} -> ok

Types:
\[
\text{Id} = \text{integer()}
\]
See \textit{external documentation}.

\textbf{getCurrentId()} -> integer()
See \textit{external documentation}.

\textbf{getOsDescription()} -> unicode:charlist()
See \textit{external documentation}. 
isPlatformLittleEndian() -> boolean()
See external documentation.

isPlatform64Bit() -> boolean()
See external documentation.

displaySize() -> {Width::integer(), Height::integer()}
See external documentation.

setCursor(Cursor) -> ok
Types:
    Cursor = wxCursor:wxCursor()
See external documentation.
A part of the standard OpenGL Utility api. See [www.khronos.org](http://www.khronos.org)

Booleans are represented by integers 0 and 1.

**DATA TYPES**

`enum() = non_neg_integer()`

   See wx/include/gl.hrl or glu.hrl

`matrix() = matrix12() | matrix16()`

`matrix12() = {float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}`

`matrix16() = {float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}`

`mem() = binary() | tuple()`

   Memory block

`vertex() = {float(), float(), float()}`

**Exports**

`tesselate(Normal, Vs::[Vs]) -> {Triangles, VertexPos}`

Types:

   Normal = vertex()
   Vs = vertex()
   Triangles = [integer()]
   VertexPos = binary()

General purpose polygon triangulation. The first argument is the normal and the second a list of vertex positions. Returned is a list of indecies of the vertices and a binary (64bit native float) containing an array of vertex positions, it starts with the vertices in Vs and may contain newly created vertices in the end.

`build1DMipmapLevels(Target, InternalFormat, Width, Format, Type, Level, Base, Max, Data) -> integer()`

Types:

   Target = enum()
   InternalFormat = integer()
   Width = integer()
   Format = enum()
   Type = enum()
   Level = integer()
   Base = integer()
Max = integer()
Data = binary()

Builds a subset of one-dimensional mipmap levels

glu:build1DMipmapLevels builds a subset of prefiltered one-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture mapped primitives.

See external documentation.

build1DMipmaps(Target, InternalFormat, Width, Format, Type, Data) -> integer()

Types:
Target = enum()
InternalFormat = integer()
Width = integer()
Format = enum()
Type = enum()
Data = binary()

Builds a one-dimensional mipmap

glu:build1DMipmaps builds a series of prefiltered one-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture mapped primitives.

See external documentation.

build2DMipmapLevels(Target, InternalFormat, Width, Height, Format, Type, Level, Base, Max, Data) -> integer()

Types:
Target = enum()
InternalFormat = integer()
Width = integer()
Height = integer()
Format = enum()
Type = enum()
Level = integer()
Base = integer()
Max = integer()
Data = binary()

Builds a subset of two-dimensional mipmap levels

glu:build2DMipmapLevels builds a subset of prefiltered two-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture mapped primitives.

See external documentation.

build2DMipmaps(Target, InternalFormat, Width, Height, Format, Type, Data) -> integer()

Types:
Target = enum()
Builds a two-dimensional mipmap

`glu:build2DMipmaps` builds a series of prefiltered two-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture-mapped primitives.

See external documentation.

```erlang
build3DMipmapLevels(Target, InternalFormat, Width, Height, Depth, Format, Type, Level, Base, Max, Data) -> integer()
```

Types:

- `Target` = `enum()`
- `InternalFormat` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Depth` = `integer()`
- `Format` = `enum()`
- `Type` = `enum()`
- `Level` = `integer()`
- `Base` = `integer()`
- `Max` = `integer()`
- `Data` = `binary()`

Builds a subset of three-dimensional mipmap levels

`glu:build3DMipmapLevels` builds a subset of prefiltered three-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture mapped primitives.

See external documentation.

```erlang
build3DMipmaps(Target, InternalFormat, Width, Height, Depth, Format, Type, Data) -> integer()
```

Types:

- `Target` = `enum()`
- `InternalFormat` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Depth` = `integer()`
- `Format` = `enum()`
- `Type` = `enum()`
- `Data` = `binary()`

Builds a three-dimensional mipmap
glu:build3DMipmaps builds a series of prefiltered three-dimensional texture maps of decreasing resolutions called a mipmap. This is used for the antialiasing of texture-mapped primitives.

See external documentation.

checkExtension(ExtName, ExtString) -> 0 | 1
Types:
  ExtName = string()
  ExtString = string()
Determines if an extension name is supported

glu:checkExtension returns ?GLU_TRUE if ExtName is supported otherwise ?GLU_FALSE is returned.

See external documentation.

cylinder(Quad, Base, Top, Height, Slices, Stacks) -> ok
Types:
  Quad = integer()
  Base = float()
  Top = float()
  Height = float()
  Slices = integer()
  Stacks = integer()
Draw a cylinder

glu:cylinder draws a cylinder oriented along the $z$ axis. The base of the cylinder is placed at $z=0$ and the top at $z=\text{height}$. Like a sphere, a cylinder is subdivided around the $z$ axis into slices and along the $z$ axis into stacks.

See external documentation.

deleteQuadric(Quad) -> ok
Types:
  Quad = integer()
Destroy a quadrics object

glu:deleteQuadric destroys the quadrics object (created with glu:newQuadric/0) and frees any memory it uses. Once glu:deleteQuadric has been called, Quad cannot be used again.

See external documentation.

disk(Quad, Inner, Outer, Slices, Loops) -> ok
Types:
  Quad = integer()
  Inner = float()
  Outer = float()
  Slices = integer()
  Loops = integer()
Draw a disk
glu:disk renders a disk on the \( z = 0 \) plane. The disk has a radius of \( \text{Outer} \) and contains a concentric circular hole with a radius of \( \text{Inner} \). If \( \text{Inner} \) is 0, then no hole is generated. The disk is subdivided around the \( z \) axis into slices (like pizza slices) and also about the \( z \) axis into rings (as specified by \( \text{Slices} \) and \( \text{Loops} \), respectively).

See external documentation.

errrorString(Error) -> string()
Types:
   Error = enum()
Produce an error string from a GL or GLU error code

glu:errorMessage produces an error string from a GL or GLU error code. The string is in ISO Latin 1 format. For example, glu:errorMessage(?GLU_OUT_OF_MEMORY) returns the string out of memory.

See external documentation.

getString(Name) -> string()
Types:
   Name = enum()
Return a string describing the GLU version or GLU extensions

glu:getString returns a pointer to a static string describing the GLU version or the GLU extensions that are supported.

See external documentation.

lookAt(EyeX, EyeY, EyeZ, CenterX, CenterY, CenterZ, UpX, UpY, UpZ) -> ok
Types:
   EyeX = float()
   EyeY = float()
   EyeZ = float()
   CenterX = float()
   CenterY = float()
   CenterZ = float()
   UpX = float()
   UpY = float()
   UpZ = float()
Define a viewing transformation

glu:lookAt creates a viewing matrix derived from an eye point, a reference point indicating the center of the scene, and an \( \text{UP} \) vector.

See external documentation.

newQuadric() -> integer()
Create a quadrics object

glu:newQuadric creates and returns a pointer to a new quadrics object. This object must be referred to when calling quadrics rendering and control functions. A return value of 0 means that there is not enough memory to allocate the object.

See external documentation.
ortho2D(Left, Right, Bottom, Top) -> ok
Types:
    Left = float()
    Right = float()
    Bottom = float()
    Top = float()
Define a 2D orthographic projection matrix
glu:ortho2D sets up a two-dimensional orthographic viewing region. This is equivalent to calling gl:ortho/6 with near=-1 and far=1.
See external documentation.

partialDisk(Quad, Inner, Outer, Slices, Loops, Start, Sweep) -> ok
Types:
    Quad = integer()
    Inner = float()
    Outer = float()
    Slices = integer()
    Loops = integer()
    Start = float()
    Sweep = float()
Draw an arc of a disk
glu:partialDisk renders a partial disk on the z=0 plane. A partial disk is similar to a full disk, except that only the subset of the disk from \( \text{Start} \) through \( \text{Start} + \text{Sweep} \) is included (where 0 degrees is along the +f\( \text{y} \) axis, 90 degrees along the +\( x \) axis, 180 degrees along the -\( y \) axis, and 270 degrees along the -\( x \) axis).
See external documentation.

perspective(Fovy, Aspect, ZNear, ZFar) -> ok
Types:
    Fovy = float()
    Aspect = float()
    ZNear = float()
    ZFar = float()
Set up a perspective projection matrix
glu:perspective specifies a viewing frustum into the world coordinate system. In general, the aspect ratio in glu:perspective should match the aspect ratio of the associated viewport. For example, aspect=2.0 means the viewer's angle of view is twice as wide in \( x \) as it is in \( y \). If the viewport is twice as wide as it is tall, it displays the image without distortion.
See external documentation.

pickMatrix(X, Y, DelX, DelY, Viewport) -> ok
Types:
    X = float()
    Y = float()
DelX = float()
DelY = float()
Viewport = {integer(), integer(), integer(), integer()}

Define a picking region

glu:pickMatrix creates a projection matrix that can be used to restrict drawing to a small region of the viewport. This is typically useful to determine what objects are being drawn near the cursor. Use glu:pickMatrix to restrict drawing to a small region around the cursor. Then, enter selection mode (with gl:renderMode/1) and rerender the scene. All primitives that would have been drawn near the cursor are identified and stored in the selection buffer.

See external documentation.

project(ObjX, ObjY, ObjZ, Model, Proj, View) -> {integer(), WinX::float(), WinY::float(), WinZ::float()}

Types:
ObjX = float()
ObjY = float()
ObjZ = float()
Model = matrix()
Proj = matrix()
View = {integer(), integer(), integer(), integer()}

Map object coordinates to window coordinates

glu:project transforms the specified object coordinates into window coordinates using Model, Proj, and View. The result is stored in WinX, WinY, and WinZ. A return value of ?GLU_TRUE indicates success, a return value of ?GLU_FALSE indicates failure.

See external documentation.

quadricDrawStyle(Quad, Draw) -> ok

Types:
Quad = integer()
Draw = enum()

Specify the draw style desired for quadrics

glu:quadricDrawStyle specifies the draw style for quadrics rendered with Quad. The legal values are as follows:

See external documentation.

quadricNormals(Quad, Normal) -> ok

Types:
Quad = integer()
Normal = enum()

Specify what kind of normals are desired for quadrics

glu:quadricNormals specifies what kind of normals are desired for quadrics rendered with Quad. The legal values are as follows:

See external documentation.
quadricOrientation(Quad, Orientation) -> ok
Types:
  Quad = integer()
  Orientation = enum()
Specify inside/outside orientation for quadrics

quadricTexture(Quad, Texture) -> ok
Types:
  Quad = integer()
  Texture = 0 | 1
Specify if texturing is desired for quadrics

glu:quadricOrientation specifies what kind of orientation is desired for quadrics rendered with Quad. The Orientation values are as follows:
See external documentation.

glu:quadricTexture specifies if texture coordinates should be generated for quadrics rendered with Quad. If the value of Texture is ?GLU_TRUE, then texture coordinates are generated, and if Texture is ?GLU_FALSE, they are not. The initial value is ?GLU_FALSE.
See external documentation.

scaleImage(Format, WIn, HIn, TypeIn, DataIn, WOut, HOut, TypeOut, DataOut) -> integer()
Types:
  Format = enum()
  WIn = integer()
  HIn = integer()
  TypeIn = enum()
  DataIn = binary()
  WOut = integer()
  HOut = integer()
  TypeOut = enum()
  DataOut = mem()
Scale an image to an arbitrary size

 glu:scaleImage scales a pixel image using the appropriate pixel store modes to unpack data from the source image and pack data into the destination image.
See external documentation.

draw a sphere

sphere(Quad, Radius, Slices, Stacks) -> ok
Types:
  Quad = integer()
  Radius = float()
  Slices = integer()
  Stacks = integer()
Draw a sphere
**glu:sphere** draws a sphere of the given radius centered around the origin. The sphere is subdivided around the \( z \) axis into slices and along the \( z \) axis into stacks (similar to lines of longitude and latitude).

See **external** documentation.

```erlang
unProject(WinX, WinY, WinZ, Model, Proj, View) -> {integer(), ObjX::float(), ObjY::float(), ObjZ::float()}
```

Types:

- \( \text{WinX} = \text{float()} \)
- \( \text{WinY} = \text{float()} \)
- \( \text{WinZ} = \text{float()} \)
- \( \text{Model} = \text{matrix()} \)
- \( \text{Proj} = \text{matrix()} \)
- \( \text{View} = \{\text{integer()}, \text{integer()}, \text{integer()}, \text{integer()}\} \)

Map window coordinates to object coordinates

**glu:unProject** maps the specified window coordinates into object coordinates using \( \text{Model} \), \( \text{Proj} \), and \( \text{View} \). The result is stored in \( \text{ObjX} \), \( \text{ObjY} \), and \( \text{ObjZ} \). A return value of \( \text{?GLU_TRUE} \) indicates success; a return value of \( \text{?GLU_FALSE} \) indicates failure.

See **external** documentation.

```erlang
unProject4(WinX, WinY, WinZ, ClipW, Model, Proj, View, NearVal, FarVal) ->
\{integer(), ObjX::float(), ObjY::float(), ObjZ::float(), ObjW::float()\}
```

Types:

- \( \text{WinX} = \text{float()} \)
- \( \text{WinY} = \text{float()} \)
- \( \text{WinZ} = \text{float()} \)
- \( \text{ClipW} = \text{float()} \)
- \( \text{Model} = \text{matrix()} \)
- \( \text{Proj} = \text{matrix()} \)
- \( \text{View} = \{\text{integer()}, \text{integer()}, \text{integer()}, \text{integer()}\} \)
- \( \text{NearVal} = \text{float()} \)
- \( \text{FarVal} = \text{float()} \)

See **unProject/6**
Erlang module

Standard OpenGL api. See www.khronos.org
Booleans are represented by integers 0 and 1.

DATA TYPES

clamp() = float()
  0.0..1.0
enum() = non_neg_integer()

See wx/include/gl.hrl

matrix() = matrix12() | matrix16()

matrix12() = {float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

matrix16() = {float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

mem() = binary() | tuple()

  Memory block

offset() = non_neg_integer()

  Offset in memory block

Exports

clearIndex(C) -> ok
Types:
  C = float()
Specify the clear value for the color index buffers

gl:clearIndex specifies the index used by gl:clear/1 to clear the color index buffers. C is not clamped. Rather, C is converted to a fixed-point value with unspecified precision to the right of the binary point. The integer part of this value is then masked with 2^m-1, where m is the number of bits in a color index stored in the frame buffer.

See external documentation.

clearColor(Red, Green, Blue, Alpha) -> ok
Types:
  Red = clamp()
  Green = clamp()
  Blue = clamp()
  Alpha = clamp()

Specify clear values for the color buffers
gl:clearColor specifies the red, green, blue, and alpha values used by \texttt{gl:clear/1} to clear the color buffers. Values specified by \texttt{gl:clearColor} are clamped to the range \([0, 1]\).

See external documentation.

\textbf{clear}(\texttt{Mask}) -> \texttt{ok}
Types:
\begin{verbatim}
    Mask = integer()
\end{verbatim}
Clear buffers to preset values
\texttt{gl:clear} sets the bitplane area of the window to values previously selected by \texttt{gl:clearColor}, \texttt{gl:clearDepth}, and \texttt{gl:clearStencil}. Multiple color buffers can be cleared simultaneously by selecting more than one buffer at a time using \texttt{gl:drawBuffer/1}.

See external documentation.

\textbf{indexMask}(\texttt{Mask}) -> \texttt{ok}
Types:
\begin{verbatim}
    Mask = integer()
\end{verbatim}
Control the writing of individual bits in the color index buffers
\texttt{gl:indexMask} controls the writing of individual bits in the color index buffers. The least significant \(n\) bits of \texttt{Mask}, where \(n\) is the number of bits in a color index buffer, specify a mask. Where a 1 (one) appears in the mask, it's possible to write to the corresponding bit in the color index buffer (or buffers). Where a 0 (zero) appears, the corresponding bit is write-protected.

See external documentation.

\textbf{colorMask}(\texttt{Red}, \texttt{Green}, \texttt{Blue}, \texttt{Alpha}) -> \texttt{ok}
Types:
\begin{verbatim}
    Red = 0 | 1
    Green = 0 | 1
    Blue = 0 | 1
    Alpha = 0 | 1
\end{verbatim}
Enable and disable writing of frame buffer color components
\texttt{gl:colorMask} and \texttt{gl:colorMaski} specify whether the individual color components in the frame buffer can or cannot be written. \texttt{gl:colorMaski} sets the mask for a specific draw buffer, whereas \texttt{gl:colorMask} sets the mask for all draw buffers. If \texttt{Red} is \texttt{?GL_FALSE}, for example, no change is made to the red component of any pixel in any of the color buffers, regardless of the drawing operation attempted.

See external documentation.

\textbf{alphaFunc}(\texttt{Func}, \texttt{Ref}) -> \texttt{ok}
Types:
\begin{verbatim}
    Func = enum()
    Ref = clamp()
\end{verbatim}
Specify the alpha test function
The alpha test discards fragments depending on the outcome of a comparison between an incoming fragment’s alpha value and a constant reference value. \texttt{gl:alphaFunc} specifies the reference value and the comparison function. The
comparison is performed only if alpha testing is enabled. By default, it is not enabled. (See `gl:enable/1` and `gl:enable/1` of `?GL_ALPHA_TEST`.)

See external documentation.

`blendFunc(Sfactor, Dfactor) -> ok`

Types:

- `Sfactor = enum()`
- `Dfactor = enum()`

Specify pixel arithmetic

Pixels can be drawn using a function that blends the incoming (source) RGBA values with the RGBA values that are already in the frame buffer (the destination values). Blending is initially disabled. Use `gl:enable/1` and `gl:enable/1` with argument `?GL_BLEND` to enable and disable blending.

See external documentation.

`logicOp(Opcode) -> ok`

Types:

- `Opcode = enum()`

Specify a logical pixel operation for rendering

`gl:logicOp` specifies a logical operation that, when enabled, is applied between the incoming RGBA color and the RGBA color at the corresponding location in the frame buffer. To enable or disable the logical operation, call `gl:enable/1` and `gl:enable/1` using the symbolic constant `?GL_COLOR_LOGIC_OP`. The initial value is disabled.

See external documentation.

`cullFace(Mode) -> ok`

Types:

- `Mode = enum()`

Specify whether front- or back-facing facets can be culled

`gl:cullFace` specifies whether front- or back-facing facets are culled (as specified by `mode`) when facet culling is enabled. Facet culling is initially disabled. To enable and disable facet culling, call the `gl:enable/1` and `gl:enable/1` commands with the argument `?GL_CULL_FACE`. Facets include triangles, quadrilaterals, polygons, and rectangles.

See external documentation.

`frontFace(Mode) -> ok`

Types:

- `Mode = enum()`

Define front- and back-facing polygons

In a scene composed entirely of opaque closed surfaces, back-facing polygons are never visible. Eliminating these invisible polygons has the obvious benefit of speeding up the rendering of the image. To enable and disable elimination of back-facing polygons, call `gl:enable/1` and `gl:enable/1` with argument `?GL_CULL_FACE`.

See external documentation.

`pointSize(Size) -> ok`

Types:
Size = float()
Specify the diameter of rasterized points

\texttt{gl:pointSize} specifies the rasterized diameter of points. If point size mode is disabled (see \texttt{gl:enable/1} with parameter \texttt{?GL_PROGRAM_POINT_SIZE}), this value will be used to rasterize points. Otherwise, the value written to the shading language built-in variable \texttt{gl_PointSize} will be used.

See external documentation.

\texttt{lineWidth(Width) -> ok}
Types:
\begin{verbatim}
     Width = float()
\end{verbatim}

Specify the width of rasterized lines

\texttt{gl:lineWidth} specifies the rasterized width of both aliased and antialiased lines. Using a line width other than 1 has different effects, depending on whether line antialiasing is enabled. To enable and disable line antialiasing, call \texttt{gl:enable/1} and \texttt{gl:enable/1} with argument \texttt{?GL_LINE_SMOOTH}. Line antialiasing is initially disabled.

See external documentation.

\texttt{lineStipple(Factor, Pattern) -> ok}
Types:
\begin{verbatim}
     Factor = integer()
     Pattern = integer()
\end{verbatim}

Specify the line stipple pattern

Line stippling masks out certain fragments produced by rasterization; those fragments will not be drawn. The masking is achieved by using three parameters: the 16-bit line stipple pattern \texttt{Pattern}, the repeat count \texttt{Factor}, and an integer stipple counter \texttt{s}.

See external documentation.

\texttt{polygonMode(Face, Mode) -> ok}
Types:
\begin{verbatim}
     Face = enum()
     Mode = enum()
\end{verbatim}

Select a polygon rasterization mode

\texttt{gl:polygonMode} controls the interpretation of polygons for rasterization. \texttt{Face} describes which polygons \texttt{Mode} applies to: both front and back-facing polygons (?)GL\_FRONT\_AND\_BACK\ ). The polygon mode affects only the final rasterization of polygons. In particular, a polygon's vertices are lit and the polygon is clipped and possibly culled before these modes are applied.

See external documentation.

\texttt{polygonOffset(Factor, Units) -> ok}
Types:
\begin{verbatim}
     Factor = float()
     Units = float()
\end{verbatim}

Set the scale and units used to calculate depth values
When `GL_POLYGON_OFFSET_FILL`, `GL_POLYGON_OFFSET_LINE`, or `GL_POLYGON_OFFSET_POINT` is enabled, each fragment's depth value will be offset after it is interpolated from the depth values of the appropriate vertices. The value of the offset is \( \text{factor} \times DZ + r \times \text{units} \), where \( DZ \) is a measurement of the change in depth relative to the screen area of the polygon, and \( r \) is the smallest value that is guaranteed to produce a resolvable offset for a given implementation. The offset is added before the depth test is performed and before the value is written into the depth buffer.

See external documentation.

**polygonStipple(Mask) -> ok**

Types:

\[
\text{Mask} = \text{binary()}
\]

Set the polygon stippling pattern

Polygon stippling, like line stippling (see `gl:lineStipple/2`), masks out certain fragments produced by rasterization, creating a pattern. Stippling is independent of polygon antialiasing.

See external documentation.

**getPolygonStipple() -> binary()**

Return the polygon stipple pattern

`gl:getPolygonStipple` returns to `Pattern` a 32×32 polygon stipple pattern. The pattern is packed into memory as if `gl:readPixels/7` with both height and width of 32, type of `GL_BITMAP`, and format of `GL_COLOR_INDEX` were called, and the stipple pattern were stored in an internal 32×32 color index buffer. Unlike `gl:readPixels/7`, however, pixel transfer operations (shift, offset, pixel map) are not applied to the returned stipple image.

See external documentation.

**edgeFlag(Flag) -> ok**

Types:

\[
\text{Flag} = 0 | 1
\]

Flag edges as either boundary or nonboundary

Each vertex of a polygon, separate triangle, or separate quadrilateral specified between a `gl:'begin'/1` pair is marked as the start of either a boundary or nonboundary edge. If the current edge flag is true when the vertex is specified, the vertex is marked as the start of a boundary edge. Otherwise, the vertex is marked as the start of a nonboundary edge. `gl:edgeFlag` sets the edge flag bit to `GL_TRUE` if `Flag` is `GL_TRUE` and to `GL_FALSE` otherwise.

See external documentation.

**edgeFlagv(Flag) -> ok**

Types:

\[
\text{Flag} = \{\text{Flag}::0 | 1\}
\]

Equivalent to `edgeFlag(Flag)`.

**scissor(X, Y, Width, Height) -> ok**

Types:

\[
X = \text{integer()}
\]
Y = integer()
Width = integer()
Height = integer()

Define the scissor box

`gl:scissor` defines a rectangle, called the scissor box, in window coordinates. The first two arguments, X and Y, specify the lower left corner of the box. Width and Height specify the width and height of the box.

See external documentation.

`clipPlane(Plane, Equation) -> ok`

Types:
- `Plane = enum()`
- `Equation = {float(), float(), float(), float()}`

Specify a plane against which all geometry is clipped

Geometry is always clipped against the boundaries of a six-plane frustum in x, y, and z. `gl:clipPlane` allows the specification of additional planes, not necessarily perpendicular to the x, y, or z axis, against which all geometry is clipped. To determine the maximum number of additional clipping planes, call `gl:getbooleanv/1` with argument `?GL_MAX_CLIP_PLANES`. All implementations support at least six such clipping planes. Because the resulting clipping region is the intersection of the defined half-spaces, it is always convex.

See external documentation.

`getClipPlane(Plane) -> {float(), float(), float(), float()}`

Types:
- `Plane = enum()`

Return the coefficients of the specified clipping plane

`gl:getClipPlane` returns in `Equation` the four coefficients of the plane equation for `Plane`.

See external documentation.

`drawBuffer(Mode) -> ok`

Types:
- `Mode = enum()`

Specify which color buffers are to be drawn into

When colors are written to the frame buffer, they are written into the color buffers specified by `gl:drawBuffer`. The specifications are as follows:

See external documentation.

`readBuffer(Mode) -> ok`

Types:
- `Mode = enum()`

Select a color buffer source for pixels

`gl:readBuffer` specifies a color buffer as the source for subsequent `gl:readPixels/7`, `gl:copyTexImage1D/7`, `gl:copyTexImage2D/8`, `gl:copyTexSubImage1D/6`, `gl:copyTexSubImage2D/8`, and `gl:copyTexSubImage3D/9` commands. Mode accepts one of twelve or more predefined values. In a fully configured system, `?GL_FRONT`, `?GL_LEFT`, and `?GL_FRONT_LEFT` all name the front left buffer, `?GL_FRONT_RIGHT` and `?GL_RIGHT` name the
front right buffer, and \texttt{GL_BACK_LEFT} and \texttt{GL_BACK} name the back left buffer. Further more, the constants \texttt{GL_COLOR_ATTACHMENT}\texttt{i} may be used to indicate the \texttt{i}th color attachment where \texttt{i} ranges from zero to the value of \texttt{GL_MAX_COLOR_ATTACHMENTS} minus one.

See external documentation.

\texttt{enable(Cap) \rightarrow ok}

Types:
\begin{verbatim}
    Cap = enum()
\end{verbatim}

Enable or disable server-side GL capabilities

\texttt{gl:enable} and \texttt{gl:enable/1} enable and disable various capabilities. Use \texttt{gl:isEnabled/1} or \texttt{gl:getBooleanv/1} to determine the current setting of any capability. The initial value for each capability with the exception of \texttt{GL_DITHER} and \texttt{GL_MULTISAMPLE} is \texttt{GL_FALSE}. The initial value for \texttt{GL_DITHER} and \texttt{GL_MULTISAMPLE} is \texttt{GL_TRUE}.

See external documentation.

\texttt{disable(Cap) \rightarrow ok}

Types:
\begin{verbatim}
    Cap = enum()
\end{verbatim}

See \texttt{enable/1}

\texttt{isEnabled(Cap) \rightarrow 0 \mid 1}

Types:
\begin{verbatim}
    Cap = enum()
\end{verbatim}

Test whether a capability is enabled

\texttt{gl:isEnabled} returns \texttt{GL_TRUE} if \texttt{Cap} is an enabled capability and returns \texttt{GL_FALSE} otherwise. Boolean states that are indexed may be tested with \texttt{gl:isEnabled/1}. For \texttt{gl:isEnabled/1}, \texttt{Index} specifies the index of the capability to test. \texttt{Index} must be between zero and the count of indexed capabilities for \texttt{Cap}. Initially all capabilities except \texttt{GL_DITHER} are disabled; \texttt{GL_DITHER} is initially enabled.

See external documentation.

\texttt{enableClientState(Cap) \rightarrow ok}

Types:
\begin{verbatim}
    Cap = enum()
\end{verbatim}

Enable or disable client-side capability

\texttt{gl:enableClientState} and \texttt{gl:enableClientState/1} enable or disable individual client-side capabilities. By default, all client-side capabilities are disabled. Both \texttt{gl:enableClientState} and \texttt{gl:enableClientState/1} take a single argument, \texttt{Cap}, which can assume one of the following values:

See external documentation.

\texttt{disableClientState(Cap) \rightarrow ok}

Types:
\begin{verbatim}
    Cap = enum()
\end{verbatim}

See \texttt{enableClientState/1}
getBooleanv(Pname) -> [0 | 1]
Types:
   Pname = enum()
Return the value or values of a selected parameter
These four commands return values for simple state variables in GL. Pname is a symbolic constant indicating the state variable to be returned, and Params is a pointer to an array of the indicated type in which to place the returned data. See external documentation.

getDoublev(Pname) -> [float()]
Types:
   Pname = enum()
See getBooleanv/1

getFloatv(Pname) -> [float()]
Types:
   Pname = enum()
See getBooleanv/1

getIntegerv(Pname) -> [integer()]
Types:
   Pname = enum()
See getBooleanv/1

pushAttrib(Mask) -> ok
Types:
   Mask = integer()
Push and pop the server attribute stack
.gl:pushAttrib takes one argument, a mask that indicates which groups of state variables to save on the attribute stack. Symbolic constants are used to set bits in the mask. Mask is typically constructed by specifying the bitwise-or of several of these constants together. The special mask ?GL_ALL_ATTRIB_BITS can be used to save all stackable states.
See external documentation.

popAttrib() -> ok
See pushAttrib/1

pushClientAttrib(Mask) -> ok
Types:
   Mask = integer()
Push and pop the client attribute stack
.gl:pushClientAttrib takes one argument, a mask that indicates which groups of client-state variables to save on the client attribute stack. Symbolic constants are used to set bits in the mask. Mask is
typically constructed by specifying the bitwise-or of several of these constants together. The special mask `GL_CLIENT_ALL_ATTRIB_BITS` can be used to save all stackable client state.

See external documentation.

```
popClientAttrib() -> ok
See pushClientAttrib/1
```

```
renderMode(Mode) -> integer()
Types:
   Mode = enum()
Set rasterization mode
gl:renderMode sets the rasterization mode. It takes one argument, Mode, which can assume one of three predefined values:
See external documentation.

```
getError() -> enum()
Return error information
gl:getError returns the value of the error flag. Each detectable error is assigned a numeric code and symbolic name. When an error occurs, the error flag is set to the appropriate error code value. No other errors are recorded until gl:getError is called, the error code is returned, and the flag is reset to `?GL_NO_ERROR`. If a call to gl:getError returns `?GL_NO_ERROR`, there has been no detectable error since the last call to gl:getError, or since the GL was initialized.
See external documentation.

```
getString(Name) -> string()
Types:
   Name = enum()
Return a string describing the current GL connection
gl:getString returns a pointer to a static string describing some aspect of the current GL connection. Name can be one of the following:
See external documentation.

```
finish() -> ok
Block until all GL execution is complete
gl:finish does not return until the effects of all previously called GL commands are complete. Such effects include all changes to GL state, all changes to connection state, and all changes to the frame buffer contents.
See external documentation.

```
flush() -> ok
Force execution of GL commands in finite time
Different GL implementations buffer commands in several different locations, including network buffers and the graphics accelerator itself. gl:flush empties all of these buffers, causing all issued commands to be executed as
quickly as they are accepted by the actual rendering engine. Though this execution may not be completed in any particular time period, it does complete in finite time.

See external documentation.

```
hint(Target, Mode) -> ok
Types:
    Target = enum()
    Mode = enum()
```

Specify implementation-specific hints

Certain aspects of GL behavior, when there is room for interpretation, can be controlled with hints. A hint is specified with two arguments. Target is a symbolic constant indicating the behavior to be controlled, and Mode is another symbolic constant indicating the desired behavior. The initial value for each Target is ?GL_DONT_CARE. Mode can be one of the following:

See external documentation.

```
clearDepth(Depth) -> ok
Types:
    Depth = clamp()
```

Specify the clear value for the depth buffer

`gl:clearDepth` specifies the depth value used by `gl:clear/1` to clear the depth buffer. Values specified by `gl:clearDepth` are clamped to the range [0, 1].

See external documentation.

```
depthFunc(Func) -> ok
Types:
    Func = enum()
```

Specify the value used for depth buffer comparisons

`gl:depthFunc` specifies the function used to compare each incoming pixel depth value with the depth value present in the depth buffer. The comparison is performed only if depth testing is enabled. (See `gl:enable/1` and `gl:enable/1` of ?GL_DEPTH_TEST.)

See external documentation.

```
depthMask(Flag) -> ok
Types:
    Flag = 0 | 1
```

Enable or disable writing into the depth buffer

`gl:depthMask` specifies whether the depth buffer is enabled for writing. If Flag is ?GL_FALSE, depth buffer writing is disabled. Otherwise, it is enabled. Initially, depth buffer writing is enabled.

See external documentation.

```
depthRange(Near_val, Far_val) -> ok
Types:
    Near_val = clamp()
```

Ericsson AB. All Rights Reserved.: wxErlang | 755
Far_val = clamp()

Specify mapping of depth values from normalized device coordinates to window coordinates

After clipping and division by w, depth coordinates range from -1 to 1, corresponding to the near and far clipping planes. **gl:depthRange** specifies a linear mapping of the normalized depth coordinates in this range to window depth coordinates. Regardless of the actual depth buffer implementation, window coordinate depth values are treated as though they range from 0 through 1 (like color components). Thus, the values accepted by **gl:depthRange** are both clamped to this range before they are accepted.

See **external documentation**.

clearAccum(Red, Green, Blue, Alpha) -> ok

Types:
- Red = float()
- Green = float()
- Blue = float()
- Alpha = float()

Specify clear values for the accumulation buffer

**gl:clearAccum** specifies the red, green, blue, and alpha values used by **gl:clear/1** to clear the accumulation buffer.

See **external documentation**.

accum(Op, Value) -> ok

Types:
- Op = enum()
- Value = float()

Operate on the accumulation buffer

The accumulation buffer is an extended-range color buffer. Images are not rendered into it. Rather, images rendered into one of the color buffers are added to the contents of the accumulation buffer after rendering. Effects such as antialiasing (of points, lines, and polygons), motion blur, and depth of field can be created by accumulating images generated with different transformation matrices.

See **external documentation**.

matrixMode(Mode) -> ok

Types:
- Mode = enum()

Specify which matrix is the current matrix

**gl:matrixMode** sets the current matrix mode. **Mode** can assume one of four values:

See **external documentation**.

ortho(Left, Right, Bottom, Top, Near_val, Far_val) -> ok

Types:
- Left = float()
- Right = float()
- Bottom = float()
- Top = float()
Near_val = float()

Far_val = float()

Multiply the current matrix with an orthographic matrix

gl:ortho describes a transformation that produces a parallel projection. The current matrix (see gl:matrixMode/1) is multiplied by this matrix and the result replaces the current matrix, as if gl:mulMatrixd/1 were called with the following matrix as its argument:

See external documentation.

frustum(Left, Right, Bottom, Top, Near_val, Far_val) -> ok

Types:

- Left = float()
- Right = float()
- Bottom = float()
- Top = float()
- Near_val = float()
- Far_val = float()

Multiply the current matrix by a perspective matrix

gl:frustum describes a perspective matrix that produces a perspective projection. The current matrix (see gl:matrixMode/1) is multiplied by this matrix and the result replaces the current matrix, as if gl:mulMatrixd/1 were called with the following matrix as its argument:

See external documentation.

viewport(X, Y, Width, Height) -> ok

Types:

- X = integer()
- Y = integer()
- Width = integer()
- Height = integer()

Set the viewport

gl:viewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (x nd y nd) be normalized device coordinates. Then the window coordinates (x w y w) are computed as follows:

See external documentation.

pushMatrix() -> ok

Push and pop the current matrix stack

There is a stack of matrices for each of the matrix modes. In ?GL_MODELVIEW mode, the stack depth is at least 32. In the other modes, ?GL_COLOR, ?GL_PROJECTION, and ?GL_TEXTURE, the depth is at least 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

See external documentation.

popMatrix() -> ok

See pushMatrix/0
loadIdentity() -> ok
Replace the current matrix with the identity matrix

`gl:loadIdentity` replaces the current matrix with the identity matrix. It is semantically equivalent to calling `gl:loadMatrixd/1` with the identity matrix.

See external documentation.

loadMatrixd(M) -> ok
Types:
  M = matrix()
Replace the current matrix with the specified matrix

`gl:loadMatrix` replaces the current matrix with the one whose elements are specified by M. The current matrix is the projection matrix, modelview matrix, or texture matrix, depending on the current matrix mode (see `gl:matrixMode/1`).

See external documentation.

loadMatrixf(M) -> ok
Types:
  M = matrix()
See loadMatrixd/1

multMatrixd(M) -> ok
Types:
  M = matrix()
Multiply the current matrix with the specified matrix

`gl:multMatrix` multiplies the current matrix with the one specified using M, and replaces the current matrix with the product.

See external documentation.

multMatrixf(M) -> ok
Types:
  M = matrix()
See multMatrixd/1

rotated(Angle, X, Y, Z) -> ok
Types:
  Angle = float()
  X = float()
  Y = float()
  Z = float()
Multiply the current matrix by a rotation matrix

`gl:rotate` produces a rotation of Angle degrees around the vector (x y z). The current matrix (see `gl:matrixMode/1`) is multiplied by a rotation matrix with the product replacing the current matrix, as if `gl:multMatrixd/1` were called with the following matrix as its argument:
See external documentation.

rotatef(Angle, X, Y, Z) -> ok
Types:
   Angle = float()
   X = float()
   Y = float()
   Z = float()
See rotated/4

scaled(X, Y, Z) -> ok
Types:
   X = float()
   Y = float()
   Z = float()
Multiply the current matrix by a general scaling matrix

scalef(X, Y, Z) -> ok
Types:
   X = float()
   Y = float()
   Z = float()
See scaled/3

translated(X, Y, Z) -> ok
Types:
   X = float()
   Y = float()
   Z = float()
Multiply the current matrix by a translation matrix

translatef(X, Y, Z) -> ok
Types:
   X = float()
   Y = float()
   Z = float()
See `translated/3`

`isList(List) -> 0 | 1`

Types:
   
   List = integer()

Determine if a name corresponds to a display list

`gl:isList` returns ?GL_TRUE if `List` is the name of a display list and returns ?GL_FALSE if it is not, or if an error occurs.

See external documentation.

`deleteLists(List, Range) -> ok`

Types:
   
   List = integer()
   Range = integer()

Delete a contiguous group of display lists

`gl:deleteLists` causes a contiguous group of display lists to be deleted. `List` is the name of the first display list to be deleted, and `Range` is the number of display lists to delete. All display lists `d` with `list<= d<= list+range-1` are deleted.

See external documentation.

`genLists(Range) -> integer()`

Types:
   
   Range = integer()

Generate a contiguous set of empty display lists

`gl:genLists` has one argument, `Range`. It returns an integer `n` such that `Range` contiguous empty display lists, named `n`, `n+1`, ..., `n+range-1`, are created. If `Range` is 0, if there is no group of `Range` contiguous names available, or if any error is generated, no display lists are generated, and 0 is returned.

See external documentation.

`newList(List, Mode) -> ok`

Types:
   
   List = integer()
   Mode = enum()

Create or replace a display list

Display lists are groups of GL commands that have been stored for subsequent execution. Display lists are created with `gl:newList`. All subsequent commands are placed in the display list, in the order issued, until `gl:endList/0` is called.

See external documentation.

`endList() -> ok`

`gl:BeginList`

See external documentation.
callList(List) -> ok
Types:
   List = integer()
Execute a display list

`gl:callList` causes the named display list to be executed. The commands saved in the display list are executed in order, just as if they were called without using a display list. If `List` has not been defined as a display list, `gl:callList` is ignored.

See external documentation.

callLists(Lists) -> ok
Types:
   Lists = [integer()]
Execute a list of display lists

`gl:callLists` causes each display list in the list of names passed as `Lists` to be executed. As a result, the commands saved in each display list are executed in order, just as if they were called without using a display list. Names of display lists that have not been defined are ignored.

See external documentation.

listBase(Base) -> ok
Types:
   Base = integer()
set the display-list base for

`gl:callLists/1`

`gl:callLists/1` specifies an array of offsets. Display-list names are generated by adding `Base` to each offset. Names that reference valid display lists are executed; the others are ignored.

See external documentation.

begin(Mode) -> ok
Types:
   Mode = enum()
Delimit the vertices of a primitive or a group of like primitives

`gl:'begin` and `gl:'begin/1` delimit the vertices that define a primitive or a group of like primitives. `gl:'begin` accepts a single argument that specifies in which of ten ways the vertices are interpreted. Taking `n` as an integer count starting at one, and `N` as the total number of vertices specified, the interpretations are as follows:

See external documentation.

close() -> ok
See 'begin/1'

vertex2d(X, Y) -> ok
Types:
   X = float()
   Y = float()
Specify a vertex

`gl:vertex` commands are used within `gl:'begin'/1 / gl:'begin'/1 pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when `gl:vertex` is called.

See external documentation.

`vertex2f(X, Y) -> ok`

Types:

- `X = float()`
- `Y = float()`

See `vertex2d/2`

`vertex2i(X, Y) -> ok`

Types:

- `X = integer()`
- `Y = integer()`

See `vertex2d/2`

`vertex2s(X, Y) -> ok`

Types:

- `X = integer()`
- `Y = integer()`

See `vertex2d/2`

`vertex3d(X, Y, Z) -> ok`

Types:

- `X = float()`
- `Y = float()`
- `Z = float()`

See `vertex2d/2`

`vertex3f(X, Y, Z) -> ok`

Types:

- `X = float()`
- `Y = float()`
- `Z = float()`

See `vertex2d/2`

`vertex3i(X, Y, Z) -> ok`

Types:

- `X = integer()`
- `Y = integer()`
- `Z = integer()`
See \texttt{vertex2d/2}

\textbf{vertex3s(X, Y, Z) -&gt; ok} \\
Types:
\begin{itemize}
  \item \texttt{X = integer()}
  \item \texttt{Y = integer()}
  \item \texttt{Z = integer()}
\end{itemize}
\textbf{See \texttt{vertex2d/2}}

\textbf{vertex4d(X, Y, Z, W) -&gt; ok} \\
Types:
\begin{itemize}
  \item \texttt{X = float()}
  \item \texttt{Y = float()}
  \item \texttt{Z = float()}
  \item \texttt{W = float()}
\end{itemize}
\textbf{See \texttt{vertex2d/2}}

\textbf{vertex4f(X, Y, Z, W) -&gt; ok} \\
Types:
\begin{itemize}
  \item \texttt{X = float()}
  \item \texttt{Y = float()}
  \item \texttt{Z = float()}
  \item \texttt{W = float()}
\end{itemize}
\textbf{See \texttt{vertex2d/2}}

\textbf{vertex4i(X, Y, Z, W) -&gt; ok} \\
Types:
\begin{itemize}
  \item \texttt{X = integer()}
  \item \texttt{Y = integer()}
  \item \texttt{Z = integer()}
  \item \texttt{W = integer()}
\end{itemize}
\textbf{See \texttt{vertex2d/2}}

\textbf{vertex4s(X, Y, Z, W) -&gt; ok} \\
Types:
\begin{itemize}
  \item \texttt{X = integer()}
  \item \texttt{Y = integer()}
  \item \texttt{Z = integer()}
  \item \texttt{W = integer()}
\end{itemize}
\textbf{See \texttt{vertex2d/2}}

\textbf{vertex2dv(V) -&gt; ok} \\
Types:
V = {X::float(), Y::float()}
Equivalent to vertex2d(X, Y).

vertex2fv(V) -> ok
Types:
  V = {X::float(), Y::float()}
Equivalent to vertex2f(X, Y).

vertex2iv(V) -> ok
Types:
  V = {X::integer(), Y::integer()}
Equivalent to vertex2i(X, Y).

vertex2sv(V) -> ok
Types:
  V = {X::integer(), Y::integer()}
Equivalent to vertex2s(X, Y).

vertex3dv(V) -> ok
Types:
  V = {X::float(), Y::float(), Z::float()}
Equivalent to vertex3d(X, Y, Z).

vertex3fv(V) -> ok
Types:
  V = {X::float(), Y::float(), Z::float()}
Equivalent to vertex3f(X, Y, Z).

vertex3iv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to vertex3i(X, Y, Z).

vertex3sv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to vertex3s(X, Y, Z).

vertex4dv(V) -> ok
Types:
  V = {X::float(), Y::float(), Z::float(), W::float()}
Equivalent to vertex4d(X, Y, Z, W).
vertex4fv(V) -> ok
Types:
  V = \{X::float(), Y::float(), Z::float(), W::float()\}
Equivalent to vertex4f(X, Y, Z, W).

vertex4iv(V) -> ok
Types:
  V = \{X::integer(), Y::integer(), Z::integer(), W::integer()\}
Equivalent to vertex4i(X, Y, Z, W).

vertex4sv(V) -> ok
Types:
  V = \{X::integer(), Y::integer(), Z::integer(), W::integer()\}
Equivalent to vertex4s(X, Y, Z, W).

normal3b(Nx, Ny, Nz) -> ok
Types:
  Nx = integer()
  Ny = integer()
  Nz = integer()
Set the current normal vector
The current normal is set to the given coordinates whenever \texttt{gl:}\texttt{\textunderscore normal} is issued. Byte, short, or integer arguments are converted to floating-point format with a linear mapping that maps the most positive representable integer value to 1.0 and the most negative representable integer value to -1.0.
See \texttt{external} documentation.

normal3d(Nx, Ny, Nz) -> ok
Types:
  Nx = float()
  Ny = float()
  Nz = float()
See \texttt{normal3b}/3

normal3f(Nx, Ny, Nz) -> ok
Types:
  Nx = float()
  Ny = float()
  Nz = float()
See \texttt{normal3b}/3

normal3i(Nx, Ny, Nz) -> ok
Types:
  Nx = integer()
\begin{verbatim}
Ny = integer()
Nz = integer()
See normal3b/3

normal3s(Nx, Ny, Nz) -> ok
Types:
    \(Nx = integer()\)
    \(Ny = integer()\)
    \(Nz = integer()\)
See normal3b/3

normal3bv(V) -> ok
Types:
    \(V = \{Nx::integer(), Ny::integer(), Nz::integer()\}\)
Equivalent to normal3b(Nx, Ny, Nz).

normal3dv(V) -> ok
Types:
    \(V = \{Nx::float(), Ny::float(), Nz::float()\}\)
Equivalent to normal3d(Nx, Ny, Nz).

normal3fv(V) -> ok
Types:
    \(V = \{Nx::float(), Ny::float(), Nz::float()\}\)
Equivalent to normal3f(Nx, Ny, Nz).

normal3iv(V) -> ok
Types:
    \(V = \{Nx::integer(), Ny::integer(), Nz::integer()\}\)
Equivalent to normal3i(Nx, Ny, Nz).

normal3sv(V) -> ok
Types:
    \(V = \{Nx::integer(), Ny::integer(), Nz::integer()\}\)
Equivalent to normal3s(Nx, Ny, Nz).

indexed(C) -> ok
Types:
    \(C = float()\)
Set the current color index
\end{verbatim}

\textit{gl:} indexed updates the current (single-valued) color index. It takes one argument, the new value for the current color index.
See external documentation.

\[
\begin{align*}
\text{indexf}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \text{float()} \\
\text{See indexd/1}
\end{align*}
\]

\[
\begin{align*}
\text{indexi}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \text{integer()} \\
\text{See indexd/1}
\end{align*}
\]

\[
\begin{align*}
\text{indexs}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \text{integer()} \\
\text{See indexd/1}
\end{align*}
\]

\[
\begin{align*}
\text{indexub}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \text{integer()} \\
\text{See indexd/1}
\end{align*}
\]

\[
\begin{align*}
\text{indexdv}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \{C::\text{float()}) \\
\text{Equivalent to indexd}(C).
\end{align*}
\]

\[
\begin{align*}
\text{indexfv}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \{C::\text{float()}) \\
\text{Equivalent to indexf}(C).
\end{align*}
\]

\[
\begin{align*}
\text{indexiv}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \{C::\text{integer()}) \\
\text{Equivalent to indexi}(C).
\end{align*}
\]

\[
\begin{align*}
\text{indexsv}(C) & \rightarrow \text{ok} \\
\text{Types:} & \\
\quad C & = \{C::\text{integer()}) \\
\text{Equivalent to indexs}(C).
\end{align*}
\]
indexubv(C) -> ok
Types:
  C = {C::integer()}
Equivalent to indexub(C).

color3b(Red, Green, Blue) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
Set the current color
The GL stores both a current single-valued color index and a current four-valued RGBA color. \texttt{gl:color} sets a new four-valued RGBA color. \texttt{gl:color} has two major variants: \texttt{gl:color3} and \texttt{gl:color4}. \texttt{gl:color3} variants specify new red, green, and blue values explicitly and set the current alpha value to 1.0 (full intensity) implicitly. \texttt{gl:color4} variants specify all four color components explicitly.
See \texttt{external} documentation.

color3d(Red, Green, Blue) -> ok
Types:
  Red = float()
  Green = float()
  Blue = float()
See \texttt{color3b/3}

color3f(Red, Green, Blue) -> ok
Types:
  Red = float()
  Green = float()
  Blue = float()
See \texttt{color3b/3}

color3i(Red, Green, Blue) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
See \texttt{color3b/3}

color3s(Red, Green, Blue) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
See color3b/3

color3ub(Red, Green, Blue) -> ok
Types:
   Red = integer()
   Green = integer()
   Blue = integer()
See color3b/3

color3ui(Red, Green, Blue) -> ok
Types:
   Red = integer()
   Green = integer()
   Blue = integer()
See color3b/3

color3us(Red, Green, Blue) -> ok
Types:
   Red = integer()
   Green = integer()
   Blue = integer()
See color3b/3

color4b(Red, Green, Blue, Alpha) -> ok
Types:
   Red = integer()
   Green = integer()
   Blue = integer()
   Alpha = integer()
See color3b/3

color4d(Red, Green, Blue, Alpha) -> ok
Types:
   Red = float()
   Green = float()
   Blue = float()
   Alpha = float()
See color3b/3

color4f(Red, Green, Blue, Alpha) -> ok
Types:
   Red = float()
   Green = float()
Blue = float()
Alpha = float()

See color3b/3

color4i(Red, Green, Blue, Alpha) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
  Alpha = integer()

See color3b/3

color4s(Red, Green, Blue, Alpha) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
  Alpha = integer()

See color3b/3

color4ub(Red, Green, Blue, Alpha) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
  Alpha = integer()

See color3b/3

color4ui(Red, Green, Blue, Alpha) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
  Alpha = integer()

See color3b/3

color4us(Red, Green, Blue, Alpha) -> ok
Types:
  Red = integer()
  Green = integer()
  Blue = integer()
  Alpha = integer()
color3bv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3b(Red, Green, Blue).

color3dv(V) -> ok
Types:
  V = {Red::float(), Green::float(), Blue::float()}
Equivalent to color3d(Red, Green, Blue).

color3fv(V) -> ok
Types:
  V = {Red::float(), Green::float(), Blue::float()}
Equivalent to color3f(Red, Green, Blue).

color3iv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3i(Red, Green, Blue).

color3sv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3s(Red, Green, Blue).

color3ubv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3ub(Red, Green, Blue).

color3uiv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3ui(Red, Green, Blue).

color3usv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to color3us(Red, Green, Blue).

color4bv(V) -> ok
Types:
  V = {Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()}
Equivalent to \texttt{color4b(Red, Green, Blue, Alpha)}.

\begin{verbatim}
color4dv(V) -> ok
Types:
  V = \{Red::float(), Green::float(), Blue::float(), Alpha::float()\}
Equivalent to \texttt{color4d(Red, Green, Blue, Alpha)}.

color4fv(V) -> ok
Types:
  V = \{Red::float(), Green::float(), Blue::float(), Alpha::float()\}
Equivalent to \texttt{color4f(Red, Green, Blue, Alpha)}.

color4iv(V) -> ok
Types:
  V = \{Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()\}
Equivalent to \texttt{color4i(Red, Green, Blue, Alpha)}.

color4sv(V) -> ok
Types:
  V = \{Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()\}
Equivalent to \texttt{color4s(Red, Green, Blue, Alpha)}.

color4ubv(V) -> ok
Types:
  V = \{Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()\}
Equivalent to \texttt{color4ub(Red, Green, Blue, Alpha)}.

color4uiv(V) -> ok
Types:
  V = \{Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()\}
Equivalent to \texttt{color4ui(Red, Green, Blue, Alpha)}.

color4usv(V) -> ok
Types:
  V = \{Red::integer(), Green::integer(), Blue::integer(), Alpha::integer()\}
Equivalent to \texttt{color4us(Red, Green, Blue, Alpha)}.

texCoord1d(S) -> ok
Types:
  S = float()
Set the current texture coordinates
\end{verbatim}
gl:texCoord specifies texture coordinates in one, two, three, or four dimensions. gl:texCoord1 sets the current texture coordinates to (s 0 0 1); a call to gl:texCoord2 sets them to (s t 0 1). Similarly, gl:texCoord3 specifies the texture coordinates as (s t r 1), and gl:texCoord4 defines all four components explicitly as (s t r q).

See external documentation.

texCoord1f(S) -> ok
Types:
  S = float()  
See texCoord1d/1

texCoord1i(S) -> ok
Types:
  S = integer()  
See texCoord1d/1

texCoord1s(S) -> ok
Types:
  S = integer()  
See texCoord1d/1

texCoord2d(S, T) -> ok
Types:
  S = float()  
  T = float()  
See texCoord1d/1

texCoord2f(S, T) -> ok
Types:
  S = float()  
  T = float()  
See texCoord1d/1

texCoord2i(S, T) -> ok
Types:
  S = integer()  
  T = integer()  
See texCoord1d/1

texCoord2s(S, T) -> ok
Types:
  S = integer()  
  T = integer()  
See texCoord1d/1
texCoord3d(S, T, R) -> ok
Types:
    S = float()
    T = float()
    R = float()
See texCoord1d/1

texCoord3f(S, T, R) -> ok
Types:
    S = float()
    T = float()
    R = float()
See texCoord1d/1

texCoord3i(S, T, R) -> ok
Types:
    S = integer()
    T = integer()
    R = integer()
See texCoord1d/1

texCoord3s(S, T, R) -> ok
Types:
    S = integer()
    T = integer()
    R = integer()
See texCoord1d/1

texCoord4d(S, T, R, Q) -> ok
Types:
    S = float()
    T = float()
    R = float()
    Q = float()
See texCoord1d/1

texCoord4f(S, T, R, Q) -> ok
Types:
    S = float()
    T = float()
    R = float()
    Q = float()
See texCoord1d/1
texCoord4i(S, T, R, Q) -> ok
Types:
  S = integer()
  T = integer()
  R = integer()
  Q = integer()
See texCoord1d/1

texCoord4s(S, T, R, Q) -> ok
Types:
  S = integer()
  T = integer()
  R = integer()
  Q = integer()
See texCoord1d/1

texCoord1dv(V) -> ok
Types:
  V = {S::float()}
Equivalent to texCoord1d(S).

texCoord1fv(V) -> ok
Types:
  V = {S::float()}
Equivalent to texCoord1f(S).

texCoord1iv(V) -> ok
Types:
  V = {S::integer()}
Equivalent to texCoord1i(S).

texCoord1sv(V) -> ok
Types:
  V = {S::integer()}
Equivalent to texCoord1s(S).

texCoord2dv(V) -> ok
Types:
  V = {S::float(), T::float()}
Equivalent to texCoord2d(S, T).

texCoord2fv(V) -> ok
Types:
\[ V = \{S::\text{float}, T::\text{float}\} \]
Equivalent to \text{TexCoord2f}(S, T).

\text{TexCoord2iv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{integer}, T::\text{integer}\} \]
Equivalent to \text{TexCoord2i}(S, T).

\text{TexCoord2sv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{integer}, T::\text{integer}\} \]
Equivalent to \text{TexCoord2s}(S, T).

\text{TexCoord3dv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{float}, T::\text{float}, R::\text{float}\} \]
Equivalent to \text{TexCoord3d}(S, T, R).

\text{TexCoord3fv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{float}, T::\text{float}, R::\text{float}\} \]
Equivalent to \text{TexCoord3f}(S, T, R).

\text{TexCoord3iv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{integer}, T::\text{integer}, R::\text{integer}\} \]
Equivalent to \text{TexCoord3i}(S, T, R).

\text{TexCoord3sv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{integer}, T::\text{integer}, R::\text{integer}\} \]
Equivalent to \text{TexCoord3s}(S, T, R).

\text{TexCoord4dv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{float}, T::\text{float}, R::\text{float}, Q::\text{float}\} \]
Equivalent to \text{TexCoord4d}(S, T, R, Q).

\text{TexCoord4fv}(V) \to \text{ok}
Types:
\[ V = \{S::\text{float}, T::\text{float}, R::\text{float}, Q::\text{float}\} \]
Equivalent to \text{TexCoord4f}(S, T, R, Q).
texCoord4iv(V) -> ok
Types:
   V = {S::integer(), T::integer(), R::integer(), Q::integer()}
Equivalent to texCoord4i(S, T, R, Q).

texCoord4sv(V) -> ok
Types:
   V = {S::integer(), T::integer(), R::integer(), Q::integer()}
Equivalent to texCoord4s(S, T, R, Q).

rasterPos2d(X, Y) -> ok
Types:
   X = float()
   Y = float()
Specify the raster position for pixel operations
The GL maintains a 3D position in window coordinates. This position, called the raster position, is used to position pixel and bitmap write operations. It is maintained with subpixel accuracy. See gl:bitmap/7, gl:drawPixels/5, and gl:copyPixels/5.
See external documentation.

rasterPos2f(X, Y) -> ok
Types:
   X = float()
   Y = float()
See rasterPos2d/2

rasterPos2i(X, Y) -> ok
Types:
   X = integer()
   Y = integer()
See rasterPos2d/2

rasterPos2s(X, Y) -> ok
Types:
   X = integer()
   Y = integer()
See rasterPos2d/2

rasterPos3d(X, Y, Z) -> ok
Types:
   X = float()
   Y = float()
   Z = float()
See `rasterPos2d/2`

`rasterPos3f(X, Y, Z) -> ok`
Types:

```
X = float()
Y = float()
Z = float()
```

See `rasterPos2d/2`

`rasterPos3i(X, Y, Z) -> ok`
Types:

```
X = integer()
Y = integer()
Z = integer()
```

See `rasterPos2d/2`

`rasterPos3s(X, Y, Z) -> ok`
Types:

```
X = integer()
Y = integer()
Z = integer()
```

See `rasterPos2d/2`

`rasterPos4d(X, Y, Z, W) -> ok`
Types:

```
X = float()
Y = float()
Z = float()
W = float()
```

See `rasterPos2d/2`

`rasterPos4f(X, Y, Z, W) -> ok`
Types:

```
X = float()
Y = float()
Z = float()
W = float()
```

See `rasterPos2d/2`

`rasterPos4i(X, Y, Z, W) -> ok`
Types:

```
X = integer()
Y = integer()
```
\[ Z = \text{integer()} \]
\[ W = \text{integer()} \]

See \texttt{rasterPos2d/2}

\texttt{rasterPos4s(X, Y, Z, W) \rightarrow ok}

Types:
\[
\begin{align*}
X &= \text{integer()} \\
Y &= \text{integer()} \\
Z &= \text{integer()} \\
W &= \text{integer()} \\
\end{align*}
\]

See \texttt{rasterPos2d/2}

\texttt{rasterPos2dv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{float()}, Y::\text{float()}\}
\]
Equivalent to \texttt{rasterPos2d(X, Y)}.

\texttt{rasterPos2fv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{float()}, Y::\text{float()}\}
\]
Equivalent to \texttt{rasterPos2f(X, Y)}.

\texttt{rasterPos2iv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{integer()}, Y::\text{integer()}\}
\]
Equivalent to \texttt{rasterPos2i(X, Y)}.

\texttt{rasterPos2sv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{integer()}, Y::\text{integer()}\}
\]
Equivalent to \texttt{rasterPos2s(X, Y)}.

\texttt{rasterPos3dv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{float()}, Y::\text{float()}, Z::\text{float()}\}
\]
Equivalent to \texttt{rasterPos3d(X, Y, Z)}.

\texttt{rasterPos3fv(V) \rightarrow ok}

Types:
\[
V = \{X::\text{float()}, Y::\text{float()}, Z::\text{float()}\}
\]
Equivalent to \texttt{rasterPos3f(X, Y, Z)}. 
rasterPos3iv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to rasterPos3i(X, Y, Z).

rasterPos3sv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to rasterPos3s(X, Y, Z).

rasterPos4dv(V) -> ok
Types:
  V = {X::float(), Y::float(), Z::float(), W::float()}
Equivalent to rasterPos4d(X, Y, Z, W).

rasterPos4fv(V) -> ok
Types:
  V = {X::float(), Y::float(), Z::float(), W::float()}
Equivalent to rasterPos4f(X, Y, Z, W).

rasterPos4iv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer(), W::integer()}
Equivalent to rasterPos4i(X, Y, Z, W).

rasterPos4sv(V) -> ok
Types:
  V = {X::integer(), Y::integer(), Z::integer(), W::integer()}
Equivalent to rasterPos4s(X, Y, Z, W).

rectd(X1, Y1, X2, Y2) -> ok
Types:
  X1 = float()
  Y1 = float()
  X2 = float()
  Y2 = float()

Draw a rectangle

gl:rect supports efficient specification of rectangles as two corner points. Each rectangle command takes four arguments, organized either as two consecutive pairs of (x y) coordinates or as two pointers to arrays, each containing an (x y) pair. The resulting rectangle is defined in the z=0 plane.

See external documentation.
rectf(X1, Y1, X2, Y2) -> ok
Types:
   X1 = float()
   Y1 = float()
   X2 = float()
   Y2 = float()
See rectd/4

recti(X1, Y1, X2, Y2) -> ok
Types:
   X1 = integer()
   Y1 = integer()
   X2 = integer()
   Y2 = integer()
See rectd/4

rects(X1, Y1, X2, Y2) -> ok
Types:
   X1 = integer()
   Y1 = integer()
   X2 = integer()
   Y2 = integer()
See rectd/4

rectdv(V1, V2) -> ok
Types:
   V1 = {float(), float()}
   V2 = {float(), float()}
See rectd/4

rectfv(V1, V2) -> ok
Types:
   V1 = {float(), float()}
   V2 = {float(), float()}
See rectd/4

rectiv(V1, V2) -> ok
Types:
   V1 = {integer(), integer()}
   V2 = {integer(), integer()}
See rectd/4
rectsv(V1, V2) -> ok

Types:

V1 = {integer(), integer()}
V2 = {integer(), integer()}

See rectd/4

vertexPointer(Size, Type, Stride, Ptr) -> ok

Types:

Size = integer()
Type = enum()
Stride = integer()
Ptr = offset() | mem()

Define an array of vertex data

gl:vertexPointer specifies the location and data format of an array of vertex coordinates to use when rendering. Size specifies the number of coordinates per vertex, and must be 2, 3, or 4. Type specifies the data type of each coordinate, and Stride specifies the byte stride from one vertex to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. (Single-array storage may be more efficient on some implementations; see gl:interleavedArrays/3.)

See external documentation.

normalPointer(Type, Stride, Ptr) -> ok

Types:

Type = enum()
Stride = integer()
Ptr = offset() | mem()

Define an array of normals

gl:normalPointer specifies the location and data format of an array of normals to use when rendering. Type specifies the data type of each normal coordinate, and Stride specifies the byte stride from one normal to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. (Single-array storage may be more efficient on some implementations; see gl:interleavedArrays/3.)

See external documentation.

colorPointer(Size, Type, Stride, Ptr) -> ok

Types:

Size = integer()
Type = enum()
Stride = integer()
Ptr = offset() | mem()

Define an array of colors

gl:colorPointer specifies the location and data format of an array of color components to use when rendering. Size specifies the number of components per color, and must be 3 or 4. Type specifies the data type of each color component, and Stride specifies the byte stride from one color to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. (Single-array storage may be more efficient on some implementations; see gl:interleavedArrays/3.)
See **external** documentation.

```erlang
indexPointer(Type, Stride, Ptr) -> ok
```

**Types:**
- `Type = enum()`  
- `Stride = integer()`  
- `Ptr = offset() | mem()`

Define an array of color indexes

**gl:indexPointer** specifies the location and data format of an array of color indexes to use when rendering. **Type** specifies the data type of each color index and **Stride** specifies the byte stride from one color index to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.

See **external** documentation.

```erlang
texCoordPointer(Size, Type, Stride, Ptr) -> ok
```

**Types:**
- `Size = integer()`  
- `Type = enum()`  
- `Stride = integer()`  
- `Ptr = offset() | mem()`

Define an array of texture coordinates

**gl:texCoordPointer** specifies the location and data format of an array of texture coordinates to use when rendering. **Size** specifies the number of coordinates per texture coordinate set, and must be 1, 2, 3, or 4. **Type** specifies the data type of each texture coordinate, and **Stride** specifies the byte stride from one texture coordinate set to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays. (Single-array storage may be more efficient on some implementations; see **gl:interleavedArrays/3**.)

See **external** documentation.

```erlang
edgeFlagPointer(Stride, Ptr) -> ok
```

**Types:**
- `Stride = integer()`  
- `Ptr = offset() | mem()`

Define an array of edge flags

**gl:edgeFlagPointer** specifies the location and data format of an array of boolean edge flags to use when rendering. **Stride** specifies the byte stride from one edge flag to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.

See **external** documentation.

```erlang
arrayElement(I) -> ok
```

**Types:**
- `I = integer()`

Render a vertex using the specified vertex array element

**gl:arrayElement** commands are used within **gl:'begin'/1**/ **gl:'begin'/1** pairs to specify vertex and attribute data for point, line, and polygon primitives. If ?GL_VERTEX_ARRAY is enabled when **gl:arrayElement** is called, a single
vertex is drawn, using vertex and attribute data taken from location \( I \) of the enabled arrays. If \(?\text{GL\_VERTEX\_ARRAY}\) is not enabled, no drawing occurs but the attributes corresponding to the enabled arrays are modified.

See external documentation.

drawArrays(Mode, First, Count) -> ok
Types:
   Mode = enum()
   First = integer()
   Count = integer()

Render primitives from array data

\( \text{gl\_drawArrays} \) specifies multiple geometric primitives with very few subroutine calls. Instead of calling a GL procedure to pass each individual vertex, normal, texture coordinate, edge flag, or color, you can prespecify separate arrays of vertices, normals, and colors and use them to construct a sequence of primitives with a single call to \( \text{gl\_drawArrays} \).

See external documentation.

drawElements(Mode, Count, Type, Indices) -> ok
Types:
   Mode = enum()
   Count = integer()
   Type = enum()
   Indices = offset() | mem()

Render primitives from array data

\( \text{gl\_drawElements} \) specifies multiple geometric primitives with very few subroutine calls. Instead of calling a GL function to pass each individual vertex, normal, texture coordinate, edge flag, or color, you can prespecify separate arrays of vertices, normals, and so on, and use them to construct a sequence of primitives with a single call to \( \text{gl\_drawElements} \).

See external documentation.

interleavedArrays(Format, Stride, Pointer) -> ok
Types:
   Format = enum()
   Stride = integer()
   Pointer = offset() | mem()

Simultaneously specify and enable several interleaved arrays

\( \text{gl\_interleavedArrays} \) lets you specify and enable individual color, normal, texture and vertex arrays whose elements are part of a larger aggregate array element. For some implementations, this is more efficient than specifying the arrays separately.

See external documentation.

shadeModel(Mode) -> ok
Types:
   Mode = enum()
Select flat or smooth shading

GL primitives can have either flat or smooth shading. Smooth shading, the default, causes the computed colors of vertices to be interpolated as the primitive is rasterized, typically assigning different colors to each resulting pixel fragment. Flat shading selects the computed color of just one vertex and assigns it to all the pixel fragments generated by rasterizing a single primitive. In either case, the computed color of a vertex is the result of lighting if lighting is enabled, or it is the current color at the time the vertex was specified if lighting is disabled.

See external documentation.

lightf(Light, Pname, Param) -> ok
Types:
    Light = enum()
    Pname = enum()
    Param = float()
Set light source parameters

gl:light sets the values of individual light source parameters. Light names the light and is a symbolic name of the form ?GL_LIGHTi, where i ranges from 0 to the value of ?GL_MAX_LIGHTS - 1. Pname specifies one of ten light source parameters, again by symbolic name. Param is either a single value or a pointer to an array that contains the new values.

See external documentation.

lighti(Light, Pname, Param) -> ok
Types:
    Light = enum()
    Pname = enum()
    Param = integer()
See lightf/3

lightfv(Light, Pname, Params) -> ok
Types:
    Light = enum()
    Pname = enum()
    Params = tuple()
See lightf/3

lightiv(Light, Pname, Params) -> ok
Types:
    Light = enum()
    Pname = enum()
    Params = tuple()
See lightf/3

getLightfv(Light, Pname) -> {float(), float(), float(), float()}
Types:
    Light = enum()
Return light source parameter values

gl:getLight returns in Params the value or values of a light source parameter. Light names the light and is a symbolic name of the form ?GL_LIGHT i where i ranges from 0 to the value of ?GL_MAX_LIGHTS - 1. ?GL_MAX_LIGHTS is an implementation dependent constant that is greater than or equal to eight. Pname specifies one of ten light source parameters, again by symbolic name.

See [external documentation](#).

gl:getLightiv(Light, Pname) -> {integer(), integer(), integer(), integer()}

Types:
- Light = enum()
- Pname = enum()

See [gl:getLightfv/2](#).

Set the lighting model parameters

gl:lightModel sets the lighting model parameter. Pname names a parameter and Params gives the new value.

There are three lighting model parameters:

See [external documentation](#).

Set the lighting model parameters

gl:lightModelf(Pname, Param) -> ok

Types:
- Pname = enum()
- Param = float()

See [gl:lightModelf/2](#).

Set the lighting model parameters

gl:lightModelfv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See [gl:lightModelf/2](#).

Set the lighting model parameters

gl:lightModeliv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See [gl:lightModelf/2](#).

Set the lighting model parameters

gl:lightModelf(Pname, Param) -> ok

Types:
- Pname = enum()
- Param = integer()

See [gl:lightModelf/2](#).

Set the lighting model parameters

gl:lightModelfv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See [gl:lightModelf/2](#).

Set the lighting model parameters

gl:lightModeliv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See [gl:lightModelf/2](#).
Specify material parameters for the lighting model

`gl:material` assigns values to material parameters. There are two matched sets of material parameters. One, the front-facing set, is used to shade points, lines, bitmaps, and all polygons (when two-sided lighting is disabled), or just front-facing polygons (when two-sided lighting is enabled). The other set, back-facing, is used to shade back-facing polygons only when two-sided lighting is enabled. Refer to the `gl:lightModelf/2` reference page for details concerning one- and two-sided lighting calculations.

See external documentation.

```erlang
materiali(Face, Pname, Param) -> ok
```

Types:

- `Face = enum()
- Pname = enum()
- Param = integer()

See `material/3`

```erlang
materialfv(Face, Pname, Params) -> ok
```

Types:

- `Face = enum()
- Pname = enum()
- Params = tuple()

See `material/3`

```erlang
materialiv(Face, Pname, Params) -> ok
```

Types:

- `Face = enum()
- Pname = enum()
- Params = tuple()

See `material/3`

```erlang
getMaterialfv(Face, Pname) -> {float(), float(), float(), float()}
```

Types:

- `Face = enum()
- Pname = enum()

Return material parameters

`gl:getMaterial` returns in `Params` the value or values of parameter `Pname` of material `Face`. Six parameters are defined:

See external documentation.

```erlang
getMaterialiv(Face, Pname) -> {integer(), integer(), integer(), integer()}
```

Types:
colorMaterial(Face, Mode) -> ok

Types:
   Face = enum()
   Mode = enum()

Cause a material color to track the current color

gl:colorMaterial specifies which material parameters track the current color. When !GL_COLOR_MATERIAL is enabled, the material parameter or parameters specified by Mode , of the material or materials specified by Face , track the current color at all times.

See external documentation.

pixelZoom(Xfactor, Yfactor) -> ok

Types:
   Xfactor = float()
   Yfactor = float()

Specify the pixel zoom factors

gl:pixelZoom specifies values for the x and y zoom factors. During the execution of gl:drawPixels/5 or gl:copyPixels/5, if ( xr, yr) is the current raster position, and a given element is in the mth row and nth column of the pixel rectangle, then pixels whose centers are in the rectangle with corners at

See external documentation.

pixelStoref(Pname, Param) -> ok

Types:
   Pname = enum()
   Param = float()

Set pixel storage modes

gl:pixelStore sets pixel storage modes that affect the operation of subsequent gl:readPixels/7 as well as the unpacking of texture patterns (see gl:texImage1D/8, gl:texImage2D/9, gl:texImage3D/10, gl:texSubImage1D/7, gl:texSubImage2D/7, gl:compressedTexImage1D/7, gl:compressedTexImage2D/8, gl:compressedTexImage3D/9, gl:compressedTexSubImage1D/7, gl:compressedTexSubImage2D/9, or gl:compressedTexSubImage1D/7).

See external documentation.

pixelStorei(Pname, Param) -> ok

Types:
   Pname = enum()
   Param = integer()

See pixelStoref/2
pixelTransferf(Pname, Param) -> ok
Types:
   Pname = enum()
   Param = float()

Set pixel transfer modes

`gl:pixelTransfer` sets pixel transfer modes that affect the operation of subsequent `gl:copyPixels/5`, `gl:copyTexImage1D/7`, `gl:copyTexImage2D/8`, `gl:copyTexSubImage1D/6`, `gl:copyTexSubImage2D/8`, `gl:copyTexSubImage3D/9`, `gl:drawPixels/5`, `gl:readPixels/7`, `gl:texImage1D/8`, `gl:texImage2D/9`, `gl:texImage3D/10`, `gl:texSubImage1D/7`, `gl:texSubImage2D/8`, and `gl:texSubImage3D/9` commands. Additionally, if the ARB_imaging subset is supported, the routines `gl:colorTable/6`, `gl:colorSubTable/6`, `gl:convolutionFilter1D/6`, `gl:convolutionFilter2D/7`, `gl:histogram/4`, `gl:minmax/3`, and `gl:separableFilter2D/8` are also affected.

The algorithms that are specified by pixel transfer modes operate on pixels after they are read from the frame buffer (`gl:copyPixels/5`, `gl:copyTexImage1D/7`, `gl:copyTexImage2D/8`, `gl:copyTexSubImage1D/6`, `gl:copyTexSubImage2D/8`, `gl:copyTexSubImage3D/9`, and `gl:readPixels/7`), or unpacked from client memory (`gl:drawPixels/5`, `gl:texImage1D/8`, `gl:texImage2D/9`, `gl:texImage3D/10`, `gl:texSubImage1D/7`, `gl:texSubImage2D/8`, and `gl:texSubImage3D/9`). Pixel transfer operations happen in the same order, and in the same manner, regardless of the command that resulted in the pixel operation. Pixel storage modes (see `gl:pixelStoref/2`) control the unpacking of pixels being read from client memory and the packing of pixels being written back into client memory.

See external documentation.

pixelTransferi(Pname, Param) -> ok
Types:
   Pname = enum()
   Param = integer()

See `pixelTransferf/2`

pixelMapfv(Map, Mapsize, Values) -> ok
Types:
   Map = enum()
   Mapsize = integer()
   Values = binary()

Set up pixel transfer maps

`gl:pixelMap` sets up translation tables, or maps, used by `gl:copyPixels/5`, `gl:copyTexImage1D/7`, `gl:copyTexImage2D/8`, `gl:copyTexSubImage1D/6`, `gl:copyTexSubImage2D/8`, `gl:copyTexSubImage3D/9`, `gl:drawPixels/5`, `gl:readPixels/7`, `gl:texImage1D/8`, `gl:texImage2D/9`, `gl:texImage3D/10`, `gl:texSubImage1D/7`, `gl:texSubImage2D/8`, and `gl:texSubImage3D/9`. Additionally, if the ARB_imaging subset is supported, the routines `gl:colorTable/6`, `gl:colorSubTable/6`, `gl:convolutionFilter1D/6`, `gl:convolutionFilter2D/7`, `gl:histogram/4`, `gl:minmax/3`, and `gl:separableFilter2D/8` use these maps. Use of these maps is described completely in the `gl:pixelTransferf/2` reference page, and partly in the reference pages for the pixel and texture image commands. Only the specification of the maps is described in this reference page.

See external documentation.

pixelMapuiv(Map, Mapsize, Values) -> ok
Types:
   Map = enum()
   Mapsize = integer()
Values = binary()
See pixelMapfv/3

pixelMapusv(Map, Mapsize, Values) -> ok
Types:
  Map = enum()
  Mapsize = integer()
  Values = binary()
See pixelMapfv/3

getAddress(Map, Values) -> ok
Types:
  Map = enum()
  Values = mem()

Return the specified pixel map
See the gl:pixelMapfv/3 reference page for a description of the acceptable values for the Map parameter.
See external documentation.

getAddressuiv(Map, Values) -> ok
Types:
  Map = enum()
  Values = mem()
See glGetPixelMapfv/2

getAddressusv(Map, Values) -> ok
Types:
  Map = enum()
  Values = mem()
See glGetPixelMapfv/2

bitmap(Width, Height, Xorig, Yorig, Xmove, Ymove, Bitmap) -> ok
Types:
  Width = integer()
  Height = integer()
  Xorig = float()
  Yorig = float()
  Xmove = float()
  Ymove = float()
A bitmap is a binary image. When drawn, the bitmap is positioned relative to the current raster position, and frame buffer pixels corresponding to 1's in the bitmap are written using the current raster color or index. Frame buffer pixels corresponding to 0's in the bitmap are not modified.

See external documentation.

readPixels(X, Y, Width, Height, Format, Type, Pixels) -> ok

Types:
- X = integer()
- Y = integer()
- Width = integer()
- Height = integer()
- Format = enum()
- Type = enum()
- Pixels = mem()

Read a block of pixels from the frame buffer

gl:readPixels returns pixel data from the frame buffer, starting with the pixel whose lower left corner is at location \((X, Y)\), into client memory starting at location Data. Several parameters control the processing of the pixel data before it is placed into client memory. These parameters are set with gl:pixelStorei/2. This reference page describes the effects on gl:readPixels of most, but not all of the parameters specified by these three commands.

See external documentation.

drawPixels(Width, Height, Format, Type, Pixels) -> ok

Types:
- Width = integer()
- Height = integer()
- Format = enum()
- Type = enum()
- Pixels = offset() | mem()

Write a block of pixels to the frame buffer

gl:drawPixels reads pixel data from memory and writes it into the frame buffer relative to the current raster position, provided that the raster position is valid. Use gl:rasterPos2d/2 or gl:windowPos2d/2 to set the current raster position; use gl:getBooleanv/1 with argument ?GL_CURRENT_RASTER_POSITION_VALID to determine if the specified raster position is valid, and gl:getBooleanv/1 with argument ?GL_CURRENT_RASTER_POSITION to query the raster position.

See external documentation.

copyPixels(X, Y, Width, Height, Type) -> ok

Types:
- X = integer()
- Y = integer()
- Width = integer()
Copy pixels in the frame buffer

`gl:copyPixels` copies a screen-aligned rectangle of pixels from the specified frame buffer location to a region relative to the current raster position. Its operation is well defined only if the entire pixel source region is within the exposed portion of the window. Results of copies from outside the window, or from regions of the window that are not exposed, are hardware dependent and undefined.

See external documentation.

```
stencilFunc(Func, Ref, Mask) -> ok
```

Types:
```
   Func = enum()
   Ref = integer()
   Mask = integer()
```

Set front and back function and reference value for stencil testing

Stenciling, like depth-buffering, enables and disables drawing on a per-pixel basis. Stencil planes are first drawn into using GL drawing primitives, then geometry and images are rendered using the stencil planes to mask out portions of the screen. Stenciling is typically used in multipass rendering algorithms to achieve special effects, such as decals, outlining, and constructive solid geometry rendering.

See external documentation.

```
stencilMask(Mask) -> ok
```

Types:
```
   Mask = integer()
```

Control the front and back writing of individual bits in the stencil planes

`gl:stencilMask` controls the writing of individual bits in the stencil planes. The least significant n bits of `Mask`, where n is the number of bits in the stencil buffer, specify a mask. Where a 1 appears in the mask, it's possible to write to the corresponding bit in the stencil buffer. Where a 0 appears, the corresponding bit is write-protected. Initially, all bits are enabled for writing.

See external documentation.

```
stencilOp(Fail, Zfail, Zpass) -> ok
```

Types:
```
   Fail = enum()
   Zfail = enum()
   Zpass = enum()
```

Set front and back stencil test actions

Stenciling, like depth-buffering, enables and disables drawing on a per-pixel basis. You draw into the stencil planes using GL drawing primitives, then render geometry and images, using the stencil planes to mask out portions of the screen. Stenciling is typically used in multipass rendering algorithms to achieve special effects, such as decals, outlining, and constructive solid geometry rendering.

See external documentation.
clearStencil(S) -> ok

Types:
    S = integer()

Specify the clear value for the stencil buffer

gl:clearStencil specifies the index used by gl:clear/1 to clear the stencil buffer. S is masked with 2^m - 1, where m is the number of bits in the stencil buffer.

See external documentation.

texGend(Coord, Pname, Param) -> ok

Types:
    Coord = enum()
    Pname = enum()
    Param = float()

Control the generation of texture coordinates

gl:texGen selects a texture-coordinate generation function or supplies coefficients for one of the functions. Coord names one of the (s, t, r, q) texture coordinates; it must be one of the symbols ?GL_S, ?GL_T, ?GL_R, or ?GL_Q. Pname must be one of three symbolic constants: ?GL_TEXTURE_GEN_MODE, ?GL_OBJECT_PLANE, or ?GL_EYE_PLANE. If Pname is ?GL_TEXTURE_GEN_MODE, then Params chooses a mode, one of ?GL_OBJECT_LINEAR, ?GL_EYE_LINEAR, ?GL_SPHERE_MAP, ?GL_NORMAL_MAP, or ?GL_REFLECTION_MAP. If Pname is either ?GL_OBJECT_PLANE or ?GL_EYE_PLANE, Params contains coefficients for the corresponding texture generation function.

See external documentation.

texGenf(Coord, Pname, Param) -> ok

Types:
    Coord = enum()
    Pname = enum()
    Param = float()

See texGend/3

texGeni(Coord, Pname, Param) -> ok

Types:
    Coord = enum()
    Pname = enum()
    Param = integer()

See texGend/3

texGendv(Coord, Pname, Params) -> ok

Types:
    Coord = enum()
    Pname = enum()
    Params = tuple()

See texGend/3
texGenfv(Coord, Pname, Params) -> ok
Types:
  Coord = enum()
  Pname = enum()
  Params = tuple()
See texGend/3

texGeniv(Coord, Pname, Params) -> ok
Types:
  Coord = enum()
  Pname = enum()
  Params = tuple()
See texGend/3

getTexGendv(Coord, Pname) -> {float(), float(), float(), float()}
Types:
  Coord = enum()
  Pname = enum()
Return texture coordinate generation parameters

gl:getTexGen returns in Params selected parameters of a texture coordinate generation function that was specified using gl:texGend/3. Coord names one of the (s, t, r, q) texture coordinates, using the symbolic constant ?GL_S, ?GL_T, ?GL_R, or ?GL_Q.
See external documentation.

gl:getTexGenfv(Coord, Pname) -> {float(), float(), float(), float()}
Types:
  Coord = enum()
  Pname = enum()
See getTexGendv/2

getTexGeniv(Coord, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Coord = enum()
  Pname = enum()
See getTexGendv/2

texEnvf(Target, Pname, Param) -> ok
Types:
  Target = enum()
  Pname = enum()
  Param = float()
See external documentation.
texEnvi(Target, Pname, Param) -> ok
Types:
   Target = enum()
   Pname = enum()
   Param = integer()

glTexEnvi
See external documentation.

texEnvfv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = tuple()

Set texture environment parameters


See external documentation.

texEnviv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = tuple()

See texEnvfv/3

gTexEnvfv(Target, Pname) -> {float(), float(), float(), float()}
Types:
   Target = enum()
   Pname = enum()

Return texture environment parameters

gl:getTexEnv returns in Params selected values of a texture environment that was specified with gl:texEnvfv/3. Target specifies a texture environment.

See external documentation.

gTexEnviv(Target, Pname) -> {integer(), integer(), integer(), integer()}
Types:
   Target = enum()
   Pname = enum()

See getTexEnvfv/2
texParameterf(Target, Pname, Param) -> ok
Types:
   Target = enum()
   Pname = enum()
   Param = float()

Set texture parameters

gl:texParameter assigns the value or values in Params to the texture parameter specified as Pname.
Target defines the target texture, either ?GL_TEXTURE_1D, ?GL_TEXTURE_2D, ?GL_TEXTURE_1D_ARRAY, ?GL_TEXTURE_2D_ARRAY, ?GL_TEXTURE_RECTANGLE, or ?GL_TEXTURE_3D. The following symbols are accepted in Pname:
See external documentation.

texParameteri(Target, Pname, Param) -> ok
Types:
   Target = enum()
   Pname = enum()
   Param = integer()
See texParameterf/3

texParameterfv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = tuple()
See texParameterf/3

texParameteriv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = tuple()
See texParameterf/3

getTexParameterfv(Target, Pname) -> {float(), float(), float(), float()}
Types:
   Target = enum()
   Pname = enum()

Return texture parameter values

gl:getTexParameter returns in Params the value or values of the texture parameter specified as Pname. Target defines the target texture. ?GL_TEXTURE_1D, ?GL_TEXTURE_2D, ?GL_TEXTURE_3D, ?GL_TEXTURE_1D_ARRAY, ?GL_TEXTURE_2D_ARRAY, ?GL_TEXTURE_RECTANGLE, ?GL_TEXTURE_CUBE_MAP, ?GL_TEXTURE_CUBE_MAP_ARRAY specify one-, two-, or three-dimensional, one-dimensional array, two-dimensional array, rectangle, cube-mapped or cube-mapped array texturing, respectively. Pname accepts the same symbols as gl:texParameterf/3, with the same interpretations:
See external documentation.

getTexParameteriv(Target, Pname) -> \{integer(), integer(), integer(), integer()\}
Types:
  Target = enum()
  Pname = enum()
See getTexParameterfv/2

gtexLevelParameterfv(Target, Level, Pname) -> \{float()\}
Types:
  Target = enum()
  Level = integer()
  Pname = enum()
Return texture parameter values for a specific level of detail.

gltexLevelParameter returns in Params texture parameter values for a specific
level-of-detail value, specified as Level. Target defines the target texture,
See external documentation.

gtexLevelParameteriv(Target, Level, Pname) -> \{integer()\}
Types:
  Target = enum()
  Level = integer()
  Pname = enum()
See getTexLevelParameterfv/3

texImage1D(Target, Level, InternalFormat, Width, Border, Format, Type, Pixels) -> ok
Types:
  Target = enum()
  Level = integer()
  InternalFormat = integer()
  Width = integer()
  Border = integer()
  Format = enum()
  Type = enum()
  Pixels = offset() | mem()
Specify a one-dimensional texture image
Texturing maps a portion of a specified texture image onto each graphical primitive for which texturing is enabled. To enable and disable one-dimensional texturing, call `gl:enable/1` and `gl:enable/1` with argument `?GL_TEXTURE_1D`. See external documentation.

```erlang
texImage2D(Target, Level, InternalFormat, Width, Height, Border, Format, Type, Pixels) -> ok
```

Types:
- `Target = enum()`
- `Level = integer()`
- `InternalFormat = integer()`
- `Width = integer()`
- `Height = integer()`
- `Border = integer()`
- `Format = enum()`
- `Type = enum()`
- `Pixels = offset() | mem()`

Specify a two-dimensional texture image

Texturing allows elements of an image array to be read by shaders. See external documentation.

```erlang
gl:getTexImage(Target, Level, Format, Type, Pixels) -> ok
```

Types:
- `Target = enum()`
- `Level = integer()`
- `Format = enum()`
- `Type = enum()`
- `Pixels = mem()`

Return a texture image into `Img`. `Target` specifies whether the desired texture image is one specified by `gl:texImage1D/8` (?GL_TEXTURE_1D), `gl:texImage2D/9` (?GL_TEXTURE_1D_ARRAY, ?GL_TEXTURE_RECTANGLE, ?GL_TEXTURE_2D or any of ?GL_TEXTURE_CUBE_MAP_*), or `gl:texImage3D/10` (?GL_TEXTURE_2D_ARRAY, ?GL_TEXTURE_3D). `Level` specifies the level-of-detail number of the desired image. `Format` and `Type` specify the format and type of the desired image array. See the reference page for `gl:texImage1D/8` for a description of the acceptable values for the `Format` and `Type` parameters, respectively. See external documentation.

```erlang
gl:genTextures(N) -> [integer()]
```

Types:
- `N = integer()`

Generate texture names

`gl:genTextures` returns `N` texture names in `Textures`. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to `gl:genTextures`. 
See external documentation.

`deleteTextures(Textures) -> ok`

Types:

- `Textures = [integer()]`

Delete named textures

`gl:deleteTextures` deletes N textures named by the elements of the array `Textures`. After a texture is deleted, it has no contents or dimensionality, and its name is free for reuse (for example by `gl:genTextures/1`). If a texture that is currently bound is deleted, the binding reverts to 0 (the default texture).

See external documentation.

`bindTexture(Target, Texture) -> ok`

Types:

- `Target = enum()`
- `Texture = integer()`

Bind a named texture to a texturing target

`gl:bindTexture` lets you create or use a named texture. Calling `gl:bindTexture` with `Target` set to `?GL_TEXTURE_1D`, `?GL_TEXTURE_2D`, `?GL_TEXTURE_3D`, or `?GL_TEXTURE_1D_ARRAY`, `?GL_TEXTURE_2D_ARRAY`, `?GL_TEXTURE_RECTANGLE`, `?GL_TEXTURE_CUBE_MAP`, `?GL_TEXTURE_2D_MULTISAMPLE` or `?GL_TEXTURE_2D_MULTISAMPLE_ARRAY` and `Texture` set to the name of the new texture binds the texture name to the target. When a texture is bound to a target, the previous binding for that target is automatically broken.

See external documentation.

`prioritizeTextures(Textures, Priorities) -> ok`

Types:

- `Textures = [integer()]`
- `Priorities = [clamp()]`

Set texture residence priority

`gl:prioritizeTextures` assigns the N texture priorities given in `Priorities` to the N textures named in `Textures`.

See external documentation.

`areTexturesResident(Textures) -> {0 | 1, Residences::[0 | 1]}`

Types:

- `Textures = [integer()]`

Determine if textures are loaded in texture memory

GL establishes a working set of textures that are resident in texture memory. These textures can be bound to a texture target much more efficiently than textures that are not resident.

See external documentation.

`isTexture(Texture) -> 0 | 1`

Types:
Texture = integer()

Determine if a name corresponds to a texture

`gl:isTexture` returns `?GL_TRUE` if `Texture` is currently the name of a texture. If `Texture` is zero, or is a non-zero value that is not currently the name of a texture, or if an error occurs, `gl:isTexture` returns `?GL_FALSE`.

See external documentation.

texSubImage1D(Target, Level, Xoffset, Width, Format, Type, Pixels) -> ok

Types:
- Target = `enum()`
- Level = `integer()`
- Xoffset = `integer()`
- Width = `integer()`
- Format = `enum()`
- Type = `enum()`
- Pixels = `offset()` | `mem()`

`glTexSubImage`  
See external documentation.

texSubImage2D(Target, Level, Xoffset, Yoffset, Width, Height, Format, Type, Pixels) -> ok

Types:
- Target = `enum()`
- Level = `integer()`
- Xoffset = `integer()`
- Yoffset = `integer()`
- Width = `integer()`
- Height = `integer()`
- Format = `enum()`
- Type = `enum()`
- Pixels = `offset()` | `mem()`

`glTexSubImage`  
See external documentation.

copyTexImage1D(Target, Level, Internalformat, X, Y, Width, Border) -> ok

Types:
- Target = `enum()`
- Level = `integer()`
- Internalformat = `enum()`
- X = `integer()`
- Y = `integer()`
- Width = `integer()`
- Border = `integer()`
Copy pixels into a 1D texture image

`gl:copyTexImage1D` defines a one-dimensional texture image with pixels from the current `?GL_READ_BUFFER`. See external documentation.

```erlang
copyTexImage1D(Target, Level, Internalformat, X, Y, Width, Height, Border) -> ok
```

Types:
- `Target` = `enum()`
- `Level` = `integer()`
- `Internalformat` = `enum()`
- `X` = `integer()`
- `Y` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Border` = `integer()`

Copy pixels into a 2D texture image

`gl:copyTexImage2D` defines a two-dimensional texture image, or cube-map texture image with pixels from the current `?GL_READ_BUFFER`. See external documentation.

```erlang
copyTexImage2D(Target, Level, Internalformat, X, Y, Width, Height, Border) -> ok
```

Types:
- `Target` = `enum()`
- `Level` = `integer()`
- `Internalformat` = `enum()`
- `X` = `integer()`
- `Y` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Border` = `integer()`

Copy a one-dimensional texture subimage

`gl:copyTexSubImage1D` replaces a portion of a one-dimensional texture image with pixels from the current `?GL_READ_BUFFER` (rather than from main memory, as is the case for `gl:texSubImage1D/7`). See external documentation.

```erlang
copyTexSubImage1D(Target, Level, XOffset, X, Y, Width) -> ok
```

Types:
- `Target` = `enum()`
- `Level` = `integer()`
- `XOffset` = `integer()`
- `X` = `integer()`
- `Y` = `integer()`
- `Width` = `integer()`

Copy a two-dimensional texture subimage

`gl:copyTexSubImage2D` replaces a portion of a two-dimensional texture image with pixels from the current `?GL_READ_BUFFER`. See external documentation.

```erlang
copyTexSubImage2D(Target, Level, XOffset, YOffset, X, Y, Width, Height) -> ok
```

Types:
- `Target` = `enum()`
- `Level` = `integer()`
- `XOffset` = `integer()`
- `YOffset` = `integer()`
- `X` = `integer()`
- `Y` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
Copy a two-dimensional texture subimage

`gl:copyTexSubImage2D` replaces a rectangular portion of a two-dimensional texture image or cube-map texture image with pixels from the current `?GL_READ_BUFFER` (rather than from main memory, as is the case for `gl:texSubImage1D/7`).

See external documentation.

```erlang
map1d(Target, U1, U2, Stride, Order, Points) -> ok
```

Types:

- `Target = enum()`
- `U1 = float()`
- `U2 = float()`
- `Stride = integer()`
- `Order = integer()`
- `Points = binary()`

See external documentation.

```erlang
map1f(Target, U1, U2, Stride, Order, Points) -> ok
```

Types:

- `Target = enum()`
- `U1 = float()`
- `U2 = float()`
- `Stride = integer()`
- `Order = integer()`
- `Points = binary()`

See external documentation.

```erlang
map2d(Target, U1, U2, Ustride, Uorder, V1, V2, Vstride, Vorder, Points) -> ok
```

Types:

- `Target = enum()`
- `U1 = float()`
- `U2 = float()`
- `Ustride = integer()`
- `Uorder = integer()`
- `V1 = float()`
- `V2 = float()`
- `Vstride = integer()`
- `Vorder = integer()`
- `Points = binary()`

See external documentation.
map2f(Target, U1, U2, Ustride, Uorder, V1, V2, Vstride, Vorder, Points) -> ok
Types:
  Target = enum()
  U1 = float()
  U2 = float()
  Ustride = integer()
  Uorder = integer()
  V1 = float()
  V2 = float()
  Vstride = integer()
  Vorder = integer()
  Points = binary()

See external documentation.

glMap

glMap

glMapdv(Target, Query, V) -> ok
Types:
  Target = enum()
  Query = enum()
  V = mem()

Return evaluator parameters

gl:map1d/6 and gl:map1d/6 define evaluators. gl:getMap returns evaluator parameters. Target chooses a map, Query selects a specific parameter, and V points to storage where the values will be returned.

See external documentation.

glMapfv(Target, Query, V) -> ok
Types:
  Target = enum()
  Query = enum()
  V = mem()

See glMapdv/3

glMapiv(Target, Query, V) -> ok
Types:
  Target = enum()
  Query = enum()
  V = mem()

See glMapdv/3

evalCoord1d(U) -> ok
Types:
  U = float()
Evaluate enabled one- and two-dimensional maps

`gl:evalCoord1` evaluates enabled one-dimensional maps at argument `U`. `gl:evalCoord2` does the same for two-dimensional maps using two domain values, `U` and `V`. To define a map, call `gl:map1d/6` and `gl:map1d/6`; to enable and disable it, call `gl:enable/1` and `gl:enable/1`.

See external documentation.

```erlang
evalCoord1f(U) -> ok
Types:
    U = float()
See evalCoord1d/1
```

```erlang
evalCoord1dv(U) -> ok
Types:
    U = {U::float()}
Equivalent to evalCoord1d(U).
```

```erlang
evalCoord1fv(U) -> ok
Types:
    U = {U::float()}
Equivalent to evalCoord1f(U).
```

```erlang
evalCoord2d(U, V) -> ok
Types:
    U = float()
    V = float()
See evalCoord1d/1
```

```erlang
evalCoord2f(U, V) -> ok
Types:
    U = float()
    V = float()
See evalCoord1d/1
```

```erlang
evalCoord2dv(U) -> ok
Types:
    U = {U::float(), V::float()}
Equivalent to evalCoord2d(U, V).
```

```erlang
evalCoord2fv(U) -> ok
Types:
    U = {U::float(), V::float()}
Equivalent to evalCoord2f(U, V).
```
mapGrid1d(Un, U1, U2) -> ok
Types:
    Un = integer()
    U1 = float()
    U2 = float()
Define a one- or two-dimensional mesh

mapGrid1f(Un, U1, U2) -> ok
Types:
    Un = integer()
    U1 = float()
    U2 = float()
See mapGrid1d/3

mapGrid2d(Un, U1, U2, Vn, V1, V2) -> ok
Types:
    Un = integer()
    U1 = float()
    U2 = float()
    Vn = integer()
    V1 = float()
    V2 = float()
See mapGrid1d/3

mapGrid2f(Un, U1, U2, Vn, V1, V2) -> ok
Types:
    Un = integer()
    U1 = float()
    U2 = float()
    Vn = integer()
    V1 = float()
    V2 = float()
See mapGrid1d/3

evalPoint1(I) -> ok
Types:
    I = integer()
Generate and evaluate a single point in a mesh

gl:mapGrid1d/3 and gl:evalMesh1/3 are used in tandem to efficiently generate and evaluate a series of evenly spaced map domain values. gl:evalPoint can be used to evaluate a single grid point in the same gridspace that is traversed by gl:evalMesh1/3. Calling gl:evalPoint1 is equivalent to calling glEvalCoord1( i,Δu+u ); where Δu=(u-u1)/n

See external documentation.

evalPoint2(I, J) -> ok

Types:
   I = integer()
   J = integer()

See evalPoint1/1

evalMesh1(Mode, I1, I2) -> ok

Types:
   Mode = enum()
   I1 = integer()
   I2 = integer()

Compute a one- or two-dimensional grid of points or lines

gl:mapGrid1d/3 and gl:evalMesh are used in tandem to efficiently generate and evaluate a series of evenly-spaced map domain values. gl:evalMesh steps through the integer domain of a one- or two-dimensional grid, whose range is the domain of the evaluation maps specified by gl:map1d/6 and gl:map1d/6. Mode determines whether the resulting vertices are connected as points, lines, or filled polygons.

See external documentation.

evalMesh2(Mode, I1, I2, J1, J2) -> ok

Types:
   Mode = enum()
   I1 = integer()
   I2 = integer()
   J1 = integer()
   J2 = integer()

See evalMesh1/3

fogf(Pname, Param) -> ok

Types:
   Pname = enum()
   Param = float()

Specify fog parameters

Fog is initially disabled. While enabled, fog affects rasterized geometry, bitmaps, and pixel blocks, but not buffer clear operations. To enable and disable fog, call gl:enable/1 and gl:enable/1 with argument ?GL_FOG.

See external documentation.
fogi(Pname, Param) -> ok

Types:
- Pname = enum()
- Param = integer()

See fogf/2

fogfv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See fogf/2

fogiv(Pname, Params) -> ok

Types:
- Pname = enum()
- Params = tuple()

See fogf/2

feedbackBuffer(Size, Type, Buffer) -> ok

Types:
- Size = integer()
- Type = enum()
- Buffer = mem()

Controls feedback mode

The gl:feedbackBuffer function controls feedback. Feedback, like selection, is a GL mode. The mode is selected by calling gl:renderMode/1 with ?GL_FEEDBACK. When the GL is in feedback mode, no pixels are produced by rasterization. Instead, information about primitives that would have been rasterized is fed back to the application using the GL.

See external documentation.

passThrough(Token) -> ok

Types:
- Token = float()

Place a marker in the feedback buffer

See external documentation.

selectBuffer(Size, Buffer) -> ok

Types:
- Size = integer()
- Buffer = mem()

Establish a buffer for selection mode values

gl:selectBuffer has two arguments: Buffer is a pointer to an array of unsigned integers, and Size indicates the size of the array. Buffer returns values from the name stack (see gl:initNames/0, gl:loadName/1, gl:pushName/1
gl

) when the rendering mode is ?GL_SELECT (see gl:renderMode/1). gl:selectBuffer must be issued before selection mode is enabled, and it must not be issued while the rendering mode is ?GL_SELECT.

See external documentation.

initNames() -> ok
Initialize the name stack

The name stack is used during selection mode to allow sets of rendering commands to be uniquely identified. It consists of an ordered set of unsigned integers. gl:initNames causes the name stack to be initialized to its default empty state.

See external documentation.

loadName(Name) -> ok
Types:
    Name = integer()
Load a name onto the name stack

The name stack is used during selection mode to allow sets of rendering commands to be uniquely identified. It consists of an ordered set of unsigned integers and is initially empty.

See external documentation.

pushName(Name) -> ok
Types:
    Name = integer()
Push and pop the name stack

The name stack is used during selection mode to allow sets of rendering commands to be uniquely identified. It consists of an ordered set of unsigned integers and is initially empty.

See external documentation.

popName() -> ok
See pushName/1

blendColor(Red, Green, Blue, Alpha) -> ok
Types:
    Red = clamp()
    Green = clamp()
    Blue = clamp()
    Alpha = clamp()
Set the blend color

The ?GL_BLEND_COLOR may be used to calculate the source and destination blending factors. The color components are clamped to the range [0 1] before being stored. See gl:blendFunc/2 for a complete description of the blending operations. Initially the ?GL_BLEND_COLOR is set to (0, 0, 0, 0).

See external documentation.
blendEquation(Mode) -> ok
Types:

    Mode = enum()

Specify the equation used for both the RGB blend equation and the Alpha blend equation.

The blend equations determine how a new pixel (the "source" color) is combined with a pixel already in the framebuffer (the "destination" color). This function sets both the RGB blend equation and the alpha blend equation to a single equation. gl:blendEquationi specifies the blend equation for a single draw buffer whereas gl:blendEquation sets the blend equation for all draw buffers.

See external documentation.

drawRangeElements(Mode, Start, End, Count, Type, Indices) -> ok
Types:

    Mode = enum()
    Start = integer()
    End = integer()
    Count = integer()
    Type = enum()
    Indices = offset() | mem()

Render primitives from array data.

gl:drawRangeElements is a restricted form of gl:drawElements/4. Mode, Start, End, and Count match the corresponding arguments to gl:drawElements/4, with the additional constraint that all values in the arrays Count must lie between Start and End, inclusive.

See external documentation.

texImage3D(Target, Level, InternalFormat, Width, Height, Depth, Border, Format, Type, Pixels) -> ok
Types:

    Target = enum()
    Level = integer()
    InternalFormat = integer()
    Width = integer()
    Height = integer()
    Depth = integer()
    Border = integer()
    Format = enum()
    Type = enum()
    Pixels = offset() | mem()

Specify a three-dimensional texture image.

Texturing maps a portion of a specified texture image onto each graphical primitive for which texturing is enabled. To enable and disable three-dimensional texturing, call gl:enable/1 and gl:enable/1 with argument ?GL_TEXTURE_3D.

See external documentation.
texSubImage3D(Target, Level, Xoffset, Yoffset, Zoffset, Width, Height, Depth, Format, Type, Pixels) -> ok

Types:
  Target = enum()
  Level = integer()
  Xoffset = integer()
  Yoffset = integer()
  Zoffset = integer()
  Width = integer()
  Height = integer()
  Depth = integer()
  Format = enum()
  Type = enum()
  Pixels = offset() | mem()

glTexSubImage
See external documentation.

copyTexSubImage3D(Target, Level, Xoffset, Yoffset, Zoffset, X, Y, Width, Height) -> ok

Types:
  Target = enum()
  Level = integer()
  Xoffset = integer()
  Yoffset = integer()
  Zoffset = integer()
  X = integer()
  Y = integer()
  Width = integer()
  Height = integer()

Copy a three-dimensional texture subimage
gl:copyTexSubImage3D replaces a rectangular portion of a three-dimensional texture image with pixels from the current ?GL_READ_BUFFER (rather than from main memory, as is the case for gl:texSubImage1D/7).
See external documentation.

colorTable(Target, Internalformat, Width, Format, Type, Table) -> ok

Types:
  Target = enum()
  Internalformat = enum()
  Width = integer()
  Format = enum()
  Type = enum()
  Table = offset() | mem()

Define a color lookup table
gl:colorTable may be used in two ways: to test the actual size and color resolution of a lookup table given a particular set of parameters, or to load the contents of a color lookup table. Use the targets ?GL_PROXY_* for the first case and the other targets for the second case.

See external documentation.

colorTableParameterfv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = {float(), float(), float(), float()}
Set color lookup table parameters

gl:colorTableParameter is used to specify the scale factors and bias terms applied to color components when they are loaded into a color table. Target indicates which color table the scale and bias terms apply to; it must be set to ?GL_COLOR_TABLE, ?GL_POST_CONVOLUTION_COLOR_TABLE, or ?GL_POST_COLOR_MATRIX_COLOR_TABLE.

See external documentation.

colorTableParameteriv(Target, Pname, Params) -> ok
Types:
   Target = enum()
   Pname = enum()
   Params = {integer(), integer(), integer(), integer()}
See colorTableParameterfv/3

copyColorTable(Target, Internalformat, X, Y, Width) -> ok
Types:
   Target = enum()
   Internalformat = enum()
   X = integer()
   Y = integer()
   Width = integer()
Copy pixels into a color table

gl:copyColorTable loads a color table with pixels from the current ?GL_READ_BUFFER (rather than from main memory, as is the case for gl:colorTable/6).

See external documentation.

getColorTable(Target, Format, Type, Table) -> ok
Types:
   Target = enum()
   Format = enum()
   Type = enum()
   Table = mem()
Retrieve contents of a color lookup table
**gl:getColorTable** returns in Table the contents of the color table specified by Target. No pixel transfer operations are performed, but pixel storage modes that are applicable to **gl:readPixels/7** are performed.

See external documentation.

```erlang
gl:getColorTableParameterfv(Target, Pname) -> {float(), float(), float(), float()}
Types:
  Target = enum()
  Pname = enum()
Get color lookup table parameters
Returns parameters specific to color table Target.
See external documentation.

gl:getColorTableParameteriv(Target, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Target = enum()
  Pname = enum()
See gl:getColorTableParameterfv/2

colorSubTable(Target, Start, Count, Format, Type, Data) -> ok
Types:
  Target = enum()
  Start = integer()
  Count = integer()
  Format = enum()
  Type = enum()
  Data = offset() | mem()
Respecify a portion of a color table

**gl:colorSubTable** is used to respecify a contiguous portion of a color table previously defined using **gl:colorTable/6**. The pixels referenced by Data replace the portion of the existing table from indices Start to start +count-1, inclusive. This region may not include any entries outside the range of the color table as it was originally specified. It is not an error to specify a subtexture with width of 0, but such a specification has no effect.
See external documentation.

copyColorSubTable(Target, Start, X, Y, Width) -> ok
Types:
  Target = enum()
  Start = integer()
  X = integer()
  Y = integer()
  Width = integer()
Respecify a portion of a color table
gl:copyColorSubTable is used to respecify a contiguous portion of a color table previously defined using gl:colorTable/6. The pixels copied from the framebuffer replace the portion of the existing table from indices Start to start+x-1, inclusive. This region may not include any entries outside the range of the color table, as was originally specified. It is not an error to specify a subtexture with width of 0, but such a specification has no effect.

See external documentation.

convolutionFilter1D(Target, Internalformat, Width, Format, Type, Image) -> ok
Types:
  Target = enum()
  Internalformat = enum()
  Width = integer()
  Format = enum()
  Type = enum()
  Image = offset() | mem()

Define a one-dimensional convolution filter

gl:convolutionFilter1D builds a one-dimensional convolution filter kernel from an array of pixels.
See external documentation.

convolutionFilter2D(Target, Internalformat, Width, Height, Format, Type, Image) -> ok
Types:
  Target = enum()
  Internalformat = enum()
  Width = integer()
  Height = integer()
  Format = enum()
  Type = enum()
  Image = offset() | mem()

Define a two-dimensional convolution filter

gl:convolutionFilter2D builds a two-dimensional convolution filter kernel from an array of pixels.
See external documentation.

convolutionParameterf(Target, Pname, Params) -> ok
Types:
  Target = enum()
  Pname = enum()
  Params = tuple()

Set convolution parameters

gl:convolutionParameter sets the value of a convolution parameter.
See external documentation.

convolutionParameterfv(Target::enum(), Pname::enum(), Params) -> ok
Types:
Params = {Params::tuple()}
Equivalent to convolutionParameterf(Target, Pname, Params).

convolutionParameteri(Target, Pname, Params) -> ok
Types:
  Target = enum()
  Pname = enum()
  Params = tuple()
See convolutionParameterf/3

convolutionParameteriv(Target::enum(), Pname::enum(), Params) -> ok
Types:
  Params = {Params::tuple()}
Equivalent to convolutionParameteri(Target, Pname, Params).

copyConvolutionFilter1D(Target, Internalformat, X, Y, Width) -> ok
Types:
  Target = enum()
  Internalformat = enum()
  X = integer()
  Y = integer()
  Width = integer()
Copy pixels into a one-dimensional convolution filter

gl:copyConvolutionFilter1D defines a one-dimensional convolution filter kernel with pixels from the current ?GL_READ_BUFFER (rather than from main memory, as is the case for gl:convolutionFilter1D/6).
See external documentation.

copyConvolutionFilter2D(Target, Internalformat, X, Y, Width, Height) -> ok
Types:
  Target = enum()
  Internalformat = enum()
  X = integer()
  Y = integer()
  Width = integer()
  Height = integer()
Copy pixels into a two-dimensional convolution filter

gl:copyConvolutionFilter2D defines a two-dimensional convolution filter kernel with pixels from the current ?GL_READ_BUFFER (rather than from main memory, as is the case for gl:convolutionFilter2D/7).
See external documentation.

getConvolutionFilter(Target, Format, Type, Image) -> ok
Types:
  Target = enum()
Get current 1D or 2D convolution filter kernel

`gl:getConvolutionFilter` returns the current 1D or 2D convolution filter kernel as an image. The one- or two-dimensional image is placed in `Image` according to the specifications in `Format` and `Type`. No pixel transfer operations are performed on this image, but the relevant pixel storage modes are applied.

See external documentation.

```erlang
getConvolutionParameterfv(Target, Pname) -> {float(), float(), float(), float()}
```

Types:

- **Target** = `enum()`
- **Pname** = `enum()`

Get convolution parameters

`gl:getConvolutionParameter` retrieves convolution parameters. `Target` determines which convolution filter is queried. `Pname` determines which parameter is returned:

See external documentation.

```erlang
getConvolutionParameteriv(Target, Pname) -> {integer(), integer(), integer(), integer()}
```

Types:

- **Target** = `enum()`
- **Pname** = `enum()`

See `getConvolutionParameterfv/2`

```erlang
separableFilter2D(Target, Internalformat, Width, Height, Format, Type, Row, Column) -> ok
```

Types:

- **Target** = `enum()`  
- **Internalformat** = `enum()`  
- **Width** = `integer()`  
- **Height** = `integer()`  
- **Format** = `enum()`  
- **Type** = `enum()`  
- **Row** = `offset()` | `mem()`  
- **Column** = `offset()` | `mem()`

Define a separable two-dimensional convolution filter

`gl:separableFilter2D` builds a two-dimensional separable convolution filter kernel from two arrays of pixels.

See external documentation.

```erlang
getHistogram(Target, Reset, Format, Type, Values) -> ok
```

Types:
Target = enum()
Reset = 0 | 1
Format = enum()
Type = enum()
Values = mem()

Get histogram table

`gl:getHistogram` returns the current histogram table as a one-dimensional image with the same width as the histogram. No pixel transfer operations are performed on this image, but pixel storage modes that are applicable to 1D images are honored.

See external documentation.

`getHistogramParameterfv(Target, Pname) -> {float()}
Types:
  Target = enum()
Pname = enum()

Get histogram parameters

`gl:getHistogramParameter` is used to query parameter values for the current histogram or for a proxy. The histogram state information may be queried by calling `gl:getHistogramParameter` with a `Target` of `GL_HISTOGRAM` (to obtain information for the current histogram table) or `GL_PROXY_HISTOGRAM` (to obtain information from the most recent proxy request) and one of the following values for the Pname argument:

See external documentation.

`getHistogramParameteriv(Target, Pname) -> {integer()}
Types:
  Target = enum()
Pname = enum()

Get minimum and maximum pixel values

`gl:getMinmax` returns the accumulated minimum and maximum pixel values (computed on a per-component basis) in a one-dimensional image of width 2. The first set of return values are the minima, and the second set of return values are the maxima. The format of the return values is determined by `Format`, and their type is determined by `Types`.

See external documentation.

`getMinmaxParameterfv(Target, Pname) -> {float()}
Types:
Target = enum()
Pname = enum()

Get minmax parameters

\texttt{gl:getMinmaxParameter} retrieves parameters for the current minmax table by setting \texttt{Pname} to one of the following values:

See \texttt{external} documentation.

\texttt{getMinmaxParameteriv(Target, Pname) -> \{integer()\}}

Types:
\begin{itemize}
  \item \texttt{Target = enum()}
  \item \texttt{Pname = enum()}
\end{itemize}

See \texttt{getMinmaxParameterfv/2}

\texttt{histogram(Target, Width, Internalformat, Sink) -> ok}

Types:
\begin{itemize}
  \item \texttt{Target = enum()}
  \item \texttt{Width = integer()}
  \item \texttt{Internalformat = enum()}
  \item \texttt{Sink = 0 | 1}
\end{itemize}

Define histogram table

When \texttt{?GL_HISTOGRAM} is enabled, RGBA color components are converted to histogram table indices by clamping to the range [0,1], multiplying by the width of the histogram table, and rounding to the nearest integer. The table entries selected by the RGBA indices are then incremented. (If the internal format of the histogram table includes luminance, then the index derived from the R color component determines the luminance table entry to be incremented.) If a histogram table entry is incremented beyond its maximum value, then its value becomes undefined. (This is not an error.)

See \texttt{external} documentation.

\texttt{minmax(Target, Internalformat, Sink) -> ok}

Types:
\begin{itemize}
  \item \texttt{Target = enum()}
  \item \texttt{Internalformat = enum()}
  \item \texttt{Sink = 0 | 1}
\end{itemize}

Define minmax table

When \texttt{?GL_MINMAX} is enabled, the RGBA components of incoming pixels are compared to the minimum and maximum values for each component, which are stored in the two-element minmax table. (The first element stores the minima, and the second element stores the maxima.) If a pixel component is greater than the corresponding component in the maximum element, then the maximum element is updated with the pixel component value. If a pixel component is less than the corresponding component in the minimum element, then the minimum element is updated with the pixel component value. (In both cases, if the internal format of the minmax table includes luminance, then the R color component of incoming pixels is used for comparison.) The contents of the minmax table may be retrieved at a later time by calling \texttt{gl:getMinmax/5}. The minmax operation is enabled or disabled by calling \texttt{gl:enable/1} or \texttt{gl:enable/1}, respectively, with an argument of \texttt{?GL_MINMAX}.

See \texttt{external} documentation.
resetHistogram(Target) -> ok
Types:
  Target = enum()
Reset histogram table entries to zero
gl:resetHistogram resets all the elements of the current histogram table to zero.
See external documentation.

resetMinmax(Target) -> ok
Types:
  Target = enum()
Reset minmax table entries to initial values
gl:resetMinmax resets the elements of the current minmax table to their initial values: the maximum element receives the minimum possible component values, and the minimum element receives the maximum possible component values.
See external documentation.

activeTexture(Texture) -> ok
Types:
  Texture = enum()
Select active texture unit
gl:activeTexture selects which texture unit subsequent texture state calls will affect. The number of texture units an implementation supports is implementation dependent, but must be at least 80.
See external documentation.

sampleCoverage(Value, Invert) -> ok
Types:
  Value = clamp()
  Invert = 0 | 1
Specify multisample coverage parameters
Multisampling samples a pixel multiple times at various implementation-dependent subpixel locations to generate antialiasing effects. Multisampling transparently antialiases points, lines, polygons, and images if it is enabled.
See external documentation.

compressedTexImage3D(Target, Level, Internalformat, Width, Height, Depth, Border, ImageSize, Data) -> ok
Types:
  Target = enum()
  Level = integer()
  Internalformat = enum()
  Width = integer()
  Height = integer()
  Depth = integer()
Border = integer()
ImageSize = integer()
Data = offset() | mem()

Specify a three-dimensional texture image in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.

compressedTexImage2D(Target, Level, Internalformat, Width, Height, Border, ImageSize, Data) -> ok
Types:
  Target = enum()
  Level = integer()
  Internalformat = enum()
  Width = integer()
  Height = integer()
  Border = integer()
  ImageSize = integer()
  Data = offset() | mem()

Specify a two-dimensional texture image in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.

compressedTexImage1D(Target, Level, Internalformat, Width, Border, ImageSize, Data) -> ok
Types:
  Target = enum()
  Level = integer()
  Internalformat = enum()
  Width = integer()
  Border = integer()
  ImageSize = integer()
  Data = offset() | mem()

Specify a one-dimensional texture image in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.

compressedTexSubImage3D(Target, Level, Xoffset, Yoffset, Zoffset, Width, Height, Depth, Format, ImageSize, Data) -> ok
Types:
  Target = enum()
  Level = integer()
  Xoffset = integer()
Specify a three-dimensional texture subimage in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.

\[
\text{compressedTexSubImage2D}(\text{Target}, \text{Level}, \text{Xoffset}, \text{Yoffset}, \text{Width}, \text{Height}, \text{Format}, \text{ImageSize}, \text{Data}) \to \text{ok}
\]

Types:
\[
\begin{align*}
\text{Target} & = \text{enum()} \\
\text{Level} & = \text{integer()} \\
\text{Xoffset} & = \text{integer()} \\
\text{Yoffset} & = \text{integer()} \\
\text{Width} & = \text{integer()} \\
\text{Height} & = \text{integer()} \\
\text{Format} & = \text{enum()} \\
\text{ImageSize} & = \text{integer()} \\
\text{Data} & = \text{offset()} \mid \text{mem()} \\
\end{align*}
\]

Specify a two-dimensional texture subimage in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.

\[
\text{compressedTexSubImage1D}(\text{Target}, \text{Level}, \text{Xoffset}, \text{Width}, \text{Format}, \text{ImageSize}, \text{Data}) \to \text{ok}
\]

Types:
\[
\begin{align*}
\text{Target} & = \text{enum()} \\
\text{Level} & = \text{integer()} \\
\text{Xoffset} & = \text{integer()} \\
\text{Width} & = \text{integer()} \\
\text{Format} & = \text{enum()} \\
\text{ImageSize} & = \text{integer()} \\
\text{Data} & = \text{offset()} \mid \text{mem()} \\
\end{align*}
\]

Specify a one-dimensional texture subimage in a compressed format
Texturing allows elements of an image array to be read by shaders.
See external documentation.
getCompressedTexImage(Target, Lod, Img) -> ok
Types:
    Target = enum()
    Lod = integer()
    Img = mem()

Return a compressed texture image
gl:getCompressedTexImage returns the compressed texture image associated with Target and Lod into Img. Img should be an array of GL_TEXTURE_COMPRESSED_IMAGE_SIZE bytes. Target specifies whether the desired texture image was one specified by gl:texImage1D/8 (?GL_TEXTURE_1D), gl:texImage2D/9 (?GL_TEXTURE_2D or any of ?GL_TEXTURE_CUBE_MAP_9), or gl:texImage3D/10 (?GL_TEXTURE_3D). Lod specifies the level-of-detail number of the desired image.
See external documentation.

clientActiveTexture(Texture) -> ok
Types:
    Texture = enum()

Select active texture unit
gl:clientActiveTexture selects the vertex array client state parameters to be modified by gl:texCoordPointer/4, and enabled or disabled with gl:enableClientState/1 or gl:enableClientState/1, respectively, when called with a parameter of ?GL_TEXTURE_COORD_ARRAY.
See external documentation.

multiTexCoord1d(Target, S) -> ok
Types:
    Target = enum()
    S = float()

Set the current texture coordinates
gl:multiTexCoord specifies texture coordinates in one, two, three, or four dimensions. gl:multiTexCoord1 sets the current texture coordinates to (s 0 0 1); a call to gl:multiTexCoord2 sets them to (s t 0 1). Similarly, gl:multiTexCoord3 specifies the texture coordinates as (s t r 1), and gl:multiTexCoord4 defines all four components explicitly as (s t r q).
See external documentation.

multiTexCoord1dv(Target::enum(), V) -> ok
Types:
    V = {S::float()}

Equivalent to multiTexCoord1d(Target, S).

multiTexCoord1f(Target, S) -> ok
Types:
    Target = enum()
    S = float()

See multiTexCoord1d/2
multiTexCoord1fv(Target::enum(), V) -> ok
Types:
  V = {S::float()}
Equivalent to multiTexCoord1f(Target, S).

display

multiTexCoord1i(Target, S) -> ok
Types:
  Target = enum()
  S = integer()
See multiTexCoord1f/2

display

multiTexCoord1iv(Target::enum(), V) -> ok
Types:
  V = {S::integer()}
Equivalent to multiTexCoord1i(Target, S).

display

multiTexCoord1s(Target, S) -> ok
Types:
  Target = enum()
  S = integer()
See multiTexCoord1f/2

display

multiTexCoord1sv(Target::enum(), V) -> ok
Types:
  V = {S::integer()}
Equivalent to multiTexCoord1s(Target, S).

display

multiTexCoord2d(Target, S, T) -> ok
Types:
  Target = enum()
  S = float()  T = float()
See multiTexCoord1d/2

display

multiTexCoord2dv(Target::enum(), V) -> ok
Types:
  V = {S::float(), T::float()}
Equivalent to multiTexCoord2d(Target, S, T).

display

multiTexCoord2f(Target, S, T) -> ok
Types:
  Target = enum()
  S = float()
T = float()
See multiTexCoord1d/2

multiTexCoord2fv(Target::enum(), V) -> ok
Types:
  V = {S::float(), T::float()}
Equivalent to multiTexCoord2f(Target, S, T).

multiTexCoord2i(Target, S, T) -> ok
Types:
  Target = enum()
  S = integer()
  T = integer()
See multiTexCoord1d/2

multiTexCoord2iv(Target::enum(), V) -> ok
Types:
  V = {S::integer(), T::integer()}
Equivalent to multiTexCoord2i(Target, S, T).

multiTexCoord2s(Target, S, T) -> ok
Types:
  Target = enum()
  S = integer()
  T = integer()
See multiTexCoord1d/2

multiTexCoord2sv(Target::enum(), V) -> ok
Types:
  V = {S::integer(), T::integer()}
Equivalent to multiTexCoord2s(Target, S, T).

multiTexCoord3d(Target, S, T, R) -> ok
Types:
  Target = enum()
  S = float()
  T = float()
  R = float()
See multiTexCoord1d/2

multiTexCoord3dv(Target::enum(), V) -> ok
Types:
  V = {S::float(), T::float(), R::float()}
Equivalent to `multiTexCoord3d(Target, S, T, R)`.

`multiTexCoord3f(Target, S, T, R)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{Target} & = \text{enum()} \\
   \text{S} & = \text{float()} \\
   \text{T} & = \text{float()} \\
   \text{R} & = \text{float()}
   \end{align*}
   \]
See `multiTexCoord1d/2`

`multiTexCoord3fv(Target::enum(), V)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{V} & = \{\text{S::float()}, \text{T::float()}, \text{R::float()}
   \end{align*}
   \]
Equivalent to `multiTexCoord3f(Target, S, T, R)`.

`multiTexCoord3i(Target, S, T, R)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{Target} & = \text{enum()} \\
   \text{S} & = \text{integer()} \\
   \text{T} & = \text{integer()} \\
   \text{R} & = \text{integer()}
   \end{align*}
   \]
See `multiTexCoord1d/2`

`multiTexCoord3iv(Target::enum(), V)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{V} & = \{\text{S::integer()}, \text{T::integer()}, \text{R::integer()}
   \end{align*}
   \]
Equivalent to `multiTexCoord3i(Target, S, T, R)`.

`multiTexCoord3s(Target, S, T, R)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{Target} & = \text{enum()} \\
   \text{S} & = \text{integer()} \\
   \text{T} & = \text{integer()} \\
   \text{R} & = \text{integer()}
   \end{align*}
   \]
See `multiTexCoord1d/2`

`multiTexCoord3sv(Target::enum(), V)` -> ok
Types:
   
   \[
   \begin{align*}
   \text{V} & = \{\text{S::integer()}, \text{T::integer()}, \text{R::integer()}
   \end{align*}
   \]
Equivalent to `multiTexCoord3s(Target, S, T, R)`.

`multiTexCoord4d(Target, S, T, R, Q)` -> ok
Types:
Target = enum()
S = float()
T = float()
R = float()
Q = float()
See multiTexCoord1d/2

multiTexCoord4dv(Target::enum(), V) -> ok
Types:
  V = {S::float(), T::float(), R::float(), Q::float()}
Equivalent to multiTexCoord4d(Target, S, T, R, Q).

multiTexCoord4f(Target, S, T, R, Q) -> ok
Types:
  Target = enum()
  S = float()
  T = float()
  R = float()
  Q = float()
See multiTexCoord1d/2

multiTexCoord4fv(Target::enum(), V) -> ok
Types:
  V = {S::float(), T::float(), R::float(), Q::float()}
Equivalent to multiTexCoord4f(Target, S, T, R, Q).

multiTexCoord4i(Target, S, T, R, Q) -> ok
Types:
  Target = enum()
  S = integer()
  T = integer()
  R = integer()
  Q = integer()
See multiTexCoord1d/2

multiTexCoord4iv(Target::enum(), V) -> ok
Types:
  V = {S::integer(), T::integer(), R::integer(), Q::integer()}
Equivalent to multiTexCoord4i(Target, S, T, R, Q).

multiTexCoord4s(Target, S, T, R, Q) -> ok
Types:
  Target = enum()
See `multiTexCoord1d/2`

```erlang
multiTexCoord4sv(Target::enum(), V) -> ok
```

Types:
```
V = {S::integer(), T::integer(), R::integer(), Q::integer()}
```

Equivalent to `multiTexCoord4s(Target, S, T, R, Q)`.

```erlang
loadTransposeMatrixf(M) -> ok
```

Types:
```
M = matrix()
```

Replace the current matrix with the specified row-major ordered matrix

`gl:loadTransposeMatrix` replaces the current matrix with the one whose elements are specified by `M`. The current matrix is the projection matrix, modelview matrix, or texture matrix, depending on the current matrix mode (see `gl:matrixMode/1`).

See external documentation.

```erlang
loadTransposeMatrixd(M) -> ok
```

Types:
```
M = matrix()
```

See `loadTransposeMatrixf/1`

```erlang
multTransposeMatrixf(M) -> ok
```

Types:
```
M = matrix()
```

Multiply the current matrix with the specified row-major ordered matrix

`gl:multTransposeMatrix` multiplies the current matrix with the one specified using `M`, and replaces the current matrix with the product.

See external documentation.

```erlang
multTransposeMatrixd(M) -> ok
```

Types:
```
M = matrix()
```

See `multTransposeMatrixf/1`

```erlang
blendFuncSeparate(SfactorRGB, DfactorRGB, SfactorAlpha, DfactorAlpha) -> ok
```

Types:
```
SfactorRGB = enum()
DfactorRGB = enum()
SfactorAlpha = enum()
DfactorAlpha = enum()
```
DfactorAlpha = enum()

Specify pixel arithmetic for RGB and alpha components separately.

Pixels can be drawn using a function that blends the incoming (source) RGBA values with the RGBA values that are already in the frame buffer (the destination values). Blending is initially disabled. Use gl:enable/1 and gl:enable/1 with argument ?GL_BLEND to enable and disable blending.

See external documentation.

multiDrawArrays(Mode, First, Count) -> ok

Types:

   Mode = enum()
   First = [integer()] | mem()
   Count = [integer()] | mem()

Render multiple sets of primitives from array data.

gl:multiDrawArrays specifies multiple sets of geometric primitives with very few subroutine calls. Instead of calling a GL procedure to pass each individual vertex, normal, texture coordinate, edge flag, or color, you can prespecify separate arrays of vertices, normals, and colors and use them to construct a sequence of primitives with a single call to gl:multiDrawArrays.

See external documentation.

pointParameterf(Pname, Param) -> ok

Types:

   Pname = enum()
   Param = float()

Specify point parameters.

The following values are accepted for Pname:

See external documentation.

pointParameterfv(Pname, Params) -> ok

Types:

   Pname = enum()
   Params = tuple()

See pointParameterf/2.

pointParameteri(Pname, Param) -> ok

Types:

   Pname = enum()
   Param = integer()

See pointParameterf/2.

pointParameteriv(Pname, Params) -> ok

Types:

   Pname = enum()
   Params = tuple()
See `pointParameterf/2`

```erlang
fogCoordf(Coord) -> ok
Types:
    Coord = float()
Set the current fog coordinates

egl:fogCoord specifies the fog coordinate that is associated with each vertex and the current raster position. The value specified is interpolated and used in computing the fog color (see `egl:fogf/2`).
See external documentation.

fogCoordfv(Coord) -> ok
Types:
    Coord = [float()]
Equivalent to `fogCoordf(Coord)`.

fogCoordd(Coord) -> ok
Types:
    Coord = float()
See `fogCoordf/1`

fogCoorddv(Coord) -> ok
Types:
    Coord = [float()]
Equivalent to `fogCoordd(Coord)`.

fogCoordPointer(Type, Stride, Pointer) -> ok
Types:
    Type = enum()
    Stride = integer()
    Pointer = offset() | mem()
Define an array of fog coordinates
egl:fogCoordPointer specifies the location and data format of an array of fog coordinates to use when rendering. Type specifies the data type of each fog coordinate, and Stride specifies the byte stride from one fog coordinate to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.
See external documentation.

secondaryColor3b(Red, Green, Blue) -> ok
Types:
    Red = integer()
    Green = integer()
    Blue = integer()
Set the current secondary color
The GL stores both a primary four-valued RGBA color and a secondary four-valued RGBA color (where alpha is always set to 0.0) that is associated with every vertex.

See *external* documentation.

```erlang
secondaryColor3bv(V) -> ok
Types:
    V = {Red::integer(), Green::integer(), Blue::integer()}
Equivalent to secondaryColor3b(Red, Green, Blue).

secondaryColor3d(Red, Green, Blue) -> ok
Types:
    Red = float()
    Green = float()
    Blue = float()
See secondaryColor3b/3

secondaryColor3dv(V) -> ok
Types:
    V = {Red::float(), Green::float(), Blue::float()}
Equivalent to secondaryColor3d(Red, Green, Blue).

secondaryColor3f(Red, Green, Blue) -> ok
Types:
    Red = float()
    Green = float()
    Blue = float()
See secondaryColor3b/3

secondaryColor3fv(V) -> ok
Types:
    V = {Red::float(), Green::float(), Blue::float()}
Equivalent to secondaryColor3f(Red, Green, Blue).

secondaryColor3i(Red, Green, Blue) -> ok
Types:
    Red = integer()
    Green = integer()
    Blue = integer()
See secondaryColor3b/3

secondaryColor3iv(V) -> ok
Types:
    V = {Red::integer(), Green::integer(), Blue::integer()}
```
Equivalent to \texttt{secondaryColor3b(Red, Green, Blue)}.

\texttt{secondaryColor3s(Red, Green, Blue)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{Red = integer()}
  \item \texttt{Green = integer()}
  \item \texttt{Blue = integer()}
\end{itemize}

See \texttt{secondaryColor3b/3}

\texttt{secondaryColor3sv(V)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{V = \{Red::integer(), Green::integer(), Blue::integer()\}}
\end{itemize}

Equivalent to \texttt{secondaryColor3s(Red, Green, Blue)}.

\texttt{secondaryColor3ub(Red, Green, Blue)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{Red = integer()}
  \item \texttt{Green = integer()}
  \item \texttt{Blue = integer()}
\end{itemize}

See \texttt{secondaryColor3b/3}

\texttt{secondaryColor3ubv(V)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{V = \{Red::integer(), Green::integer(), Blue::integer()\}}
\end{itemize}

Equivalent to \texttt{secondaryColor3ub(Red, Green, Blue)}.

\texttt{secondaryColor3ui(Red, Green, Blue)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{Red = integer()}
  \item \texttt{Green = integer()}
  \item \texttt{Blue = integer()}
\end{itemize}

See \texttt{secondaryColor3b/3}

\texttt{secondaryColor3uiu(V)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{V = \{Red::integer(), Green::integer(), Blue::integer()\}}
\end{itemize}

Equivalent to \texttt{secondaryColor3ui(Red, Green, Blue)}.

\texttt{secondaryColor3uis(Red, Green, Blue)} \rightarrow \texttt{ok}

Types:

\begin{itemize}
  \item \texttt{Red = integer()}
  \item \texttt{Green = integer()}
  \item \texttt{Blue = integer()}
\end{itemize}
See `secondaryColor3b/3`

`secondaryColor3usv(V) -> ok`
Types:

\[ V = \{\text{Red}::\text{integer}(), \text{Green}::\text{integer}(), \text{Blue}::\text{integer}()\} \]
Equivalent to `secondaryColor3us(Red, Green, Blue)`.

`secondaryColorPointer(Size, Type, Stride, Pointer) -> ok`
Types:

\[
\begin{align*}
\text{Size} &= \text{integer}() \\
\text{Type} &= \text{enum}() \\
\text{Stride} &= \text{integer}() \\
\text{Pointer} &= \text{offset}() \mid \text{mem}()
\end{align*}
\]

Define an array of secondary colors

`gl:secondaryColorPointer` specifies the location and data format of an array of color components to use when rendering. `Size` specifies the number of components per color, and must be 3. `Type` specifies the data type of each color component, and `Stride` specifies the byte stride from one color to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.

See [external documentation](#).

`windowPos2d(X, Y) -> ok`
Types:

\[
\begin{align*}
X &= \text{float}() \\
Y &= \text{float}()
\end{align*}
\]

Specify the raster position in window coordinates for pixel operations

The GL maintains a 3D position in window coordinates. This position, called the raster position, is used to position pixel and bitmap write operations. It is maintained with subpixel accuracy. See `gl:bitmap/7`, `gl:drawPixels/5`, and `gl:copyPixels/5`.

See [external documentation](#).

`windowPos2dv(V) -> ok`
Types:

\[ V = \{X::\text{float}(), Y::\text{float}()\} \]
Equivalent to `windowPos2d(X, Y)`.

`windowPos2f(X, Y) -> ok`
Types:

\[
\begin{align*}
X &= \text{float}() \\
Y &= \text{float}()
\end{align*}
\]
See `windowPos2d/2`

`windowPos2fv(V) -> ok`
Types:
V = \{X::float(), Y::float()\}
Equivalent to `windowPos2f(X, Y)`.

`windowPos2i(X, Y)` -> ok
Types:
\[X = \text{integer}()\]
\[Y = \text{integer}()\]
See `windowPos2d/2`

`windowPos2iv(V)` -> ok
Types:
\[V = \{X::integer(), Y::integer()\}\]
Equivalent to `windowPos2i(X, Y)`.

`windowPos2s(X, Y)` -> ok
Types:
\[X = \text{integer}()\]
\[Y = \text{integer}()\]
See `windowPos2d/2`

`windowPos2sv(V)` -> ok
Types:
\[V = \{X::integer(), Y::integer()\}\]
Equivalent to `windowPos2s(X, Y)`.

`windowPos3d(X, Y, Z)` -> ok
Types:
\[X = \text{float}()\]
\[Y = \text{float}()\]
\[Z = \text{float}()\]
See `windowPos2d/2`

`windowPos3dv(V)` -> ok
Types:
\[V = \{X::float(), Y::float(), Z::float()\}\]
Equivalent to `windowPos3d(X, Y, Z)`.

`windowPos3f(X, Y, Z)` -> ok
Types:
\[X = \text{float}()\]
\[Y = \text{float}()\]
\[Z = \text{float}()\]
See `windowPos2d/2`
windowPos3fv(V) -> ok
Types:
   V = {X::float(), Y::float(), Z::float()}
Equivalent to windowPos3f(X, Y, Z).

windowPos3i(X, Y, Z) -> ok
Types:
   X = integer()
   Y = integer()
   Z = integer()
See windowPos2d/2

windowPos3iv(V) -> ok
Types:
   V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to windowPos3i(X, Y, Z).

windowPos3s(X, Y, Z) -> ok
Types:
   X = integer()
   Y = integer()
   Z = integer()
See windowPos2d/2

windowPos3sv(V) -> ok
Types:
   V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to windowPos3s(X, Y, Z).

genQueries(N) -> [integer()]
Types:
   N = integer()
Generate query object names

gl:genQueries returns N query object names in Ids. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to gl:genQueries.
See external documentation.

deleteQueries(Ids) -> ok
Types:
   Ids = [integer()]
Delete named query objects
gl:deleteQueries deletes \( N \) query objects named by the elements of the array \( \text{Ids} \). After a query object is deleted, it has no contents, and its name is free for reuse (for example by \( \text{gl:genQueries/1} \)).

See external documentation.

\[
\text{isQuery}(\text{Id}) \rightarrow 0 \mid 1
\]

Types:

\[
\text{Id} = \text{integer()}
\]

Determine if a name corresponds to a query object.

\( \text{gl:isQuery} \) returns \?GL_TRUE\ if \( \text{Id} \) is currently the name of a query object. If \( \text{Id} \) is zero, or is a non-zero value that is not currently the name of a query object, or if an error occurs, \( \text{gl:isQuery} \) returns \?GL_FALSE\.

See external documentation.

\[
\text{beginQuery}(\text{Target}, \text{Id}) \rightarrow \text{ok}
\]

Types:

\[
\begin{align*}
\text{Target} & = \text{enum()} \\
\text{Id} & = \text{integer()}
\end{align*}
\]

Delimit the boundaries of a query object.

\( \text{gl:beginQuery} \) and \( \text{gl:beginQuery/2} \) delimit the boundaries of a query object. \( \text{Query} \) must be a name previously returned from a call to \( \text{gl:genQueries/1} \). If a query object with name \( \text{Id} \) does not yet exist it is created with the type determined by \( \text{Target} \). \( \text{Target} \) must be one of \?GL_SAMPLES_PASSED\, \?GL_ANY_SAMPLES_PASSED\, \?GL_PRIMITIVES_GENERATED\, \?GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN\, or \?GL_TIME_ELAPSED\.

The behavior of the query object depends on its type and is as follows.

See external documentation.

\[
\text{endQuery}(\text{Target}) \rightarrow \text{ok}
\]

Types:

\[
\text{Target} = \text{enum()}
\]

See \text{beginQuery/2}.

\[
\text{getQueryiv}(\text{Target}, \text{Pname}) \rightarrow \text{integer()}
\]

Types:

\[
\begin{align*}
\text{Target} & = \text{enum()} \\
\text{Pname} & = \text{enum()}
\end{align*}
\]

\( \text{glGetQuery} \) returns in \( \text{Params} \) a selected parameter of the query object.

See external documentation.

\[
\text{getQueryObjectiv}(\text{Id}, \text{Pname}) \rightarrow \text{integer()}
\]

Types:

\[
\begin{align*}
\text{Id} & = \text{integer()} \\
\text{Pname} & = \text{enum()}
\end{align*}
\]

Return parameters of a query object.

\( \text{gl:getQueryObject} \) returns in \( \text{Params} \) a selected parameter of the query object specified by \( \text{Id} \).
getQueryObjectui(Id, Pname) -> integer()

Types:
  Id = integer()
  Pname = enum()

See getQueryObjectiv/2

bindBuffer(Target, Buffer) -> ok

Types:
  Target = enum()
  Buffer = integer()

Bind a named buffer object

gl:bindBuffer binds a buffer object to the specified buffer binding point. Calling gl:bindBuffer with Target set to one of the accepted symbolic constants and Buffer set to the name of a buffer object binds that buffer object name to the target. If no buffer object with name Buffer exists, one is created with that name. When a buffer object is bound to a target, the previous binding for that target is automatically broken.

See external documentation.

deleteBuffers(Buffers) -> ok

Types:
  Buffers = [integer()]

Delete named buffer objects

gl:deleteBuffers deletes N buffer objects named by the elements of the array Buffers. After a buffer object is deleted, it has no contents, and its name is free for reuse (for example by gl:genBuffers/1). If a buffer object that is currently bound is deleted, the binding reverts to 0 (the absence of any buffer object).

See external documentation.

genBuffers(N) -> [integer()]

Types:
  N = integer()

Generate buffer object names

gl:genBuffers returns N buffer object names in Buffers. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to gl:genBuffers.

See external documentation.

isBuffer(Buffer) -> 0 | 1

Types:
  Buffer = integer()

Determine if a name corresponds to a buffer object

gl:isBuffer returns ?GL_TRUE if Buffer is currently the name of a buffer object. If Buffer is zero, or is a non-zero value that is not currently the name of a buffer object, or if an error occurs, gl:isBuffer returns ?GL_FALSE.
bufferData(Target, Size, Data, Usage) -> ok
Types:
   Target = enum()
   Size = integer()
   Data = offset() | mem()
   Usage = enum()

Creates and initializes a buffer object's data store

See external documentation.

bufferSubData(Target, Offset, Size, Data) -> ok
Types:
   Target = enum()
   Offset = integer()
   Size = integer()
   Data = offset() | mem()

Updates a subset of a buffer object's data store

See external documentation.

gl:bufferSubData redefines some or all of the data store for the buffer object currently bound to Target. Data starting at byte offset Offset and extending for Size bytes is copied to the data store from the memory pointed to by Data. An error is thrown if Offset and Size together define a range beyond the bounds of the buffer object's data store.

getBufferSubData(Target, Offset, Size, Data) -> ok
Types:
   Target = enum()
   Offset = integer()
   Size = integer()
   Data = mem()

Returns a subset of a buffer object's data store

See external documentation.

gl:getBufferSubData returns some or all of the data from the buffer object currently bound to Target. Data starting at byte offset Offset and extending for Size bytes is copied from the data store to the memory pointed to by Data. An error is thrown if the buffer object is currently mapped, or if Offset and Size together define a range beyond the bounds of the buffer object's data store.

See external documentation.

gl:getParameteriv(Target, Pname) -> integer()
Types:
   Target = enum()
Return parameters of a buffer object
gl:getBufferParameteriv returns in Data a selected parameter of the buffer object specified by Target.
See external documentation.

blendEquationSeparate(ModeRGB, ModeAlpha) -> ok
Types:
  ModeRGB = enum()
  ModeAlpha = enum()
Set the RGB blend equation and the alpha blend equation separately
The blend equations determines how a new pixel (the "source" color) is combined with a pixel already in the framebuffer (the "destination" color). These functions specific one blend equation for the RGB-color components and one blend equation for the alpha component. gl:blendEquationSeparatei specifies the blend equations for a single draw buffer whereas gl:blendEquationSeparate sets the blend equations for all draw buffers.
See external documentation.

drawBuffers(Bufs) -> ok
Types:
  Bufs = [enum()]
Specifies a list of color buffers to be drawn into
gl:drawBuffers defines an array of buffers into which outputs from the fragment shader data will be written. If a fragment shader writes a value to one or more user defined output variables, then the value of each variable will be written into the buffer specified at a location within Bufs corresponding to the location assigned to that user defined output. The draw buffer used for user defined outputs assigned to locations greater than or equal to N is implicitly set to ?GL_NONE and any data written to such an output is discarded.
See external documentation.

stencilOpSeparate(Face, Sfail, Dpfail, Dppass) -> ok
Types:
  Face = enum()
  Sfail = enum()
  Dpfail = enum()
  Dppass = enum()
Set front and/or back stencil test actions
Stenciling, like depth-buffering, enables and disables drawing on a per-pixel basis. You draw into the stencil planes using GL drawing primitives, then render geometry and images, using the stencil planes to mask out portions of the screen. Stenciling is typically used in multipass rendering algorithms to achieve special effects, such as decals, outlining, and constructive solid geometry rendering.
See external documentation.

stencilFuncSeparate(Face, Func, Ref, Mask) -> ok
Types:
  Face = enum()
Func = \texttt{enum()}

Ref = \texttt{integer()}

Mask = \texttt{integer()}

Set front and/or back function and reference value for stencil testing

Stenciling, like depth-buffering, enables and disables drawing on a per-pixel basis. You draw into the stencil planes using GL drawing primitives, then render geometry and images, using the stencil planes to mask out portions of the screen. Stenciling is typically used in multipass rendering algorithms to achieve special effects, such as decals, outlining, and constructive solid geometry rendering.

See external documentation.

\texttt{stencilMaskSeparate(Face, Mask) -> ok}

Types:

\begin{itemize}
  \item Face = \texttt{enum()}
  \item Mask = \texttt{integer()}
\end{itemize}

Control the front and/or back writing of individual bits in the stencil planes

\texttt{gl:stencilMaskSeparate} controls the writing of individual bits in the stencil planes. The least significant \(n\) bits of \texttt{Mask}, where \(n\) is the number of bits in the stencil buffer, specify a mask. Where a \(1\) appears in the mask, it's possible to write to the corresponding bit in the stencil buffer. Where a \(0\) appears, the corresponding bit is write-protected. Initially, all bits are enabled for writing.

See external documentation.

\texttt{attachShader(Program, Shader) -> ok}

Types:

\begin{itemize}
  \item Program = \texttt{integer()}
  \item Shader = \texttt{integer()}
\end{itemize}

Attaches a shader object to a program object

In order to create a complete shader program, there must be a way to specify the list of things that will be linked together. Program objects provide this mechanism. Shaders that are to be linked together in a program object must first be attached to that program object. \texttt{gl:attachShader} attaches the shader object specified by \texttt{Shader} to the program object specified by \texttt{Program}. This indicates that \texttt{Shader} will be included in link operations that will be performed on \texttt{Program}.

See external documentation.

\texttt{bindAttribLocation(Program, Index, Name) -> ok}

Types:

\begin{itemize}
  \item Program = \texttt{integer()}
  \item Index = \texttt{integer()}
  \item Name = \texttt{string()}
\end{itemize}

Associates a generic vertex attribute index with a named attribute variable

\texttt{gl:bindAttribLocation} is used to associate a user-defined attribute variable in the program object specified by \texttt{Program} with a generic vertex attribute index. The name of the user-defined attribute variable is passed as a null terminated string in \texttt{Name}. The generic vertex attribute index to be bound to this variable is specified by \texttt{Index}. When \texttt{Program} is made part of current state, values provided via the generic vertex attribute \texttt{Index} will modify the value of the user-defined attribute variable specified by \texttt{Name}.
See external documentation.

```erlang
compileShader(Shader) -> ok
Types:
  Shader = integer()
Compiles a shader object

gl:compileShader compiles the source code strings that have been stored in the shader object specified by Shader.
See external documentation.
```

```erlang
createProgram() -> integer()
Creates a program object

gl:createProgram creates an empty program object and returns a non-zero value by which it can be referenced. A program object is an object to which shader objects can be attached. This provides a mechanism to specify the shader objects that will be linked to create a program. It also provides a means for checking the compatibility of the shaders that will be used to create a program (for instance, checking the compatibility between a vertex shader and a fragment shader). When no longer needed as part of a program object, shader objects can be detached.
See external documentation.
```

```erlang
createShader(Type) -> integer()
Types:
  Type = enum()
Creates a shader object

gl:createShader creates an empty shader object and returns a non-zero value by which it can be referenced. A shader object is used to maintain the source code strings that define a shader. ShaderType indicates the type of shader to be created. Five types of shader are supported. A shader of type ?GL_VERTEX_SHADER is a shader that is intended to run on the programmable vertex processor. A shader of type ?GL_TESS_CONTROL_SHADER is a shader that is intended to run on the programmable tessellation processor in the control stage. A shader of type ?GL_TESS_EVALUATION_SHADER is a shader that is intended to run on the programmable tessellation processor in the evaluation stage. A shader of type ?GL_GEOMETRY_SHADER is a shader that is intended to run on the programmable geometry processor. A shader of type ?GL_FRAGMENT_SHADER is a shader that is intended to run on the programmable fragment processor.
See external documentation.
```

```erlang
deleteProgram(Program) -> ok
Types:
  Program = integer()
Deletes a program object

gl:deleteProgram frees the memory and invalidates the name associated with the program object specified by Program. This command effectively undoes the effects of a call to gl:createProgram/0.
See external documentation.
```

```erlang
deleteShader(Shader) -> ok
Types:
```

See external documentation.
Shader = integer()

Deletes a shader object

gl:deleteShader frees the memory and invalidates the name associated with the shader object specified by Shader. This command effectively undoes the effects of a call to gl:createShader/1.

See external documentation.

detachShader(Program, Shader) -> ok

Types:
    Program = integer()
    Shader = integer()

Detaches a shader object from a program object to which it is attached

gl:detachShader detaches the shader object specified by Shader from the program object specified by Program. This command can be used to undo the effect of the command gl:attachShader/2.

See external documentation.

disableVertexAttribArray(Index) -> ok

Types:
    Index = integer()

Enable or disable a generic vertex attribute array

gl:enableVertexAttribArray enables the generic vertex attribute array specified by Index. By default, all client-side capabilities are disabled, including all generic vertex attribute arrays. If enabled, the values in the generic vertex attribute array will be accessed and used for rendering when calls are made to vertex array commands such as gl:drawArrays/3, gl:drawElements/4, gl:drawRangeElements/6, see glMultiDrawElements, or gl:multiDrawArrays/3.

See external documentation.

enableVertexAttribArray(Index) -> ok

Types:
    Index = integer()

See disableVertexAttribArray/1

getActiveAttrib(Program, Index, BufSize) -> {Size::integer(), Type::enum(), Name::string()}

Types:
    Program = integer()
    Index = integer()
    BufSize = integer()

Returns information about an active attribute variable for the specified program object

gl:getActiveAttrib returns information about an active attribute variable in the program object specified by Program. The number of active attributes can be obtained by calling gl:getProgramiv/2 with the value ?GL_ACTIVE_ATTRIBUTES. A value of 0 for Index selects the first active attribute variable. Permissible values for Index range from 0 to the number of active attribute variables minus 1.
See external documentation.

`getActiveUniform(Program, Index, BufSize) -> {Size::integer(), Type::enum(), Name::string()}`

Types:
- `Program = integer()`
- `Index = integer()`
- `BufSize = integer()`

Returns information about an active uniform variable for the specified program object.

`gl:getActiveUniform` returns information about an active uniform variable in the program object specified by `Program`. The number of active uniform variables can be obtained by calling `gl:getProgramiv/2` with the value `?GL_ACTIVE_UNIFORMS`. A value of 0 for `Index` selects the first active uniform variable. Permissible values for `Index` range from 0 to the number of active uniform variables minus 1.

See external documentation.

`getAttachedShaders(Program, MaxCount) -> [integer()]`

Types:
- `Program = integer()`
- `MaxCount = integer()`

Returns the handles of the shader objects attached to a program object.

`gl:getAttachedShaders` returns the names of the shader objects attached to `Program`. The names of shader objects that are attached to `Program` will be returned in `Shaders`. The actual number of shader names written into `Shaders` is returned in `Count`. If no shader objects are attached to `Program`, `Count` is set to 0. The maximum number of shader names that may be returned in `Shaders` is specified by `MaxCount`.

See external documentation.

`getAttribLocation(Program, Name) -> integer()`

Types:
- `Program = integer()`
- `Name = string()`

Returns the location of an attribute variable.

`gl:getAttribLocation` queries the previously linked program object specified by `Program` for the attribute variable specified by `Name` and returns the index of the generic vertex attribute that is bound to that attribute variable. If `Name` is a matrix attribute variable, the index of the first column of the matrix is returned. If the named attribute variable is not an active attribute in the specified program object or if `Name` starts with the reserved prefix "gl_", a value of -1 is returned.

See external documentation.

`getProgramiv(Program, Pname) -> integer()`

Types:
- `Program = integer()`
- `Pname = enum()`

Returns a parameter from a program object.
gl:getProgram returns in Params the value of a parameter for a specific program object. The following parameters are defined:

See external documentation.

getProgramInfoLog(Program, BufSize) -> string()
Types:
   Program = integer()
   BufSize = integer()
Returns the information log for a program object

gl:getProgramInfoLog returns the information log for the specified program object. The information log for a program object is modified when the program object is linked or validated. The string that is returned will be null terminated.

See external documentation.

getShaderiv(Shader, Pname) -> integer()
Types:
   Shader = integer()
   Pname = enum()
Returns a parameter from a shader object

gl:getShader returns in Params the value of a parameter for a specific shader object. The following parameters are defined:

See external documentation.

getShaderInfoLog(Shader, BufSize) -> string()
Types:
   Shader = integer()
   BufSize = integer()
Returns the information log for a shader object

gl:getShaderInfoLog returns the information log for the specified shader object. The information log for a shader object is modified when the shader is compiled. The string that is returned will be null terminated.

See external documentation.

getShaderSource(Shader, BufSize) -> string()
Types:
   Shader = integer()
   BufSize = integer()
Returns the source code string from a shader object

gl:getShaderSource returns the concatenation of the source code strings from the shader object specified by Shader. The source code strings for a shader object are the result of a previous call to gl:shaderSource/2. The string returned by the function will be null terminated.

See external documentation.
getUniformLocation(Program, Name) -> integer()

Types:

- **Program** = integer()
- **Name** = string()

Returns the location of a uniform variable

gl:getUniformLocation returns an integer that represents the location of a specific uniform variable within a program object. Name must be a null terminated string that contains no white space. Name must be an active uniform variable name in Program that is not a structure, an array of structures, or a subcomponent of a vector or a matrix. This function returns -1 if Name does not correspond to an active uniform variable in Program, if Name starts with the reserved prefix "gl_", or if Name is associated with an atomic counter or a named uniform block.

See external documentation.

getUniformfv(Program, Location) -> matrix()

Types:

- **Program** = integer()
- **Location** = integer()

Returns the value of a uniform variable

gl:getUniform returns in Params the value(s) of the specified uniform variable. The type of the uniform variable specified by Location determines the number of values returned. If the uniform variable is defined in the shader as a boolean, int, or float, a single value will be returned. If it is defined as a vec2, ivec2, or bvec2, two values will be returned. If it is defined as a vec3, ivec3, or bvec3, three values will be returned, and so on. To query values stored in uniform variables declared as arrays, call gl:getUniform for each element of the array. To query values stored in uniform variables declared as structures, call gl:getUniform for each field in the structure. The values for uniform variables declared as a matrix will be returned in column major order.

See external documentation.

getUniformiv(Program, Location) -> {integer(), integer(), integer(),
integer(), integer(), integer(), integer(), integer(), integer(), integer(),
integer(), integer(), integer(), integer(), integer(), integer()}

Types:

- **Program** = integer()
- **Location** = integer()

See getUniformfv/2

getVertexAttribdv(Index, Pname) -> {float(), float(), float(), float()}

Types:

- **Index** = integer()
- **Pname** = enum()

Return a generic vertex attribute parameter

gl:getVertexAttrib returns in Params the value of a generic vertex attribute parameter. The generic vertex attribute to be queried is specified by Index, and the parameter to be queried is specified by Pname.

See external documentation.
getVertexAttribfv(Index, Pname) -> \{float(), float(), float(), float()\}
Types:
  Index = integer()
  Pname = enum()
See getVertexAttribdv/2

getVertexAttribiv(Index, Pname) -> \{integer(), integer(), integer(), integer()\}
Types:
  Index = integer()
  Pname = enum()
See getVertexAttribdv/2

isProgram(Program) -> 0 | 1
Types:
  Program = integer()
Determines if a name corresponds to a program object

isShader(Shader) -> 0 | 1
Types:
  Shader = integer()
Determines if a name corresponds to a shader object

linkProgram(Program) -> ok
Types:
  Program = integer()
Links a program object

See external documentation.
**shaderSource(Shader, String) -> ok**

Types:

- `Shader = integer()`
- `String = iolist()`

Replaces the source code in a shader object

`gl:shaderSource` sets the source code in `Shader` to the source code in the array of strings specified by `String`. Any source code previously stored in the shader object is completely replaced. The number of strings in the array is specified by `Count`. If `Length` is `?NULL`, each string is assumed to be null terminated. If `Length` is a value other than `?NULL`, it points to an array containing a string length for each of the corresponding elements of `String`. Each element in the `Length` array may contain the length of the corresponding string (the null character is not counted as part of the string length) or a value less than 0 to indicate that the string is null terminated. The source code strings are not scanned or parsed at this time; they are simply copied into the specified shader object.

See [external documentation](#).

**useProgram(Program) -> ok**

Types:

- `Program = integer()`

Installs a program object as part of current rendering state

`gl:useProgram` installs the program object specified by `Program` as part of current rendering state. One or more executables are created in a program object by successfully attaching shader objects to it with `gl:attachShader/2`, successfully compiling the shader objects with `gl:compileShader/1`, and successfully linking the program object with `gl:linkProgram/1`.

See [external documentation](#).

**uniform1f(Location, V0) -> ok**

Types:

- `Location = integer()`
- `V0 = float()`

Specify the value of a uniform variable for the current program object

`gl:uniform` modifies the value of a uniform variable or a uniform variable array. The location of the uniform variable to be modified is specified by `Location`, which should be a value returned by `gl:getUniformLocation/2`. `gl:uniform` operates on the program object that was made part of current state by calling `gl:useProgram/1`.

See [external documentation](#).

**uniform2f(Location, V0, V1) -> ok**

Types:

- `Location = integer()`
- `V0 = float()`
- `V1 = float()`

See `uniform1f/2`

**uniform3f(Location, V0, V1, V2) -> ok**

Types:

- `Location = integer()`
V0 = float()
V1 = float()
V2 = float()
See `uniform1f/2`

`uniform4f(Location, V0, V1, V2, V3) -> ok`
Types:
   Location = integer()
   V0 = float()
   V1 = float()
   V2 = float()
   V3 = float()
See `uniform1f/2`

`uniform1i(Location, V0) -> ok`
Types:
   Location = integer()
   V0 = integer()
See `uniform1f/2`

`uniform2i(Location, V0, V1) -> ok`
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
See `uniform1f/2`

`uniform3i(Location, V0, V1, V2) -> ok`
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
   V2 = integer()
See `uniform1f/2`

`uniform4i(Location, V0, V1, V2, V3) -> ok`
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
   V2 = integer()
   V3 = integer()
See `uniform1f/2`
uniform1fv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [float()]
See uniform1f/2

uniform2fv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{float(), float()}]
See uniform1f/2

uniform3fv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{float(), float(), float()}]
See uniform1f/2

uniform4fv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{float(), float(), float(), float()}]
See uniform1f/2

uniform1iv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [integer()]
See uniform1f/2

uniform2iv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{integer(), integer()}]
See uniform1f/2

uniform3iv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{integer(), integer(), integer()}]
See uniform1f/2
uniform4iv(Location, Value) -> ok
Types:
    Location = integer()
    Value = [{integer(), integer(), integer(), integer()}]
See uniform1f/2

uniformMatrix2fv(Location, Transpose, Value) -> ok
Types:
    Location = integer()
    Transpose = 0 | 1
    Value = [{float(), float(), float(), float()}]
See uniform1f/2

uniformMatrix3fv(Location, Transpose, Value) -> ok
Types:
    Location = integer()
    Transpose = 0 | 1
    Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float()}]
See uniform1f/2

uniformMatrix4fv(Location, Transpose, Value) -> ok
Types:
    Location = integer()
    Transpose = 0 | 1
    Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}]
See uniform1f/2

validateProgram(Program) -> ok
Types:
    Program = integer()
Validates a program object

gl:validateProgram checks to see whether the executables contained in Program can execute given the current OpenGL state. The information generated by the validation process will be stored in Program’s information log. The validation information may consist of an empty string, or it may be a string containing information about how the current program object interacts with the rest of current OpenGL state. This provides a way for OpenGL implementers to convey more information about why the current program is inefficient, suboptimal, failing to execute, and so on.
See external documentation.

vertexAttrib1d(Index, X) -> ok
Types:
    Index = integer()
X = float()
Specifies the value of a generic vertex attribute
The gl:vertexAttrib family of entry points allows an application to pass generic vertex attributes in numbered locations.
See external documentation.

vertexAttrib1dv(Index::integer(), V) -> ok
Types:
   V = {X::float()}
Equivalent to vertexAttrib1d(Index, X).

vertexAttrib1f(Index, X) -> ok
Types:
   Index = integer()
   X = float()
See vertexAttrib1d/2

vertexAttrib1fv(Index::integer(), V) -> ok
Types:
   V = {X::float()}
Equivalent to vertexAttrib1f(Index, X).

vertexAttrib1sv(Index::integer(), V) -> ok
Types:
   V = {X::integer()}
Equivalent to vertexAttrib1s(Index, X).

vertexAttrib2d(Index, X, Y) -> ok
Types:
   Index = integer()
   X = float()
   Y = float()
See vertexAttrib1d/2

vertexAttrib2dv(Index::integer(), V) -> ok
Types:
   V = {X::float(), Y::float()}
Equivalent to `vertexAttrib2d(Index, X, Y)`.

`vertexAttrib2f(Index, X, Y) -> ok`

Types:
- `Index = integer()`
- `X = float()`
- `Y = float()`

See `vertexAttrib1d/2`

`vertexAttrib2fv(Index::integer(), V) -> ok`

Types:
- `V = {X::float(), Y::float()}`

Equivalent to `vertexAttrib2f(Index, X, Y)`.

`vertexAttrib2s(Index, X, Y) -> ok`

Types:
- `Index = integer()`
- `X = integer()`
- `Y = integer()`

See `vertexAttrib1d/2`

`vertexAttrib2sv(Index::integer(), V) -> ok`

Types:
- `V = {X::integer(), Y::integer()}`

Equivalent to `vertexAttrib2s(Index, X, Y)`.

`vertexAttrib3d(Index, X, Y, Z) -> ok`

Types:
- `Index = integer()`
- `X = float()`
- `Y = float()`
- `Z = float()`

See `vertexAttrib1d/2`

`vertexAttrib3dv(Index::integer(), V) -> ok`

Types:
- `V = {X::float(), Y::float(), Z::float()}`

Equivalent to `vertexAttrib3d(Index, X, Y, Z)`.

`vertexAttrib3f(Index, X, Y, Z) -> ok`

Types:
- `Index = integer()`
- `X = float()`
Y = float(
Z = float()
See `vertexAttrib1d/2`

`vertexAttrib3fv(Index::integer(), V) -> ok`  
Types:  
    V = {X::float(), Y::float(), Z::float()}
Equivalent to `vertexAttrib3f(Index, X, Y, Z)`.

`vertexAttrib3s(Index, X, Y, Z) -> ok`  
Types:  
    Index = integer()  
    X = integer()  
    Y = integer()  
    Z = integer()  
See `vertexAttrib1d/2`

`vertexAttrib3sv(Index::integer(), V) -> ok`  
Types:  
    V = {X::integer(), Y::integer(), Z::integer()}
Equivalent to `vertexAttrib3s(Index, X, Y, Z)`.

`vertexAttrib4Nbv(Index, V) -> ok`  
Types:  
    Index = integer()  
    V = {integer(), integer(), integer(), integer()}
See `vertexAttrib1d/2`

`vertexAttrib4Niv(Index, V) -> ok`  
Types:  
    Index = integer()  
    V = {integer(), integer(), integer(), integer()}
See `vertexAttrib1d/2`

`vertexAttrib4Nsv(Index, V) -> ok`  
Types:  
    Index = integer()  
    V = {integer(), integer(), integer(), integer()}
See `vertexAttrib1d/2`

`vertexAttrib4Nub(Index, X, Y, Z, W) -> ok`  
Types:  
    Index = integer()
X = integer()
Y = integer()
Z = integer()
W = integer()
See vertexAttrib1d/2

vertexAttrib4Nubv(Index::integer(), V) -> ok
Types:
   V = {X::integer(), Y::integer(), Z::integer(), W::integer()}
Equivalent to vertexAttrib4Nub(Index, X, Y, Z, W).

vertexAttrib4Nuiv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See vertexAttrib1d/2

vertexAttrib4Nusv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See vertexAttrib1d/2

vertexAttrib4bv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See vertexAttrib1d/2

vertexAttrib4d[Index, X, Y, Z, W] -> ok
Types:
   Index = integer()
   X = float()
   Y = float()
   Z = float()
   W = float()
See vertexAttrib1d/2

vertexAttrib4dv[Index::integer(), V] -> ok
Types:
   V = {X::float(), Y::float(), Z::float(), W::float()}
Equivalent to vertexAttrib4d[Index, X, Y, Z, W].
vertexAttrib4f(Index, X, Y, Z, W) -> ok
Types:
   Index = integer()
   X = float()
   Y = float()
   Z = float()
   W = float()
See `vertexAttrib1d/2`

vertexAttrib4fv(Index::integer(), V) -> ok
Types:
   V = {X::float(), Y::float(), Z::float(), W::float()}
Equivalent to `vertexAttrib4f(Index, X, Y, Z, W)`.

vertexAttrib4iv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See `vertexAttrib1d/2`

vertexAttrib4s(Index, X, Y, Z, W) -> ok
Types:
   Index = integer()
   X = integer()
   Y = integer()
   Z = integer()
   W = integer()
See `vertexAttrib1d/2`

vertexAttrib4sv(Index::integer(), V) -> ok
Types:
   V = {X::integer(), Y::integer(), Z::integer(), W::integer()}
Equivalent to `vertexAttrib4s(Index, X, Y, Z, W)`.

vertexAttrib4ubv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See `vertexAttrib1d/2`

vertexAttrib4uiv(Index, V) -> ok
Types:
   Index = integer()
V = \{integer(), integer(), integer(), integer()\}

See `vertexAttrib1d/2`

`vertexAttrib4usv(Index, V) -> ok`

Types:
- `Index = integer()`
- `V = \{integer(), integer(), integer(), integer()\}`

See `vertexAttrib1d/2`

`vertexAttribPointer(Index, Size, Type, Normalized, Stride, Pointer) -> ok`

Types:
- `Index = integer()`
- `Size = integer()`
- `Type = enum()`
- `Normalized = 0 | 1`
- `Stride = integer()`
- `Pointer = offset() | mem()`

Define an array of generic vertex attribute data

`gl:vertexAttribPointer`, `gl:vertexAttribIPointer` and `gl:vertexAttribLPointer` specify the location and data format of the array of generic vertex attributes at index `Index` to use when rendering. `Size` specifies the number of components per attribute and must be 1, 2, 3, 4, or `?GL_BGRA`. `Type` specifies the data type of each component, and `Stride` specifies the byte stride from one attribute to the next, allowing vertices and attributes to be packed into a single array or stored in separate arrays.

See external documentation.

`uniformMatrix2x3fv(Location, Transpose, Value) -> ok`

Types:
- `Location = integer()`
- `Transpose = 0 | 1`
- `Value = \{(float(), float(), float(), float(), float(), float())\}`

See `uniform1f/2`

`uniformMatrix3x2fv(Location, Transpose, Value) -> ok`

Types:
- `Location = integer()`
- `Transpose = 0 | 1`
- `Value = \{(float(), float(), float(), float(), float(), float())\}`

See `uniform1f/2`

`uniformMatrix2x4fv(Location, Transpose, Value) -> ok`

Types:
- `Location = integer()`
- `Transpose = 0 | 1`
Value = [{float(), float(), float(), float(), float(), float(), float(), float()}]

See uniform1fv

uniformMatrix4x2fv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

See uniform1fv

uniformMatrix3x4fv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

See uniform1fv

uniformMatrix4x3fv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

See uniform1fv

colorMaski(Index, R, G, B, A) -> ok

Types:
  Index = integer()
  R = 0 | 1
  G = 0 | 1
  B = 0 | 1
  A = 0 | 1

glColorMaski

See external documentation.

getBooleani_v(Target, Index) -> [0 | 1]

Types:
  Target = enum()
  Index = integer()

See getBooleanv/1
### getIntegeri_v

*Signature:* `getIntegeri_v(Target, Index) -> [integer()]`

*Types:*

- `Target` = `enum()`  
- `Index` = `integer()`

*See:* `getBooleanv/1`

### enablei

*Signature:* `enablei(Target, Index) -> ok`

*Types:*

- `Target` = `enum()`  
- `Index` = `integer()`

*See:* `enable/1`

### disablei

*Signature:* `disablei(Target, Index) -> ok`

*Types:*

- `Target` = `enum()`  
- `Index` = `integer()`

*See:* `enable/1`

### isEnabledi

*Signature:* `isEnabledi(Target, Index) -> 0 | 1`

*Types:*

- `Target` = `enum()`  
- `Index` = `integer()`

*glEnablei*

*See:* `enable/1`

### beginTransformFeedback

*Signature:* `beginTransformFeedback(PrimitiveMode) -> ok`

*Types:*

- `PrimitiveMode` = `enum()`

*Start transform feedback operation*

Transform feedback mode captures the values of varying variables written by the vertex shader (or, if active, the geometry shader). Transform feedback is said to be active after a call to `gl:beginTransformFeedback` until a subsequent call to `gl:beginTransformFeedback/1` . Transform feedback commands must be paired.

*See:* `external` documentation

### endTransformFeedback

*Signature:* `endTransformFeedback() -> ok`

*See:* `beginTransformFeedback/1`

### bindBufferRange

*Signature:* `bindBufferRange(Target, Index, Buffer, Offset, Size) -> ok`

*Types:*

- `Target` = `enum()`  
- `Index` = `integer()`  
- `Buffer` = `integer()`
**Offset** = integer()

**Size** = integer()

Bind a range within a buffer object to an indexed buffer target

`gl:bindBufferRange` binds a range the buffer object `Buffer` represented by `Offset` and `Size` to the binding point at index `Index` of the array of targets specified by `Target`. Each `Target` represents an indexed array of buffer binding points, as well as a single general binding point that can be used by other buffer manipulation functions such as `gl:bindBuffer/2` or see `glMapBuffer`. In addition to binding a range of `Buffer` to the indexed buffer binding target, `gl:bindBufferBase` also binds the range to the generic buffer binding point specified by `Target`.

See [external documentation](#).

`unbindBufferBase(Target, Index, Buffer) -> ok`

Types:

- **Target** = enum()
- **Index** = integer()
- **Buffer** = integer()

Bind a buffer object to an indexed buffer target

`gl:bindBufferBase` binds the buffer object `Buffer` to the binding point at index `Index` of the array of targets specified by `Target`. Each `Target` represents an indexed array of buffer binding points, as well as a single general binding point that can be used by other buffer manipulation functions such as `gl:bindBuffer/2` or see `glMapBuffer`. In addition to binding `Buffer` to the indexed buffer binding target, `gl:bindBufferBase` also binds `Buffer` to the generic buffer binding point specified by `Target`.

See [external documentation](#).

`transformFeedbackVaryings(Program, Varyings, BufferMode) -> ok`

Types:

- **Program** = integer()
- **Varyings** = iolist()
- **BufferMode** = enum()

Specify values to record in transform feedback buffers

The names of the vertex or geometry shader outputs to be recorded in transform feedback mode are specified using `gl:transformFeedbackVaryings`. When a geometry shader is active, transform feedback records the values of selected geometry shader output variables from the emitted vertices. Otherwise, the values of the selected vertex shader outputs are recorded.

See [external documentation](#).

`getTransformFeedbackVarying(Program, Index, BufSize) -> {Size::integer(), Type::enum(), Name::string()}`

Types:

- **Program** = integer()
- **Index** = integer()
- **BufSize** = integer()

Retrieve information about varying variables selected for transform feedback

Information about the set of varying variables in a linked program that will be captured during transform feedback may be retrieved by calling `gl:getTransformFeedbackVarying`. `gl:getTransformFeedbackVarying`
gl

provides information about the varying variable selected by Index. An Index of 0 selects the first varying variable specified in the Varyings array passed to gl:transformFeedbackVaryings/3, and an Index of GL_TRANSFORM_FEEDBACK_VARYINGS-1 selects the last such variable.

See external documentation.

clampColor(Target, Clamp) -> ok
Types:
   Target = enum()
   Clamp = enum()

specify whether data read via
gl:readPixels/7 should be clamped

gl:clampColor controls color clamping that is performed during gl:readPixels/7. Target must be GL_CLAMP_READ_COLOR. If Clamp is GL_TRUE, read color clamping is enabled; if Clamp is GL_FALSE, read color clamping is disabled. If Clamp is GL_FIXED_ONLY, read color clamping is enabled only if the selected read buffer has fixed point components and disabled otherwise.

See external documentation.

beginConditionalRender(Id, Mode) -> ok
Types:
   Id = integer()
   Mode = enum()

Start conditional rendering

Conditional rendering is started using gl:beginConditionalRender and ended using gl:endConditionalRender. During conditional rendering, all vertex array commands, as well as gl:clear/1 and gl:clearBufferiv/3 have no effect if the (?GL_SAMPLES_PASSED) result of the query object Id is zero, or if the (?GL_ANY_SAMPLES_PASSED) result is ?GL_FALSE. The results of commands setting the current vertex state, such as gl:vertexAttrib1d/2 are undefined. If the (?GL_SAMPLES_PASSED) result is non-zero or if the (?GL_ANY_SAMPLES_PASSED) result is ?GL_TRUE, such commands are not discarded. The Id parameter to gl:beginConditionalRender must be the name of a query object previously returned from a call to gl:genQueries/1. Mode specifies how the results of the query object are to be interpreted. If Mode is ?GL_QUERY_WAIT, the GL waits for the results of the query to be available and then uses the results to determine if subsequent rendering commands are discarded. If Mode is ?GL_QUERY_NO_WAIT, the GL may choose to unconditionally execute the subsequent rendering commands without waiting for the query to complete.

See external documentation.

endConditionalRender() -> ok
See beginConditionalRender/2

vertexAttribIPointer(Index, Size, Type, Stride, Pointer) -> ok
Types:
   Index = integer()
   Size = integer()
   Type = enum()
   Stride = integer()
   Pointer = offset() | mem()
glVertexAttribIPointer
See external documentation.

getVertexAttribIiv(Index, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Index = integer()
  Pname = enum()
See getVertexAttribdv/2

getVertexAttribIuiv(Index, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Index = integer()
  Pname = enum()

glGetVertexAttribI
See external documentation.

vertexAttribI1i(Index, X) -> ok
Types:
  Index = integer()
  X = integer()
See vertexAttrib1d/2

vertexAttribI2i(Index, X, Y) -> ok
Types:
  Index = integer()
  X = integer()
  Y = integer()
See vertexAttrib1d/2

vertexAttribI3i(Index, X, Y, Z) -> ok
Types:
  Index = integer()
  X = integer()
  Y = integer()
  Z = integer()
See vertexAttrib1d/2

vertexAttribI4i(Index, X, Y, Z, W) -> ok
Types:
  Index = integer()
  X = integer()
Y = integer()
Z = integer()
W = integer()

See `vertexAttrib1d/2`

`vertexAttrib1ui(Index, X) -> ok`
Types:
  Index = integer()
  X = integer()

See `vertexAttrib1d/2`

`vertexAttrib2ui(Index, X, Y) -> ok`
Types:
  Index = integer()
  X = integer()
  Y = integer()

See `vertexAttrib1d/2`

`vertexAttrib3ui(Index, X, Y, Z) -> ok`
Types:
  Index = integer()
  X = integer()
  Y = integer()
  Z = integer()

See `vertexAttrib1d/2`

`vertexAttrib4ui(Index, X, Y, Z, W) -> ok`
Types:
  Index = integer()
  X = integer()
  Y = integer()
  Z = integer()
  W = integer()

See `vertexAttrib1d/2`

`vertexAttrib1iv(Index::integer(), V) -> ok`
Types:
  V = {X::integer()}

Equivalent to `vertexAttrib1ii(Index, X)`.

`vertexAttrib2iv(Index::integer(), V) -> ok`
Types:
  V = {X::integer(), Y::integer()}

860 | Ericsson AB. All Rights Reserved.: wxErlang
Equivalent to `vertexAttribI2i(Index, X, Y).

`vertexAttribI3iv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer(), Y::integer(), Z::integer()\} \]  
Equivalent to `vertexAttribI3i(Index, X, Y, Z).

`vertexAttribI4iv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer(), Y::integer(), Z::integer(), W::integer()\} \]  
Equivalent to `vertexAttribI4i(Index, X, Y, Z, W).

`vertexAttribI1uiv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer()\} \]  
Equivalent to `vertexAttribI1ui(Index, X).

`vertexAttribI2uiv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer(), Y::integer()\} \]  
Equivalent to `vertexAttribI2ui(Index, X, Y).

`vertexAttribI3uiv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer(), Y::integer(), Z::integer()\} \]  
Equivalent to `vertexAttribI3ui(Index, X, Y, Z).

`vertexAttribI4uiv(Index::integer(), V) -> ok`  
Types:  
\[ V = \{X::integer(), Y::integer(), Z::integer(), W::integer()\} \]  
Equivalent to `vertexAttribI4ui(Index, X, Y, Z, W).

`vertexAttribI4bv(Index, V) -> ok`  
Types:  
\[ Index = integer() \]  
\[ V = \{integer(), integer(), integer(), integer()\} \]  
See `vertexAttrib1d/2`

`vertexAttribI4sv(Index, V) -> ok`  
Types:  
\[ Index = integer() \]  
\[ V = \{integer(), integer(), integer(), integer()\} \]  
See `vertexAttrib1d/2`
vertexAttribI4ubv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See vertexAttrib1f/2

vertexAttribI4usv(Index, V) -> ok
Types:
   Index = integer()
   V = {integer(), integer(), integer(), integer()}
See vertexAttrib1f/2

gGetUniformuiv(Program, Location) -> {integer(), integer(), integer(),
integer(), integer(), integer(), integer(), integer(), integer(), integer(),
integer(), integer(), integer(), integer(), integer()}  
Types:
   Program = integer()
   Location = integer()
See getUniformfv/2

bindFragDataLocation(Program, Color, Name) -> ok
Types:
   Program = integer()
   Color = integer()
   Name = string()
Bind a user-defined varying out variable to a fragment shader color number
gl:bindFragDataLocation explicitly specifies the binding of the user-defined varying out variable Name to
fragment shader color number ColorNumber for program Program. If Name was bound previously, its assigned
binding is replaced with ColorNumber. Name must be a null-terminated string. ColorNumber must be less than
?GL_MAX_DRAW_BUFFERS.
See external documentation.

gGetFragDataLocation(Program, Name) -> integer()
Types:
   Program = integer()
   Name = string()
Query the bindings of color numbers to user-defined varying out variables
gl:getFragDataLocation retrieves the assigned color number binding for the user-defined varying out variable
Name for program Program. Program must have previously been linked. Name must be a null-terminated string.
If Name is not the name of an active user-defined varying out fragment shader variable within Program, -1 will
be returned.
See external documentation.
uniform1ui(Location, V0) -> ok
Types:
   Location = integer()
   V0 = integer()
See uniform1f/2

uniform2ui(Location, V0, V1) -> ok
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
See uniform1f/2

uniform3ui(Location, V0, V1, V2) -> ok
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
   V2 = integer()
See uniform1f/2

uniform4ui(Location, V0, V1, V2, V3) -> ok
Types:
   Location = integer()
   V0 = integer()
   V1 = integer()
   V2 = integer()
   V3 = integer()
See uniform1f/2

uniform1uiv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [integer()]
See uniform1f/2

uniform2uiv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [{integer(), integer()}]
See uniform1f/2
uniform3uiv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{integer(), integer(), integer()}]
See uniform1f/2

uniform4uiv(Location, Value) -> ok
Types:
  Location = integer()
  Value = [{integer(), integer(), integer(), integer()}]
See uniform1f/2

texParameterIiv(Target, Pname, Params) -> ok
Types:
  Target = enum()
  Pname = enum()
  Params = tuple()
See texParameterf/3

texParameterIuiv(Target, Pname, Params) -> ok
Types:
  Target = enum()
  Pname = enum()
  Params = tuple()

glTexParameterI
See external documentation.

getTexParameterIiv(Target, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Target = enum()
  Pname = enum()
See getTexParameterfv/2

getTexParameterIuiv(Target, Pname) -> {integer(), integer(), integer(), integer()}
Types:
  Target = enum()
  Pname = enum()

glGetTexParameterI
See external documentation.
clearBufferiv(Buffer, Drawbuffer, Value) -> ok

Types:
  Buffer = enum()
  Drawbuffer = integer()
  Value = tuple()

Clear individual buffers of the currently bound draw framebuffer

gl:clearBuffer* clears the specified buffer to the specified value(s). If Buffer is ?GL_COLOR, a particular draw buffer ?GL_DRAWBUFFER I is specified by passing I as DrawBuffer. In this case, Value points to a four-element vector specifying the R, G, B and A color to clear that draw buffer to. If Buffer is one of ?GL_FRONT, ?GL_BACK, ?GL_LEFT, ?GL_RIGHT, or ?GL_FRONT_AND_BACK, identifying multiple buffers, each selected buffer is cleared to the same value. Clamping and conversion for fixed-point color buffers are performed in the same fashion as gl:clearColor/4.

See external documentation.

clearBufferuiv(Buffer, Drawbuffer, Value) -> ok

Types:
  Buffer = enum()
  Drawbuffer = integer()
  Value = tuple()

See clearBufferiv/3

clearBufferfv(Buffer, Drawbuffer, Value) -> ok

Types:
  Buffer = enum()
  Drawbuffer = integer()
  Value = tuple()

See clearBufferiv/3

clearBufferfi(Buffer, Drawbuffer, Depth, Stencil) -> ok

Types:
  Buffer = enum()
  Drawbuffer = integer()
  Depth = float()
  Stencil = integer()

glClearBufferfi

See external documentation.

getStringi(Name, Index) -> string()

Types:
  Name = enum()
  Index = integer()

See getString/1
drawArraysInstanced(Mode, First, Count, Primcount) -> ok
Types:
   Mode = enum()
   First = integer()
   Count = integer()
   Primcount = integer()

glDrawArraysInstance
See external documentation.

drawElementsInstanced(Mode, Count, Type, Indices, Primcount) -> ok
Types:
   Mode = enum()
   Count = integer()
   Type = enum()
   Indices = offset() | mem()
   Primcount = integer()

glDrawElementsInstance
See external documentation.

texBuffer(Target, Internalformat, Buffer) -> ok
Types:
   Target = enum()
   Internalformat = enum()
   Buffer = integer()

Attach the storage for a buffer object to the active buffer texture

egl:texBuffer attaches the storage for the buffer object named Buffer to the active buffer texture, and specifies the internal format for the texel array found in the attached buffer object. If Buffer is zero, any buffer object attached to the buffer texture is detached and no new buffer object is attached. If Buffer is non-zero, it must be the name of an existing buffer object. Target must be ?GL_TEXTURE_BUFFER. Internalformat specifies the storage format, and must be one of the following sized internal formats:

See external documentation.

primitiveRestartIndex(Index) -> ok
Types:
   Index = integer()

Specify the primitive restart index

egl:primitiveRestartIndex specifies a vertex array element that is treated specially when primitive restarting is enabled. This is known as the primitive restart index.

See external documentation.

getInteger64i_v(Target, Index) -> [integer()]
Types:
   Target = enum()
Index = integer()
See getBooleanv/1

getBufferParameteri64v(Target, Pname) -> [integer()]
Types:
  Target = enum()
  Pname = enum()
glGetBufferParameteri64v
See external documentation.

framebufferTexture(Target, Attachment, Texture, Level) -> ok
Types:
  Target = enum()
  Attachment = enum()
  Texture = integer()
  Level = integer()
Attach a level of a texture object as a logical buffer to the currently bound framebuffer object

gl:framebufferTexture, gl:framebufferTexture1D, gl:framebufferTexture2D, and
gl:framebufferTexture attach a selected mipmap level or image of a texture object as one
of the logical buffers of the framebuffer object currently bound to Target. Target must be
?GL_DRAW_FRAMEBUFFER, ?GL_READ_FRAMEBUFFER, or ?GL_FRAMEBUFFER. ?GL_FRAMEBUFFER is
equivalent to ?GL_DRAW_FRAMEBUFFER.
See external documentation.

vertexAttribDivisor(Index, Divisor) -> ok
Types:
  Index = integer()
  Divisor = integer()
Modify the rate at which generic vertex attributes advance during instanced rendering

gl:vertexAttribDivisor modifies the rate at which generic vertex attributes advance when rendering multiple
instances of primitives in a single draw call. If Divisor is zero, the attribute at slot Index advances once per vertex.
If Divisor is non-zero, the attribute advances once per Divisor instances of the set(s) of vertices being rendered.
An attribute is referred to as instanced if its ?GL_VERTEX_ATTRIB_ARRAY_DIVISOR value is non-zero.
See external documentation.

minSampleShading(Value) -> ok
Types:
  Value = clamp()
Specifies minimum rate at which sample shaing takes place

gl:minSampleShading specifies the rate at which samples are shaded within a covered pixel. Sample-rate
shading is enabled by calling gl:enable/1 with the parameter ?GL_SAMPLE_SHADING. If ?GL_MULTISAMPLE or
?GL_SAMPLE_SHADING is disabled, sample shading has no effect. Otherwise, an implementation must provide at
least as many unique color values for each covered fragment as specified by Value times Samples where Samples
is the value of ?GL_SAMPLES for the current framebuffer. At least 1 sample for each covered fragment is generated.
blendEquationi(Buf, Mode) -> ok
Types:
  Buf = integer()
  Mode = enum()
See blendEquation/1

blendEquationSeparatei(Buf, ModeRGB, ModeAlpha) -> ok
Types:
  Buf = integer()
  ModeRGB = enum()
  ModeAlpha = enum()
See blendEquationSeparate/2

blendFunci(Buf, Src, Dst) -> ok
Types:
  Buf = integer()
  Src = enum()
  Dst = enum()
glBlendFunci
See external documentation.

blendFuncSeparatei(Buf, SrcRGB, DstRGB, SrcAlpha, DstAlpha) -> ok
Types:
  Buf = integer()
  SrcRGB = enum()
  DstRGB = enum()
  SrcAlpha = enum()
  DstAlpha = enum()
See blendFuncSeparate/4

loadTransposeMatrixfARB(M) -> ok
Types:
  M = matrix()
gLLoadTransposeMatrixARB
See external documentation.

loadTransposeMatrixdARB(M) -> ok
Types:
  M = matrix()
gLLoadTransposeMatrixARB
See external documentation.

multTransposeMatrixfARB(M) -> ok
Types:
   M = matrix()
glMultTransposeMatrixARB
See external documentation.

multTransposeMatrixdARB(M) -> ok
Types:
   M = matrix()
glMultTransposeMatrixARB
See external documentation.

weightbvARB(Weights) -> ok
Types:
   Weights = [integer()]
glWeightARB
See external documentation.

weightsvARB(Weights) -> ok
Types:
   Weights = [integer()]
glWeightARB
See external documentation.

weightivARB(Weights) -> ok
Types:
   Weights = [integer()]
glWeightARB
See external documentation.

weightfvARB(Weights) -> ok
Types:
   Weights = [float()]
glWeightARB
See external documentation.

weightdvARB(Weights) -> ok
Types:
   Weights = [float()]
glWeightARB
See external documentation.

weightubvARB(Weights) -> ok
Types:
    \texttt{Weights} = [\texttt{integer()}]
glWeightARB
See external documentation.

weightusvARB(Weights) -> ok
Types:
    \texttt{Weights} = [\texttt{integer()}]
glWeightARB
See external documentation.

weightuivARB(Weights) -> ok
Types:
    \texttt{Weights} = [\texttt{integer()}]
glWeightARB
See external documentation.

vertexBlendARB(Count) -> ok
Types:
    \texttt{Count} = \texttt{integer()}
glVertexBlenARB
See external documentation.

currentPaletteMatrixARB(Index) -> ok
Types:
    \texttt{Index} = \texttt{integer()}
glCurrentPaletteMatrixARB
See external documentation.

matrixIndexubvARB(Indices) -> ok
Types:
    \texttt{Indices} = [\texttt{integer()}]
glMatrixIndexARB
See external documentation.

matrixIndexusvARB(Indices) -> ok
Types:
    \texttt{Indices} = [\texttt{integer()}]
glMatrixIndexARB
See **external** documentation.

`matrixIndexuivARB(Indices) -> ok`
Types:

- `Indices = [integer()]`
`glMatrixIndexARB`

See **external** documentation.

`programStringARB(Target, Format, String) -> ok`
Types:

- `Target = enum()`  
- `Format = enum()`  
- `String = string()`
`glProgramStringARB`

See **external** documentation.

`bindProgramARB(Target, Program) -> ok`
Types:

- `Target = enum()`  
- `Program = integer()`
`glBindProgramARB`

See **external** documentation.

`deleteProgramsARB(Programs) -> ok`
Types:

- `Programs = [integer()]`
`glDeleteProgramsARB`

See **external** documentation.

`genProgramsARB(N) -> [integer()]`
Types:

- `N = integer()`
`glGenProgramsARB`

See **external** documentation.

`programEnvParameter4dARB(Target, Index, X, Y, Z, W) -> ok`
Types:

- `Target = enum()`  
- `Index = integer()`  
- `X = float()`  
- `Y = float()`  
- `Z = float()`
\[
W = \text{float()}
\]

\text{glProgramEnvParameterARB}

See \texttt{external} documentation.

\text{programEnvParameter4dvARB(Target, Index, Params) -> ok}

Types:

\begin{itemize}
\item \texttt{Target = \text{enum()}}
\item \texttt{Index = \text{integer()}}
\item \texttt{Params = \{\text{float()}, \text{float()}, \text{float()}, \text{float()\}}}
\end{itemize}

\text{glProgramEnvParameterARB}

See \texttt{external} documentation.

\text{programEnvParameter4fARB(Target, Index, X, Y, Z, W) -> ok}

Types:

\begin{itemize}
\item \texttt{Target = \text{enum()}}
\item \texttt{Index = \text{integer()}}
\item \texttt{X = \text{float()}}
\item \texttt{Y = \text{float()}}
\item \texttt{Z = \text{float()}}
\item \texttt{W = \text{float()}}
\end{itemize}

\text{glProgramEnvParameterARB}

See \texttt{external} documentation.

\text{programEnvParameter4fvARB(Target, Index, Params) -> ok}

Types:

\begin{itemize}
\item \texttt{Target = \text{enum()}}
\item \texttt{Index = \text{integer()}}
\item \texttt{Params = \{\text{float()}, \text{float()}, \text{float()}, \text{float()\}}}
\end{itemize}

\text{glProgramEnvParameterARB}

See \texttt{external} documentation.

\text{programLocalParameter4dARB(Target, Index, X, Y, Z, W) -> ok}

Types:

\begin{itemize}
\item \texttt{Target = \text{enum()}}
\item \texttt{Index = \text{integer()}}
\item \texttt{X = \text{float()}}
\item \texttt{Y = \text{float()}}
\item \texttt{Z = \text{float()}}
\item \texttt{W = \text{float()}}
\end{itemize}

\text{glProgramLocalParameterARB}

See \texttt{external} documentation.
programLocalParameter4dvARB(Target, Index, Params) -> ok
Types:
  Target = enum()
  Index = integer()
  Params = {float(), float(), float(), float()}

See external documentation.

programLocalParameter4fARB(Target, Index, X, Y, Z, W) -> ok
Types:
  Target = enum()
  Index = integer()
  X = float()
  Y = float()
  Z = float()
  W = float()

See external documentation.

programLocalParameter4fvARB(Target, Index, Params) -> ok
Types:
  Target = enum()
  Index = integer()
  Params = {float(), float(), float(), float()}

See external documentation.

getProgramEnvParameterdvARB(Target, Index) -> {float(), float(), float(), float()}
Types:
  Target = enum()
  Index = integer()

See external documentation.

getProgramEnvParameterfvARB(Target, Index) -> {float(), float(), float(), float()}
Types:
  Target = enum()
  Index = integer()

See external documentation.
getProgramLocalParameterdvARB(Target, Index) -> {float(), float(), float(), float()}
Types:
   Target = enum()
   Index = integer()
getProgramLocalParameterARB
See external documentation.

getProgramLocalParameterfvARB(Target, Index) -> {float(), float(), float(), float()}
Types:
   Target = enum()
   Index = integer()
getProgramLocalParameterARB
See external documentation.

getProgramStringARB(Target, Pname, String) -> ok
Types:
   Target = enum()
   Pname = enum()
   String = mem()
getProgramStringARB
See external documentation.

gGetBufferParameterivARB(Target, Pname) -> [integer()]
Types:
   Target = enum()
   Pname = enum()
getBufferParameterARB
See external documentation.

deleteObjectARB(Obj) -> ok
Types:
   Obj = integer()
deleteObjectARB
See external documentation.

gGetHandleARB(Pname) -> integer()
Types:
   Pname = enum()
gGetHandleARB
See external documentation.

874 | Ericsson AB. All Rights Reserved.: wxErlang
detachObjectARB(ContainerObj, AttachedObj) -> ok
Types:

  ContainerObj = integer()
  AttachedObj = integer()

gIDetachObjectARB
See external documentation.

createShaderObjectARB(ShaderType) -> integer()
Types:

  ShaderType = enum()

gCreateShaderObjectARB
See external documentation.

shaderSourceARB(ShaderObj, String) -> ok
Types:

  ShaderObj = integer()
  String = iolist()

gShaderSourceARB
See external documentation.

compileShaderARB(ShaderObj) -> ok
Types:

  ShaderObj = integer()

gCompileShaderARB
See external documentation.

createProgramObjectARB() -> integer()

gCreateProgramObjectARB
See external documentation.

attachObjectARB(ContainerObj, Obj) -> ok
Types:

  ContainerObj = integer()
  Obj = integer()

gAttachObjectARB
See external documentation.

linkProgramARB(ProgramObj) -> ok
Types:

  ProgramObj = integer()

gLinkProgramARB
See **external** documentation.

`useProgramObjectARB(ProgramObj) -> ok`

Types:

- `ProgramObj = integer()`

`glUseProgramObjectARB`

See **external** documentation.

`validateProgramARB(ProgramObj) -> ok`

Types:

- `ProgramObj = integer()`

`glValidateProgramARB`

See **external** documentation.

`getObjectParameterfvARB(Obj, Pname) -> float()`

Types:

- `Obj = integer()`
- `Pname = enum()`

`glGetObjectParameterARB`

See **external** documentation.

`getObjectParameterivARB(Obj, Pname) -> integer()`

Types:

- `Obj = integer()`
- `Pname = enum()`

`glGetObjectParameterARB`

See **external** documentation.

`getInfoLogARB(Obj, MaxLength) -> string()`

Types:

- `Obj = integer()`
- `MaxLength = integer()`

`glGetInfoLogARB`

See **external** documentation.

`getAttachedObjectsARB(ContainerObj, MaxCount) -> [integer()]`

Types:

- `ContainerObj = integer()`
- `MaxCount = integer()`

`glGetAttachedObjectsARB`

See **external** documentation.
getUniformLocationARB(ProgramObj, Name) -> integer()
Types:
  ProgramObj = integer()
  Name = string()

See external documentation.

getActiveUniformARB(ProgramObj, Index, MaxLength) -> {Size::integer(), Type::enum(), Name::string()}
Types:
  ProgramObj = integer()
  Index = integer()
  MaxLength = integer()

See external documentation.

getUniformfvARB(ProgramObj, Location) -> matrix()
Types:
  ProgramObj = integer()
  Location = integer()

See external documentation.

getUniformivARB(ProgramObj, Location) -> {integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer(), integer()}
Types:
  ProgramObj = integer()
  Location = integer()

See external documentation.

getShaderSourceARB(Obj, MaxLength) -> string()
Types:
  Obj = integer()
  MaxLength = integer()

See external documentation.

bindAttribLocationARB(ProgramObj, Index, Name) -> ok
Types:
  ProgramObj = integer()
  Index = integer()
Name = string()
glBindAttribLocationARB
See external documentation.

getActiveAttribARB(ProgramObj, Index, MaxLength) -> {Size::integer(),
Type::enum(), Name::string()}
Types:
    ProgramObj = integer()
    Index = integer()
    MaxLength = integer()
gGetActiveAttribARB
See external documentation.

getAttribLocationARB(ProgramObj, Name) -> integer()
Types:
    ProgramObj = integer()
    Name = string()
gGetAttribLocationARB
See external documentation.

isRenderbuffer(Renderbuffer) -> 0 | 1
Types:
    Renderbuffer = integer()
Determine if a name corresponds to a renderbuffer object

    gl:isRenderbuffer returns ?GL_TRUE if Renderbuffer is currently the name of a renderbuffer object. If
    Renderbuffer is zero, or if Renderbuffer is not the name of a renderbuffer object, or if an error occurs,
    gl:isRenderbuffer returns ?GL_FALSE. If Renderbuffer is a name returned by gl:genRenderbuffers/1, by
    that has not yet been bound through a call to gl:bindRenderbuffer/2 or gl:framebufferRenderbuffer/4, then the name
    is not a renderbuffer object and gl:isRenderbuffer returns ?GL_FALSE.
See external documentation.

bindRenderbuffer(Target, Renderbuffer) -> ok
Types:
    Target = enum()
    Renderbuffer = integer()
Bind a renderbuffer to a renderbuffer target

    gl:bindRenderbuffer binds the renderbuffer object with name Renderbuffer to the renderbuffer target
    specified by Target. Target must be ?GL_RENDERBUFFER. Renderbuffer is the name of a renderbuffer
    object previously returned from a call to gl:genRenderbuffers/1, or zero to break the existing binding of a renderbuffer
    object to Target.
See external documentation.
deleteRenderbuffers(Renderbuffers) -> ok
Types:
  Renderbuffers = [integer()]
Delete renderbuffer objects

`gl:deleteRenderbuffers` deletes the N renderbuffer objects whose names are stored in the array addressed by `Renderbuffers`. The name zero is reserved by the GL and is silently ignored, should it occur in `Renderbuffers`, as are other unused names. Once a renderbuffer object is deleted, its name is again unused and it has no contents. If a renderbuffer that is currently bound to the target `?GL_RENDERBUFFER` is deleted, it is as though `gl:bindRenderbuffer/2` had been executed with a `Target` of `?GL_RENDERBUFFER` and a `Name` of zero.

See `external` documentation.

genRenderbuffers(N) -> [integer()]
Types:
  N = integer()
Generate renderbuffer object names

`gl:genRenderbuffers` returns N renderbuffer object names in `Renderbuffers`. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to `gl:genRenderbuffers`.

See `external` documentation.

renderbufferStorage(Target, Internalformat, Width, Height) -> ok
Types:
  Target = enum()
  Internalformat = enum()
  Width = integer()
  Height = integer()
Establish data storage, format and dimensions of a renderbuffer object's image

`gl:renderbufferStorage` is equivalent to calling `gl:renderbufferStorageMultisample/5` with the `Samples` set to zero.

See `external` documentation.

getRenderbufferParameteriv(Target, Pname) -> integer()
Types:
  Target = enum()
  Pname = enum()
Retrieve information about a bound renderbuffer object

See external documentation.

isFramebuffer(Framebuffer) -> 0 | 1
Types:
    Framebuffer = integer()
Determine if a name corresponds to a framebuffer object

\texttt{gl:isFramebuffer} returns \texttt{?GL_TRUE} if Framebuffer is currently the name of a framebuffer object. If Framebuffer is zero, or if \texttt{?Framebuffer} is not the name of a framebuffer object, or if an error occurs, \texttt{gl:isFramebuffer} returns \texttt{?GL_FALSE}. If Framebuffer is a name returned by \texttt{gl:genFramebuffers/1}, by that has not yet been bound through a call to \texttt{gl:bindFramebuffer/2}, then the name is not a framebuffer object and \texttt{gl:isFramebuffer} returns \texttt{?GL_FALSE}.

See external documentation.

bindFramebuffer(Target, Framebuffer) -> ok
Types:
    Target = enum()
    Framebuffer = integer()
Bind a framebuffer to a framebuffer target

\texttt{gl:bindFramebuffer} binds the framebuffer object with name Framebuffer to the framebuffer target specified by Target. Target must be either \texttt{?GL_DRAW_FRAMEBUFFER}, \texttt{?GL_READ_FRAMEBUFFER} or \texttt{?GL_FRAMEBUFFER}. If a framebuffer object is bound to \texttt{?GL_DRAW_FRAMEBUFFER} or \texttt{?GL_READ_FRAMEBUFFER}, it becomes the target for rendering or readback operations, respectively, until it is deleted or another framebuffer is bound to the corresponding bind point. Calling \texttt{gl:bindFramebuffer} with Target set to \texttt{?GL_FRAMEBUFFER} binds Framebuffer to both the read and draw framebuffer targets. Framebuffer is the name of a framebuffer object previously returned from a call to \texttt{gl:genFramebuffers/1}, or zero to break the existing binding of a framebuffer object to Target.

See external documentation.

deleteFramebuffers(Framebuffers) -> ok
Types:
    Framebuffers = [integer()]
Delete framebuffer objects

\texttt{gl:deleteFramebuffers} deletes the \texttt{N} framebuffer objects whose names are stored in the array addressed by Framebuffers. The name zero is reserved by the GL and is silently ignored, should it occur in Framebuffers, as are other unused names. Once a framebuffer object is deleted, its name is again unused and it has no attachments. If a framebuffer that is currently bound to one or more of the targets \texttt{?GL_DRAW_FRAMEBUFFER} or \texttt{?GL_READ_FRAMEBUFFER} is deleted, it is as though \texttt{gl:bindFramebuffer/2} had been executed with the corresponding Target and Framebuffer zero.

See external documentation.

genFramebuffers(N) -> [integer()]
Types:
    N = integer()
Generate framebuffer object names
gl:genFramebuffers returns N framebuffer object names in Ids. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to gl:genFramebuffers.

See external documentation.

checkFramebufferStatus(Target) -> enum()
Types:
  Target = enum()
Check the completeness status of a framebuffer

gl:checkFramebufferStatus queries the completeness status of the framebuffer object currently bound to Target. Target must be ?GL_DRAW_FRAMEBUFFER, ?GL_READ_FRAMEBUFFER or ?GL_FRAMEBUFFER.

?GL_FRAMEBUFFER is equivalent to ?GL_DRAW_FRAMEBUFFER.

See external documentation.

framebufferTexture1D(Target, Attachment, Textarget, Texture, Level) -> ok
Types:
  Target = enum()
  Attachment = enum()
  Textarget = enum()
  Texture = integer()
  Level = integer()

See framebufferTexture/4

framebufferTexture2D(Target, Attachment, Textarget, Texture, Level) -> ok
Types:
  Target = enum()
  Attachment = enum()
  Textarget = enum()
  Texture = integer()
  Level = integer()

See framebufferTexture/4

framebufferTexture3D(Target, Attachment, Textarget, Texture, Level, Zoffset) -> ok
Types:
  Target = enum()
  Attachment = enum()
  Textarget = enum()
  Texture = integer()
  Level = integer()
  Zoffset = integer()

See framebufferTexture/4
framebufferRenderbuffer(Target, Attachment, Renderbuffertarget, Renderbuffer) -> ok
Types:
  Target = enum()
  Attachment = enum()
  Renderbuffertarget = enum()
  Renderbuffer = integer()

Attach a renderbuffer as a logical buffer to the currently bound framebuffer object

`gl:FramebufferRenderbuffer` attaches a renderbuffer as one of the logical buffers of the currently bound framebuffer object. Renderbuffer is the name of the renderbuffer object to attach and must be either zero, or the name of an existing renderbuffer object of type `Renderbuffertarget`. If `Renderbuffer` is not zero and if `gl:FramebufferRenderbuffer` is successful, then the renderbuffer name `Renderbuffer` will be used as the logical buffer identified by `Attachment` of the framebuffer currently bound to `Target`.

See external documentation.

getFramebufferAttachmentParameteriv(Target, Attachment, Pname) -> integer()
Types:
  Target = enum()
  Attachment = enum()
  Pname = enum()

Retrieve information about attachments of a bound framebuffer object

`gl:GetFramebufferAttachmentParameteriv` returns information about attachments of a bound framebuffer object. `Target` specifies the framebuffer binding point and must be `GL_DRAW_FRAMEBUFFER`, `GL_READ_FRAMEBUFFER` or `GL_FRAMEBUFFER`. `GL_FRAMEBUFFER` is equivalent to `GL_DRAW_FRAMEBUFFER`.

See external documentation.

generateMipmap(Target) -> ok
Types:
  Target = enum()

Generate mipmaps for a specified texture target

`gl:GenerateMipmap` generates mipmaps for the texture attached to `Target` of the active texture unit. For cube map textures, a `GL_INVALID_OPERATION` error is generated if the texture attached to `Target` is not cube complete.

See external documentation.

blitFramebuffer(SrcX0, SrcY0, SrcX1, SrcY1, DstX0, DstY0, DstX1, DstY1, Mask, Filter) -> ok
Types:
  SrcX0 = integer()
  SrcY0 = integer()
  SrcX1 = integer()
  SrcY1 = integer()
  DstX0 = integer()
Copy a block of pixels from the read framebuffer to the draw framebuffer

`gl:blitFramebuffer` transfers a rectangle of pixel values from one region of the read framebuffer to another region in the draw framebuffer. `Mask` is the bitwise OR of a number of values indicating which buffers are to be copied. The values are `?GL_COLOR_BUFFER_BIT`, `?GL_DEPTH_BUFFER_BIT`, and `?GL_STENCIL_BUFFER_BIT`. The pixels corresponding to these buffers are copied from the source rectangle bounded by the locations `(SrcX0; SrcY0)` and `(SrcX1; SrcY1)` to the destination rectangle bounded by the locations `(DstX0; DstY0)` and `(DstX1; DstY1)`. The lower bounds of the rectangle are inclusive, while the upper bounds are exclusive.

See external documentation.

```erlang
renderbufferStorageMultisample(Target, Samples, Internalformat, Width, Height) -> ok
```

Types:

```
Target = enum()
Samples = integer()
Internalformat = enum()
Width = integer()
Height = integer()
```

Establish data storage, format, dimensions and sample count of a renderbuffer object's image

`gl:renderbufferStorageMultisample` establishes the data storage, format, dimensions and number of samples of a renderbuffer object's image.

See external documentation.

```erlang
framebufferTextureLayer(Target, Attachment, Texture, Level, Layer) -> ok
```

Types:

```
Target = enum()
Attachment = enum()
Texture = integer()
Level = integer()
Layer = integer()
```

See framebufferTexture/4

```erlang
framebufferTextureFaceARB(Target, Attachment, Texture, Level, Face) -> ok
```

Types:

```
Target = enum()
Attachment = enum()
Texture = integer()
Level = integer()
Face = enum()
```
flushMappedBufferRange(Target, Offset, Length) -> ok
Types:
    Target = enum()
    Offset = integer()
    Length = integer()

Indicate modifications to a range of a mapped buffer

`gl:flushMappedBufferRange` indicates that modifications have been made to a range of a mapped buffer. The buffer must previously have been mapped with the `?GL_MAP_FLUSH_EXPLICIT` flag. Offset and Length indicate the modified subrange of the mapping, in basic units. The specified subrange to flush is relative to the start of the currently mapped range of the buffer. `gl:flushMappedBufferRange` may be called multiple times to indicate distinct subranges of the mapping which require flushing.

See external documentation.

bindVertexArray(Array) -> ok
Types:
    Array = integer()

Bind a vertex array object

`gl:bindVertexArray` binds the vertex array object with name `Array`. `Array` is the name of a vertex array object previously returned from a call to `gl:genVertexArrays/1`, or zero to break the existing vertex array object binding.

See external documentation.

deleteVertexArrays(Arrays) -> ok
Types:
    Arrays = [integer()]

Delete vertex array objects

`gl:deleteVertexArrays` deletes N vertex array objects whose names are stored in the array addressed by `Arrays`. Once a vertex array object is deleted it has no contents and its name is again unused. If a vertex array object that is currently bound is deleted, the binding for that object reverts to zero and the default vertex array becomes current. Unused names in `Arrays` are silently ignored, as is the value zero.

See external documentation.

genVertexArrays(N) -> [integer()]
Types:
    N = integer()

Generate vertex array object names

`gl:genVertexArrays` returns N vertex array object names in `Arrays`. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to `gl:genVertexArrays`.

See external documentation.
isVertexArray(Array) -> 0 | 1
Types:
   Array = integer()
Determine if a name corresponds to a vertex array object

gl:isVertexArray returns ?GL_TRUE if Array is currently the name of a renderbuffer object. If Renderbuffer is zero, or if Array is not the name of a renderbuffer object, or if an error occurs, gl:isVertexArray returns ?GL_FALSE. If Array is a name returned by gl:genVertexArrays/1, by that has not yet been bound through a call to gl:bindVertexArray/1, then the name is not a vertex array object and gl:isVertexArray returns ?GL_FALSE.
See external documentation.

getUniformIndices(Program, UniformNames) -> [integer()]
Types:
   Program = integer()
   UniformNames = iolist()
Retrieve the index of a named uniform block

gl:getUniformIndices retrieves the indices of a number of uniforms within Program.
See external documentation.

getActiveUniformsiv(Program, UniformIndices, Pname) -> [integer()]
Types:
   Program = integer()
   UniformIndices = [integer()]
   Pname = enum()

glGetActiveUniforms
See external documentation.

getActiveUniformName(Program, UniformIndex, BufSize) -> string()
Types:
   Program = integer()
   UniformIndex = integer()
   BufSize = integer()
Query the name of an active uniform

gl:getActiveUniformName returns the name of the active uniform at UniformIndex within Program. If UniformName is not NULL, up to BufSize characters (including a nul-terminator) will be written into the array whose address is specified by UniformName. If Length is not NULL, the number of characters that were (or would have been) written into UniformName (not including the nul-terminator) will be placed in the variable whose address is specified in Length. If Length is NULL, no length is returned. The length of the longest uniform name in Program is given by the value of ?GL_ACTIVE_UNIFORM_MAX_LENGTH, which can be queried with glGetProgramiv/2.
See external documentation.
getUniformBlockIndex(Program, UniformBlockName) -> integer()

Types:
  Program = integer()
  UniformBlockName = string()

Retrieve the index of a named uniform block

gl:getUniformBlockIndex retrieves the index of a uniform block within Program.
See external documentation.

getActiveUniformBlockiv(Program, UniformBlockIndex, Pname, Params) -> ok

Types:
  Program = integer()
  UniformBlockIndex = integer()
  Pname = enum()
  Params = mem()

Query information about an active uniform block

gl:getActiveUniformBlockiv retrieves information about an active uniform block within Program.
See external documentation.

getActiveUniformBlockName(Program, UniformBlockIndex, BufSize) -> string()

Types:
  Program = integer()
  UniformBlockIndex = integer()
  BufSize = integer()

Retrieve the name of an active uniform block

gl:getActiveUniformBlockName retrieves the name of the active uniform block at UniformBlockIndex within Program.
See external documentation.

uniformBlockBinding(Program, UniformBlockIndex, UniformBlockBinding) -> ok

Types:
  Program = integer()
  UniformBlockIndex = integer()
  UniformBlockBinding = integer()

Assign a binding point to an active uniform block

Binding points for active uniform blocks are assigned using gl:uniformBlockBinding. Each of a program's active uniform blocks has a corresponding uniform buffer binding point. Program is the name of a program object for which the command gl:linkProgram/1 has been issued in the past.
See external documentation.

copyBufferSubData(ReadTarget, WriteTarget, ReadOffset, WriteOffset, Size) -> ok

Types:
Copy part of the data store of a buffer object to the data store of another buffer object

gl:copyBufferSubData copies part of the data store attached to ReadTarget to the data store attached to WriteTarget. The number of basic machine units indicated by Size is copied from the source, at offset ReadOffset to the destination at WriteOffset, also in basic machine units.

See external documentation.

drawElementsBaseVertex(Mode, Count, Type, Indices, Basevertex) -> ok

Types:
- Mode = enum()
- Count = integer()
- Type = enum()
- Indices = offset() | mem()
- Basevertex = integer()

Render primitives from array data with a per-element offset

gl:drawElementsBaseVertex behaves identically to gl:drawElements/4 except that the \( i \)th element transferred by the corresponding draw call will be taken from element Indices \([i]+\) Basevertex of each enabled array. If the resulting value is larger than the maximum value representable by Type, it is as if the calculation were upconverted to 32-bit unsigned integers (with wrapping on overflow conditions). The operation is undefined if the sum would be negative.

See external documentation.

drawRangeElementsBaseVertex(Mode, Start, End, Count, Type, Indices, Basevertex) -> ok

Types:
- Mode = enum()
- Start = integer()
- End = integer()
- Count = integer()
- Type = enum()
- Indices = offset() | mem()
- Basevertex = integer()

Render primitives from array data with a per-element offset

gl:drawRangeElementsBaseVertex is a restricted form of gl:drawElementsBaseVertex/5. Mode, Start, End, Count and Basevertex match the corresponding arguments to gl:drawElementsBaseVertex/5, with the additional constraint that all values in the array Indices must lie between Start and End, inclusive, prior to adding Basevertex. Index values lying outside the range \([Start, End]\) are treated in the same way as gl:drawElementsBaseVertex/5. The \( i \)th element transferred by the corresponding draw call will be taken from element Indices \([i]+Basevertex\) of each enabled array. If the resulting value is larger than the maximum value...
representable by Type, it is as if the calculation were upconverted to 32-bit unsigned integers (with wrapping on overflow conditions). The operation is undefined if the sum would be negative.

See external documentation.

drawElementsInstancedBaseVertex(Mode, Count, Type, Indices, Primcount, Basevertex) -> ok
Types:
   Mode = enum()
   Count = integer()
   Type = enum()
   Indices = offset() | mem()
   Primcount = integer()
   Basevertex = integer()

Render multiple instances of a set of primitives from array data with a per-element offset

$gl$::drawElementsInstancedBaseVertex behaves identically to $gl$::drawElementsInstanced/5 except that the ith element transferred by the corresponding draw call will be taken from element Indices [i] + Basevertex of each enabled array. If the resulting value is larger than the maximum value representable by Type, it is as if the calculation were upconverted to 32-bit unsigned integers (with wrapping on overflow conditions). The operation is undefined if the sum would be negative.

See external documentation.

provokingVertex(Mode) -> ok
Types:
   Mode = enum()

Specify the vertex to be used as the source of data for flat shaded varyings

Flatshading a vertex shader varying output means to assign all vertices of the primitive the same value for that output. The vertex from which these values is derived is known as the provoking vertex and $gl$::provokingVertex specifies which vertex is to be used as the source of data for flat shaded varyings.

See external documentation.

fenceSync(Condition, Flags) -> integer()
Types:
   Condition = enum()
   Flags = integer()

Create a new sync object and insert it into the GL command stream

$gl$::fenceSync creates a new fence sync object, inserts a fence command into the GL command stream and associates it with that sync object, and returns a non-zero name corresponding to the sync object.

See external documentation.

isSync(Sync) -> 0 | 1
Types:
   Sync = integer()

Determine if a name corresponds to a sync object
gl:isSync returns ?GL_TRUE if Sync is currently the name of a sync object. If Sync is not the name of a sync object, or if an error occurs, gl:isSync returns ?GL_FALSE. Note that zero is not the name of a sync object.

See external documentation.

deleteSync(Sync) -> ok
Types:
  Sync = integer()
Delete a sync object

gl:deleteSync deletes the sync object specified by Sync. If the fence command corresponding to the specified sync object has completed, or if no gl:waitSync/3 or gl:clientWaitSync/3 commands are blocking on Sync, the object is deleted immediately. Otherwise, Sync is flagged for deletion and will be deleted when it is no longer associated with any fence command and is no longer blocking any gl:waitSync/3 or gl:clientWaitSync/3 command. In either case, after gl:deleteSync returns, the name Sync is invalid and can no longer be used to refer to the sync object.

See external documentation.

clientWaitSync(Sync, Flags, Timeout) -> enum()
Types:
  Sync = integer()
  Flags = integer()
  Timeout = integer()
Block and wait for a sync object to become signaled

gl:clientWaitSync causes the client to block and wait for the sync object specified by Sync to become signaled. If Sync is signaled when gl:clientWaitSync is called, gl:clientWaitSync returns immediately, otherwise it will block and wait for up to Timeout nanoseconds for Sync to become signaled.

See external documentation.

waitSync(Sync, Flags, Timeout) -> ok
Types:
  Sync = integer()
  Flags = integer()
  Timeout = integer()
Instruct the GL server to block until the specified sync object becomes signaled

gl:waitSync causes the GL server to block and wait until Sync becomes signaled. Sync is the name of an existing sync object upon which to wait. Flags and Timeout are currently not used and must be set to zero and the special value ?GL_TIMEOUT_IGNORED, respectively

Flags and Timeout are placeholders for anticipated future extensions of sync object capabilities. They must have these reserved values in order that existing code calling gl:waitSync operate properly in the presence of such extensions.

See external documentation.

getInteger64v(Pname) -> [integer()]
Types:
  Pname = enum()
See `getBooleanv/1`

```erlang
getSynciv(Sync, Pname, BufSize) -> [integer()]
```
Types:
- `Sync` = `integer()`
- `Pname` = `enum()`
- `BufSize` = `integer()`

Query the properties of a sync object

`gl:getSynciv` retrieves properties of a sync object. `Sync` specifies the name of the sync object whose properties to retrieve.

See external documentation.

```erlang
texImage2DMultisample(Target, Samples, Internalformat, Width, Height, Fixedsamplelocations) -> ok
```
Types:
- `Target` = `enum()`
- `Samples` = `integer()`
- `Internalformat` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Fixedsamplelocations` = 0 | 1

Establish the data storage, format, dimensions, and number of samples of a multisample texture's image

`gl:texImage2DMultisample` establishes the data storage, format, dimensions and number of samples of a multisample texture's image.

See external documentation.

```erlang
texImage3DMultisample(Target, Samples, Internalformat, Width, Height, Depth, Fixedsamplelocations) -> ok
```
Types:
- `Target` = `enum()`
- `Samples` = `integer()`
- `Internalformat` = `integer()`
- `Width` = `integer()`
- `Height` = `integer()`
- `Depth` = `integer()`
- `Fixedsamplelocations` = 0 | 1

Establish the data storage, format, dimensions, and number of samples of a multisample texture's image

`gl:texImage3DMultisample` establishes the data storage, format, dimensions and number of samples of a multisample texture's image.

See external documentation.

```erlang
getMultisamplefv(Pname, Index) -> {float(), float()}
```
Types:
gl:getMultisamplefv queries the location of a given sample. Pname specifies the sample parameter to retrieve and must be `GL_SAMPLE_POSITION`. Index corresponds to the sample for which the location should be returned. The sample location is returned as two floating-point values in `Val[0]` and `Val[1]`, each between 0 and 1, corresponding to the X and Y locations respectively in the GL pixel space of that sample. (0.5, 0.5) this corresponds to the pixel center. Index must be between zero and the value of `GL_SAMPLES - 1`.

See external documentation.

```erlang
sampleMaski(Index, Mask) -> ok
```

Types:
- `Index = integer()`
- `Mask = integer()`

Set the value of a sub-word of the sample mask

`gl:sampleMaski` sets one 32-bit sub-word of the multi-word sample mask, `GL_SAMPLE_MASK_VALUE`.

See external documentation.

```erlang
namedStringARB(Type, Name, String) -> ok
```

Types:
- `Type = enum()`
- `Name = string()`
- `String = string()`

`glNamedStringARB` See external documentation.

```erlang
deleteNamedStringARB(Name) -> ok
```

Types:
- `Name = string()`

`glDeleteNamedStringARB` See external documentation.

```erlang
compileShaderIncludeARB(Shader, Path) -> ok
```

Types:
- `Shader = integer()`
- `Path = iolist()`

`glCompileShaderIncludeARB` See external documentation.

```erlang
isNamedStringARB(Name) -> 0 | 1
```

Types:
- `Name = string()`
**glIsNamedStringARB**

See external documentation.

getNamedStringARB(Name, BufSize) -> string()

Types:

- **Name** = string()
- **BufSize** = integer()

glGetNamedStringARB

See external documentation.

getNamedStringivARB(Name, Pname) -> integer()

Types:

- **Name** = string()
- **Pname** = enum()

glGetNamedStringARB

See external documentation.

bindFragDataLocationIndexed(Program, ColorNumber, Index, Name) -> ok

Types:

- **Program** = integer()
- **ColorNumber** = integer()
- **Index** = integer()
- **Name** = string()

glBindFragDataLocationIndexed

See external documentation.

getFragDataIndex(Program, Name) -> integer()

Types:

- **Program** = integer()
- **Name** = string()

Query the bindings of color indices to user-defined varying out variables

**gl:getFragDataIndex** returns the index of the fragment color to which the variable Name was bound when the program object Program was last linked. If Name is not a varying out variable of Program, or if an error occurs, -1 will be returned.

See external documentation.

genSamplers(Count) -> [integer()]

Types:

- **Count** = integer()

Generate sampler object names
gl:genSamplers returns \( N \) sampler object names in `Samplers`. There is no guarantee that the names form a contiguous set of integers; however, it is guaranteed that none of the returned names was in use immediately before the call to `gl:genSamplers`.

See external documentation.

`deleteSamplers(Samplers)` -> `ok`

Types:

\[
\text{Samplers} = \{\text{integer}\}\]

Delete named sampler objects

`gl:deleteSamplers` deletes \( N \) sampler objects named by the elements of the array `Ids`. After a sampler object is deleted, its name is again unused. If a sampler object that is currently bound to a sampler unit is deleted, it is as though `gl:bindSampler/2` is called with unit set to the unit the sampler is bound to and sampler zero. Unused names in samplers are silently ignored, as is the reserved name zero.

See external documentation.

`isSampler(Sampler)` -> 0 | 1

Types:

\[
\text{Sampler} = \text{integer}\]

Determine if a name corresponds to a sampler object

`gl:isSampler` returns `GL_TRUE` if `Id` is currently the name of a sampler object. If `Id` is zero, or is a non-zero value that is not currently the name of a sampler object, or if an error occurs, `gl:isSampler` returns `GL_FALSE`.

See external documentation.

`bindSampler(Unit, Sampler)` -> `ok`

Types:

\[
\text{Unit} = \text{integer}\]
\[
\text{Sampler} = \text{integer}\]

Bind a named sampler to a texturing target

`gl:bindSampler` binds `Sampler` to the texture unit at index `Unit`. `Sampler` must be zero or the name of a sampler object previously returned from a call to `gl:genSamplers/I`. `Unit` must be less than the value of `GL_MAX_COMBINED_TEXTURE_IMAGE_UNITS`.

See external documentation.

`samplerParameteri(Sampler, Pname, Param)` -> `ok`

Types:

\[
\text{Sampler} = \text{integer}\]
\[
\text{Pname} = \text{enum}\]
\[
\text{Param} = \text{integer}\]

Set sampler parameters

`gl:samplerParameter` assigns the value or values in `Params` to the sampler parameter specified as `Pname`. `Sampler` specifies the sampler object to be modified, and must be the name of a sampler object previously returned from a call to `gl:genSamplers/I`. The following symbols are accepted in `Pname`:

See external documentation.
samplerParameteriv(Sampler, Pname, Param) -> ok
Types:
   Sampler = integer()
Pname = enum()
   Param = [integer()]
See samplerParameteri/3

samplerParameterf(Sampler, Pname, Param) -> ok
Types:
   Sampler = integer()
Pname = enum()
   Param = float()
See samplerParameteri/3

samplerParameterfv(Sampler, Pname, Param) -> ok
Types:
   Sampler = integer()
Pname = enum()
   Param = [float()]
See samplerParameteri/3

samplerParameterIiv(Sampler, Pname, Param) -> ok
Types:
   Sampler = integer()
Pname = enum()
   Param = [integer()]
See samplerParameteri/3

samplerParameterIuiv(Sampler, Pname, Param) -> ok
Types:
   Sampler = integer()
Pname = enum()
   Param = [integer()]

glSamplerParameterI
See external documentation.

glGetSamplerParameteriv(Sampler, Pname) -> [integer()]
Types:
   Sampler = integer()
Pname = enum()
Return sampler parameter values
gl:getSamplerParameter returns in Params the value or values of the sampler parameter specified as Pname. Sampler defines the target sampler, and must be the name of an existing sampler object, returned from a previous call to gl:genSamplers/1. Pname accepts the same symbols as gl:samplerParameteri/3, with the same interpretations:

See external documentation.

getSamplerParameterIiv(Sampler, Pname) -> [integer()]
Types:
  Sampler = integer()
  Pname = enum()
See getSamplerParameteriv/2

getsamplerParameterfv(Sampler, Pname) -> [float()]
Types:
  Sampler = integer()
  Pname = enum()
See getSamplerParameteriv/2

getsamplerParameterIuiv(Sampler, Pname) -> [integer()]
Types:
  Sampler = integer()
  Pname = enum()
glGetSamplerParameterI
See external documentation.

queryCounter(Id, Target) -> ok
Types:
  Id = integer()
  Target = enum()
Record the GL time into a query object after all previous commands have reached the GL server but have not yet necessarily executed.

gl:queryCounter causes the GL to record the current time into the query object named Id. Target must be ?GL_TIMESTAMP. The time is recorded after all previous commands on the GL client and server state and the framebuffer have been fully realized. When the time is recorded, the query result for that object is marked available. gl:queryCounter timer queries can be used within a gl:beginQuery2 / gl:beginQuery2 block where the target is ?GL_TIME_ELAPSED and it does not affect the result of that query object.
See external documentation.

getQueryObjecti64v(Id, Pname) -> integer()
Types:
  Id = integer()
  Pname = enum()
glGetQueryObjecti64v
See external documentation.
getQueryObjectui64v(Id, Pname) -> integer()
Types:
   Id = integer()
   Pname = enum()

See external documentation.

drawArraysIndirect(Mode, Indirect) -> ok
Types:
   Mode = enum()
   Indirect = offset() | mem()

Render primitives from array data, taking parameters from memory

gl:drawArraysIndirect specifies multiple geometric primitives with very few subroutine calls.
gl:drawArraysIndirect behaves similarly to gl:drawArraysInstancedBaseInstance/5, except that the parameters to gl:drawArraysInstancedBaseInstance/5 are stored in memory at the address given by Indirect.

See external documentation.

drawElementsIndirect(Mode, Type, Indirect) -> ok
Types:
   Mode = enum()
   Type = enum()
   Indirect = offset() | mem()

Render indexed primitives from array data, taking parameters from memory

gl:drawElementsIndirect specifies multiple indexed geometric primitives with very few subroutine calls.
gl:drawElementsIndirect behaves similarly to gl:drawElementsInstancedBaseVertexBaseInstance/7, except that the parameters to gl:drawElementsInstancedBaseVertexBaseInstance/7 are stored in memory at the address given by Indirect.

See external documentation.

uniform1d(Location, X) -> ok
Types:
   Location = integer()
   X = float()

See uniform1f/2

uniform2d(Location, X, Y) -> ok
Types:
   Location = integer()
   X = float()
   Y = float()

See uniform1f/2
uniform3d(Location, X, Y, Z) -> ok
Types:
   Location = integer()
   X = float()
   Y = float()
   Z = float()
See uniform1f/2

uniform4d(Location, X, Y, Z, W) -> ok
Types:
   Location = integer()
   X = float()
   Y = float()
   Z = float()
   W = float()
See uniform1f/2

uniform1dv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [float()]
See uniform1f/2

uniform2dv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [{float(), float()}]
See uniform1f/2

uniform3dv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [{float(), float(), float()}]
See uniform1f/2

uniform4dv(Location, Value) -> ok
Types:
   Location = integer()
   Value = [{float(), float(), float(), float()}]
See uniform1f/2

uniformMatrix2dv(Location, Transpose, Value) -> ok
Types:
uniformMatrix3dv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float()}]

See uniform1f/2

uniformMatrix4dv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}]

See uniform1f/2

uniformMatrix2x3dv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float()}]

See uniform1f/2

uniformMatrix2x4dv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float()}]

See uniform1f/2

uniformMatrix3x2dv(Location, Transpose, Value) -> ok

Types:
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float()}]

See uniform1f/2
uniformMatrix3x4dv(Location, Transpose, Value) -> ok
Types:
  - Location = integer()
  - Transpose = 0 | 1
  - Value = [{float(), float(), float(), float(), float(), float(), float(),
               float(), float(), float(), float(), float(), float()}]
See `uniform1fv2`

uniformMatrix4x2dv(Location, Transpose, Value) -> ok
Types:
  - Location = integer()
  - Transpose = 0 | 1
  - Value = [{float(), float(), float(), float(), float(), float(), float(),
               float(), float()}]
See `uniform1fv2`

uniformMatrix4x3dv(Location, Transpose, Value) -> ok
Types:
  - Location = integer()
  - Transpose = 0 | 1
  - Value = [{float(), float(), float(), float(), float(), float(), float(),
               float(), float(), float(), float(), float(), float()}]
See `uniform1fv2`

getUniformdv(Program, Location) -> matrix()
Types:
  - Program = integer()
  - Location = integer()
See `getUniformfv2`

getSubroutineUniformLocation(Program, Shadertype, Name) -> integer()
Types:
  - Program = integer()
  - Shadertype = enum()
  - Name = string()
Retrieve the location of a subroutine uniform of a given shader stage within a program

`gl:getSubroutineUniformLocation` returns the location of the subroutine uniform variable `Name`
in the shader stage of type `Shadertype` attached to `Program`, with behavior otherwise identical to
`gl:getUniformLocation2`.

See `external` documentation.

getSubroutineIndex(Program, Shadertype, Name) -> integer()
Types:
Program = integer()
Shadertype = enum()
Name = string()

Retrieve the index of a subroutine uniform of a given shader stage within a program
gl:getSubroutineIndex returns the index of a subroutine uniform within a shader stage attached to a program object. Program contains the name of the program to which the shader is attached. Shadertype specifies the stage from which to query shader subroutine index. Name contains the null-terminated name of the subroutine uniform whose name to query.

See external documentation.

gl:getActiveSubroutineUniformName(Program, Shadertype, Index, Bufsize) -> string()
Types:
    Program = integer()
    Shadertype = enum()
    Index = integer()
    Bufsize = integer()

Query the name of an active shader subroutine uniform
gl:getActiveSubroutineUniformName retrieves the name of an active shader subroutine uniform. Program contains the name of the program containing the uniform. Shadertype specifies the stage for which which the uniform location, given by Index, is valid. Index must be between zero and the value of ?GL_ACTIVE_SUBROUTINE_UNIFORMS minus one for the shader stage.

See external documentation.

gl:getActiveSubroutineName(Program, Shadertype, Index, Bufsize) -> string()
Types:
    Program = integer()
    Shadertype = enum()
    Index = integer()
    Bufsize = integer()

Query the name of an active shader subroutine
gl:getActiveSubroutineName queries the name of an active shader subroutine uniform from the program object given in Program. Index specifies the index of the shader subroutine uniform within the shader stage given by Stage, and must between zero and the value of ?GL_ACTIVE_SUBROUTINES minus one for the shader stage.

See external documentation.

uniformSubroutinesuiv(Shadertype, Indices) -> ok
Types:
    Shadertype = enum()
    Indices = [integer()]

Load active subroutine uniforms
gl:uniformSubroutines loads all active subroutine uniforms for shader stage Shadertype of the current program with subroutine indices from Indices, storing Indices[i] into the uniform at location I. Count
must be equal to the value of $\text{GL_ACTIVE_SUBROUTINE_UNIFORM_LOCATIONS}$ for the program currently in use at shader stage $\text{Shadertype}$. Furthermore, all values in $\text{Indices}$ must be less than the value of $\text{GL_ACTIVE_SUBROUTINES}$ for the shader stage.

See external documentation.

```
getUniformSubroutineuiv(Shadertype, Location) -> \{integer(), integer(),
   integer(), integer(), integer(), integer(), integer(), integer(),
   integer(), integer(), integer(), integer(), integer(), integer()

Types:
   Shadertype = \text{enum}()
   Location = \text{integer}()
```

Retrieve the value of a subroutine uniform of a given shader stage of the current program

$\text{gl:getUniformSubroutine}$ retrieves the value of the subroutine uniform at location $\text{Location}$ for shader stage $\text{Shadertype}$ of the current program. $\text{Location}$ must be less than the value of $\text{GL_ACTIVE_SUBROUTINE_UNIFORM_LOCATIONS}$ for the shader currently in use at shader stage $\text{Shadertype}$. The value of the subroutine uniform is returned in $\text{Values}$.

See external documentation.

```
getProgramStageiv(Program, Shadertype, Pname) -> integer()

Types:
   Program = \text{integer}()
   Shadertype = \text{enum}()
   Pname = \text{enum}()
```

Retrieve properties of a program object corresponding to a specified shader stage

$\text{gl:getProgramStage}$ queries a parameter of a shader stage attached to a program object. $\text{Program}$ contains the name of the program to which the shader is attached. $\text{Shadertype}$ specifies the stage from which to query the parameter. $\text{Pname}$ specifies which parameter should be queried. The value or values of the parameter to be queried is returned in the variable whose address is given in $\text{Values}$.

See external documentation.

```
patchParameteri(Pname, Value) -> ok

Types:
   Pname = \text{enum}()
   Value = \text{integer}()
```

Specifies the parameters for patch primitives

$\text{gl:patchParameter}$ specifies the parameters that will be used for patch primitives. $\text{Pname}$ specifies the parameter to modify and must be either $\text{GL_PATCH_VERTICES}$, $\text{GL_PATCH_DEFAULT_OUTER_LEVEL}$ or $\text{GL_PATCH_DEFAULT_INNER_LEVEL}$. For $\text{gl:patchParameteri}$, $\text{Value}$ specifies the new value for the parameter specified by $\text{Pname}$. For $\text{gl:patchParameterfv}$, $\text{Values}$ specifies the address of an array containing the new values for the parameter specified by $\text{Pname}$.

See external documentation.

```
patchParameterfv(Pname, Values) -> ok

Types:
```
Pname = enum()
Values = [float()]
See patchParameteri/2

bindTransformFeedback(Target, Id) -> ok
Types:
  Target = enum()
  Id = integer()
Bind a transform feedback object

bindTransformFeedback binds the transform feedback object with name Id to the current GL state. Id must be a name previously returned from a call to gl:genTransformFeedbacks/1. If Id has not previously been bound, a new transform feedback object with name Id and initialized with with the default transform state vector is created.

See external documentation.

deleteTransformFeedbacks(Ids) -> ok
Types:
  Ids = [integer()]
Delete transform feedback objects

deleteTransformFeedbacks deletes the N transform feedback objects whose names are stored in the array Ids. Unused names in Ids are ignored, as is the name zero. After a transform feedback object is deleted, its name is again unused and it has no contents. If an active transform feedback object is deleted, its name immediately becomes unused, but the underlying object is not deleted until it is no longer active.

See external documentation.

genTransformFeedbacks(N) -> [integer()]
Types:
  N = integer()
Reserve transform feedback object names

genTransformFeedbacks returns N previously unused transform feedback object names in Ids. These names are marked as used, for the purposes of gl:genTransformFeedbacks only, but they acquire transform feedback state only when they are first bound.

See external documentation.

isTransformFeedback(Id) -> 0 | 1
Types:
  Id = integer()
Determine if a name corresponds to a transform feedback object

isTransformFeedback returns ?GL_TRUE if Id is currently the name of a transform feedback object. If Id is zero, or if ?id is not the name of a transform feedback object, or if an error occurs, gl:isTransformFeedback returns ?GL_FALSE. If Id is a name returned by gl:genTransformFeedbacks/1, but that has not yet been bound through a call to gl:bindTransformFeedback/2, then the name is not a transform feedback object and gl:isTransformFeedback returns ?GL_FALSE.

See external documentation.
pauseTransformFeedback() -> ok
Pause transform feedback operations

gl:pauseTransformFeedback pauses transform feedback operations on the currently active transform feedback
object. When transform feedback operations are paused, transform feedback is still considered active and changing
most transform feedback state related to the object results in an error. However, a new transform feedback object may
be bound while transform feedback is paused.
See external documentation.

resumeTransformFeedback() -> ok
Resume transform feedback operations

gl:resumeTransformFeedback resumes transform feedback operations on the currently active transform
feedback object. When transform feedback operations are paused, transform feedback is still considered active and
changing most transform feedback state related to the object results in an error. However, a new transform feedback
object may be bound while transform feedback is paused.
See external documentation.

drawTransformFeedback(Mode, Id) -> ok
Types:
  Mode = enum()
  Id = integer()
Render primitives using a count derived from a transform feedback object

gl:drawTransformFeedback draws primitives of a type specified by Mode using a count retrieved from
the transform feedback specified by Id. Calling gl:drawTransformFeedback is equivalent to calling
gl:drawArrays/3 with Mode as specified, First set to zero, and Count set to the number of vertices captured on
vertex stream zero the last time transform feedback was active on the transform feedback object named by Id.
See external documentation.

drawTransformFeedbackStream(Mode, Id, Stream) -> ok
Types:
  Mode = enum()
  Id = integer()
  Stream = integer()
Render primitives using a count derived from a specified stream of a transform feedback object

gl:drawTransformFeedbackStream draws primitives of a type specified by Mode using a count retrieved
from the transform feedback stream specified by Stream of the transform feedback object specified by Id. Calling
gl:drawTransformFeedbackStream is equivalent to calling gl:drawArrays/3 with Mode as specified, First
set to zero, and Count set to the number of vertices captured on vertex stream Stream the last time transform
feedback was active on the transform feedback object named by Id.
See external documentation.

beginQueryIndexed(Target, Index, Id) -> ok
Types:
  Target = enum()
  Index = integer()
Id = integer()

glBeginQueryIndexed

See external documentation.

endQueryIndexed(Target, Index) -> ok

Types:
  Target = enum()
  Index = integer()

Delimit the boundaries of a query object on an indexed target

gl:beginQueryIndexed and gl:endQueryIndexed/2 delimit the boundaries of a query object. Query
must be a name previously returned from a call to gl:genQueries/1. If a query object with name
Id does not yet exist it is created with the type determined by Target. Target must be
one of ?GL_SAMPLES_PASSED, ?GL_ANY_SAMPLES_PASSED, ?GL_PRIMITIVES_GENERATED, ?
GL_TRANSFORM_FEEDBACK_PRIMITIVES_WRITTEN, or ?GL_TIME_ELAPSED. The behavior of the query
object depends on its type and is as follows.

See external documentation.

getQueryIndexediv(Target, Index, Pname) -> integer()

Types:
  Target = enum()
  Index = integer()
  Pname = enum()

Return parameters of an indexed query object target

gl:getQueryIndexediv returns in Params a selected parameter of the indexed query object target specified by
Target and Index. Index specifies the index of the query object target and must be between zero and a target-
specific maximum.

See external documentation.

releaseShaderCompiler() -> ok

Release resources consumed by the implementation's shader compiler

gl:releaseShaderCompiler provides a hint to the implementation that it may free internal resources associated
with its shader compiler. gl:compileShader/1 may subsequently be called and the implementation may at that time
reallocate resources previously freed by the call to gl:releaseShaderCompiler.

See external documentation.

shaderBinary(Shaders, Binaryformat, Binary) -> ok

Types:
  Shaders = [integer()]
  Binaryformat = enum()
  Binary = binary()

Load pre-compiled shader binaries
gl:shaderBinary loads pre-compiled shader binary code into the Count shader objects whose handles are given in Shaders. Binary points to Length bytes of binary shader code stored in client memory. BinaryFormat specifies the format of the pre-compiled code.

See external documentation.

getShaderPrecisionFormat(Shadertype, Precisiontype) -> {Range::[integer()], integer()}, Precision::integer()}
Types:
  Shadertype = enum()
  Precisiontype = enum()
Retrieve the range and precision for numeric formats supported by the shader compiler

getProgramBinary(Program, BufSize) -> {BinaryFormat::enum(),
  Binary::binary()}
Types:
  Program = integer()
  BufSize = integer()
Return a binary representation of a program object's compiled and linked executable source

See external documentation.
programBinary(Program, BinaryFormat, Binary) -> ok
Types:
  Program = integer()
  BinaryFormat = enum()
  Binary = binary()

Load a program object with a program binary

gl:programBinary loads a program object with a program binary previously returned from gl:getProgramBinary/2. BinaryFormat and Binary must be those returned by a previous call to gl:getProgramBinary/2, and Length must be the length returned by gl:getProgramBinary/2, or by gl:getProgramiv/2 when called with Pname set to ?GL_PROGRAM_BINARY_LENGTH. If these conditions are not met, loading the program binary will fail and Program’s ?GL_LINK_STATUS will be set to ?GL_FALSE.

See external documentation.

programParameteri(Program, Pname, Value) -> ok
Types:
  Program = integer()
  Pname = enum()
  Value = integer()

Specify a parameter for a program object

gl:programParameter specifies a new value for the parameter named by Pname for the program object Program.

See external documentation.

useProgramStages(Pipeline, Stages, Program) -> ok
Types:
  Pipeline = integer()
  Stages = integer()
  Program = integer()

Bind stages of a program object to a program pipeline

gl:useProgramStages binds executables from a program object associated with a specified set of shader stages to the program pipeline object given by Pipeline. Pipeline specifies the program pipeline object to which to bind the executables. Stages contains a logical combination of bits indicating the shader stages to use within Program with the program pipeline object Pipeline. Stages must be a logical combination of ?GL_VERTEX_SHADER_BIT, ?GL_TESS_CONTROL_SHADER_BIT, ?GL_TESS_EVALUATION_SHADER_BIT, ?GL_GEOMETRY_SHADER_BIT, and ?GL_FRAGMENT_SHADER_BIT. Additionally, the special value ?GL_ALL_SHADER_BITS may be specified to indicate that all executables contained in Program should be installed in Pipeline.

See external documentation.

activeShaderProgram(Pipeline, Program) -> ok
Types:
  Pipeline = integer()
  Program = integer()

Set the active program object for a program pipeline object
gl:activeShaderProgram sets the linked program named by Program to be the active program for the program pipeline object Pipeline. The active program in the active program pipeline object is the target of calls to gl:uniform1f/2 when no program has been made current through a call to gl:useProgram/1.

See external documentation.

createShaderProgramv(Type, Strings) -> integer()
Types:
  Type = enum()
  Strings = iolist()
glCreateShaderProgramv
See external documentation.

bindProgramPipeline(Pipeline) -> ok
Types:
  Pipeline = integer()
Bind a program pipeline to the current context

gl:bindProgramPipeline binds a program pipeline object to the current context. Pipeline must be a name previously returned from a call to gl:genProgramPipelines/1. If no program pipeline exists with name Pipeline then a new pipeline object is created with that name and initialized to the default state vector.

See external documentation.

deleteProgramPipelines(Pipelines) -> ok
Types:
  Pipelines = [integer()]
Delete program pipeline objects

gl:deleteProgramPipelines deletes the N program pipeline objects whose names are stored in the array Pipelines. Unused names in Pipelines are ignored, as is the name zero. After a program pipeline object is deleted, its name is again unused and it has no contents. If program pipeline object that is currently bound is deleted, the binding for that object reverts to zero and no program pipeline object becomes current.

See external documentation.

genProgramPipelines(N) -> [integer()]
Types:
  N = integer()
Reserve program pipeline object names

gl:genProgramPipelines returns N previously unused program pipeline object names in Pipelines. These names are marked as used, for the purposes of gl:genProgramPipelines only, but they acquire program pipeline state only when they are first bound.

See external documentation.

isProgramPipeline(Pipeline) -> 0 | 1
Types:
  Pipeline = integer()
Determine if a name corresponds to a program pipeline object

`gl:isProgramPipeline` returns `?GL_TRUE` if `Pipeline` is currently the name of a program pipeline object. If `Pipeline` is zero, or if `pipeline` is not the name of a program pipeline object, or if an error occurs, `gl:isProgramPipeline` returns `?GL_FALSE`. If `Pipeline` is a name returned by `gl:genProgramPipelines/1`, but that has not yet been bound through a call to `gl:bindProgramPipeline/1`, then the name is not a program pipeline object and `gl:isProgramPipeline` returns `?GL_FALSE`.

See `external` documentation.

`getProgramPipelineiv(Pipeline, Pname) -> integer()`

Types:

- `Pipeline = integer()`
- `Pname = enum()`

Retrieve properties of a program pipeline object

`gl:getProgramPipelineiv` retrieves the value of a property of the program pipeline object `Pipeline`. `Pname` specifies the name of the parameter whose value to retrieve. The value of the parameter is written to the variable whose address is given by `Params`.

See `external` documentation.

`programUniform1i(Program, Location, V0) -> ok`

Types:

- `Program = integer()`
- `Location = integer()`
- `V0 = integer()`

Specify the value of a uniform variable for a specified program object

`gl:programUniform` modifies the value of a uniform variable or a uniform variable array. The location of the uniform variable to be modified is specified by `Location`, which should be a value returned by `gl:getUniformLocation/2`. `gl:programUniform` operates on the program object specified by `Program`.

See `external` documentation.

`programUniform1iv(Program, Location, Value) -> ok`

Types:

- `Program = integer()`
- `Location = integer()`
- `Value = [integer()]`

See `programUniform1i/3`

`programUniform1f(Program, Location, V0) -> ok`

Types:

- `Program = integer()`
- `Location = integer()`
- `V0 = float()`

See `programUniform1i/3`
programUniform1fv(\(\text{Program, Location, Value}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{Value} & = [\text{float()}]
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)

programUniform1fv1d(\(\text{Program, Location, V0}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{V0} & = \text{float()}
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)

programUniform1dvv(\(\text{Program, Location, Value}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{Value} & = [\text{float()}]
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)

programUniform1ui(\(\text{Program, Location, V0}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{V0} & = \text{integer()}
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)

programUniform1uv(\(\text{Program, Location, Value}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{Value} & = [\text{integer()}]
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)

programUniform2i(\(\text{Program, Location, V0, V1}\)) \(\rightarrow\) \(\text{ok}\)

Types:

\[
\begin{align*}
\text{Program} & = \text{integer()} \\
\text{Location} & = \text{integer()} \\
\text{V0} & = \text{integer()} \\
\text{V1} & = \text{integer()}
\end{align*}
\]

See \(\text{programUniform1i}\) \(\text{3}\)
programUniform2iv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{integer(), integer()}]
See programUniform1iv/3

programUniform2f(Program, Location, V0, V1) -> ok
Types:
  Program = integer()
  Location = integer()
  V0 = float()
  V1 = float()
See programUniform1iv/3

programUniform2fv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{float(), float()}]
See programUniform1iv/3

programUniform2d(Program, Location, V0, V1) -> ok
Types:
  Program = integer()
  Location = integer()
  V0 = float()
  V1 = float()
See programUniform1iv/3

programUniform2dv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{float(), float()}]
See programUniform1iv/3

programUniform2ui(Program, Location, V0, V1) -> ok
Types:
  Program = integer()
  Location = integer()
  V0 = integer()
  V1 = integer()
See `programUniform1i/3`

```
programUniform2uiv(Program, Location, Value) -> ok
Types:
    Program = integer()
    Location = integer()
    Value = [{integer(), integer()}]
See programUniform1i/3
```

```
programUniform3i(Program, Location, V0, V1, V2) -> ok
Types:
    Program = integer()
    Location = integer()
    V0 = integer()
    V1 = integer()
    V2 = integer()
See programUniform1i/3
```

```
programUniform3iv(Program, Location, Value) -> ok
Types:
    Program = integer()
    Location = integer()
    Value = [{integer(), integer(), integer()}]
See programUniform1i/3
```

```
programUniform3f(Program, Location, V0, V1, V2) -> ok
Types:
    Program = integer()
    Location = integer()
    V0 = float()
    V1 = float()
    V2 = float()
See programUniform1i/3
```

```
programUniform3fv(Program, Location, Value) -> ok
Types:
    Program = integer()
    Location = integer()
    Value = [{float(), float(), float()}]
See programUniform1i/3
```

```
programUniform3d(Program, Location, V0, V1, V2) -> ok
Types:
    Program = integer()
    Location = integer()
    V0 = float()
    V1 = float()
    V2 = float()
See programUniform1i/3
```
programUniform3dv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{float(), float(), float()}]
See programUniform1i/3

programUniform3uii(Program, Location, V0, V1, V2) -> ok
Types:
  Program = integer()
  Location = integer()
  V0 = integer()
  V1 = integer()
  V2 = integer()
See programUniform1i/3

programUniform3uiiv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{integer(), integer(), integer()}]
See programUniform1i/3

programUniform4i(Program, Location, V0, V1, V2, V3) -> ok
Types:
  Program = integer()
  Location = integer()
  V0 = integer()
  V1 = integer()
  V2 = integer()
  V3 = integer()
See programUniform1i/3

programUniform4iv(Program, Location, Value) -> ok
Types:
  Program = integer()
Location = integer()
Value = [{integer(), integer(), integer(), integer()}]

See programUniform1i/3

programUniform4f(Program, Location, V0, V1, V2, V3) -> ok
Types:

  Program = integer()
  Location = integer()
  V0 = float()
  V1 = float()
  V2 = float()
  V3 = float()

See programUniform1i/3

programUniform4fv(Program, Location, Value) -> ok
Types:

  Program = integer()
  Location = integer()
  Value = [{float(), float(), float(), float()}]

See programUniform1i/3

programUniform4d(Program, Location, V0, V1, V2, V3) -> ok
Types:

  Program = integer()
  Location = integer()
  V0 = float()
  V1 = float()
  V2 = float()
  V3 = float()

See programUniform1i/3

programUniform4dv(Program, Location, Value) -> ok
Types:

  Program = integer()
  Location = integer()
  Value = [{float(), float(), float(), float()}]

See programUniform1i/3

programUniform4ui(Program, Location, V0, V1, V2, V3) -> ok
Types:

  Program = integer()
  Location = integer()
  V0 = integer()
\begin{verbatim}
V1 = integer()
V2 = integer()
V3 = integer()

See programUniform1i/3

programUniform4uiv(Program, Location, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Value = [{integer(), integer(), integer(), integer()}]

See programUniform1i/3

programUniformMatrix2fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float()}]

See programUniform1i/3

programUniformMatrix3fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(),
            float(), float()}]

See programUniform1i/3

programUniformMatrix4fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(),
            float(), float(), float(), float(), float(), float(), float(), float(),
            float(), float()}]

See programUniform1i/3

programUniformMatrix2dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
\end{verbatim}
programUniformMatrix3dv(Program, Location, Transpose, Value) -> ok

Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

See programUniform1i3

programUniformMatrix4dv(Program, Location, Transpose, Value) -> ok

Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}

See programUniform1i3

programUniformMatrix2x3fv(Program, Location, Transpose, Value) -> ok

Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float()}

See programUniform1i3

programUniformMatrix3x2fv(Program, Location, Transpose, Value) -> ok

Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float()}

See programUniform1i3

programUniformMatrix2x4fv(Program, Location, Transpose, Value) -> ok

Types:
  Program = integer()
  Location = integer()
Transpose = 0 | 1
Value = [\{float(), float(), float(), float(), float(), float(), float()\}]

See programUniform1i3

programUniformMatrix4x2fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [\{float(), float(), float(), float(), float(), float(), float(), float()\}]

See programUniform1i3

programUniformMatrix3x4fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [\{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()\}]

See programUniform1i3

programUniformMatrix4x3fv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [\{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()\}]

See programUniform1i3

programUniformMatrix2x3dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [\{float(), float(), float(), float(), float(), float()\}]

See programUniform1i3

programUniformMatrix3x2dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
programUniformMatrix2x4dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float()}}
See programUniform1i/3

programUniformMatrix4x2dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float()}}
See programUniform1i/3

programUniformMatrix3x4dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}}
See programUniform1i/3

programUniformMatrix4x3dv(Program, Location, Transpose, Value) -> ok
Types:
  Program = integer()
  Location = integer()
  Transpose = 0 | 1
  Value = [{float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float(), float()}}
See programUniform1i/3

validateProgramPipeline(Pipeline) -> ok
Types:
  Pipeline = integer()
Validate a program pipeline object against current GL state

`gl:validateProgramPipeline` instructs the implementation to validate the shader executables contained in `Pipeline` against the current GL state. The implementation may use this as an opportunity to perform any internal shader modifications that may be required to ensure correct operation of the installed shaders given the current GL state.

See [external documentation](#).

**getProgramPipelineInfoLog(Pipeline, BufSize) -> string()**

Types:

- `Pipeline` = `integer()`
- `BufSize` = `integer()`

Retrieve the info log string from a program pipeline object

`gl:getProgramPipelineInfoLog` retrieves the info log for the program pipeline object `Pipeline`. The info log, including its null terminator, is written into the array of characters whose address is given by `InfoLog`. The maximum number of characters that may be written into `InfoLog` is given by `BufSize`, and the actual number of characters written into `InfoLog` is returned in the integer whose address is given by `Length`. If `Length` is `NULL`, no length is returned.

See [external documentation](#).

**vertexAttribL1d(Index, X) -> ok**

Types:

- `Index` = `integer()`
- `X` = `float()`

`gIAttribL`

See [external documentation](#).

**vertexAttribL2d(Index, X, Y) -> ok**

Types:

- `Index` = `integer()`
- `X` = `float()`
- `Y` = `float()`

`gIAttribL`

See [external documentation](#).

**vertexAttribL3d(Index, X, Y, Z) -> ok**

Types:

- `Index` = `integer()`
- `X` = `float()`
- `Y` = `float()`
- `Z` = `float()`

`gIAttribL`

See [external documentation](#).
vertexAttribL4d(Index, X, Y, Z, W) -> ok
Types:
   Index = integer()
   X = float()
   Y = float()
   Z = float()
   W = float()

See external documentation.

glVertexAttribL4d(Index, X, Y, Z, W) -> ok

See external documentation.

vertexAttribL1dv(Index::integer(), V) -> ok
Types:
   V = {X::float()}

Equivalent to vertexAttribL1d(Index, X).

vertexAttribL2dv(Index::integer(), V) -> ok
Types:
   V = {X::float(), Y::float()}

Equivalent to vertexAttribL2d(Index, X, Y).

vertexAttribL3dv(Index::integer(), V) -> ok
Types:
   V = {X::float(), Y::float(), Z::float()}

Equivalent to vertexAttribL3d(Index, X, Y, Z).

vertexAttribL4dv(Index::integer(), V) -> ok
Types:
   V = {X::float(), Y::float(), Z::float(), W::float()}

Equivalent to vertexAttribL4d(Index, X, Y, Z, W).

vertexAttribLPointer(Index, Size, Type, Stride, Pointer) -> ok
Types:
   Index = integer()
   Size = integer()
   Type = enum()
   Stride = integer()
   Pointer = offset() | mem()

See external documentation.

getVertexAttribLdv(Index, Pname) -> {float(), float(), float(), float(), float()}
Types:
   Index = integer()
Pname = enum()
gGetVertexAttribL
See external documentation.

viewportArrayv(First, V) -> ok
Types:
  First = integer()
  V = [{float(), float(), float(), float()}]
gViewportArrayv
See external documentation.

viewportIndexedf(Index, X, Y, W, H) -> ok
Types:
  Index = integer()
  X = float()
  Y = float()
  W = float()
  H = float()
Set a specified viewport
gl:viewportIndexedf and gl:viewportIndexedfv specify the parameters for a single viewport. Index specifies the index of the viewport to modify. Index must be less than the value of ?GL_MAX_VIEWPORTS. For gl:viewportIndexedf, X, Y, W, and H specify the left, bottom, width and height of the viewport in pixels, respectively. For gl:viewportIndexedfv, V contains the address of an array of floating point values specifying the left (x), bottom (y), width (w), and height (h) of each viewport, in that order. x and y give the location of the viewport's lower left corner, and w and h give the width and height of the viewport, respectively. The viewport specifies the affine transformation of x and y from normalized device coordinates to window coordinates. Let (x nd y nd) be normalized device coordinates. Then the window coordinates (x w y w) are computed as follows:
See external documentation.

viewportIndexedfv(Index, V) -> ok
Types:
  Index = integer()
  V = {float(), float(), float(), float()}
See viewportIndexedf/5

scissorArrayv(First, V) -> ok
Types:
  First = integer()
  V = [{integer(), integer(), integer(), integer()}]
gScissorArrayv
See external documentation.
scissorIndexed(Index, Left, Bottom, Width, Height) -> ok
Types:
  Index = integer()
  Left = integer()
  Bottom = integer()
  Width = integer()
  Height = integer()
glScissorIndexed
See external documentation.

scissorIndexedv(Index, V) -> ok
Types:
  Index = integer()
  V = {integer(), integer(), integer(), integer()}
glScissorIndexed
See external documentation.

depthRangeArrayv(First, V) -> ok
Types:
  First = integer()
  V = [{clamp()}, {clamp()}]
glDepthRangeArrayv
See external documentation.

depthRangeIndexed(Index, N, F) -> ok
Types:
  Index = integer()
  N = clamp()
  F = clamp()
glDepthRangeIndexed
See external documentation.

getFloati_v(Target, Index) -> [float()]
Types:
  Target = enum()
  Index = integer()
See getBooleanv/1

getDoublei_v(Target, Index) -> [float()]
Types:
  Target = enum()
  Index = integer()
See `getBooleanv/1`

debugMessageControlARB(Source, Type, Severity, Ids, Enabled) -> ok
Types:
  Source = enum()
  Type = enum()
  Severity = enum()
  Ids = [integer()]
  Enabled = 0 | 1
glDebugMessageControlARB
See external documentation.

debugMessageInsertARB(Source, Type, Id, Severity, Buf) -> ok
Types:
  Source = enum()
  Type = enum()
  Id = integer()
  Severity = enum()
  Buf = string()
glDebugMessageInsertARB
See external documentation.

getDebugMessageLogARB(Count, Bufsize) -> {integer(), Sources::[enum()],
Types::[enum()], Ids::[integer()], Severities::[enum()], MessageLog::
[string()]}
Types:
  Count = integer()
  Bufsize = integer()
glGetDebugMessageLogARB
See external documentation.

getGraphicsResetStatusARB() -> enum()
glGetGraphicsResetStatusARB
See external documentation.

drawArraysInstancedBaseInstance(Mode, First, Count, Primcount, Baseinstance)
-> ok
Types:
  Mode = enum()
  First = integer()
  Count = integer()
  Primcount = integer()
  Baseinstance = integer()
Draw multiple instances of a range of elements with offset applied to instanced attributes

**gl:drawArraysInstancedBaseInstance** behaves identically to **gl:drawArrays/3** except that **Primcount** instances of the range of elements are executed and the value of the internal counter **InstanceID** advances for each iteration. **InstanceID** is an internal 32-bit integer counter that may be read by a vertex shader as **gl_InstanceID**.

See external documentation.

```erlang
drawElementsInstancedBaseInstance(Mode, Count, Type, Indices, Primcount, Baseinstance) -> ok
drawElementsInstancedBaseVertexBaseInstance(Mode, Count, Type, Indices, Primcount, Basevertex, Baseinstance) -> ok
drawTransformFeedbackInstanced(Mode, Id, Primcount) -> ok
```

Types:
- **Mode** = enum()
- **Count** = integer()
- **Type** = enum()
- **Indices** = offset() | mem()
- **Primcount** = integer()
- **Baseinstance** = integer()
- **Basevertex** = integer()
Mode = enum()
Id = integer()
Primcount = integer()
glDrawTransformFeedbackInstance
See external documentation.

drawTransformFeedbackStreamInstanced(Mode, Id, Stream, Primcount) -> ok
Types:
  Mode = enum()
  Id = integer()
  Stream = integer()
  Primcount = integer()
glDrawTransformFeedbackStreamInstance
See external documentation.

getInternalformativ(Target, Internalformat, Pname, BufSize) -> [integer()]
Types:
  Target = enum()
  Internalformat = enum()
  Pname = enum()
  BufSize = integer()
glGetInternalformat
See external documentation.

bindImageTexture(Unit, Texture, Level, Layered, Layer, Access, Format) -> ok
Types:
  Unit = integer()
  Texture = integer()
  Level = integer()
  Layered = 0 | 1
  Layer = integer()
  Access = enum()
  Format = enum()

Bind a level of a texture to an image unit
gl:bindImageTexture binds a single level of a texture to an image unit for the purpose of reading and writing it from shaders. Unit specifies the zero-based index of the image unit to which to bind the texture level. Texture specifies the name of an existing texture object to bind to the image unit. If Texture is zero, then any existing binding to the image unit is broken. Level specifies the level of the texture to bind to the image unit.
See external documentation.

memoryBarrier(Barriers) -> ok
Types:
  Barriers = integer()
Defines a barrier ordering memory transactions

`gl::memoryBarrier` defines a barrier ordering the memory transactions issued prior to the command relative to those issued after the barrier. For the purposes of this ordering, memory transactions performed by shaders are considered to be issued by the rendering command that triggered the execution of the shader. `Barriers` is a bitfield indicating the set of operations that are synchronized with shader stores; the bits used in `Barriers` are as follows:

See external documentation.

```latex
texStorage1D(Target, Levels, Internalformat, Width) -> ok
```

Types:

```latex
Target = enum()
Levels = integer()
Internalformat = enum()
Width = integer()
```

Simultaneously specify storage for all levels of a one-dimensional texture

`gl::texStorage1D` specifies the storage requirements for all levels of a one-dimensional texture simultaneously. Once a texture is specified with this command, the format and dimensions of all levels become immutable unless it is a proxy texture. The contents of the image may still be modified, however, its storage requirements may not change. Such a texture is referred to as an immutable-format texture.

See external documentation.

```latex
texStorage2D(Target, Levels, Internalformat, Width, Height) -> ok
```

Types:

```latex
Target = enum()
Levels = integer()
Internalformat = enum()
Width = integer()
Height = integer()
```

Simultaneously specify storage for all levels of a two-dimensional or one-dimensional array texture

`gl::texStorage2D` specifies the storage requirements for all levels of a two-dimensional texture or one-dimensional texture array simultaneously. Once a texture is specified with this command, the format and dimensions of all levels become immutable unless it is a proxy texture. The contents of the image may still be modified, however, its storage requirements may not change. Such a texture is referred to as an immutable-format texture.

See external documentation.

```latex
texStorage3D(Target, Levels, Internalformat, Width, Height, Depth) -> ok
```

Types:

```latex
Target = enum()
Levels = integer()
Internalformat = enum()
Width = integer()
Height = integer()
Depth = integer()
```

Simultaneously specify storage for all levels of a three-dimensional, two-dimensional array or cube-map array texture
**gl::texStorage3D** specifies the storage requirements for all levels of a three-dimensional, two-dimensional array or cube-map array texture simultaneously. Once a texture is specified with this command, the format and dimensions of all levels become immutable unless it is a proxy texture. The contents of the image may still be modified, however, its storage requirements may not change. Such a texture is referred to as an *immutable-format* texture.

See **external documentation**.

```erlang
depthBoundsEXT(Zmin, Zmax) -> ok
```

Types:

- `Zmin = clamp()`
- `Zmax = clamp()`

**glDepthBoundsEXT**

See **external documentation**.

```erlang
stencilClearTagEXT(StencilTagBits, StencilClearTag) -> ok
```

Types:

- `StencilTagBits = integer()`
- `StencilClearTag = integer()`

**glStencilClearTagEXT**

See **external documentation**.