Package ‘conductor’

May 8, 2026

Type Package

Title Create Tours in 'Shiny' Apps Using 'Shepherd.js'

Version 0.1.2

Description Enable the use of 'Shepherd.js' to create tours in 'Shiny' applications.

License MIT + file LICENSE

Encoding UTF-8

Imports htmltools, R6, shiny

Suggests altdoc, shinytest2, testthat

RoxygenNote 7.2.1

URL https://github.com/etiennebacher/conductor

BugReports https://github.com/etiennebacher/conductor/issues

Config/testthat/edition 3

NeedsCompilation no

Author Etienne Bacher [aut, cre]

Maintainer Etienne Bacher <etienne.bacher@protonmail.com>

Repository CRAN

Date/Publication 2026-03-16 12:10:02 UTC

Contents

Conductor *Create a "conductor" tour*

Description

In addition to this page, you can also directly access the documentation of shepherd.js here: https://docs.shepherdjs.dev.

Methods

Public methods:

- Conductor$new()
- Conductor$init()
- Conductor$start()
- Conductor$step()
- Conductor$updateStepOptions()
- Conductor$show()
- Conductor$remove()
- Conductor$moveNext()
- Conductor$moveBack()
- Conductor$cancel()
- Conductor$complete()
- Conductor$hide()
- Conductor$getCurrentStep()
- Conductor$getHighlightedElement()
- Conductor$isOpen()
- Conductor$isActive()
- Conductor$clone()

Method new():

Usage:

```
Conductor$new(
  exitOnEsc = TRUE,
  keyboardNavigation = TRUE,
  useModalOverlay = TRUE,
  classPrefix = NULL,
  tourName = NULL,
  stepsContainer = NULL,
  modalContainer = NULL,
  confirmCancel = FALSE,
  confirmCancelMessage = NULL,
  defaultStepOptions = NULL,
  mathjax = FALSE,
  progress = FALSE,
```

```
  onComplete = NULL,
  onCancel = NULL,
  onHide = NULL,
  onShow = NULL,
  onStart = NULL,
  onActive = NULL,
  onInactive = NULL
)
```

Arguments:

`exitOnEsc` Allow closing the tour by pressing "Escape". Default is `TRUE`.

`keyboardNavigation` Allow navigating the tour with keyboard arrows. Default is `TRUE`.

`useModalOverlay` Highlight the tour popover and the element (if specified). Default is `TRUE`.

`classPrefix` Add a prefix to the classes of the tour. This allows having different CSS for each tour.

`tourName` An (optional) name to give to the tour.

`stepsContainer` An optional container element for the steps. If `NULL` (default), the steps will be appended to `document.body`.

`modalContainer` An optional container element for the modal. If `NULL` (default), the modal will be appended to `document.body`.

`confirmCancel` Ask confirmation to cancel the tour. Default is `FALSE`.

`confirmCancelMessage` Message in the popup that ask confirmation to close the tour (works only if `confirmCancel = TRUE`).

`defaultStepOptions` A nested list of options to apply to the entire tour. See `Details`.

`mathjax` Enable MathJax? Default is `FALSE`. This requires importing MathJax, for example with `shiny::withMathJax()`.

`progress` Show a step counter in each step? Default is `FALSE`.

`onComplete` A JavaScript code to run when the tour is completed.

`onCancel` A JavaScript code to run when the tour is cancelled.

`onHide` A JavaScript code to run when the tour is hidden.

`onShow` A JavaScript code to run when the tour is shown.

`onStart` A JavaScript code to run when the tour starts.

`onActive` A JavaScript code to run when the tour is active.

`onInactive` A JavaScript code to run when the tour is inactive.

Details: Create a new `Conductor` object.

Returns: A `Conductor` object.

Method `init()`**:**

Usage:

```
Conductor$init(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Initialise `Conductor`.

Method `start()`**:**

Usage:

```
Conductor$start(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Start `Conductor`.

Method `step()`**:**

Usage:

```
Conductor$step(
  title = NULL,
  text = NULL,
  el = NULL,
  position = NULL,
  arrow = TRUE,
  tabId = NULL,
  tab = NULL,
  canClickTarget = TRUE,
  advanceOn = NULL,
  scrollTo = NULL,
  cancelIcon = NULL,
  showOn = NULL,
  id = NULL,
  buttons = NULL,
  classes = NULL,
  highlightClass = NULL,
  onComplete = NULL,
  onCancel = NULL,
  onHide = NULL,
  onShow = NULL
)
```

Arguments:

`title` Title of the popover.

`text` Text of the popover.

`el` The element to highlight. It can be an id (for example `#mynav`), a class (for instance `.navbar`), or a general tag (for example `button`). If `NULL` (default) or if the selector is not found, the popover will appear in the center of the page.

`position` Position of the popover relative to the element. Possible values are: 'auto', 'auto-start', 'auto-end', 'top', 'top-start', 'top-end', 'bottom', 'bottom-start', 'bottom-end', 'right', 'right-start', 'right-end', 'left', 'left-start', 'left-end'.

`arrow` Add an arrow pointing towards the highlighted element. Default is `TRUE`.

`tabId` Id of the `tabsetPanel()`.

`tab` Name of the tab that contains the element.

`canClickTarget` Allow the highlighted element to be clicked. Default is `TRUE`.

`advanceOn` An action on the page which should advance the tour to the next step. It should be a list with a string selector and an event name.

`scrollTo` Should the element be scrolled to when this step is shown? Default is `TRUE`.

`cancelIcon` A list of two elements: `enabled` is a boolean indicating whether a "close" icon should be displayed (default is `TRUE`); `label` is the label to add for `aria-label`.

`showOn` Either a boolean or a JavaScript expression that returns `true` or `false`. It indicates whether the step should be displayed in the tour.

`id` Name of the step (optional).

`buttons` A list of lists. Each "sublist" contains the information for one button. There are six possible arguments for each button: action ("back" or "next"), text (name of the button), secondary (`TRUE`/`FALSE`), disabled (`TRUE`/`FALSE`), label (aria-label of the button), and classes (for finer CSS customization).

`classes` A character vector of extra classes to add to the step's content element.

`highlightClass` An extra class to apply to `el` when it is highlighted. Only one extra class is accepted.

`onComplete` Some JavaScript code to run when the step is complete (only for the last step).

`onCancel` Some JavaScript code to run when the step is cancelled.

`onHide` Some JavaScript code to run when the step is hidden.

`onShow` Some JavaScript code to run when the step is shown.

Details: Add a step in a `Conductor` tour.

Method `updateStepOptions()`**:**

Usage:

```
Conductor$updateStepOptions(
  step = NULL,
  title = NULL,
  text = NULL,
  el = NULL,
  position = NULL,
  arrow = TRUE,
  tabId = NULL,
  tab = NULL,
  canClickTarget = TRUE,
  advanceOn = NULL,
  scrollTo = TRUE,
  cancelIcon = NULL,
  showOn = NULL,
  id = NULL,
  buttons = NULL,
  classes = NULL,
  highlightClass = NULL,
  session = NULL
)
```

Arguments:

`step` Id of the step (optional). If `NULL` (default), the current step is used.

`title` Title of the popover.

`text` Text of the popover.

`el` The element to highlight. It can be an id (for example `#mynav`), a class (for instance `.navbar`), or a general tag (for example `button`). If `NULL` (default) or if the selector is not found, the popover will appear in the center of the page.

`position` Position of the popover relative to the element. Possible values are: 'auto', 'auto-start', 'auto-end', 'top', 'top-start', 'top-end', 'bottom', 'bottom-start', 'bottom-end', 'right', 'right-start', 'right-end', 'left', 'left-start', 'left-end'.

`arrow` Add an arrow pointing towards the highlighted element. Default is `TRUE`.

`tabId` Id of the `tabsetPanel()`.

`tab` Name of the tab that contains the element.

`canClickTarget` Allow the highlighted element to be clicked. Default is `TRUE`.

`advanceOn` An action on the page which should advance shepherd to the next step. It should be a list with a string selector and an event name.

`scrollTo` Should the element be scrolled to when this step is shown? Default is `TRUE`.

`cancelIcon` A list of two elements: `enabled` is a boolean indicating whether a "close" icon should be displayed (default is `TRUE`); `label` is the label to add for `aria-label`.

`showOn` Either a boolean or a JavaScript expression that returns `true` or `false`. It indicates whether the step should be displayed in the tour.

`id` Name of the step (optional).

`buttons` A list of lists. Each "sublist" contains the information for one button. There are six possible arguments for each button: action ("back" or "next"), text (name of the button), secondary (`TRUE`/`FALSE`), disabled (`TRUE`/`FALSE`), label (aria-label of the button), and classes (for finer CSS customization).

`classes` A character vector of extra classes to add to the step's content element.

`highlightClass` An extra class to apply to `el` when it is highlighted. Only one extra class is accepted.

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

`onShow` Some JavaScript code to run when the step is shown.

`onHide` Some JavaScript code to run when the step is hidden.

`onCancel` Some JavaScript code to run when the step is cancelled.

`onComplete` Some JavaScript code to run when the step is complete (only for the last step).

Details: Modify the options of a specific step.

Method `show()`**:**

Usage:

```
Conductor$show(step = NULL, session = NULL)
```

Arguments:

`step` Either the id of the step to show (defined in `$step()`) or its number.

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Show a specific step.

Method `remove()`**:**

Usage:

```
Conductor$remove(step = NULL, session = NULL)
```

Arguments:

`step` A character vector with the id(s) of the step(s) to remove (defined in `$step()`).

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Remove specific step(s).

Method `moveNext()`**:**

Usage:

```
Conductor$moveNext(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Advances the tour to the next step.

Method `moveBack()`**:**

Usage:

```
Conductor$moveBack(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Shows the previous step.

Method `cancel()`**:**

Usage:

```
Conductor$cancel(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Cancels the tour.

Method `complete()`**:**

Usage:

```
Conductor$complete(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Completes the tour.

Method `hide()`**:**

Usage:

```
Conductor$hide(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Hides the current step.

Method `getCurrentStep()`**:**

Usage:

```
Conductor$getCurrentStep(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Get the id of the current step. If no `id` was specified in `$step()`, a random id is generated.

Method `getHighlightedElement()`**:**

Usage:

```
Conductor$getHighlightedElement(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Returns the id of the highlighted element of the current step. If this element has no id, it returns its class.

Method `isOpen()`**:**

Usage:

```
Conductor$isOpen(step = NULL, session = NULL)
```

Arguments:

`step` Id of the step (optional). If `NULL` (default), the current step is used.

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Returns a value `TRUE` or `FALSE` indicating whether the step is open.

Method `isActive()`**:**

Usage:

```
Conductor$isActive(session = NULL)
```

Arguments:

`session` A valid Shiny session. If `NULL` (default), the function attempts to get the session with `shiny::getDefaultReactiveDomain()`.

Details: Returns a value `TRUE` or `FALSE` indicating whether the tour is active.

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
Conductor$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

useConductor *Dependencies*

Description

Include dependencies, place anywhere in the shiny UI.

Usage

```
useConductor()

use_conductor()
```

Examples

```
library(shiny)
library(conductor)

ui <- fluidPage(
 useConductor()
 # also works:
 # use_conductor()
)

server <- function(input, output){}

if(interactive()) shinyApp(ui, server)
```

Index