Package 'howler'

July 21, 2022

Type Package

Title 'Shiny' Extension of 'howler.js'

Version 0.2.0

Description Audio interactivity within 'shiny' applications using 'howler.js'. Enables the status of the audio player to be sent from the UI to the server, and events such as playing and pausing the audio can be triggered from the server.

License MIT + file LICENSE

URL https://github.com/ashbaldry/howler, https://github.com/goldfire/howler.js

BugReports https://github.com/ashbaldry/howler/issues

Encoding UTF-8

Imports shiny, htmlwidgets

Suggests rmarkdown, knitr, shinytest2, globals, testthat (>= 3.0.0)

Language en-GB

RoxygenNote 7.2.0

VignetteBuilder knitr

NeedsCompilation no

Author Ashley Baldry [aut, cre], James Simpson [aut] (Creator of howler.js)

Maintainer Ashley Baldry <arbaldry91@gmail.com>

Repository CRAN

Date/Publication 2022-07-21 17:50:05 UTC

R topics documented:

`howler` *Create a Howler Audio Player*

Description

`howler` is used to initialise the 'howler.js' framework by adding all of the specified tracks to the player, and can be run by either including UI buttons or server-side actions.

Usage

```
howler(
  tracks,
  options = list(),
  track_formats = NULL,
  auto_continue = FALSE,
  auto_loop = FALSE,
  seek_ping_rate = 1000,
  elementId = NULL
)
```

Arguments

`tracks`	A named vector of file paths to sounds. If multiple file extensions are included, then use a named list instead, with each list item containing each extension of the sound.
`options`	A named list of options to add to the player. For a full list of options see https://github.com/goldfire/howler.js
`track_formats`	An optional list of formats of the sounds. By default 'howler' will guess the format to play in. Must be the same length as tracks
`auto_continue`	If there are multiple files, would you like to auto play the next file after the current one has finished? Defaults to `TRUE`
`auto_loop`	Once all files have been played, would you like to restart playing the playlist? Defaults to `FALSE`
`seek_ping_rate`	Number of milliseconds between each update of 'input${id}_seek' while playing. Default is set to 1000. If set to 0, then 'input${id}_seek' will not exist.
`elementId`	HTML id tag to be given to the howler player element

Details

All buttons associated with the `howler` should be given the same `id` argument. This is to ensure that the buttons are linked to the player.

i.e. If `howler(id = "howler")`, then `howlerPlayButton(id = "howler")`

Value

A shiny.tag containing all of the required options for a `Howl` JS object to be initialised in a shiny application.

On the server side there will be up to four additional objects available as inputs:

`{id}_playing` A logical value as to whether or not the `howler` is playing audio

`{id}_track` Basename of the file currently loaded

`{id}_seek` (If `seek_ping_rate > 0`) the current time (in seconds) of the track loaded

`{id}_duration` The duration (in seconds) of the track loaded

See Also

howlerButton, howlerServer

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    title = "howler.js Player",
    howler(elementId = "howler", c(sound = "audio/sound.mp3")),
    howlerPlayPauseButton("howler")
  )

  server <- function(input, output) {
  }

  runShiny(ui, server)
}

## Not run:
# Multiple file formats
howler(
  elementId = "howler",
  list(
    track_1 = c("audio/sound.webm", "audio/sound.mp3"),
    track_2 = c("audio/sound2.webm", "audio/sound2.mp3"),
  )
)

## End(Not run)
```

howler-shiny *Shiny bindings for howler*

Description

Output and render functions for using howler within Shiny applications and interactive Rmd documents.

Usage

```
howlerOutput(outputId)

renderHowler(expr, env = parent.frame(), quoted = FALSE)
```

Arguments

outputId	output variable to read from
expr	An expression that generates a howler
env	The environment in which to evaluate expr.
quoted	Is expr a quoted expression (with quote())? This is useful if you want to save an expression in a variable.

Value

An output or render function that enables the use of the widget within Shiny applications.

howlerButton *Audio Buttons*

Description

Buttons that can be used to interact with the howler.

howlerPlayButton, howlerPauseButton, howlerPlayPauseButton and howlerStopButton will all be applied to the current track.

howlerBackButton and howlerForwardButton will change the track position by a specified amount of time.

howlerPreviousButton and howlerNextButton will play the previous/following track supplied to the player.

howlerVolumeDownButton and howlerVolumeUpButton will change the volume of the player by a specified percentage.

howlerButton is a customisable version of any of the above individual button.

Usage

```
howlerButton(howler_id, button_type = HOWLER_BUTTON_TYPES, ...)

howlerPlayButton(howler_id)

howlerPauseButton(howler_id)

howlerPlayPauseButton(howler_id)

howlerStopButton(howler_id)

howlerBackButton(howler_id, seek_change = 10)

howlerForwardButton(howler_id, seek_change = 10)

howlerPreviousButton(howler_id)

howlerNextButton(howler_id)

howlerVolumeUpButton(howler_id, volume_change = 0.1)

howlerVolumeDownButton(howler_id, volume_change = 0.1)

howlerVolumeToggleButton(howler_id)
```

Arguments

howler_id	ID given to the howler player.
button_type	Type of button to create. Available buttons are in the details, default set to play_pause.
...	Attributes/Inner tags added to the button
seek_change	Time (in seconds) to move forward/backward the track when clicked. Default is 10 seconds
volume_change	How much to change the volume by. Default is 10%.

Details

The following button_type are available to create:

play_pause (default) Switch between playing and pausing the track

play Resumes the current track

pause Pauses the current track

stop Stops current track, when played will start from beginning

previous,next Switches to the previous/following track

volumedown,volumeup Decreases/Increases the volume by 10% (If using howlerButton include the attribute `data-volume-change`)

back,forward Seek forward/backwards 10s (If using howlerButton include the attribute `data-seek-change` with negative values to go backwards)

When using a play_pause button, the icon will toggle between the play and pause button depending on whether or not the track is playing.

Value

An HTML tag containing the audio button.

An additional input will be available in the server side in the form {id}_{button_type}. For example howlerBackButton("howler") will create an input element of input$howler_back. All of these will work in the same way as actionButton

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    tile = "howler.js Player",
    howler(elementId = "howler", "audio/sound.mp3"),
    howlerPreviousButton("howler"),
    howlerBackButton("howler"),
    howlerPlayPauseButton("howler"),
    howlerForwardButton("howler"),
    howlerNextButton("howler"),
    howlerVolumeDownButton("howler"),
    howlerVolumeUpButton("howler")
  )

  server <- function(input, output) {
  }

  shinyApp(ui, server)
}
```

---

howlerCurrentTrack *Current Track*

Description

A way to display track information in the UI without having to communicate with the server.

Usage

```
howlerCurrentTrack(id, ...)

howlerSeekTime(id, ...)

howlerDurationTime(id, ...)
```

Arguments

id	ID given to the current track label. For it to work with the howler, the ID must match that of the howler.
...	Other attributes to give to the HTML tag.

Value

A div tag that will be linked to the howler to show the current track.

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    title = "howler.js Player",
    howler(elementId = "sound", "audio/sound.mp3"),
    howlerCurrentTrack("sound"),
    p(
      howlerSeekTime("sound"),
      "/",
      howlerDurationTime("sound")
    ),
    howlerPlayPauseButton("sound")
  )

  server <- function(input, output, session) {
  }

  shinyApp(ui, server)
}
```

---

howlerModule *Howler.js Module*

Description

A simple module containing a howler player and a default set of howler buttons. The module also contains the current position of the track being played and the duration of the track.

Usage

```
howlerModuleUI(id, files, ..., include_current_track = TRUE, width = "300px")

howlerBasicModuleUI(id, files, ..., width = "300px")

howlerModuleServer(id)
```

Arguments

id	ID to give to the namespace of the module. The howler player will have the ID {id}-howler.
files	Files that will be used in the player. This can either be a single vector, or a list where different formats of the same file are kept in each element of the list.
...	Further arguments to send to howler
include_current_track	Logical, should the current track be included in the UI of the module?
width	Width (in pixels) of the player. Defaults to 400px.

Value

The UI will provide a player with a play/pause button, previous and next buttons, duration information and a volume slider.

The server-side module will return a list of reactive objects:

playing Logical value whether or not the player is currently playing

track Name of the track currently loaded

duration Duration (in seconds) of the track currently loaded

seek Current position (in seconds) of the track currently loaded

Examples

```
if (interactive()) {
  ui <- fluidPage(
    title = "howler.js Module",
    howlerModuleUI("howl", c("audio/track1.mp3", "audio/track2.mp3"))
  )

  server <- function(input, output, session) {
    moduleServer("howl", howlerModuleServer)
  }

  shinyApp(ui, server)
}
```

---

howlerSeekSlider *Seek Slider*

Description

A UI element that can be included with a howler to manually change the location of the track.

Usage

```
howlerSeekSlider(id)
```

Arguments

id	ID given to the volume slider. For it to work with the howler, the ID must match that of the howler.

Value

A slider element of class howler-seek-slider that will display the position of the current track playing.

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    title = "howler.js Player",
    howler(elementId = "sound", "audio/sound.mp3"),
    howlerPlayPauseButton("sound"),
    howlerSeekSlider("sound")
  )

  server <- function(input, output, session) {
  }

  shinyApp(ui, server)
}
```

---

howlerServer *Update howler.js Server-Side*

Description

Change the state of the howler player from the server.

playHowl, pauseHowl and stopHowl will all be applied to the current track.

changeTrack will update the track to the file specified.

addTrack will add a new track to the specified player.

Usage

```
changeTrack(id, track, session = getDefaultReactiveDomain())

addTrack(id, track, play_track = FALSE, session = getDefaultReactiveDomain())

playHowl(id, session = getDefaultReactiveDomain())

pauseHowl(id, session = getDefaultReactiveDomain())
```

```
stopHowl(id, session = getDefaultReactiveDomain())

seekHowl(id, seek, session = getDefaultReactiveDomain())
```

Arguments

id	ID of the howler to update
track	Either the track name of the file to change to, or the index of the file to play. If the file is not included in the player nothing will happen.
session	Shiny session
play_track	Logical, should the new track be played on addition?
seek	Time (in seconds) to set the position of the track

Value

Updates the the state of the specified howler in the shiny application.

Examples

```
if (interactive()) {
  library(shiny)

  tracks <- c("audio/track1.mp3", "audio/track2.mp3")

  ui <- fluidPage(
    title = "howler.js Player",
    selectInput("track", "Select Track", basename(tracks)),
    howler(elementId = "howler", tracks),
    howlerPlayPauseButton("howler")
  )

  server <- function(input, output) {
    observeEvent(input$track, changeHowlerTrack("howler", input$track))
  }

  runShiny(ui, server)
}
```

---

howlerVolumeSlider *Volume Slider*

Description

A more user friendly way to adjust the volume of the howler than by using buttons. There are still volume up/down buttons, but a slider can be moved up/down as required.

Usage

```
howlerVolumeSlider(id, volume = 1, button = TRUE)
```

Arguments

id	ID given to the volume slider. For it to work with the howler, the ID must match that of the howler.
volume	Initial volume to set the player at. Defaults at 100%
button	Logical, should a mute toggle button be included next to the slider? Default is TRUE

Details

If using howlerVolumeSlider, avoid using the volume buttons, as this will cause duplicate IDs to appear in the shiny application and prevents the slider from working properly.

Value

A volume slider with a howlerVolumeDownButton and a howlerVolumeUpButton either side.

Examples

```
if (interactive()) {
  library(shiny)

  ui <- fluidPage(
    title = "howler.js Player",
    howler(elementId = "sound", "audio/sound.mp3"),
    howlerPlayPauseButton("sound"),
    howlerVolumeSlider("sound")
  )

  server <- function(input, output, session) {
  }

  shinyApp(ui, server)
}
```

---

runHowlerExample *Run* {howler} *Example Applications*

Description

Run {howler} Example Applications

Usage

```
runHowlerExample(example = "basic", display.mode = "showcase", ...)

availableHowlerExamples()
```

Arguments

example	Name of the example to load. Current examples include: **basic** Basic example of howler in use **full** Basic example of using all buttons available in howler **module** Example of using the howlerModule **server** Example showing server-side functionality
display.mode	The mode in which to display the application. By default set to "showcase" to show code behind the example.
...	Optional arguments to send to shiny::runApp

Value

This function does not return a value; interrupt R to stop the application (usually by pressing Ctrl+C or Esc).

Examples

```
availableHowlerExamples()

if (interactive()) {
  library(shiny)
  library(howler)

  runHowlerExample("basic")
}
```

Index