

RUBY/TK - CHECKBUTTON WIDGET

http://www.tutorialspoint.com/ruby/rubyTk_checkbutton.htm

Copyright © tutorialspoint.com

Description:

A **Checkbutton** is like a regular button, except that not only can the user press it, which will invoke a command callback, but it also holds a binary value of some kind *i. e. , atoggle*. Checkbuttons are used all the time when a user is asked to choose between, e.g., two different values for an option.

A checkbutton can display a textual string, bitmap or image. If text is displayed, it must all be in a single font, but it can occupy multiple lines on the screen (if it contains newlines or if wrapping occurs because of the *wraplength* option) and one of the characters may optionally be underlined using the *underline* option.

A checkbutton has all of the behavior of a simple button, including the following: it can display itself in either of three different ways, according to the state option; it can be made to appear raised, sunken, or flat; it can be made to flash; and it invokes a Tcl command whenever mouse button 1 is clicked over the checkbutton.

Syntax:

Here is a simple syntax to create this widget:

```
TkCheckButton.new(root) {  
    ....Standard Options....  
    ....Widget-specific Options....  
}
```

Standard Options:

- activebackground
- activeforeground
- anchor
- background
- bitmap
- borderwidth
- compound
- cursor
- disabledforeground
- font
- foreground
- highlightbackground
- highlightcolor
- highlightthickness
- image
- justify
- padx
- pady

- relief
- takefocus
- text
- textvariable
- underline
- wraplength

These options have been described in previous chapter.

Widget-specific Options:

SN	Options with Description
1	<p>command => String</p> <p>Specifies a Ruby command to associate with the button. This command is typically invoked when mouse button 1 is released over the button window. Here you can associate a Ruby method to be executed against mouse click. Built in function which can be called using command option:</p> <ul style="list-style-type: none"> • deselect: Deselects the checkbutton and sets the associated variable to its "off" value. • flash: Flashes the checkbutton. This is accomplished by redisplaying the checkbutton several times, alternating between active and normal colors. • select: Selects the checkbutton and sets the associated variable to its "on" value. • toggle: Toggles the selection state of the button, redisplaying it and modifying its associated variable to reflect the new state.
2	<p>height => Integer</p> <p>Specifies a desired height for the button.</p>
3	<p>indicatoron => Boolean</p> <p>Specifies whether or not the indicator should be drawn. Must be a proper boolean value. If <i>false</i>, the <i>relief</i> option is ignored and the widget's relief is always sunken if the widget is selected and raised otherwise.</p>
4	<p>offvalue => Integer</p> <p>Specifies value to store in the button's associated variable whenever this button is deselected. Defaults to 0.</p>
5	<p>onvalue => Integer</p> <p>Specifies value to store in the button's associated variable whenever this button is selected. Defaults to 1</p>
6	<p>selectcolor => String</p> <p>Specifies a background color to use when the button is selected. If <i>indicatoron</i> is true, then the color applies to the indicator. If <i>indicatoron</i> is false, this color is used as the background for the entire widget, in place of <i>background</i> or <i>activebackground</i>, whenever</p>

the widget is selected.

7 **selectimage** => Image

Specifies an image to display *inplaceoftheimageoption* when the checkbox is selected. This option is ignored unless the image option has been specified.

8 **state** => String

Specifies one of three states for the button: *normal*, *active*, or *disabled*. In normal state the button is displayed using the *foreground* and *background* options. The active state is typically used when the pointer is over the button. In active state the button is displayed using the *activeforeground* and *activebackground* options. Disabled state means that the button should be insensitive.

9 **variable** => Variable

Specifies name of global variable to set to indicate whether or not this button is selected. Defaults to the name of the button within its parent

10 **width** => Integer

Specifies a desired width for the button.

Event Bindings:

Ruby/Tk automatically creates class bindings for checkboxes that give them the following default behavior:

- A checkbox activates whenever the mouse passes over it and deactivates whenever the mouse leaves the checkbox.
- When mouse button 1 is pressed over a checkbox it is invoked
itsselectionstatetogglesandthecommandassociatedwiththebuttonisinvoked, ifthereisone.
- When a checkbox has the input focus, the space key causes the checkbox to be invoked.

If the checkbox's state is *disabled* then none of the above actions occur: the checkbox is completely non-responsive.

Examples:

```
require 'tk'

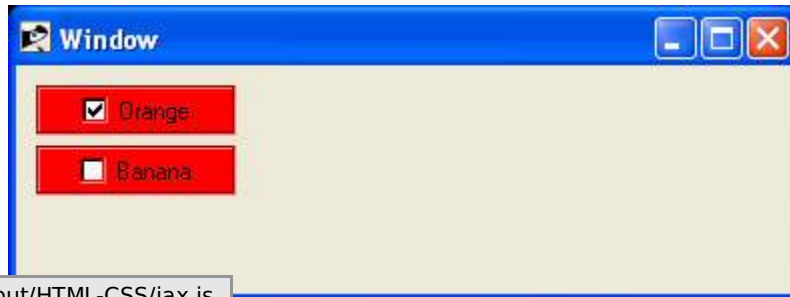
root = TkRoot.new
root.title = "Window"

CkhButton1 = TkCheckButton.new(root) do
  text "Orange"
  indicatoron "true"
  background "red"
  relief "groove"
  height 2
  width 2
  onvalue 'Orange'
  place('height' => 25, 'width' => 100, 'x' => 10, 'y'=> 10)
  command (select)
end

CkhButton2 = TkCheckButton.new(root) do
  text "Banana"
  background "red"
```

```
relief "groove"  
height 2  
width 2  
onvalue 'Banana'  
place('height' => 25, 'width' => 100, 'x' => 10, 'y'=> 40)  
end  
Tk.mainloop
```

This will produce the following result:



Loading [MathJax]/jax/output/HTML-CSS/jax.js