

Package: EZRShiny (via r-universe)

October 11, 2026

Type Package

Title Build Consistent 'shiny' App Layouts with Less Code

Version 0.1.0

Description Short, consistent wrappers around 'shiny', 'bslib' and 'shinyjs' for building multi-page 'shiny' apps. One function call builds the page with a navigation bar, logo, title and dark mode switch. Tab functions nest pages up to three levels deep, with optional sidebars. Input functions create full-width inputs with optional information tooltips, and helpers switch buttons on and off and show or hide tabs from the server. Includes a starter app template and an example app that explores the Titanic passenger data.

License GPL-3

Encoding UTF-8

Language en-US

Depends R (>= 4.1.0)

Imports bsicons, bslib (>= 0.7.0), colourpicker, DT, ggiraph (>= 0.9.1), here, plotly (>= 4.12.0), shiny (>= 1.8.0), shinyjs, shinyWidgets, utils

Suggests datasets, ggplot2, htmltools, knitr, rmarkdown, testthat (>= 3.0.0), withr

VignetteBuilder knitr

Config/testthat/edition 3

Roxygen list(markdown = TRUE)

URL <https://github.com/vari-bbc/EZRShiny>

BugReports <https://github.com/vari-bbc/EZRShiny/issues>

Config/roxygen2/version 8.1.0

Config/pak/sysreqs libcairo2-dev cmake libfontconfig1-dev libfreetype6-dev make libicu-dev libpng-dev libuv1-dev libssl-dev zlib1g-dev

Repository <https://vari-bbc.r-universe.dev>

Date/Publication 2026-09-30 17:16:59 UTC
RemoteUrl https://github.com/vari-bbc/ezrshiny
RemoteRef HEAD
RemoteSha 28549b0127ab886addff3cfde1a0b6acc484ccce

Contents

addTooltip	2
createEZApp	3
deactivateItems	4
hideNavTabs	5
navInputs	6
navOutputs	8
navPieces	9
navSpanText	10
runExampleApp	10
sourceFunctions	11
startSection	12
subTabs	13
topTabs	14
UINav	16
Index	18

addTooltip	<i>Add an information tooltip to a label</i>
------------	--

Description

Puts an info icon after the label. Hovering over the label or the icon shows tooltipText. Every nav*() input and output function calls this for you through its tooltipText argument. Call it directly only when building your own inputs.

Usage

addTooltip(label, tooltipText)

Arguments

label	The label: a string or a 'shiny' tag.
tooltipText	Text to show on hover. NA or NULL means no tooltip, and the label is returned unchanged.

Value

The label, with a tooltip if one was given.

Examples

```
addTooltip("Upload data", "CSV files only")
addTooltip("Upload data", NA)
```

createEZApp

Start a new app from the EZRShiny template

Description

Creates a folder holding a ready-to-edit app laid out the standard way:

Usage

```
createEZApp(path, appName = "My App", overwrite = FALSE)
```

Arguments

path	Folder to create the app in. It is created if needed.
appName	Name shown in the app's navigation bar.
overwrite	TRUE to replace an existing app.R in path.

Details

- app.R: the app, with numbered sections for loading packages, the UI and the server, and comments explaining what goes where.
- Functions/: put your helper .R files here. app.R loads them all with `sourceFunctions()`.
- www/: put images such as logos here. Give just the file name to `UINav(logoFile = ...)`.
- Necessary_Files/: put files the app needs at start, such as a data template users can download.

Open app.R and run it with `shiny::runApp("<path>")` or the Run App button.

Value

The path to the new app.R, invisibly.

Examples

```
appFolder <- file.path(tempdir(), "myFirstApp")
createEZApp(appFolder, appName = "My First App")
list.files(appFolder)

if (interactive()) {
  shiny::runApp(appFolder)
}
```

deactivateItems	<i>Switch inputs on and off</i>
-----------------	---------------------------------

Description

Greys out inputs so they can't be used, or makes them usable again. Use this in the server to stop users clicking a button or download before the app is ready for it, for example deactivating every download at start and activating each one once there is something to download.

Usage

```
deactivateItems(itemIDs)
```

```
activateItems(itemIDs)
```

Arguments

`itemIDs` IDs of the inputs to switch, as a character vector.

Details

Needs `shinyjs::useShinyjs()` in the UI, which `UINav()` adds for you.

Value

NULL, invisibly. Called for its effect on the page.

Examples

```
if (interactive()) {
  shiny::shinyApp(
    ui = UINav(
      appName = "Demo",
      singleTab("Home",
        navCheckbox("ready", "Ready to download?"),
        navDownload("theDownload", "Download")
      )
    ),
    server = function(input, output, session) {
      shiny::observe({
        if (isTRUE(input$ready)) {
          activateItems("theDownload")
        } else {
          deactivateItems("theDownload")
        }
      })
    }
  )
}
```

hideNavTabs

*Hide and show tabs***Description**

Hides tabs until they are useful, for example the results tabs before any data has been uploaded, and shows them again from the server.

Usage

```
hideNavTabs(rootID, tabIDs)
```

```
showNavTabs(rootID, tabIDs)
```

Arguments

rootID ID of the row of tabs.

tabIDs IDs of the tabs to hide or show, as a character vector.

Details

rootID is the id of the `biLevelTab()`, `triSubTab()` or `triSubSidebarTab()` holding the tabs, or "root" for the top-level tabs in the navigation bar. tabIDs are the tabs to hide or show:

- Sub tabs: their values. These default to the title with spaces removed, and spaces are removed here too, so the titles themselves work.
- Top-level tabs (rootID = "root"): their titles, exactly as written, spaces included.

showNavTabs() also selects the first tab in tabIDs, briefly selecting the second first, so the tab's contents are drawn straight away.

Value

NULL, invisibly. Called for its effect on the page.

Examples

```
if (interactive()) {
  shiny::shinyApp(
    ui = UINav(
      appName = "Demo",
      biLevelTab("Results",
        subTab("Start", navButton("go", "Show the results")),
        subTab("Table", navOutputText("tableText")),
        subTab("Plot", navOutputText("plotText"))
      )
    ),
    server = function(input, output, session) {
      hideNavTabs("Results", c("Table", "Plot"))
    }
  )
}
```

```

    shiny::observeEvent(input$go, {
      showNavTabs("Results", c("Table", "Plot"))
    })
  }
)
}

```

navInputs

Standard inputs

Description

Full-width inputs for sidebars and tabs. Each is a thin wrapper around a 'shiny', 'bslib', 'shinyWidgets' or 'colourpicker' input, uses the same argument names, and takes an optional `tooltipText` that adds an info icon to the label (see [addTooltip\(\)](#)).

Usage

```
navButton(inputId, label, tooltipText = NA)
```

```

navSelect(
  inputId,
  label,
  choices = NULL,
  selected = choices[1],
  multiple = FALSE,
  create = FALSE,
  tooltipText = NA
)

```

```
navUpload(inputId, label, multiple = FALSE, tooltipText = NA)
```

```
navCheckbox(inputId, label, value = FALSE, tooltipText = NA)
```

```
navDownload(inputId, label, tooltipText = NA)
```

```
navText(inputId, label, value = "", tooltipText = NA)
```

```
navNumeric(inputId, label, value = 1, min = NA, max = NA, tooltipText = NA)
```

```
navColor(inputId, label, value = "white", tooltipText = NA)
```

```
navDate(inputId, label, range = FALSE, tooltipText = NA)
```

Arguments

`inputId` The input's ID. Read it in the server as `input$<inputId>`.

label	Label shown above or beside the input.
tooltipText	Optional text for an info tooltip on the label.
choices	Choices to offer. Can be left NULL and filled in later from the server with <code>shiny::updateSelectizeInput()</code> .
selected	Choices selected at start. Defaults to the first choice.
multiple	TRUE to allow more than one choice (<code>navSelect()</code>) or file (<code>navUpload()</code>).
create	TRUE to let the user type in options that aren't in choices.
value	Starting value: text for <code>navText()</code> , a number for <code>navNumeric()</code> , TRUE/FALSE for whether <code>navCheckbox()</code> starts on, and a color name or hex code for <code>navColor()</code> .
min, max	Smallest and largest allowed values. NA for no limit.
range	TRUE to pick a start and end date, FALSE for one date.

Details

Function	Builds	Read in the server as
<code>navButton()</code>	<code>bslib::input_task_button()</code>	<code>input\$id</code> (click count)
<code>navSelect()</code>	<code>shiny::selectizeInput()</code>	<code>input\$id</code> (chosen values)
<code>navUpload()</code>	<code>shiny::fileInput()</code>	<code>input\$id\$dataPath</code> , <code>input\$id\$name</code>
<code>navDownload()</code>	<code>shiny::downloadButton()</code>	pair with <code>output\$id <- downloadHandler(...)</code>
<code>navCheckbox()</code>	<code>bslib::input_switch()</code>	<code>input\$id</code> (TRUE/FALSE)
<code>navText()</code>	<code>shiny::textInput()</code>	<code>input\$id</code>
<code>navNumeric()</code>	<code>shiny::numericInput()</code>	<code>input\$id</code>
<code>navColor()</code>	<code>colourpicker::colourInput()</code>	<code>input\$id</code> (hex color)
<code>navDate()</code>	<code>shinyWidgets::airDatepickerInput()</code>	<code>input\$id</code> (one or two dates)

Value

A 'shiny' tag (or tag list) to place in the UI.

Examples

```
navButton("runData", "Run analysis", tooltipText = "Upload data first")
navSelect("groups", "Pick groups", choices = c("Control", "Treated"),
         multiple = TRUE)
navUpload("dataUpload", "Upload a CSV")
navDownload("resultsDownload", "Download results")
navCheckbox("logScale", "Use a log scale")
navText("plotTitle", "Plot title")
navNumeric("alpha", "Significance cutoff", value = 0.05, min = 0, max = 1)
navColor("pointColor", "Point color", value = "#D55E00")
navDate("dates", "Date range", range = TRUE)
```

navOutputs

*Standard outputs***Description**

Placeholders in the UI that the server fills in. Each is paired with a render function in the server, assigned to `output$<outputId>`:

Usage

```
navOutputTable(outputId, label = NULL, tooltipText = NA)
```

```
navOutputPlot(outputId, label = NULL, tooltipText = NA)
```

```
navOutputPlotly(outputId, label = NULL, tooltipText = NA)
```

```
navOutputGirafe(outputId, label = NULL, tooltipText = NA)
```

```
navOutputPic(outputId, label = NULL, tooltipText = NA)
```

```
sideNavOutputPic(outputId, label = NULL, tooltipText = NA)
```

```
navOutputText(outputId, label = NULL, tooltipText = NA)
```

Arguments

<code>outputId</code>	The output's ID. Fill it in the server with <code>output\$<outputId> <- render...()</code> .
<code>label</code>	Optional heading shown above the output.
<code>tooltipText</code>	Optional text for an info tooltip after the label.

Details

Function	Fill it in the server with	Size
<code>navOutputTable()</code>	<code>DT::renderDT()</code>	full width
<code>navOutputPlot()</code>	<code>shiny::renderPlot()</code>	80% wide
<code>navOutputPlotly()</code>	<code>plotly::renderPlotly()</code>	80% wide, full window height
<code>navOutputGirafe()</code>	<code>ggiraph::renderGirafe()</code>	80% wide, full window height
<code>navOutputPic()</code>	<code>shiny::renderImage()</code>	80% of the window wide, full window height
<code>sideNavOutputPic()</code>	<code>shiny::renderImage()</code>	40% wide, for beside another output
<code>navOutputText()</code>	<code>shiny::renderText()</code>	as needed

Like the inputs, each output can have a `label` shown above it and a `tooltipText` that adds an info icon after the label (or on its own, if there's no label). Use the tooltip to explain how to read a plot or table.

Value

A 'shiny' output tag to place in the UI, with its label if one was given.

Examples

```
navOutputTable("resultsTable")
navOutputPlotly("pcaPlot", label = "PCA",
  tooltipText = "Each point is one sample.")
navOutputText("statusMessage")
```

navPieces

Building blocks of the navigation bar

Description

`UINav()` puts these together for you. They're exported so you can build a custom page from the same pieces.

Usage

```
navPadding()

navBar(..., barColor = NULL)

navDarkSwitch(mode = "dark")

navItem(...)
```

Arguments

<code>...</code>	Contents: tabs for <code>navBar()</code> , anything for <code>navItem()</code> .
<code>barColor</code>	Background color of the bar. NULL uses the theme's default.
<code>mode</code>	Starting mode for the switch: "dark" or "light".

Details

- `navPadding()`: a small spacer above the navigation bar.
- `navBar()`: the navigation bar itself, with the ID "root".
- `navDarkSwitch()`: the light/dark mode switch.
- `navItem()`: wraps anything (an image, a heading, a link) so it can sit in the navigation bar or a tab strip without being a tab.

Value

A 'shiny' tag or 'bslib' navigation object.

Examples

```
navItem(shiny::strong("Version 1.0"))
navDarkSwitch("light")
```

navSpanText	<i>Text with an information tooltip</i>
-------------	---

Description

Plain text followed by an info icon. Hovering over the icon shows `tooltipText` below it. Useful for explaining a section of a page.

Usage

```
navSpanText(text, tooltipText = NA)
```

Arguments

<code>text</code>	Text to show.
<code>tooltipText</code>	Text to show when hovering over the icon.

Value

A 'shiny' span tag.

Examples

```
navSpanText("Quality control", "Samples with over 20% missing values are removed.")
```

runExampleApp	<i>Run the example Titanic app</i>
---------------	------------------------------------

Description

Opens "Titanic Explorer", a complete example app built with EZRShiny that explores who survived the Titanic, using R's built-in [datasets::Titanic](#) data. It uses every kind of tab (a single page, a page with a sidebar, a row of sub tabs and a drop-down menu of tabs), hides tabs until they are useful, switches buttons on and off, and follows the standard folder layout:

Usage

```
runExampleApp(...)
```

Arguments

<code>...</code>	Passed to shiny::runApp() , for example <code>launch.browser = TRUE</code> .
------------------	--

Details

- app.R: packages, UI and server.
- Functions/: helper functions, loaded with `sourceFunctions()`.
- Necessary_Files/: the Titanic data as a CSV, one row per person.
- www/: the logo shown in the navigation bar.

Its code is a good starting point for your own app. Find it with `system.file("examples", "titanicExplorer", package = "EZRShiny")`.

Needs the 'ggplot2' package.

Value

Does not return while the app is running.

Examples

```
# Where the example app's files are:
list.files(system.file("examples", "titanicExplorer", package = "EZRShiny"),
           recursive = TRUE)

if (interactive()) {
  runExampleApp()
}
```

sourceFunctions	<i>Source every R file in a folder</i>
-----------------	--

Description

Loads all of an app's helper functions in one call, so app.R doesn't need a `source()` line per file. Every file ending in `.R` in the folder is sourced, in alphabetical order. Subfolders are not searched.

Usage

```
sourceFunctions(functionFolderPath, envir = parent.frame())
```

Arguments

functionFolderPath	Path to the folder holding the <code>.R</code> files, for example <code>"Functions"</code> .
envir	Environment to load the functions into. Defaults to the one <code>sourceFunctions()</code> is called from, which in app.R is the global environment.

Details

A relative path is looked up from the working directory first. If it isn't found there, it is looked up from the project root with `here::here()`.

Value

The paths of the sourced files, invisibly.

Examples

```
folder <- file.path(tempdir(), "Functions")
dir.create(folder, showWarnings = FALSE)
writeLines("addOne <- function(x) x + 1", file.path(folder, "addOne.R"))

sourceFunctions(folder)
addOne(1)
```

startSection

Mark the start and end of a section of server code in the log

Description

Writes a short marker to the console so the app's log shows which part of the server is running. Put `startSection()` at the top of an `observeEvent()` and `endSection()` at the bottom, using the same name. Markers are sent as messages, so `suppressMessages()` hides them.

Usage

```
startSection(sectionName)

endSection(sectionName)
```

Arguments

`sectionName` Name of the section, shown in the log.

Value

NULL, invisibly. Called for the message it writes.

Examples

```
startSection("Read in data")
endSection("Read in data")
```

subTabs	<i>Sub tabs</i>
---------	-----------------

Description

Tabs that go inside a `biLevelTab()` or a `triLevelTab()`:

Usage

```
triSubTab(title, ..., icon = NULL, id = title, height = cardHeight())

subTab(title, ..., value = title)

subSidebarTab(title, sidebarElements = NULL, ..., value = title, icon = NULL)

triSubSidebarTab(
  title,
  sidebarElements = NULL,
  ...,
  icon = NULL,
  id = title,
  height = cardHeight()
)

subTwoColPage(leftSide, rightSide)
```

Arguments

title	Tab name shown in the tab strip or menu.
...	Contents. See the table above.
icon	Optional icon shown before the title.
id	ID of the row of sub tabs inside the tab. Defaults to the title with spaces removed.
height	Height of the card holding the tab's contents.
value	ID of the tab, used with <code>showNavTabs()</code> , <code>hideNavTabs()</code> and <code>bslib::nav_select()</code> . Defaults to the title with spaces removed.
sidebarElements	A <code>list()</code> of inputs for the sidebar.
leftSide, rightSide	Contents of the left and right columns. Wrap several items in <code>shiny::tagList()</code> .

Details

Function	Goes inside	Holds
<code>subTab()</code>	<code>biLevelTab()</code> , <code>triSubTab()</code> , <code>triSubSidebarTab()</code>	inputs and outputs

subSidebarTab()	biLevelTab(), triSubTab(), triSubSidebarTab()	a sidebar plus inputs and outputs
triSubTab()	triLevelTab()	subTab(), subSidebarTab()
triSubSidebarTab()	triLevelTab()	a sidebar plus subTab(), subSidebarTab()

subTwoColPage() isn't a tab: it splits a page into two equal columns and can go inside any tab.

Value

A 'bslib' navigation panel, or for subTwoColPage() a column layout.

Examples

```
biLevelTab("Explore",
  subSidebarTab("PCA",
    sidebarElements = list(navButton("runPCA", "Create PCA")),
    navOutputPlotly("pcaPlot")
  ),
  subTab("Table",
    subTwoColPage(navOutputTable("tableLeft"), navOutputTable("tableRight"))
  )
)
```

topTabs	<i>Top-level tabs</i>
---------	-----------------------

Description

Tabs that go directly inside `UINav()`. Pick one by how much the tab holds:

Usage

```
triLevelTab(title, ..., icon = NULL)

biLevelTab(title, ..., icon = NULL, id = title, height = cardHeight())

sidebarLevelTab(
  title,
  sidebarElements = NULL,
  ...,
  icon = NULL,
  height = cardHeight()
)

singleTab(title, ..., icon = NULL, height = cardHeight())
```

Arguments

title	Tab name shown in the navigation bar.
...	Contents. See the table above.
icon	Optional icon shown before the title, for example <code>bsicons::bs_icon("table")</code> .
id	ID of the row of sub tabs, used with showNavTabs() and hideNavTabs() . Defaults to the title with spaces removed.
height	Height of the card holding the tab's contents. Change the default for every tab with <code>options(EZRShiny.cardHeight = "70vh")</code> .
sidebarElements	A <code>list()</code> of inputs for the sidebar.

Details

Function	Holds	Put inside it
<code>singleTab()</code>	one page	inputs and outputs
<code>sidebarLevelTab()</code>	one page with a sidebar	inputs and outputs
<code>biLevelTab()</code>	a row of sub tabs	subTab() , subSidebarTab()
<code>triLevelTab()</code>	a drop-down menu of tabs with sub tabs	triSubTab() , triSubSidebarTab()

Value

A 'bslib' navigation panel or menu.

Examples

```
singleTab("About", shiny::p("This app explores the Titanic data.))

sidebarLevelTab("Plot",
  sidebarElements = list(navButton("makePlot", "Make plot")),
  navOutputPlot("thePlot")
)

biLevelTab("Explore",
  subTab("Summary", navOutputTable("summaryTable")),
  subTab("Plots", navOutputPlot("summaryPlot"))
)

triLevelTab("Analyses",
  triSubTab("Group A",
    subTab("Table", navOutputTable("tableA"))
  ),
  triSubTab("Group B",
    subTab("Table", navOutputTable("tableB"))
  )
)
```

Description

The outermost UI function: use it as `ui <- UINav(...)`. It builds a page with a navigation bar across the top holding, from left to right, any logos, the app name, one entry per tab you pass in, and a dark mode switch on the far right.

Usage

```
UINav(
  ...,
  logoFile = NULL,
  appName = NULL,
  logoHeight = "40vh",
  barColor = NULL,
  theme = bslib::bs_theme()
)
```

Arguments

<code>...</code>	Tabs to show in the navigation bar.
<code>logoFile</code>	File names of images to show at the left of the navigation bar, in order. Put the images in the app's <code>www</code> folder and give just the file name, for example <code>"logo.png"</code> (png, svg, jpg or gif). <code>NULL</code> (the default) or <code>""</code> shows no logo.
<code>appName</code>	App name shown after the logos. If left <code>NULL</code> , a global variable called <code>appName</code> is used, or no name if there isn't one.
<code>logoHeight</code>	Height of each logo, as a CSS size. A single value is used for every logo.
<code>barColor</code>	Background color of the navigation bar, as a color name or hex code, for example <code>"#005596"</code> . <code>NULL</code> uses the theme's default. Text color switches between light and dark to stay readable.
<code>theme</code>	A <code>bslib::bs_theme()</code> to style the whole app, for example <code>bslib::bs_theme(primary = "#005596", preset = "cosmo")</code> .

Details

Pass tabs made with `singleTab()`, `sidebarLevelTab()`, `biLevelTab()` or `triLevelTab()`. The navigation bar has the ID `"root"`, so the server can change tab with `bslib::nav_select("root", "<tab title>")`.

Value

A 'shiny' page to use as the app's UI.

Examples

```
ui <- UINav(  
  appName = "My App",  
  barColor = "#005596",  
  singleTab("Upload",  
    navUpload("dataUpload", "Upload a CSV")  
  ),  
  singleTab("Results",  
    navOutputTable("resultsTable")  
  )  
)  
  
if (interactive()) {  
  shiny::shinyApp(ui, function(input, output, session) {})  
}
```

Index

activateItems (deactivateItems), 4
addTooltip, 2
addTooltip(), 6

biLevelTab (topTabs), 14
biLevelTab(), 5, 13, 16
bslib::bs_theme(), 16
bslib::input_switch(), 7
bslib::input_task_button(), 7
bslib::nav_select(), 13

colourpicker::colourInput(), 7
createEZApp, 3

datasets::Titanic, 10
deactivateItems, 4
DT::renderDT(), 8

endSection (startSection), 12

ggiraph::renderGirafe(), 8

here::here(), 11
hideNavTabs, 5
hideNavTabs(), 13, 15

navBar (navPieces), 9
navButton (navInputs), 6
navCheckbox (navInputs), 6
navColor (navInputs), 6
navDarkSwitch (navPieces), 9
navDate (navInputs), 6
navDownload (navInputs), 6
navInputs, 6
navItem (navPieces), 9
navNumeric (navInputs), 6
navOutputGirafe (navOutputs), 8
navOutputPic (navOutputs), 8
navOutputPlot (navOutputs), 8
navOutputPlotly (navOutputs), 8
navOutputs, 8

navOutputTable (navOutputs), 8
navOutputText (navOutputs), 8
navPadding (navPieces), 9
navPieces, 9
navSelect (navInputs), 6
navSpanText, 10
navText (navInputs), 6
navUpload (navInputs), 6

plotly::renderPlotly(), 8

runExampleApp, 10

shiny::downloadButton(), 7
shiny::fileInput(), 7
shiny::numericInput(), 7
shiny::renderImage(), 8
shiny::renderPlot(), 8
shiny::renderText(), 8
shiny::runApp(), 10
shiny::selectizeInput(), 7
shiny::textInput(), 7
shiny::updateSelectizeInput(), 7
shinyWidgets::airDatepickerInput(), 7
showNavTabs (hideNavTabs), 5
showNavTabs(), 13, 15
sidebarLevelTab (topTabs), 14
sidebarLevelTab(), 16
sideNavOutputPic (navOutputs), 8
singleTab (topTabs), 14
singleTab(), 16
sourceFunctions, 11
sourceFunctions(), 3, 11
startSection, 12
subSidebarTab (subTabs), 13
subSidebarTab(), 15
subTab (subTabs), 13
subTab(), 15
subTabs, 13
subTwoColPage (subTabs), 13

topTabs, [14](#)
triLevelTab (topTabs), [14](#)
triLevelTab(), [13](#), [16](#)
triSubSidebarTab (subTabs), [13](#)
triSubSidebarTab(), [5](#), [15](#)
triSubTab (subTabs), [13](#)
triSubTab(), [5](#), [15](#)

UINav, [16](#)
UINav(), [4](#), [9](#), [14](#)