Issue wit getting name of modulator #600

Unanswered **bijlevel** asked this question in **Q&A**

bijlevel 3 weeks ago edited

I have a slider and a button both calling the same function to get the name. When I use

```
local mName = L(comp:getOwner():getName())
```

I get a nil value error for 'getOwner()' when clicking on the button, while clicking on the slider it works OK.

When I use

```
local mName = L(comp:getName())
```

I get a nil value error for 'getName()' on the slider and now the button works OK.

The call for the slider is on mouseUp, the call for the button is 'when the modulator value changes'

So this inconsistency(?) puzzles me. Any help much appreciated. I have attached a panel as example.

getName issue.bpanelz.zip

↑ 1

Category

Q&A

Labels

None yet

2 participants

1 comment · 4 replies

Oldest Newest Top

damiensellier 3 weeks ago edited

For mouse callbacks this is the way it works :

```
--
-- Called when a mouse is down on this component
--

whenMouseDown = function(--[[ CtrlrComponent --]] comp, --[[ MouseEvent --]] event)

local mName = L(comp:getOwner():getName())
console("Displays mName : "..(mName))

compName =
panel:getModulatorByName(mName):getComponent():getProperty("componentVisibleName")
console("Displays compName : "..(compName))
// etc

end
```

I got it working for both the slider and button on mouse down

On modulator value change it's another story because the arguments of the function are different, it's on modulator level, so it's "mod", not "comp" as you can see.

```
--
-- Called when a modulator value changes
-- @mod   http://ctrlr.org/api/class_ctrlr_modulator.html
-- @value    new numeric value of the modulator
--
nameOnValueChange = function(--[[ CtrlrModulator --]] mod, --[[ number --]] value, --[[
number --]] source)

local mName = L(mod:getName())
console("Displays mName : "..(mName))

end
```

mouseDownForCompName_1_0_2024-01-29_19-01.panel.zip

↑ 1 ☺ 4 replies

bijlevel 3 weeks ago Author

@damiensellier Thanks for your input! As I understand now (and somehow guessed) the difference is the levels of the components, e.g. '*comp*' and '*mod*'. Well that means I do have to make two separate functions, one for a slider and one for a button. No problem!

☺

damiensellier 3 weeks ago edited

Not for the button and another one for the slider BUT one for "when mod value change" and another "mouse down (on component)". It's 2 different scripts for 2 different events. One is related to the modulator value, the other one is a callback on a mouse event on the component. 2 very different things.

You can use the same for both the slider and button though.

That's why the template at the top of the default scripts are different. One passes mod argument and mod value, the other passes the component "comp" and the event triggered by the mouse.

function(--[[CtrlrComponent --]] comp, --[[MouseEvent --]] event)
VS
function(--[[CtrlrModulator --]] mod, --[[number --]] value, --[[number --]] source)

PS : source defines the way the mod has changed value, it's either from the GUI by rotating a slider for ex, internally via a script or via midi input, I don't remelber the other, you can then filter your function depending on the source.

https://github.com/damiensellier/CtrlrX/wiki/Variables-for-the-%22source%22-argument-in-LUA-scripts

☺ ❤ 1

damiensellier 3 weeks ago

Happy 600th discussion btw ;)

1

bijlevel 3 weeks ago Author

@damiensellier Yest I also noticed the remarkable thread #! And about the scripts, you're right. I ended up with one script for the button and one for the slider and they also do differ a bit in actual code. Thanks again for your input! (What time is it in your region?)

Write a reply