Package ‘sortable’

December 13, 2021

Type Package

Title Drag-and-Drop in 'shiny' Apps with 'SortableJS'

Version 0.4.5

Description Enables drag-and-drop behaviour in Shiny apps, by exposing the functionality of the 'SortableJS' \(\text{<https://sortablejs.github.io/Sortable/>}\) JavaScript library as an 'htmlwidget'. You can use this in Shiny apps and widgets, 'learnr' tutorials as well as R Markdown. In addition, provides a custom 'learnr' question type - 'question_rank()' - that allows ranking questions with drag-and-drop.

License MIT + file LICENSE

URL https://rstudio.github.io/sortable/

BugReports https://github.com/rstudio/sortable/issues

Imports htmltools, htmlwidgets, learnr (>= 0.10.0), shiny, assertthat, jsonlite, utils, ellipsis, rlang

Suggests base64, knitr, testthat (>= 2.1.0), withr, rmarkdown, magrittr, webshot, spelling, covr

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.1.2

Language en-US

Config/testthat/edition 3

NeedsCompilation no

Author Andrie de Vries [cre, aut], Barret Schloerke [aut], Kenton Russell [aut, ccp] (Original author), RStudio [cph, fnl], Lebedev Konstantin [cph] ('SortableJS', https://sortablejs.github.io/Sortable/)

Maintainer Andrie de Vries <apdevries@gmail.com>

Repository CRAN

Date/Publication 2021-12-13 10:00:02 UTC
## add_rank_list

**R topics documented:**

- add_rank_list .................................................. 2
- bucket_list ..................................................... 3
- chain_js_events ................................................. 4
- is_sortable_options ............................................ 5
- question_rank ................................................... 6
- rank_list .......................................................... 7
- render_sortable .................................................. 9
- sortable_js ....................................................... 9
- sortable_js_capture_input ..................................... 10
- sortable_options ............................................... 12
- sortable_output .................................................. 14

### Description

Since a `bucket_list` can contain more than one `rank_list`, you need an easy way to define the contents of each individual rank list. This function serves as a specification of a rank list.

### Usage

```r
add_rank_list(text, labels = NULL, input_id = NULL, ...)
```

### Arguments

- **text**
  - Text to appear at top of list.
- **labels**
  - A character vector with the text to display inside the widget. This can also be a list of html tag elements. The text content of each label or label name will be used to set the shiny `input_id` value.
- **input_id**
  - output variable to read the plot/image from.
- **...**
  - Other arguments passed to `rank_list`

### Value

A list of class `add_rank_list`
bucket_list

Create a bucket list.

Description

A bucket list can contain more than one rank_list and allows drag-and-drop of items between the different lists.

Usage

```r
bucket_list(
  header = NULL,
  ..., 
  group_name,
  group_put_max = rep(Inf, length(labels)),
  options = sortable_options(),
  class = "default-sortable",
  orientation = c("horizontal", "vertical")
)
```

Arguments

- **header**: Text that appears at the top of the bucket list. (This is encoded as an HTML <p> tag, so not strictly speaking a header.) Note that you must explicitly provide header argument, especially in the case where you want the header to be empty - to do this use `header = NULL` or `header = NA`.
- **...**: One or more specifications for a rank list, and must be defined by `add_rank_list`.
- **group_name**: Passed to SortableJS as the group name. Also the input value set in Shiny. (input[[group_name]])
- **group_put_max**: Not yet implemented
- **options**: Options to be supplied to sortable_js object. See sortable_options for more details
- **class**: A css class applied to the bucket list and rank lists. This can be used to define custom styling.
- **orientation**: Either horizontal or vertical, and specifies the layout of the components on the page.

Value

A list with class `bucket_list`

See Also

rank_list
**Examples**

```r
## -- example-bucket-list ---------------------------------------------
## bucket list

if(interactive()) {
    bucket_list(
        header = "This is a bucket list. You can drag items between the lists.",
        add_rank_list(
            text = "Drag from here",
            labels = c("a", "bb", "ccc")
        ),
        add_rank_list(
            text = "to here",
            labels = NULL
        )
    )
}

## bucket list with three columns

if(interactive()) {
    bucket_list(
        header = c("Sort these items into Letters and Numbers"),
        add_rank_list(
            text = "Drag from here",
            labels = sample(c(1:3, letters[1:2]))
        ),
        add_rank_list(
            text = "Letters"
        ),
        add_rank_list(
            text = "Numbers"
        )
    )
}

## Example of a shiny app

if (interactive()) {
    app <- system.file("shiny-examples/bucket_list/app.R", package = "sortable")
    shiny::runApp(app)
}
```

---

**chain_js_events**  
*Chain multiple JavaScript events*

**Description**

SortableJS does not have an event based system. To be able to call multiple JavaScript events under the same event execution, they need to be executed one after another.
is_sortable_options

Usage

chain_js_events(...)

Arguments

... JavaScript functions defined by htmlwidgets::JS

Value

A single JavaScript function that will call all methods provided with the event

See Also

Other JavaScript functions: sortable_js_capture_input()

Description

Check if object is sortable options.

Usage

is_sortable_options(x)

Arguments

x Object to test

Value

Logical vector. TRUE if the object inherits from sortable_options

Examples

is_sortable_options("foo") # returns FALSE
Ranking question for learnr tutorials.

Description

Add interactive ranking tasks to your learnr tutorials. The student can drag-and-drop the answer options into the desired order.

Usage

```r
question_rank(
  text,
  ..., 
  correct = "Correct!",
  incorrect = "Incorrect",
  loading = c("**Loading:** ", text, "<br/><br/><br/>")
  submit_button = "Submit Answer",
  try_again_button = "Try Again",
  allow_retry = FALSE,
  random_answer_order = TRUE,
  options = sortable_options()
)
```

Arguments

- **text**: Question or option text
- **...**: parameters passed onto `learnr::question()`.
- **correct**: For question, text to print for a correct answer (defaults to "Correct!"). For answer, a boolean indicating whether this answer is correct.
- **incorrect**: Text to print for an incorrect answer (defaults to "Incorrect") when allow_retry is FALSE.
- **loading**: Loading text to display as a placeholder while the question is loaded.
- **submit_button**: Label for the submit button. Defaults to "Submit Answer".
- **try_again_button**: Label for the try again button. Defaults to "Submit Answer".
- **allow_retry**: Allow retry for incorrect answers. Defaults to FALSE.
- **random_answer_order**: Display answers in a random order.
- **options**: Options to be supplied to `sortable_js` object. See `sortable_options` for more details.

Details

Each set of answer options must contain the same set of answer options. When the question is completed, the first correct answer will be displayed.

Note that, by default, the answer order is randomized.
rank_list

Value
A custom learnr question, with type = sortable_rank. See learnr::question().

Examples
## Example of rank problem inside a learnr tutorial
if (interactive()) {
  learnr::run_tutorial("question_rank", package = "sortable")
}

---

**Description**

Create a ranking item list using the SortableJS framework, and generates an htmlwidgets element. The elements of this list can be dragged and dropped in any order.

You can embed a ranking question inside a learnr tutorial, using question_rank().

To embed a rank_list inside a shiny app, see the Details section.

**Usage**

```r
rank_list(
  text = "", 
  labels, 
  input_id, 
  css_id = NULL, 
  options = sortable_options(), 
  class = "default-sortable"
)
```

**Arguments**

- **text**: Text to appear at top of list.
- **labels**: A character vector with the text to display inside the widget. This can also be a list of html tag elements. The text content of each label or label name will be used to set the shiny input_id value.
- **input_id**: output variable to read the plot/image from.
- **css_id**: This is the css id to use, and must be unique in your shiny app. If NULL, the function generates an id of the form rank_list_id_1, and will automatically increment for every rank_list.
- **options**: Options to be supplied to sortable js object. See sortable_options for more details.
- **class**: A css class applied to the rank list. This can be used to define custom styling.
You can embed a `rank_list` inside a Shiny app, to capture the preferred ranking order of your user. The widget automatically updates a Shiny output, with the matching `input_id`.

**See Also**

`sortable_js`, `bucket_list` and `question_rank`

**Examples**

```r
## - example-rank-list ------------------------------------------------
if (interactive()) {
  rank_list(
    text = "You can drag, drop and re-order these items:",
    labels = c("one", "two", "three", "four", "five"),
    input_id = "example_2"
  )
}
## - example-rank-list-multidrag ------------------------------------------
if (interactive()) {
  rank_list(
    text = "You can select multiple items and drag as a group:",
    labels = c("one", "two", "three", "four", "five"),
    input_id = "example_2",
    options = sortable_options(
      multiDrag = TRUE
    )
  )
}
## - example-rank-list-swap -----------------------------------------------
if (interactive()) {
  rank_list(
    text = "You can re-order these items, and notice the swapping behaviour:",
    labels = c("one", "two", "three", "four", "five"),
    input_id = "example_2",
    options = sortable_options(
      swap = TRUE
    )
  )
}
## Example of a shiny app
if (interactive()) {
  app <- system.file("shiny-examples/rank_list/app.R", package = "sortable")
  shiny::runApp(app)
}
```
**render_sortable**

Widget render function for use in Shiny.

**Description**

Widget render function for use in Shiny.

**Usage**

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

**Arguments**

- `expr`: An expression
- `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.

**sortable_js**

Creates an htmlwidget with embedded 'SortableJS' library.

**Description**

Creates an htmlwidget that provides SortableJS to use for drag-and-drop interactivity in Shiny apps and R Markdown.

**Usage**

`sortable_js( css_id, options = sortable_options(), width = 0, height = 0, elementId = NULL, preRenderHook = NULL )`

**Arguments**

- `css_id`: String `css_id` id on which to apply SortableJS. Note, `sortable_js` works with any html element, not just `ul/li`.
- `options`: Options to be supplied to `sortable_js` object. See `sortable_options` for more details
- `width`: Fixed width for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget’s container.
**height** Fixed height for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget's container.

**elementId** Use an explicit element ID for the widget (rather than an automatically generated one). Useful if you have other JavaScript that needs to explicitly discover and interact with a specific widget instance.

**preRenderHook** A function to be run on the widget, just prior to rendering. It accepts the entire widget object as input, and should return a modified widget object.

**See Also**

`sortable_options()`

**Examples**

```r
## -- example-sortable-js --------------------------------------------------
# Simple example of sortable_js.
# Important: set the tags CSS `id` equal to the sortable_js `css_id`

if (interactive()) {
  if (require(htmltools)) {
    html_print(
      tagList(
        tags$p("You can drag and reorder the items in this list:"),
        tags$ul(
          id = "example_1",
          tags$li("Move"),
          tags$li("Or drag"),
          tags$li("Each of the items"),
          tags$li("To different positions")
        ),
        sortable_js(css_id = "example_1")
      )
    )
  }
}
```

---

`sortable_js_capture_input`

Construct JavaScript method to capture Shiny inputs on change.

**Description**

This captures the state of a sortable list. It will look for a data-rank-id attribute of the first child for each element. If no? attribute exists for that particular item's first child, the inner text will be used as an identifier.
sortable_js_capture_input

Usage

sortable_js_capture_input(input_id)
sortable_js_capture_bucket_input(input_id, input_ids, css_ids)

Arguments

input_id Shiny input name to set
input_ids Set of Shiny input ids to set corresponding to the provided css_ids
css_ids Set of SortableJS css_id values to help retrieve all to set as an object

Details

This method is used with the onSort option of sortable_js. See sortable_options().

Value

A character vector with class JS_EVAL. See htmlwidgets::JS().

See Also

sortable_js and rank_list.

Other JavaScript functions: chain_js_events()

Examples

```r
## -- example-sortable-js-capture -----------------------------------------
# Simple example of sortable_js_capture.
# Important: set the tags CSS `id` equal to the sortable_js `css_id`

if(interactive()) {
  library(shiny)
  library(sortable)

  ui <- fluidPage(
    div(
      id = "sortable",
      div(id = 1, `data-rank-id` = "HELLO", class = "well", "Hello"),
      div(id = 2, `data-rank-id` = "WORLD", class = "well", "world")
    ),
   verbatimTextOutput("chosen"),
    sortable_js(
      css_id = "sortable",
      options = sortable_options(
        onSort = sortable_js_capture_input(input_id = "selected")
      )
    )
  )

  server <- function(input, output){
```
sortable_options

output$chosen <- renderPrint(input$selected)

shinyApp(ui, server)

## ------------------------------------
# For an example, see the Shiny app at
system.file("shiny-examples/drag_vars_to_plot/app.R", package = "sortable")

---

## sortable_options

### Define options to pass to a sortable object.

#### Description

Use this function to define the options for `sortable_js` and `rank_list`, which will pass these in turn to the SortableJS JavaScript library.

#### Usage

```r
sortable_options(
  ..., 
  swap = NULL, 
  multiDrag = NULL, 
  group = NULL, 
  sort = NULL, 
  delay = NULL, 
  disabled = NULL, 
  animation = NULL, 
  handle = NULL, 
  filter = NULL, 
  draggable = NULL, 
  swapThreshold = NULL, 
  invertSwap = NULL, 
  direction = NULL, 
  scrollSensitivity = NULL, 
  scrollSpeed = NULL, 
  onStart = NULL, 
  onEnd = NULL, 
  onAdd = NULL, 
  onUpdate = NULL, 
  onSort = NULL, 
  onRemove = NULL, 
  onFilter = NULL, 
  onMove = NULL, 
  onLoad = NULL
)
```
Arguments

... other arguments passed onto SortableJS

swap If TRUE, modifies the behaviour of sortable to allow for items to be swapped with each other rather than sorted. Once dragging starts, the user can drag over other items and there will be no change in the elements. However, the item that the user drops on will be swapped with the originally dropped item. See also https://github.com/SortableJS/Sortable/tree/master/plugins/Swap

multiDrag If TRUE, allows the selection of multiple items within a sortable at once, and drag them as one item. Once placed, the items will unfold into their original order, but all beside each other at the new position. See also https://github.com/SortableJS/Sortable/wiki/Dragging-Multiple-Items-in-Sortable

group To drag elements from one list into another, both lists must have the same group value. See Sortable#group-option for more details. ["name"]

sort Boolean that allows sorting inside a list. [TRUE]

delay Time in milliseconds to define when the sorting should start. [0]

disable Boolean that disables the sortable if set to true. [FALSE]

animation Millisecond duration of the animation of items when sorting [0 (no animation)]

handle CSS selector used for the drag handle selector within list items. [".my-handle"]

filter CSS selector or JS function used for elements that cannot be dragged. [".ignore-elements"]

draggable CSS selector of which items inside the element should be draggable. [".item"]

swapThreshold Percentage of the target that the swap zone will take up, as a number between 0 and 1. [1]

invertSwap Set to TRUE to set the swap zone to the sides of the target, for the effect of sorting "in between" items. [FALSE]

direction Direction of sortable ["horizontal"]

scrollSensitivity Number of pixels the mouse needs to be to an edge to start scrolling. [30]

scrollSpeed Number of pixels for the speed of scrolling. [10]

onStart, onEnd JS function called when an element dragging starts or ends

onAdd JS function called when an element is dropped into the list from another list

onUpdate JS function called when the sorting is changed within a list

onSort JS function called by any change to the list (add / update / remove)

onRemove JS function called when an element is removed from the list into another list

onFilter JS function called when an attempt is made to drag a filtered element

onMove JS function called when an item is moved in a list or between lists

onLoad JS function dispatched on the "next tick" after SortableJS has initialized

Details

Many of the SortableJS options will accept a JavaScript function. You can do this using the htmlwidgets::JS function.
**Value**

A list with class `sortable_options`

**References**

[https://github.com/sortablejs/Sortable/](https://github.com/sortablejs/Sortable/)

**See Also**

`sortable_js`

**Examples**

```r
sortable_options(sort = FALSE)
```

---

**sortable_output**  
*Widget output function for use in Shiny.*

**Description**

Widget output function for use in Shiny.

**Usage**

```r
sortable_output(input_id, width = "0px", height = "0px")
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>input_id</code></td>
<td>output variable to use for the sortable object</td>
</tr>
<tr>
<td><code>width</code></td>
<td>Fixed width for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget’s container.</td>
</tr>
<tr>
<td><code>height</code></td>
<td>Fixed height for widget (in css units). The default is NULL, which results in intelligent automatic sizing based on the widget’s container.</td>
</tr>
</tbody>
</table>
Index

* JavaScript functions
  chain_js_events, 4
  sortable_js_capture_input, 10

add_rank_list, 2, 3
bucket_list, 2, 3, 8
chain_js_events, 4, 11
htmlwidgets::JS, 5
htmlwidgets::JS(), 11
is_sortable_options, 5
learnr::question(), 6, 7
question_rank, 6, 8
question_rank(), 7
rank_list, 2, 3, 7, 11, 12
renderSortable, 9
sortable_js, 3, 6–9, 9, 11, 12, 14
sortable_js_capture_bucket_input
  (sortable_js_capture_input), 10
sortable_js_capture_input, 5, 10
sortable_options, 3, 6, 7, 9, 12
sortable_options(), 10, 11
sortable_output, 14