Package: shinyfocus (via r-universe)

May 20, 2026

Title Use Browser Focus Events in 'shiny'

Version 0.0.0.9000

Description As a user moves through a 'shiny' app, it can be useful to trigger server events when different parts of the app gain or lose focus. This package aims to make it easy to implement such events.

License MIT + file LICENSE

URL https://shinyworks.github.io/shinyfocus/, https://github.com/shinyworks/shinyfocus

BugReports https://github.com/shinyworks/shinyfocus/issues

Imports htmltools, rlang, shiny

Suggests covr, pkgload, shinytest2, testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

Config/pak/sysreqs cmake make libuv1-dev zlib1g-dev

Repository https://shinyworks.r-universe.dev

Date/Publication 2023-11-13 17:59:40 UTC

RemoteUrl https://github.com/shinyworks/shinyfocus

RemoteRef HEAD

RemoteSha 14aceaa0587ae321305a7e97af7d94765fedd8b8

Contents

on_blur *Respond when input loses focus*

Description

Set up a shiny::observeEvent() observer to trigger when the named input loses focus.

Usage

```
on_blur(
  id,
  handler_expr,
  ...,
  priority = 99999,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

id	The ID string of an input.
handler_expr	The expression to trigger whenever the specified input loses focus. This expression is quoted and executed in the calling environment.
...	Arguments passed on to shiny::observeEvent label A label for the observer or reactive, useful for debugging. suspended If TRUE, start the observer in a suspended state. If FALSE (the default), start in a non-suspended state. autoDestroy If TRUE (the default), the observer will be automatically destroyed when its domain (if any) ends. ignoreNULL Whether the action should be triggered (or value calculated, in the case of eventReactive) when the input event expression is NULL. See Details. once Whether this observeEvent should be immediately destroyed after the first time that the code in handlerExpr is run. This pattern is useful when you want to subscribe to a event that should only happen once.
priority	An integer that controls the priority with which the observer should be executed. It often makes sense for this priority to be very high to avoid conflicts.
session	The session (aka domain) in which the observer will be created and executed. The default is almost always desired.

Value

A shiny observer (see shiny::observe()).

See Also

Other observers: on_focus_change(), on_focus()

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {
  # App 1: A relatively simple ui without modules.
  shiny::shinyApp(
    ui = shiny::fluidPage(
      shinyfocus_js_dependency(),
      shiny::textInput("input1", "Input 1"),
      shiny::textInput("input2", "Input 2"),
      shiny::actionButton("go_button", "Go!"),
      shiny::textOutput("blurring")
    ),
    server = function(input, output, session) {
      # Update the value in blurring whenever input1 loses focus.
      on_blur(
        "input1",
        output$blurring <- shiny::renderText(
          paste(
            "You left input1 at",
            Sys.time()
          )
        )
      )
    }
  )

  # App 2: With module.
  blurUI <- function(id) {
    shiny::tagList(
      shiny::textInput(shiny::NS(id, "observe_me"), "Observe me"),
      shiny::textOutput(NS(id, "blurring"))
    )
  }
  blurServer <- function(id) {
    shiny::moduleServer(id, function(input, output, session) {
      on_blur(
        "observe_me",
        output$blurring <- shiny::renderText(
          paste(
            "You left observe_me at",
            Sys.time()
          )
        )
      )
    })
  }

  shiny::shinyApp(
    ui = shiny::fluidPage(
      shinyfocus_js_dependency(),
      blurUI("blur_module"),
      shiny::textInput("another_input", "Another input"),
```

```
      shiny::actionButton("go_button", "Go!")
    ),
    server = function(input, output, session) {
      blurServer("blur_module")
    }
  )
}
```

---

on_focus *Respond when input gains focus*

Description

Set up a `shiny::observeEvent()` observer to trigger when the named input gains focus.

Usage

```
on_focus(
  id,
  handler_expr,
  ...,
  priority = 99999,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`id`	The ID string of an input.
`handler_expr`	The expression to trigger whenever the specified input gains focus. This expression is quoted and executed in the calling environment.
`...`	Arguments passed on to `shiny::observeEvent` `label` A label for the observer or reactive, useful for debugging. `suspended` If `TRUE`, start the observer in a suspended state. If `FALSE` (the default), start in a non-suspended state. `autoDestroy` If `TRUE` (the default), the observer will be automatically destroyed when its domain (if any) ends. `ignoreNULL` Whether the action should be triggered (or value calculated, in the case of `eventReactive`) when the input event expression is `NULL`. See Details. `once` Whether this `observeEvent` should be immediately destroyed after the first time that the code in `handlerExpr` is run. This pattern is useful when you want to subscribe to a event that should only happen once.
`priority`	An integer that controls the priority with which the observer should be executed. It often makes sense for this priority to be very high to avoid conflicts.
`session`	The session (aka domain) in which the observer will be created and executed. The default is almost always desired.

Value

A shiny observer (see shiny::observe()).

See Also

Other observers: on_blur(), on_focus_change()

Examples

```
## Only run examples in interactive R sessions
if (interactive()) {
  # App 1: A relatively simple ui without modules.
  shiny::shinyApp(
    ui = shiny::fluidPage(
      shinyfocus_js_dependency(),
      shiny::textInput("input1", "Input 1"),
      shiny::textInput("input2", "Input 2"),
      shiny::actionButton("go_button", "Go!"),
      shiny::textOutput("focusing")
    ),
    server = function(input, output, session) {
      # Update the value in focusing whenever input1 has focus.
      on_focus(
        "input1",
        output$focusing <- shiny::renderText(
          paste(
            "You entered input1 at",
            Sys.time()
          )
        )
      )
    }
  )

  # App 2: With module.
  focusUI <- function(id) {
    shiny::tagList(
      shiny::textInput(shiny::NS(id, "observe_me"), "Observe me"),
      shiny::textOutput(NS(id, "focusing"))
    )
  }
  focusServer <- function(id) {
    shiny::moduleServer(id, function(input, output, session) {
      on_focus(
        "observe_me",
        output$focusing <- shiny::renderText(
          paste(
            "You entered observe_me at",
            Sys.time()
          )
        )
      )
```

```
  })
}

shiny::shinyApp(
  ui = shiny::fluidPage(
    shinyfocus_js_dependency(),
    focusUI("focus_module"),
    shiny::textInput("another_input", "Another input"),
    shiny::actionButton("go_button", "Go!")
  ),
  server = function(input, output, session) {
    focusServer("focus_module")
  }
)
}
```

---

on_focus_change *Respond when input changes focus*

Description

Set up a `shiny::observeEvent()` observer to trigger when the named input gains or loses focus.

Usage

```
on_focus_change(
  id,
  handler_expr,
  ...,
  change_on = c("focus", "blur"),
  priority = 99999,
  session = shiny::getDefaultReactiveDomain()
)
```

Arguments

`id`	The ID string of an input.
`handler_expr`	The expression to trigger whenever the specified input changes focus. This expression is quoted and executed in the calling environment.
`...`	Arguments passed on to `shiny::observeEvent` `label` A label for the observer or reactive, useful for debugging. `suspended` If `TRUE`, start the observer in a suspended state. If `FALSE` (the default), start in a non-suspended state. `autoDestroy` If `TRUE` (the default), the observer will be automatically destroyed when its domain (if any) ends. `ignoreNULL` Whether the action should be triggered (or value calculated, in the case of `eventReactive`) when the input event expression is `NULL`. See Details.

once Whether this observeEvent should be immediately destroyed after the first time that the code in handlerExpr is run. This pattern is useful when you want to subscribe to a event that should only happen once.

change_on A character indicating whether the observer should update when the input becomes focused and/or when the input becomes blurred.

priority An integer that controls the priority with which the observer should be executed. It often makes sense for this priority to be very high to avoid conflicts.

session The session (aka domain) in which the observer will be created and executed. The default is almost always desired.

Value

A shiny observer (see shiny::observe()).

See Also

Other observers: on_blur(), on_focus()

Examples

```
if (interactive()) {
  # App 1: A relatively simple ui without modules.
  shiny::shinyApp(
    ui = shiny::fluidPage(
      shinyfocus_js_dependency(),
      shiny::textInput("input1", "Input 1"),
      shiny::textInput("input2", "Input 2"),
      shiny::actionButton("go_button", "Go!"),
      shiny::textOutput("changing")
    ),
    server = function(input, output, session) {
      # Update the value in "changing" whenever input1 gains or loses focus.
      on_focus_change(
        "input1",
        output$changing <- shiny::renderText(
          paste(
            "You entered or left input1 at",
            Sys.time()
          )
        )
      )
    }
  )

  # App 2: With module.
  changeUI <- function(id) {
    shiny::tagList(
      shiny::textInput(shiny::NS(id, "observe_me"), "Observe me"),
      shiny::textOutput(NS(id, "changing"))
    )
  }
```

```
  changeServer <- function(id) {
    shiny::moduleServer(id, function(input, output, session) {
      on_focus_change(
        "observe_me",
        output$changing <- shiny::renderText(
          paste(
            "You entered or left observe_me at",
            Sys.time()
          )
        )
      )
    })
  }

  shiny::shinyApp(
    ui = shiny::fluidPage(
      shinyfocus_js_dependency(),
      changeUI("change_module"),
      shiny::textInput("another_input", "Another input"),
      shiny::actionButton("go_button", "Go!")
    ),
    server = function(input, output, session) {
      changeServer("change_module")
    }
  )
}
```

`shinyfocus_js_dependency` *Add the shinyfocus JavaScript to shiny*

Description

Add the `shinyfocus.js` script to a shiny app exactly once. Adding this script setups up the object used to detect changes in focus.

Usage

```
shinyfocus_js_dependency()
```

Value

An `htmltools::htmlDependency()`, which shiny uses to add the `shinyfocus.js` script exactly once.

Examples

```
shinyfocus_js_dependency()
```

Index