

Package: shinyTimer (via r-universe)

March 8, 2025

Title Customizable Timer for 'shiny' Applications

Version 0.1.0

Description Provides a customizable timer widget for 'shiny' applications. Key features include countdown and count-up mode, multiple display formats (including simple seconds, minutes-seconds, hours-minutes-seconds, and minutes-seconds-centiseconds), ability to pause, resume, and reset the timer. 'shinytimer' widget can be particularly useful for creating interactive and time-sensitive applications, tracking session times, setting time limits for tasks or quizzes, and more.

Depends R (>= 4.1.0)

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.1

Imports shiny, htmltools

Suggests R6, testthat (>= 3.0.0)

Config/pak/sysreqs make zlib1g-dev

Repository <https://maciekbanas.r-universe.dev>

RemoteUrl <https://github.com/maciekbanas/shinytimer>

RemoteRef HEAD

RemoteSha b30472684e1f02d13b8c06fb39bd86ef8861d1e9

Contents

countDown	2
countUp	3
pauseTimer	3
resetTimer	4
shinyTimer	5
updateShinyTimer	6

countDown	<i>Set shinyTimer in motion: count down</i>
-----------	---

Description

Set shinyTimer in motion: count down

Usage

```
countDown(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	The input ID corresponding to the UI element.
session	The session object from the shiny server function.

Value

No return value, called for side effects.

Examples

```
if (interactive()) {  
  library(shiny)  
  shinyApp(  
    ui = fluidPage(  
      shinyTimer("timer", label = "Countdown Timer", seconds = 20, type = "mm:ss"),  
      actionButton("start", "Start Countdown")  
    ),  
    server = function(input, output, session) {  
      observeEvent(input$start, {  
        countDown("timer")  
      })  
    }  
  )  
}
```

countUp	<i>Set shinyTimer in motion: count up</i>
---------	---

Description

Set shinyTimer in motion: count up

Usage

```
countUp(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	The input ID corresponding to the UI element.
session	The session object from the shiny server function.

Value

No return value, called for side effects.

Examples

```
if (interactive()) {  
  library(shiny)  
  shinyApp(  
    ui = fluidPage(  
      shinyTimer("timer", label = "Count Up Timer", seconds = 0, type = "mm:ss.cs"),  
      actionButton("start", "Start Counting Up")  
    ),  
    server = function(input, output, session) {  
      observeEvent(input$start, {  
        countUp("timer")  
      })  
    }  
  )  
}
```

pauseTimer	<i>Pause shinyTimer</i>
------------	-------------------------

Description

Pause shinyTimer

Usage

```
pauseTimer(inputId, session = shiny::getDefaultReactiveDomain())
```

Arguments

inputId	The input ID corresponding to the UI element.
session	The session object from the shiny server function.

Value

No return value, called for side effects.

Examples

```
if (interactive()) {  
  library(shiny)  
  shinyApp(  
    ui = fluidPage(  
      shinyTimer("timer", label = "Countdown Timer", seconds = 20, type = "mm:ss"),  
      actionButton("start", "Start Countdown"),  
      actionButton("pause", "Pause Countdown")  
    ),  
    server = function(input, output, session) {  
      observeEvent(input$start, {  
        countDown("timer")  
      })  
      observeEvent(input$pause, {  
        pauseTimer("timer")  
      })  
    }  
  )  
}
```

resetTimer

Reset shinyTimer

Description

Reset shinyTimer

Usage

```
resetTimer(  
  inputId,  
  hours = 0,  
  minutes = 0,  
  seconds = 0,  
  session = shiny::getDefaultReactiveDomain()  
)
```

Arguments

inputId	The input ID corresponding to the UI element.
hours	The new reset time in hours.
minutes	The new reset time in minutes.
seconds	The new reset time in seconds.
session	The session object from the shiny server function.

Value

No return value, called for side effects.

Examples

```
if (interactive()) {  
  library(shiny)  
  shinyApp(  
    ui = fluidPage(  
      shinyTimer("timer", label = "Countdown Timer", seconds = 20, type = "mm:ss"),  
      actionButton("reset", "Reset Timer")  
    ),  
    server = function(input, output, session) {  
      observeEvent(input$reset, {  
        resetTimer("timer", seconds = 20)  
      })  
    }  
  )  
}
```

shinyTimer

shinyTimer widget

Description

shinyTimer widget

Usage

```
shinyTimer(  
  inputId,  
  label = NULL,  
  hours = 0,  
  minutes = 0,  
  seconds = 0,  
  type = "simple",  
  background = "none",  
  ...  
)
```

Arguments

inputId	The input id.
label	The label to display above the countdown.
hours	An integer, the starting time in hours for the countdown.
minutes	An integer, the starting time in minutes for the countdown.
seconds	An integer, the starting time in seconds for the countdown.
type	The type of the countdown timer display ("simple", "mm:ss", "hh:mm:ss", "mm:ss.cs").
background	The shape of the timer's container ("none", "circle", "rectangle").
...	Any additional parameters you want to pass to the placeholder for the timer (htmltools::tags\$div).

Value

A shiny UI component for the countdown timer.

Examples

```
if (interactive()) {
  library(shiny)
  shinyApp(
    ui = fluidPage(
      shinyTimer("timer", label = "Countdown Timer", seconds = 10)
    ),
    server = function(input, output, session) {
      observeEvent(input$start, {
        countDown("timer", session)
      })
    }
  )
}
```

updateShinyTimer

Update shinyTimer widget

Description

Update shinyTimer widget

Usage

```
updateShinyTimer(
  inputId,
  hours = NULL,
  minutes = NULL,
  seconds = NULL,
  type = NULL,
```

```
    label = NULL,  
    background = NULL,  
    session = shiny::getDefaultReactiveDomain()  
  )
```

Arguments

inputId	The input ID corresponding to the UI element.
hours	The new starting time in hours for the countdown.
minutes	The new starting time in minutes for the countdown.
seconds	The new starting time in seconds for the countdown.
type	The new type of the countdown timer display ("simple", "mm:ss", "hh:mm:ss", "mm:ss.cs").
label	The new label to be displayed above the countdown timer.
background	The new shape of the timer's container ("none", "circle", "rectangle").
session	The session object from the shiny server function.

Value

No return value, called for side effects.

Examples

```
if (interactive()) {  
  library(shiny)  
  shinyApp(  
    ui = fluidPage(  
      shinyTimer("timer", label = "Countdown Timer", seconds = 10, type = "mm:ss"),  
      actionButton("update", "Update Timer")  
    ),  
    server = function(input, output, session) {  
      observeEvent(input$update, {  
        updateShinyTimer("timer", seconds = 20, type = "hh:mm:ss")  
      })  
    }  
  )  
}
```

Index

countDown, 2

countUp, 3

pauseTimer, 3

resetTimer, 4

shinyTimer, 5

updateShinyTimer, 6