Package ‘crosstalk’

November 23, 2023

Type Package
Title Inter-Widget Interactivity for HTML Widgets
Version 1.2.1
Description Provides building blocks for allowing HTML widgets to communicate with each other, with Shiny or without (i.e. static .html files). Currently supports linked brushing and filtering.
License MIT + file LICENSE
Imports htmltools (>= 0.3.6), jsonlite, lazyeval, R6
Suggests shiny, ggplot2, testthat (>= 2.1.0), sass, bslib
BugReports https://github.com/rstudio/crosstalk/issues
RoxygenNote 7.2.3
Encoding UTF-8
NeedsCompilation no

Author Joe Cheng [aut],
Carson Sievert [aut, cre] (<https://orcid.org/0000-0002-4958-2844>),
Posit Software, PBC [cph, fnd],
jQuery Foundation [cph] (jQuery library and jQuery UI library),
jQuery contributors [ctb, cph] (jQuery library; authors listed in inst/www/shared/jquery-AUTHORS.txt),
Mark Otto [ctb] (Bootstrap library),
Jacob Thornton [ctb] (Bootstrap library),
Bootstrap contributors [ctb] (Bootstrap library),
Twitter, Inc [cph] (Bootstrap library),
Brian Reavis [ctb, cph] (selectize.js library),
Kristopher Michael Kowal [ctb, cph] (es5-shim library),
es5-shim contributors [ctb, cph] (es5-shim library),
Denis Ineshin [ctb, cph] (ion.rangeSlider library),
Sami Samhuri [ctb, cph] (Javascript strftime library)

Maintainer Carson Sievert <carson@posit.co>
bscols

Arrange HTML elements or widgets in Bootstrap columns

Description

This helper function makes it easy to put HTML elements side by side. It can be called directly from the console but is especially designed to work in an R Markdown document. Warning: This will bring in all of Bootstrap!

Usage

bscols(..., widths = NA, device = c("xs", "sm", "md", "lg"))

Arguments

... htmltools tag objects, lists, text, HTML widgets, or NULL. These arguments should be unnamed.

widths The number of columns that should be assigned to each of the ... elements (the total number of columns available is always 12). The width vector will be recycled if there are more ... arguments. NA columns will evenly split the remaining columns that are left after the widths are recycled and non-NA values are subtracted.

device The class of device which is targeted by these widths; with smaller screen sizes the layout will collapse to a one-column, top-to-bottom display instead. xs: never collapse, sm: collapse below 768px, md: 992px, lg: 1200px.

Value

A browsable HTML element.
Examples

```r
library(htmltools)

# If width is unspecified, equal widths will be used
bscols(
  div(style = css(width="100\%", height="400px", background_color="red")),
  div(style = css(width="100\%", height="400px", background_color="blue"))
)

# Use NA to absorb remaining width
bscols(widths = c(2, NA, NA),
  div(style = css(width="100\%", height="400px", background_color="red")),
  div(style = css(width="100\%", height="400px", background_color="blue")),
  div(style = css(width="100\%", height="400px", background_color="green"))
)

# Recycling widths
bscols(widths = c(2, 4),
  div(style = css(width="100\%", height="400px", background_color="red")),
  div(style = css(width="100\%", height="400px", background_color="blue")),
  div(style = css(width="100\%", height="400px", background_color="red")),
  div(style = css(width="100\%", height="400px", background_color="blue"))
)
```

---

**ClientValue**

**ClientValue object**

Description

An object that can be used in a Shiny server function to get or set a crosstalk variable that exists on the client. The client copy of the variable is the canonical copy, so there is no direct "set" method that immediately changes the value; instead, there is a ‘sendUpdate’ method that sends a request to the browser to change the value, which will then cause the new value to be relayed back to the server.

This object is used to implement `SharedData` and should not need to be used directly by users.

Methods

**Public methods:**

- `ClientValue$new()`
- `ClientValue$get()`
- `ClientValue$sendUpdate()`
- `ClientValue$clone()`

**Method new():** Creates a new ClientValue object to reflect the crosstalk variable specified by 'group' and 'name'.
ClientValue

Usage:
ClientValue$new(
  name,
  group = "default",
  session = shiny::getDefaultReactiveDomain()
)

Arguments:
name  The name of the crosstalk variable.
group The name of the crosstalk variable group.
session The Shiny session to connect to; defaults to the current session.

Method get(): Read the value. This is a reactive operation akin to reading a reactive value, and so can only be done in a reactive context (e.g. in a 'shiny::reactive()', 'shiny::observe()', or 'shiny::isolate()' block).

Usage:
ClientValue$get()

Method sendUpdate(): Send a message to the browser asking it to update the crosstalk var to the given value. This update does not happen synchronously, that is, a call to 'get()' immediately following 'sendUpdate(value)' will not reflect the new value.

Usage:
ClientValue$sendUpdate(value)

Arguments:
value  The new value for the crosstalk variable. Must be serializable as JSON using 'jsonlite'.

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

Usage:
ClientValue$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.

Examples

library(shiny)

server <- function(input, output, session) {
  cv <- ClientValue$new("var1", "group1")

  r <- reactive(
    # Don't proceed unless cv$get() is a non-NULL value
    validate(need(cv$get(), message = FALSE))
    runif(cv$get())
  )

  observeEvent(input$click, {
    cv$sendUpdate(NULL)
  })
}
crosstalkLibs

})
)

crosstalkLibs  Crosstalk dependencies

Description
List of htmlDependency objects necessary for Crosstalk to function. Intended for widget authors.

Usage
crosstalkLibs()

filter_select  Categorical filter controls

Description
Creates a select box or list of checkboxes, for filtering a SharedData object based on categorical data.

Usage
filter_select(id, label, sharedData, group, allLevels = FALSE, multiple = TRUE)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>id</td>
<td>An HTML element ID; must be unique within the web page</td>
</tr>
<tr>
<td>label</td>
<td>A human-readable label</td>
</tr>
<tr>
<td>sharedData</td>
<td>SharedData object with the data to filter</td>
</tr>
<tr>
<td>group</td>
<td>A one-sided formula whose values will populate this select box. Generally this should be a character or factor column; if not, it will be coerced to character.</td>
</tr>
</tbody>
</table>
allLevels  If the vector described by group is factor-based, should all the levels be displayed as options, or only ones that are present in the data?
multiple  Can multiple values be selected?
inline  If TRUE, render checkbox options horizontally instead of vertically.
columns  Number of columns the options should be arranged into.

Examples

```r
## Only run examples in interactive R sessions
if (interactive()) {

  sd <- SharedData$new(chickwts)
  filter_select("feedtype", "Feed type", sd, "feed")
}
```

---

**filter_slider**  *Range filter control*

**Description**

Creates a slider widget that lets users filter observations based on a range of values.

**Usage**

```r
filter_slider(
  id,
  label,
  sharedData,
  column,
  step = NULL,
  round = FALSE,
  ticks = TRUE,
  animate = FALSE,
  width = NULL,
  sep = ",",
  pre = NULL,
  post = NULL,
  timeFormat = NULL,
  timezone = NULL,
  dragRange = TRUE,
  min = NULL,
  max = NULL
)
```

animation_options(}
filter_slider

```
interval = 1000,
loop = FALSE,
playButton = NULL,
pauseButton = NULL
```

Arguments

- **id**
  - An HTML element ID; must be unique within the web page
- **label**
  - A human-readable label
- **sharedData**
  - A `SharedData` object with the data to filter
- **column**
  - A one-sided formula whose values will be used for this slider. The column must be of type `Date`, `POSIXt`, or numeric.
- **step**
  - Specifies the interval between each selectable value on the slider (if `NULL`, a heuristic is used to determine the step size). If the values are dates, `step` is in days; if the values are times (POSIXt), `step` is in seconds.
- **round**
  - `TRUE` to round all values to the nearest integer; `FALSE` if no rounding is desired; or an integer to round to that number of decimal places (for example, `1` will round to the nearest 0.1, and `-2` will round to the nearest 100). Any rounding will be applied after snapping to the nearest step.
- **ticks**
  - `FALSE` to hide tick marks, `TRUE` to show them according to some simple heuristics.
- **animate**
  - `TRUE` to show simple animation controls with default settings; `FALSE` not to; or a custom settings list, such as those created using `animationOptions`.
- **width**
  - The width of the slider control (see `validateCssUnit` for valid formats)
- **sep**
  - Separator between thousands places in numbers.
- **pre**
  - A prefix string to put in front of the value.
- **post**
  - A suffix string to put after the value.
- **timeFormat**
  - Only used if the values are `Date` or `POSIXt` objects. A time format string, to be passed to the Javascript strftime library. See [https://github.com/samsonjs/strftime](https://github.com/samsonjs/strftime) for more details. The allowed format specifications are very similar, but not identical, to those for R’s `strftime` function. For Dates, the default is "%F" (like "2015-07-01"), and for POSIXt, the default is "%F %T" (like "2015-07-01 15:32:10")
- **timezone**
  - Only used if the values are POSIXt objects. A string specifying the time zone offset for the displayed times, in the format "+HHMM" or "-HHMM". If `NULL` (the default), times will be displayed in the browser’s time zone. The value "+0000" will result in UTC time.
- **dragRange**
  - This option is used only if it is a range slider (with two values). If `TRUE` (the default), the range can be dragged. In other words, the min and max can be dragged together. If `FALSE`, the range cannot be dragged.
- **min**
  - The leftmost value of the slider. By default, set to the minimal number in input data.
is.SharedData

Description

Check if an object is an instance of `SharedData` or not.

Usage

```r
is.SharedData(x)
```

Arguments

- `x` The object that may or may not be an instance of `SharedData`

Value

`logical`
**maintain_selection**  
_Synchronize Shiny brush selection with shared data_

**Description**
Waits for a brush to change, and propagates that change to the `sharedData` object.

**Usage**
```r
maintain_selection(sharedData, brushId, ownerId = "")
```

**Arguments**
- `sharedData`: The shared data instance
- `brushId`: Character vector indicating the name of the plotOutput brush
- `ownerId` (TBD)

**scale_fill_selection**  
_ggplot2 helpers_

**Description**
Add `scale_fill_selection()` or `scale_color_selection` to a ggplot to customize the scale for fill or color, respectively, for linked brushing. Use `selection_factor` to turn logical vectors representing selection, to a factor with the levels ordered for use with ggplot2 bar stacking.

**Usage**
```r
scale_fill_selection(color_false, color_true)
scale_color_selection(color_false, color_true)
selection_factor(
  x,
  na.replace = c(FALSE, NA, TRUE),
  reverse = packageVersion("ggplot2") < "2.2.0"
)
```

**Arguments**
- `color_false`: The color that should be mapped to unselected rows
- `color_true`: The color that should be mapped to selected rows
- `x`: Either a data frame with a `selected_` column, or, a logical vector indicating which rows are selected
SharedData

**Description**

An R6 class that represents a shared data frame, or sufficiently data frame-like object.

The primary use for SharedData is to be passed to Crosstalk-compatible widgets in place of a data frame. Each SharedData$new(...) call makes a new "group" of widgets that link to each other, but not to widgets in other groups. You can also use a SharedData object from Shiny code in order to react to filtering and brushing from non-widget visualizations (like ggplot2 plots).

**Methods**

**Public methods:**

- SharedData$new()
- SharedData$origData()
- SharedData$groupName()
- SharedData$key()
- SharedData$data()
- SharedData$selection()
- SharedData$clearSelection()
- SharedData$clone()

**Method** new():

**Usage:**

```r
## Not run:
sd <- SharedData$new(iris)
renderPlot({
  df <- sd$data(withSelection = TRUE, withFilter = TRUE)
  ggplot(df, aes(Sepal.Length, Sepal.Width,
                 color = selection_factor(df))) +
  geom_point() +
  scale_color_selection("#444444", "skyblue1")
})

## End(Not run)
```
SharedData$new(
  data,
  key = NULL,
  group = createUniqueId(4, prefix = "SharedData")
)

Arguments:
data  A data frame-like object, or a Shiny reactive expression that returns a data frame-like object.
key  Character vector or one-sided formula that indicates the name of the column that represents the key or ID of the data frame. These must be unique, and ideally will be something intrinsic to the data (a proper ID) rather than a transient property like row index. If NULL, then row.names(data) will be used.
group  The "identity" of the Crosstalk group that widgets will join when you pass them this SharedData object. In some cases, you will want to have multiple independent SharedData objects link up to form a single web of widgets that all share selection and filtering state; in those cases, you'll give those SharedData objects the same group name. (One example: in Shiny, ui.R and server.R might each need their own SharedData instance, even though they're intended to represent a single group.)

Method origData(): Return the data frame that was used to create this SharedData instance. If a reactive expression, evaluate the reactive expression. Equivalent to SharedData$data(FALSE, FALSE, FALSE).
  Usage:
  SharedData$origData()

Method groupName(): Returns the value of group that was used to create this instance.
  Usage:
  SharedData$groupName()

Method key(): Returns the vector of key values. Filtering is not applied.
  Usage:
  SharedData$key()

Method data(): Return the data (or read and return the data if the data is a Shiny reactive expression).
When running in Shiny, calling data() is a reactive operation that will invalidate if the selection or filter change (assuming that information was requested), or if the original data is a reactive expression that has invalidated.
  Usage:
  SharedData$data(withSelection = FALSE, withFilter = TRUE, withKey = FALSE)
Arguments:
  withSelection  If 'TRUE', add a selection_ column with logical values indicating which rows are in the current selection, or NA if no selection is currently active.
  withFilter  If 'TRUE' (the default), only return rows that are part of the current filter settings, if any.
withKey  If ‘TRUE’, add a key_ column with the key values of each row (normally not needed
since the key is either one of the other columns or else just the row names).

Method selection(): Get or set the current selection in the client.
If called without arguments, returns a logical vector of rows that are currently selected (brushed),
or NULL if no selection exists. Intended to be called from a Shiny reactive context, and invalidates
whenever the selection changes.
If called with one or two arguments, sets the selection based on the given value indirectly, by
sending the value to the web browser (assumes an active Shiny app or Shiny R Markdown docu-
ment).

Usage:
SharedData$selection(value, ownerId = "")

Arguments:
value  If provided, a logical vector of ‘nrow(origData())’ length, indicating which rows are
currently selected (brushed).
ownerId  Set this argument to the ‘outputId’ of a widget if conceptually that widget "initiated"
the selection (prevents that widget from clearing its visual selection box, which is nor-
ually cleared when the selection changes). For example, if setting the selection based on a
[shiny::plotOutput()] brush, then ‘ownerId’ should be the ‘outputId’ of that ‘plotOutput’.

Method clearSelection():  Clears the selection indirectly, by sending an instruction to the
client that it should do so.

Usage:
SharedData$clearSelection(ownerId = "")

Arguments:
ownerId  See the [SharedData$selection()] method.

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

Usage:
SharedData$clone(deep = FALSE)

Arguments:
deep  Whether to make a deep clone.
Index

animation_options (filter_slider), 6
animationOptions, 7
browsable, 2
bscols, 2
ClientValue, 3
crosstalkLibs, 5
Date, 7
filter_checkbox (filter_select), 5
filter_select, 5
filter_slider, 6
HTML, 8
htmlDependency, 5
is.SharedData, 8
maintain_selection, 9
POSIXt, 7
reactive expression, 11
scale_color_selection
  (scale_fill_selection), 9
scale_fill_selection, 9
selection_factor
  (scale_fill_selection), 9
SharedData, 3, 5, 8, 10
strftime, 7
tag, 8
validateCssUnit, 7