label	*The label for the entry box* *Allowed: String* *Default: 'Enter value'*
title	*The title for the entry box* *Allowed: String* *Default: 'Dialog box'*
type	*The type of entry widgets* *Allowed: 'entry', 'file', 'string', 'boolean', 'measure', 'quantity', 'record', 'region', 'scalar, 'untyped'* *Default: 'entry'*
value	*The initial value to insert* *Allowed: Appropriate type*
hlp	*Popup help for entry box* *Allowed: String*
helpOnLabel	*Is the popup help on the label or the entry box* *Allowed: T or F* *Default: T*
widgetset	*The widget server to use* *Allowed: widgetserver object* *Default: dws*

Returns
String

Description

Example

```
include 'dialogbox.g'
txt := dialogbox(label='Enter colour', title='Colour chooser', type='string', value='Blue');
if (!is_fail(txt)) {
   print 'Selected colour=', txt
} else {
  print 'Colour selection aborted'
}
```

The initial value of 'blue' is inserted.

1.4.10 extendoptionmenu - Tool

Package **display**
Module **widgets**

Tk widget where the (interactively extendable) menu label changes with selection include "extendoptionmenu.g"

Constructors

extendoptionmenu	*Construct an extendoptionmenu*

Functions

disabled	*Disable or enable the extendoptionmenu widget*
done	*Destroy ourselves*
extend	*Extend the menu with more items*
findlabel	*Find the index of the item with the given label*
geteventmask	*Get the event mask*
getindex	*Get the index of the current selection*
getlabel	*Get the label of the current selection*
getlabels	*Gets all of the labels associated with the menu*
getpreviousindex	*Get the index previous to the current selection*
replace	*Replace the menu with the provided items*
selectindex	*Select an item in the menu by index*
selectlabel	*Select an item in the menu by label*
setbackground	*Set the color of the background of the extendoptionmenu button*
seteventmask	*Set the event mask*
setforeground	*Set the color of the foreground of the extendoptionmenu button*
setwidth	*Set the width of the optionmenu button*

Description

This Tk widget enables you to create a flat (i.e. no submenus) menu in a specified frame. You can associate a label (that which appears on the top menu button), a name (that which appears in the menu list) and a value (whatever you like, it may be different from the label) for each menu item. When you select one of the items in the menu, the label of the menu button changes to reflect the label of the selected item.
If your list of items is very long, a menu may be too long to sensibly manage. Therefore, you can optionally choose (see constructor) to use a popup scrolling listbox rather than a menu (it's really implemented with a selectabelist widget). It can be treated like any of the built in Tk widgets, and has almost all of the same construction arguments as the button Tk widget.
Whenever a selection is made, this widget emits an event called `select`*. In the $value of this event, you can find a record with fields* `label`*,* `name` *and* `value` *containing that information for the selected item. The event generation can be disabled if you desire with the seteventmask tool function.*
Whenever the menu is replaced with the replace tool function, this widget emits an event called `replaced` *(it has no value). Note that the tool function extend*

invokes the `replace` *function, so a* `replaced` *event is generated then as well. The widget accepts an event called* `setwidth`*. It is the same as calling the setwidth tool function.*
This widget is very much like the optionmenu widget. There are two differences. First, the `extendoptionmenu` *menu can be extended interactively during use. When constructed, you specify a special symbol for the menu (defaults to the ellipsis). When the user selects this item, you can type in a new entry in a ahlinkdialogboxwidgets:dialogbox or use callback functions that the programmer provides. Second,* `extendoptionmenu` *only offers you labels for the menu items, whereas optionmenu offers you labels, names, and values for each menu item. The* `select` *event is also triggered when you add an item to the menu.*

extendoptionmenu - Constructor

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Construct an extendoptionmenu

Synopsis

extendoptionmenu(parent, labels, hlp, hlp2, nbreak, symbol, callback1, callback2, callbackdata, dialoglabel, dialogtitle, padx, pady, width, updatelabel, height, justify, font, relief, borderwidth, foreground, background, disabled, anchor, fill, widgetset)

Arguments

parent	*Parent of the menu* *Allowed: Widget*
labels	*The menu item labels* *Allowed: Vector of strings* *Default: Empty menu*
hlp	*Short popuphelp text* *Allowed: String* *Default: No help*
hlp2	*Long popuphelp text* *Allowed: String* *Default: No help*
nbreak	*The break point for menu or listbox usage* *Allowed: Integer* *Default: 20*
symbol	*Special symbol for menu extension* *Allowed: String* *Default: '...'*
callback1	*Callback function when special symbol selected* *Allowed: String* *Default: F*
callback2	*Callback function when special symbol selected* *Allowed: String* *Default: F*
callbackdata	*Data for callback2 calculations* *Allowed: Any Glish variable* *Default: F*
dialoglabel	*Label for dialogbox* *Allowed: String* *Default: 'Item'*
dialogtitle	*Title for dialogbox* *Allowed: String* *Default: 'Enter new item ¡CR¿'*
padx	*Horizontal padding around text* *Allowed: Integer* *Default: 7*
pady	*Vertical padding around text* *Allowed: Integer* *Default: 3*
width	*Width in character units* *Allowed: Integer* *Default: Auto-scale*
updatelabel	*Update menu label with selected item* *Allowed: T or F* *Default: T*
height	*Height in character units* *Allowed: Integer* *Default: 1*
justify	*Justification of text* *Allowed: String from 'center', 'left', 'right'* *Default: 'center'*
font	*Font of text* *Allowed: X font*
relief	*Edge relief* *Allowed: String from 'flat', 'ridge', 'raised', 'sunken', 'groove'*

Returns
Agent

Description

By default, the `extendoptionmenu` *will insert a "..." into the menu (you don't include it in the* `labels` *list). Use the argument* `symbol`*) to use a different special selection symbol. When the user selects this item from the menu, a few things might happen.*
First, if `callback1` *is a valid function, then it is invoked (it has no arguments), and the result is expected to be a string which is inserted into the menu (duplicates are ignored).*
Second, if `callback1` *is not a valid function, then a ahlinkdialogboxwidgets:dialogbox widget is created, and the user types the new menu item label into the entry box of this widget. If the new item is a duplicate, nothing further happens. Otherwise, if* `callback2` *is a valid function, then it is invoked with arguments* `callback2(newitem, labels, callbackdata`*. Here,* `newitem` *is the string that came from the* `dialogbox`*.* `labels` *is the list of current menu labels, and* `callbackdata` *is any Glish variable you care to pass into the* `callback2` *function. It is up to you, the programmer, whether you need to use* `callbackdata` *or not. If the result returned from* `callback2` *is* `T`*, then the* `newitem` *is placed in the menu.*
Note that the `width` *argument can be used in three ways. If* `width=-1` *(the default), the menu button will be wide enough (and of fixed width) to accomodate all of the item labels in the menu. This means that the GUI won't jitter when you write different length labels into it. If* `width=0`*, then the menu button will change its width to accomodate whatever label is written into it. If* `width > 0`*, then the button is again of fixed width, with the width that you specify.*
If there are less items in the list than `nbreak`*, you will see a flat menu. Otherwise, you will see a scrolling list box. Note that to activate the listbox you must press and release the main button. Whereas for the flatmenu, pressing is enough to activate it.*

Example

```
- include 'extendoptionmenu.g'
- labels := "m cm mm";
- f := frame();
- em := extendoptionmenu(f, labels);
```

The ellipsis is used, by default, as the menu extend symbol. A dialog box will appear if the user selects the extend symbol. If the user types in a string which is not already in the list of labels, it will be added to the menu.

Example

```
- include 'quanta.g'
- include 'extendoptionmenu.g'
- qs := quanta();
- labels := "m cm mm";
- f := frame();
- em := extendoptionmenu(f, labels, callback2=myfun, callbackdata=qs)
```

The ellipsis is used, by default, as the menu extend symbol. A dialog box will appear if the user selects the extend symbol. If the user types in a string which is not already in the list of labels, it will be added to the menu provided the function `myfun`*, which you supply, returns T. Function tt myfun is given the quanta server tool,* `qs` *for use.*

---

extendoptionmenu.extend - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Extend the menu with more items

Synopsis
extend(labels, width)

Arguments

labels	*The menu item labels* *Allowed: Vector of strings* *Default: Empty menu*
width	*Width in character units* *Allowed: Integer* *Default: Auto-scale*

Returns
Bool

Description

This tool function extends the menu with the supplied items.

Example

```
- labels := "m cm mm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.extend(labels="dm Dm");
- m.getlabels();
m cm mm dm Dm
-
```

Note that unlike when you interactively extend the menu, no checking for the validity of the new items is done. That is your responsibility.

---

extendoptionmenu.replace - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Replace the menu with the provided items

Synopsis
replace(labels, width)

Arguments

labels	*The menu item labels* *Allowed: Vector of strings* *Default: Empty menu*
width	*Width in character units* *Allowed: Integer* *Default: Auto-scale*

Returns
Bool

Description

This tool function replaces the menu with the supplied items.

Example

```
- labels := "m cm mm";
- f := frame();
- m := extendoptionmenu(f, labels);
m.getlabels();
m cm mm
- m.replace("dm Dm");
- m.getlabels();
dm Dm
-
```

---

extendoptionmenu.selectindex - Function

Package display
Module widgets
Tool extendoptionmenu

Select an item in the menu by index

Synopsis
selectindex(idx)

Arguments

idx	*The menu item index* *Allowed: Integer*

Returns
Bool

Description

This tool function selects a menu item by its index. It is equivalent to selecting the item with the GUI interface. If the index is invalid, it returns `F`.

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.selectindex(3);
- m.getlabel();
cm
-
```

---

extendoptionmenu.selectlabel - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Select an item in the menu by label

Synopsis
selectlabel(label)

Arguments

label	*The menu item label* *Allowed: Same type as the array "labels" used in construction*

Returns
Bool

Description

This tool function selects a menu item by its label. It is equivalent to selecting the item with the GUI interface. If the label is invalid, it returns `F`.

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels)
```

```
- m.selectlabel('mm');
- m.getlabel();
mm
-
```

Returns
Bool

Description

This tool function selects a menu item by its name. It is equivalent to selecting the item with the GUI interface. If the name is invalid, it returns `F`.

Example

```
- labels := "m mm cm";
- names := "A B C";
- f := frame();
- m := extendoptionmenu(f, labels, names);
- m.selectname("C");
- m.getindex();
3
-
```

---

extendoptionmenu.getindex - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Get the index of the current selection

Synopsis
getindex()

Returns
Integer or F

Description

This tool function recovers the index of the currently selected item in the menu. If the menu is empty, it returns `F`*. If no menu item has been selected by the user, it returns unity.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels)
- m.selectindex(2);
- m.getindex();
2
```

---

extendoptionmenu.getpreviousindex - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Get the index previous to the current selection

Synopsis
getpreviousindex()

Returns
Integer or F

Description

This tool function recovers the index that you selected previous to the current selection. If the menu is empty, it returns `F`*. If no menu item has been selected by the user, it returns unity.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.selectindex(2);
- m.selectindex(3);
- m.getpreviousindex();
2
```

---

extendoptionmenu.getlabel - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Get the label of the current selection

Synopsis
getlabel()

Returns
String or F

Description

This tool function recovers the label of the currently selected item in the menu. If the menu is empty, it returns `F`. *If no menu item has been selected by the user, it returns the first item.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.selectindex(2);
- m.getlabel();
mm
```

---

extendoptionmenu.getlabels - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Gets all of the labels associated with the menu

Synopsis
getlabels()

Returns
Vector of String

Description

This tool function recovers the labels associated with each item in the menu. If the menu is empty, it returns an empty array.

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.getlabels();
m mm cm ...
```

Note the ellipsis shows in the list.

---

extendoptionmenu.findlabel - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Find the index of the item with the given label

Synopsis

findlabel(label)

Arguments

label	*The menu item label* *Allowed: Same type as the array "labels" used in construction*

Returns
Integer or F

Description

This tool function find the index in the menu of the item specified by its label. If you give an invalid label, `F` *is returned.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.findlabel('cm');
3
```

---

extendoptionmenu.geteventmask - Function

Package display
Module widgets
Tool extendoptionmenu

Get the event mask

Synopsis
geteventmask()

Returns
Vector of Bool

Description

This tool function recovers the event mask. This mask indicates whether the `select` *event will be emitted or not when you select a menu item. By default the mask is all T. You can modify it with the seteventmask tool function. New items added to the menu get a T value for their mask.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.geteventmask();
[T T T T]
- m.seteventmask([T,F,F,T]);
- m.geteventmask();
[T F F T]
```

Note that an element for the ellipsis is recovered too (element 4).

---

extendoptionmenu.seteventmask - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Set the event mask

Synopsis
seteventmask(mask)

Arguments

mask	*The event mask* *Allowed: Vector of Bool*

Returns
Bool

Description

This tool function sets the event mask. This mask indicates whether the `select` *event will be emitted or not when you select a menu item. By default the mask is all T. You can get the current mask with the geteventmask tool function. If the mask is invalid, it returns* `F`*. New items added to the menu get a T value for their mask.*

Example

```
- labels := "m mm cm";
- f := frame();
- m := extendoptionmenu(f, labels);
- m.seteventmask([T,F,T,T]);
- m.geteventmask();
[T F T T]
- whenever m->select do {print $value;};
```

In this example, when you selected the second item, no event would be emitted. Note that an element for the ellipsis is required too (element 4).

---

extendoptionmenu.setforeground - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Set the color of the foreground of the extendoptionmenu button

Synopsis
setforeground(color)

Arguments

color	*The color* *Allowed: Any X color encoded as a String*

Returns
Bool

Description

This tool function sets the color of the foreground of the extendoptionmenu button.

---

extendoptionmenu.setbackground - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Set the color of the background of the extendoptionmenu button

Synopsis
setbackground(color)

Arguments

color	*The color* *Allowed: Any X color encoded as a String*

Returns
Bool

Description

This tool function sets the color of the background of the extendoptionmenu button.

---

extendoptionmenu.setwidth - Function

Package	**display**
Module	**widgets**
Tool	**extendoptionmenu**

Set the width of the optionmenu button

Synopsis
setwidth(width)

Arguments

width	*The width in character units* *Allowed: Integer*

Returns
Bool

Description

This tool function sets the width of the optionmenu button in character units.

---

extendoptionmenu.disabled - Function

Package display
Module widgets
Tool extendoptionmenu

Disable or enable the extendoptionmenu widget

Synopsis
disabled(disable)

Arguments

disable	*Disable the widget* *Allowed: T or F* *Default: T*

Returns
Bool

Description

This tool function disables or enables the extendoptionmenu.