Package ‘htmlwidgets’

December 6, 2023

Type Package

Title HTML Widgets for R

Version 1.6.4

Description A framework for creating HTML widgets that render in various contexts including the R console, 'R Markdown' documents, and 'Shiny' web applications.

License MIT + file LICENSE

URL `https://github.com/ramnathv/htmlwidgets`

BugReports `https://github.com/ramnathv/htmlwidgets/issues`

Imports grDevices, htmltools (>= 0.5.7), jsonlite (>= 0.9.16), knitr (>= 1.8), rmarkdown, yaml

Suggests testthat

Enhances shiny (>= 1.1)

VignetteBuilder knitr

Encoding UTF-8

RoxygenNote 7.2.3

NeedsCompilation no

Author Ramnath Vaidyanathan [aut, cph],
Yihui Xie [aut],
JJ Allaire [aut],
Joe Cheng [aut],
Carson Sievert [aut, cre] (<`https://orcid.org/0000-0002-4958-2844`>),
Kenton Russell [aut, cph],
Ellis Hughes [ctb],
Posit Software, PBC [cph, fnd]

Maintainer Carson Sievert <`carson@posit.co`>

Repository CRAN

Date/Publication 2023-12-06 06:00:06 UTC

R topics documented:

htmlwidgets-package 2
createWidget 3
getDependency 4
htmlwidgets-shiny 4
JS 6
onRender 6
onStaticRenderComplete 8
prependContent 9
saveWidget 10
scaffoldWidget 10
setWidgetIdSeed 11
sizingPolicy 12

Index **15**

`htmlwidgets-package` *HTML Widgets for R*

Description

The **htmlwidgets** package provides a framework for easily creating R bindings to JavaScript libraries. Widgets created using the framework can be:

Details

- Used at the R console for data analysis just like conventional R plots (via RStudio Viewer)
- Seamlessly embedded within R Markdown documents and Shiny web applications.
- Saved as standalone web pages for ad-hoc sharing via email, Dropbox, etc.

To get started creating your own HTML widgets, see the documentation available in the package vignettes:

```
vignette("develop_intro", package = "htmlwidgets")
vignette("develop_sizing", package = "htmlwidgets")
vignette("develop_advanced", package = "htmlwidgets")
```

Source code for the package is available on GitHub:

`https://github.com/ramnathv/htmlwidgets`

Author(s)

Ramnath Vaidyanathan, Joe Cheng, JJ Allaire, and Yihui Xie

createWidget — *Create an HTML Widget*

Description

Create an HTML widget based on widget YAML and JavaScript contained within the specified package.

Usage

```
createWidget(
  name,
  x,
  width = NULL,
  height = NULL,
  sizingPolicy = htmlwidgets::sizingPolicy(),
  package = name,
  dependencies = NULL,
  elementId = NULL,
  preRenderHook = NULL
)
```

Arguments

`name`	Widget name (should match the base name of the YAML and JavaScript files used to implement the widget)
`x`	Widget instance data (underlying data to render and options that govern how it's rendered). This value will be converted to JSON using `toJSON` and made available to the widget's JavaScript `renderValue` function.
`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.
`sizingPolicy`	Options that govern how the widget is sized in various containers (e.g. a standalone browser, the RStudio Viewer, a knitr figure, or a Shiny output binding). These options can be specified by calling the `sizingPolicy` function.
`package`	Package where the widget is defined (defaults to the widget name).
`dependencies`	Additional widget HTML dependencies (over and above those defined in the widget YAML). This is useful for dynamic dependencies that only exist when selected widget options are enabled (e.g. sets of map tiles or projections).
`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.

Details

For additional details on developing widgets, see package vignettes: `vignette("develop_intro", package = "htmlwidgets")`.

Value

An object of class `htmlwidget` that will intelligently print itself into HTML in a variety of contexts including the R console, within R Markdown documents, and within Shiny output bindings.

`getDependency` *Get js and css dependencies for a htmlwidget*

Description

Get js and css dependencies for a htmlwidget

Usage

```
getDependency(name, package = name)
```

Arguments

`name`	name of the widget.
`package`	name of the package, defaults to the widget name.

`htmlwidgets-shiny` *Shiny bindings for HTML widgets*

Description

Helpers to create output and render functions for using HTML widgets within Shiny applications and interactive Rmd documents.

Usage

```
shinyWidgetOutput(
  outputId,
  name,
  width,
  height,
  package = name,
  inline = FALSE,
  reportSize = TRUE,
  reportTheme = FALSE,
  fill = !inline
)

shinyRenderWidget(expr, outputFunction, env, quoted, cacheHint = "auto")
```

Arguments

`outputId`	output variable to read from
`name`	Name of widget to create output binding for
`width, height`	Must be a valid CSS unit (like `"100%"`, `"400px"`, `"auto"`) or a number, which will be coerced to a string and have `"px"` appended.
`package`	Package containing widget (defaults to `name`)
`inline`	use an inline (`span()`) or block container (`div()`) for the output
`reportSize`	Should the widget's container size be reported in the shiny session's client data?
`reportTheme`	Should the widget's container styles (e.g., colors and fonts) be reported in the shiny session's client data?
`fill`	whether or not the returned tag should be treated as a fill item, meaning that its `height` is allowed to grow/shrink to fit a fill container with an opinionated height (see `htmltools::bindFillRole()` for more). Examples of fill containers include `bslib::card()` and `bslib::card_body_fill()`.
`expr`	An expression that generates an HTML widget (or a promise of an HTML widget).
`outputFunction`	Shiny output function corresponding to this render function.
`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.
`cacheHint`	Extra information to use for optional caching using `shiny::bindCache()`.

Details

These functions are delegated to from within your widgets own shiny output and render functions. The delegation is boilerplate and always works the same for all widgets (see example below).

Value

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

Examples

```
# shiny output binding for a widget named 'foo'
fooOutput <- function(outputId, width = "100%", height = "400px") {
  htmlwidgets::shinyWidgetOutput(outputId, "foo", width, height)
}

# shiny render function for a widget named 'foo'
renderFoo <- function(expr, env = parent.frame(), quoted = FALSE) {
  if (!quoted) { expr <- substitute(expr) } # force quoted
  htmlwidgets::shinyRenderWidget(expr, fooOutput, env, quoted = TRUE)
}
```

JS — *Mark character strings as literal JavaScript code*

Description

This function `JS()` marks character vectors with a special class, so that it will be treated as literal JavaScript code when evaluated on the client-side.

Usage

```
JS(...)
```

Arguments

`...`	character vectors as the JavaScript source code (all arguments will be pasted into one character string)

Author(s)

Yihui Xie

Examples

```
library(htmlwidgets)
JS('1 + 1')
list(x = JS('function(foo) {return foo;}'), y = 1:10)
JS('function(x) {', 'return x + 1;', '}')
```

onRender — *Execute custom JavaScript code after rendering*

Description

Use this function to supplement the widget's built-in JavaScript rendering logic with additional custom JavaScript code, just for this specific widget object.

Usage

```
onRender(x, jsCode, data = NULL)
```

Arguments

`x`	An HTML Widget object
`jsCode`	Character vector containing JavaScript code (see Details)
`data`	An additional argument to pass to the `jsCode` function. This can be any R object that can be serialized to JSON. If you have multiple objects to pass to the function, use a named list.

Details

The `jsCode` parameter must contain valid JavaScript code which when evaluated returns a function.

The function will be invoked with three arguments: the first is the widget's main HTML element, and the second is the data to be rendered (the `x` parameter in `createWidget`). The third argument is the JavaScript equivalent of the R object passed into `onRender` as the `data` argument; this is an easy way to transfer e.g. data frames without having to manually do the JSON encoding.

When the function is invoked, the `this` keyword will refer to the widget instance object.

Value

The modified widget object

See Also

`onStaticRenderComplete`, for writing custom JavaScript that involves multiple widgets.

Examples

```
## Not run:
library(leaflet)

# This example uses browser geolocation. RStudio users:
# this won't work in the Viewer pane; try popping it
# out into your system web browser.
leaflet() %>% addTiles() %>%
  onRender("
    function(el, x) {
      // Navigate the map to the user's location
      this.locate({setView: true});
    }
  ")


# This example shows how you can make an R data frame available
# to your JavaScript code.

meh <- "&#x1F610;";
yikes <- "&#x1F628;";

df <- data.frame(
  lng = quakes$long,
  lat = quakes$lat,
  html = ifelse(quakes$mag < 5.5, meh, yikes),
  stringsAsFactors = FALSE
)

leaflet() %>% addTiles() %>%
  fitBounds(min(df$lng), min(df$lat), max(df$lng), max(df$lat)) %>%
  onRender("
    function(el, x, data) {
      for (var i = 0; i < data.lng.length; i++) {
        var icon = L.divIcon({className: '', html: data.html[i]});
        L.marker([data.lat[i], data.lng[i]], {icon: icon}).addTo(this);
      }
    }
  ", data = df)

## End(Not run)
```

onStaticRenderComplete

Execute JavaScript code after static render

Description

Convenience function for wrapping a JavaScript code string with a `<script>` tag and the boilerplate necessary to delay the execution of the code until after the next time htmlwidgets completes rendering any widgets that are in the page. This mechanism is designed for running code to customize widget instances, which can't be done at page load time since the widget instances will not have been created yet.

Usage

```
onStaticRenderComplete(jsCode)
```

Arguments

`jsCode`	A character vector containing JavaScript code. No R error will be raised if the code is invalid, not even on JavaScript syntax errors. However, the web browser will throw errors at runtime.

Details

Each call to `onStaticRenderComplete` will result in at most one invocation of the given code. In some edge cases in Shiny, it's possible for static rendering to happen more than once (e.g. a `renderUI` that contains static HTML widgets). `onStaticRenderComplete` calls only schedule execution for the next static render operation.

The pure JavaScript equivalent of `onStaticRenderComplete` is `HTMLWidgets.addPostRenderHandler(callback)`, where `callback` is a JavaScript function that takes no arguments.

Value

An htmltools `tags$script` object.

Examples

```
## Not run:
library(leaflet)
library(htmltools)
library(htmlwidgets)

page <- tagList(
  leaflet() %>% addTiles(),
  onStaticRenderComplete(
    "HTMLWidgets.find('.leaflet').setZoom(4);"
  )
)
print(page, browse = TRUE)

## End(Not run)
```

`prependContent` *Prepend/append extra HTML content to a widget*

Description

Use these functions to attach extra HTML content (primarily JavaScript and/or CSS styles) to a widget, for rendering in standalone mode (i.e. printing at the R console) or in a knitr document. These functions are NOT supported when running in a Shiny widget rendering function, and will result in a warning if used in that context. Multiple calls are allowed, and later calls do not undo the effects of previous calls.

Usage

```
prependContent(x, ...)

appendContent(x, ...)
```

Arguments

`x`	An HTML Widget object
`...`	Valid tags, text, and/or `HTML`, or lists thereof.

Value

A modified HTML Widget object.

saveWidget *Save a widget to an HTML file*

Description

Save a rendered widget to an HTML file (e.g. for sharing with others).

Usage

```
saveWidget(
  widget,
  file,
  selfcontained = TRUE,
  libdir = NULL,
  background = "white",
  title = class(widget)[[1]],
  knitrOptions = list()
)
```

Arguments

`widget`	Widget to save
`file`	File to save HTML into
`selfcontained`	Whether to save the HTML as a single self-contained file (with external resources base64 encoded) or a file with external resources placed in an adjacent directory.
`libdir`	Directory to copy HTML dependencies into (defaults to filename_files).
`background`	Text string giving the html background color of the widget. Defaults to white.
`title`	Text to use as the title of the generated page.
`knitrOptions`	A list of **knitr** chunk options.

scaffoldWidget *Create implementation scaffolding for an HTML widget*

Description

Add the minimal code required to implement an HTML widget to an R package.

Usage

```
scaffoldWidget(name, bowerPkg = NULL, edit = interactive())
```

Arguments

`name`	Name of widget
`bowerPkg`	Optional name of Bower package upon which this widget is based. If you specify this parameter then bower will be used to automatically download the widget's source code and dependencies and add them to the widget's YAML.
`edit`	Automatically open the widget's JavaScript source file after creating the scaffolding.

Note

This function must be executed from the root directory of the package you wish to add the widget to.

`setWidgetIdSeed` *Set the random seed for widget element ids*

Description

Set a random seed for generating widget element ids. Calling this function rather than relying on the default behavior ensures stable widget ids across sessions.

Usage

```
setWidgetIdSeed(seed, kind = NULL, normal.kind = NULL)
```

Arguments

`seed`	a single value, interpreted as an integer, or `NULL` (see 'Details').
`kind`	character or `NULL`. If `kind` is a character string, set R's RNG to the kind desired. Use `"default"` to return to the R default. See 'Details' for the interpretation of `NULL`.
`normal.kind`	character string or `NULL`. If it is a character string, set the method of Normal generation. Use `"default"` to return to the R default. `NULL` makes no change.

sizingPolicy *Create a widget sizing policy*

Description

Define the policy by which HTML widgets will be sized in various containers (e.g. Browser, RStudio Viewer, R Markdown, Shiny). Note that typically widgets can accept the default sizing policy (or override only one or two aspects of it) and get satisfactory sizing behavior via the automatic sizing logic built into the htmlwidgets framework (see the notes below for the most typical exceptions to this).

Usage

```
sizingPolicy(
  defaultWidth = NULL,
  defaultHeight = NULL,
  padding = NULL,
  viewer.defaultWidth = NULL,
  viewer.defaultHeight = NULL,
  viewer.padding = NULL,
  viewer.fill = TRUE,
  viewer.suppress = FALSE,
  viewer.paneHeight = NULL,
  browser.defaultWidth = NULL,
  browser.defaultHeight = NULL,
  browser.padding = NULL,
  browser.fill = FALSE,
  browser.external = FALSE,
  knitr.defaultWidth = NULL,
  knitr.defaultHeight = NULL,
  knitr.figure = TRUE,
  fill = NULL
)
```

Arguments

`defaultWidth`	The default width used to display the widget. This parameter specifies the default width for viewing in all contexts (browser, viewer, and knitr) unless it is specifically overridden with e.g. `browser.defaultWidth`.
`defaultHeight`	The default height used to display the widget. This parameter specifies the default height for viewing in all contexts (browser, viewer, and knitr) unless it is specifically overridden with e.g. `browser.defaultHeight`.
`padding`	Padding around the widget (in pixels). This parameter specifies the padding for viewing in all contexts (browser and viewer) unless it is specifically overriden by e.g. `browser.padding`.

`viewer.defaultWidth`
: The default width used to display the widget within the RStudio Viewer.

`viewer.defaultHeight`
: The default height used to display the widget within the RStudio Viewer.

`viewer.padding`
: Padding around the widget when displayed in the RStudio Viewer (defaults to 15 pixels).

`viewer.fill`
: When displayed in the RStudio Viewer, automatically size the widget to the viewer dimensions (note that `viewer.padding` is still applied). Default to `TRUE`.

`viewer.suppress`
: Never display the widget within the RStudio Viewer (useful for widgets that require a large amount of space for rendering). Defaults to `FALSE`.

`viewer.paneHeight`
: Request that the RStudio Viewer be forced to a specific height when displaying this widget.

`browser.defaultWidth`
: The default width used to display the widget within a standalone web browser.

`browser.defaultHeight`
: The default height used to display the widget within a standalone web browser.

`browser.padding`
: Padding around the widget when displayed in a standalone browser (defaults to 40 pixels).

`browser.fill`
: When displayed in a standalone web browser, automatically size the widget to the browser dimensions (note that `browser.padding` is still applied). Defaults to `FALSE`.

`browser.external`
: When displaying in a browser, always use an external browser (via `browseURL()`). Defaults to ‘FALSE“, which will result in the use of an internal browser within RStudio v1.1 and higher.

`knitr.defaultWidth`
: The default width used to display the widget within documents generated by knitr (e.g. R Markdown).

`knitr.defaultHeight`
: The default height used to display the widget within documents generated by knitr (e.g. R Markdown).

`knitr.figure`
: Apply the default knitr fig.width and fig.height to the widget when it’s rendered within R Markdown documents. Defaults to `TRUE`.

`fill`
: Whether or not the widget’s container should be treated as a fill item, meaning that its `height` is allowed to grow/shrink to fit a fill container with an opinionated height (see `htmltools::bindFillRole()` for more). Examples of fill containers include `bslib::card()` and `bslib::card_body_fill()`.

Details

The default HTML widget sizing policy treats the widget with the same sizing semantics as an R plot. When printed at the R console the widget is displayed within the RStudio Viewer and sized to fill the Viewer pane (modulo any padding). When rendered inside an R Markdown document the widget is sized based on the default size of figures in the document.

You might need to change the default behavior if your widget is extremely large. In this case you might specify `viewer.suppress = TRUE` and `knitr.figure = FALSE` as well provide for a larger default width and height for knitr.

You also might need to change the default behavior if you widget already incorporates padding. In this case you might specify `viewer.padding = 0`.

For additional details on widget sizing:

`vignette("develop_sizing", package = "htmlwidgets")`

Value

A widget sizing policy

Index

appendContent (prependContent), 9

browseURL(), *13*

createWidget, 3

getDependency, 4

HTML, *9*
htmltools::bindFillRole(), *5*, *13*
htmlwidgets (htmlwidgets-package), 2
htmlwidgets-package, 2
htmlwidgets-shiny, 4

JS, 6

onRender, 6
onStaticRenderComplete, *7*, 8

prependContent, 9

saveWidget, 10
scaffoldWidget, 10
setWidgetIdSeed, 11
shinyRenderWidget (htmlwidgets-shiny), 4
shinyWidgetOutput (htmlwidgets-shiny), 4
sizingPolicy, *3*, 12

tags, *8*, *9*
toJSON, *3*