Package 'r2resize'

November 12, 2025

Type Package

Title In-Text Resize for Images, Tables and Fancy Resize Containers in 'shiny', 'rmarkdown' and 'quarto' Documents

Version 2.0

Maintainer Obinna Obianom <idonshayo@gmail.com>

Description Offers a suite of tools designed to enhance the responsiveness and interactivity of web-based documents and applications created with R. It provides an automatic, configurable resizing toolbar that can be seamlessly integrated with HTML elements such as containers, images, and tables, allowing end-users to dynamically adjust their dimensions. Beyond the toolbar, the package includes a rich collection of flexible, expandable, and interactive container functionalities, such as highly customizable split-screen layouts (splitCard), versatile sizeable cards (sizeableCard), dynamic window-like elements (windowCard), visually engaging emphasis cards (empahsisCard), and sophisticated flexible and elastic card layouts (flexCard, elastiCard). Furthermore, it offers an elegant image viewer and resizer (shinyExpandImage) perfect for interactive galleries. r2resize is particularly well-suited for developers and data scientists looking to create modern, responsive, and user-friendly 'shiny' applications, 'markdown' reports, and 'quarto' documents that adapt gracefully to different screen sizes and user preferences, significantly improving the user experience.

License MIT + file LICENSE

URL https://r2resize.obi.obianom.com

BugReports https://github.com/oobianom/r2resize/issues

Depends R (> 3.6)

Imports utils, shiny, htmltools, quickcode, DT

Suggests rmarkdown, knitr, r2symbols, testthat

Encoding UTF-8

VignetteBuilder knitr

Language en-US

RoxygenNote 7.3.2

Config/testthat/edition 3

NeedsCompilation no

Author Obinna Obianom [aut, cre]

Repository CRAN

Date/Publication 2025-11-12 07:50:07 UTC

Contents

`add.JQuery` *Attach jQuery to an HTML Document for r2resize Components*

Description

This function dynamically includes the jQuery library into an HTML output. It's particularly useful when other functions within the 'r2resize' package, which rely on jQuery, are used in an environment (like a knitted R Markdown document or a Shiny application) where jQuery might not be automatically present or where a specific version is required.

Usage

```
add.JQuery(version = "3.5.1")
```

Arguments

`version`	A character string specifying the desired jQuery version (e.g., "3.5.1", "3.7.0"). Defaults to "3.5.1" if no version is provided. Ensure the version specified is available on the jQuery CDN.

Details

The 'add.JQuery' function generates a '<script>' HTML tag that links to a specified version of the jQuery library from a Content Delivery Network (CDN). By calling this function, you ensure that the necessary jQuery functionalities are available for 'r2resize''s interactive components, such as those used in 'splitCard', 'sizeableCard', or 'windowCard'. This function adds the 'html' and 'character' classes to the output, making it compatible with 'htmltools::htmlDependency' and other HTML rendering contexts in R.

Value

An HTML ‘<script>‘ tag as a character string, with ‘html‘ and ‘character‘ classes, containing the link to the jQuery library.

Note

This function is crucial for maintaining the robustness and reliability of the ‘r2resize‘ package. It acts as a fallback mechanism, ensuring that interactive functionalities remain operational even when jQuery is not explicitly included in the HTML page by default. This enhances the package's ability to gracefully handle diverse HTML environments, facilitating seamless integration and usage.

See Also

splitCard for an example of a ‘r2resize‘ component that might rely on jQuery.

Examples

```
# Attach the default jQuery version (3.5.1)
r2resize::add.JQuery()

# Attach a specific jQuery version (e.g., 3.7.0)
r2resize::add.JQuery("3.7.0")

# This function is often used implicitly by other r2resize components,
# but can be manually added to ensure jQuery is present in a custom HTML context.

# Example in a simple R Markdown chunk (won't render fully without proper setup)
## Not run:
r2resize::add.JQuery()
htmltools::div("This text requires jQuery for some interactive features.")

## End(Not run)
```

---

add.resizer *Configure and add a dynamic resizing toolbar to HTML documents*

Description

This function allows the inclusion and configuration of a responsive toolbar in HTML outputs, enabling users to dynamically resize images and tables within the document. It provides fine-grained control over the toolbar's appearance and behavior, enhancing the interactivity and user experience of R Markdown documents, Shiny applications, or any HTML output.

Usage

```
add.resizer(
  theme.color = NULL,
  position = c("top", "bottom"),
  font.size = NULL,
  font.color = NULL,
  tables = TRUE,
  images = TRUE,
  line.color = NULL,
  thumb.width = NULL,
  thumb.height = NULL,
  line.width = NULL,
  line.height = NULL,
  dim.units = "px",
  default.image.width = NULL
)
```

Arguments

theme.color	A character string specifying the theme color for the resizer and table elements (e.g., "black" or "#000000").
position	A character string indicating the position of the resize toolbar, either "top" or "bottom".
font.size	A character string specifying the font size of the page elements in pixels (e.g., "14px").
font.color	A character string specifying the font color of the page elements (e.g., "dark-blue" or "#006699").
tables	A logical value (TRUE or FALSE). If TRUE, the resize toolbar will be added to HTML tables.
images	A logical value (TRUE or FALSE). If TRUE, the resize toolbar will be added to HTML images.
line.color	A character string specifying the color of the resizer track (e.g., "red" or "#f5f5f5").
thumb.width	A numeric value specifying the width of the resizer thumb.
thumb.height	A numeric value specifying the height of the resizer thumb.
line.width	A numeric value specifying the width of the resizer track.
line.height	A numeric value specifying the height of the resizer track.
dim.units	A character string specifying the unit for the height and width of the track or thumb (e.g., "px").
default.image.width	A character string specifying the default width of all images on the page (e.g., "100%", "500px").

Details

The ‘add.resizer‘ function injects necessary CSS and JavaScript into your HTML document to create interactive resizing capabilities. It dynamically modifies the dimensions of images and tables based on user interaction with the generated toolbar. This is particularly useful for documents where content responsiveness and user-controlled viewing preferences are important.

The function relies on an internal templating system to fetch and customize the CSS and JavaScript files. Parameters like ‘theme.color‘, ‘font.size‘, ‘line.color‘, and dimension-related arguments directly influence the visual styling of the toolbar and the initial appearance of content.

For Shiny applications, this function should be called within the UI to ensure the necessary scripts and styles are loaded when the application starts. In R Markdown documents, simply including a call to ‘add.resizer()‘ will integrate the toolbar into the knitted HTML output.

Value

An HTML script tag containing the necessary CSS and JavaScript for the resizing toolbar, applied as an ‘html‘ object.

Examples for r2resize

More examples and demo pages for this function are located at this link - https://r2resize.obi.obianom.com.

See Also

splitCard, shinyExpandImage

Examples

```
# Default settings: adds resizer to both tables and images at the top
r2resize::add.resizer()

# Add resizer to only images, placed at the bottom of the page
if (interactive()) {
  shiny::shinyApp(
    ui = shiny::fluidPage(
      r2resize::add.resizer(
        tables = FALSE,
        images = TRUE,
        position = "bottom"
      ),
      shiny::tags$img(src = "https://via.placeholder.com/150", width = "100px"),
      shiny::h3("Only images will have a resizer toolbar.")
    ),
    server = function(input, output) {}
  )
}

# Add resizer to only tables with a specific theme color and font size
if (interactive()) {
  shiny::shinyApp(
```

```
    ui = shiny::fluidPage(
      r2resize::add.resizer(
        tables = TRUE,
        images = FALSE,
        theme.color = "darkgreen",
        font.size = "16px"
      ),
      shiny::h3("Table with resizer:"),
      shiny::renderTable(data.frame(
        A = 1:3,
        B = LETTERS[1:3]
      ))
    ),
    server = function(input, output) {}
  )
}

# Customize resizer line color, width, and height, and set default image width
r2resize::add.resizer(
  line.color = "#FF5733",
  line.width = 250,
  line.height = 10,
  thumb.width = 30,
  thumb.height = 30,
  dim.units = "pt",
  default.image.width = "60%"
)

# Full customization example for R Markdown or Shiny
## Not run:
  # In an R Markdown document or Shiny UI:
  add.resizer(
    theme.color = "purple",
    position = "top",
    font.size = "13px",
    font.color = "#4A148C",
    tables = TRUE,
    images = TRUE,
    line.color = "#C2185B",
    thumb.width = 28,
    thumb.height = 28,
    line.width = 200,
    line.height = 8,
    dim.units = "px",
    default.image.width = "75%"
  )

## End(Not run)
```

---

elastiCard *Hover elastic card container*

Description

Creates an automatic elastic card holder, designed for showcasing images or navigation items with a hover-based expansion effect, providing an engaging user experience.

Usage

```
elastiCard(
  ...,
  height.px = NULL,
  width.px = NULL,
  border.color = "white",
  border.width.px = 1,
  active.panel = 1
)
```

Arguments

`...`	A list of image or content containers, where each item is a named vector or list specifying properties like 'bg' (background image URL), 'icon' (Font Awesome icon name), 'title', 'subtitle', 'desc' (longer description), and 'text.color'. See examples for the expected structure of these item lists.
`height.px`	Numeric. The fixed height of the entire 'elastiCard' container in pixels. If 'NULL', the height adjusts automatically.
`width.px`	Numeric. The fixed width of the entire 'elastiCard' container in pixels. If 'NULL', the width adjusts automatically.
`border.color`	Character string. The color of the border for the entire container (e.g., "white", "#RRGGBB").
`border.width.px`	Numeric. The width of the border for the entire container in pixels.
`active.panel`	Numeric. This parameter is retained for consistency with 'flexCard' but does not affect the hover-based behavior of 'elastiCard'.

Details

The 'elastiCard' function provides a dynamic display of multiple content cards that automatically expand on hover. This effect is suitable for interactive galleries, team member profiles, or feature lists where a preview is shown and more details emerge on user interaction. Each card can display a background image, a title, a subtitle, a longer description, and customizable text colors. The hover-based elasticity makes for an intuitive and responsive design.

Value

An HTML 'section' element containing multiple elastic cards with hover-based expansion functionality, suitable for inclusion in Shiny applications or R Markdown documents.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

Ensure that ‘shiny‘ is loaded if using this function within a Shiny application. The ‘elastiCard‘ leverages CSS transitions for its hover effects, providing a smooth user experience. For best visual results, provide images with consistent aspect ratios if using ‘bg‘ properties.

See Also

`flexCard` for click-expandable flexible cards, `shinyExpandImage` for elegant image viewing and resizing.

Other Image and Container Resizing Components: `flexCard()`

Examples

```
## Not run:
if (interactive()) {
  library(shiny)
  library(r2resize)
  library(htmltools)

  ui <- fluidPage(
    tags$h2("Hover Elastic Cards - Text Only"),
    elastiCard(
      item1 = c(
        icon = "brain",
        title = "Cognitive Science",
        subtitle = "Exploring the mind",
        desc = "Lorem ipsum dolor sit amet, consectetur adipiscing elit,
                sed do eiusmod tempor incididunt ut labore et dolore magna aliqua.",
        text.color = "lightblue"
      ),
      item2 = c(
        title = "Neurobiology",
        subtitle = "The science of the nervous system",
        desc = "Ut enim ad minim veniam, quis nostrud exercitation ullamco
                laboris nisi ut aliquip ex ea commodo consequat.",
        text.color = "lightgreen"
      ),
      height.px = 300,
      width.px = 900,
      border.color = "gray",
      border.width.px = 1
    ),
    tags$br(),
    tags$h2("Hover Elastic Cards - With Background Images"),
    elastiCard(
      itemA = c(
        bg = "https://r2resize.obi.obianom.com/m/image1.jpg",
        icon = "chart-pie",
        title = "Data Visualization",
        subtitle = "Making data accessible",
        desc = "Duis aute irure dolor in reprehenderit in voluptate velit esse
```

```
                cillum dolore eu fugiat nulla pariatur."
        ),
        itemB = c(
          bg = "https://r2resize.obi.obianom.com/m/image2.jpg",
          title = "Machine Learning",
          subtitle = "AI-powered insights",
          desc = "Excepteur sint occaecat cupidatat non proident, sunt in culpa qui
                  officia deserunt mollit anim id est laborum.",
          text.color = "white"
        ),
        itemC = c(
          bg = "https://r2resize.obi.obianom.com/m/image3.jpg",
          title = "Cloud Computing",
          subtitle = "Scalable solutions",
          desc = "Sed ut perspiciatis unde omnis iste natus error sit voluptatem
                  accusantium doloremque laudantium.",
          text.color = "yellow"
        ),
        height.px = 350
      )
    )

    server <- function(input, output) {}

    shinyApp(ui, server)
  }

  ## End(Not run)
```

`empahsisCard` *Create an Emphasis Card with a Dynamic Border*

Description

Creates a container that visually emphasizes its content with a subtle, dynamic border effect. This card is designed to draw user attention to important information or sections.

Usage

```
empahsisCard(..., bg.color = NULL)
```

Arguments

`...`	The content to be placed inside the emphasis card. Can be any ‘shiny::tagList‘ or HTML content.
`bg.color`	The background color of the content area. Can be a named R color (e.g., "red", "black") or a hexadecimal color code (e.g., "#333333").

Details

The ‘empahsisCard‘ (and its alias ‘emphasisCard‘) provides a unique visual cue to highlight content. The border of the card subtly animates or changes, indicating that the content within is significant or has a special status. This is ideal for calls to action, important notices, or featured content.

Value

A ‘shiny::div‘ element representing the emphasis card with its dynamic border, ready for inclusion in a Shiny UI or R Markdown output.

More examples for r2resize

More examples and demo pages are located at this link - https://r2resize.obi.obianom.com.

Note

This function requires the ‘shiny‘ package for rendering.

See Also

emphasisCard (alias), splitCard

Other Container Functions: sizeableCard(), splitCard(), splitCard2(), windowCard()

Examples

```
if (interactive()) {
  library(shiny)
  # Simple emphasis card with basic text
  shinyApp(
    ui = fluidPage(
      h2("Emphasis Card Example"),
      empahsisCard(
        shiny::h4("Important Announcement!"),
        shiny::p("Please read this crucial message.")
      )
    ),
    server = function(input, output) {}
  )

  # Emphasis card with a custom background color and multiple elements
  shinyApp(
    ui = fluidPage(
      h2("Styled Emphasis Card"),
      empahsisCard(
        shiny::h3("Featured Product"),
       shiny::img(src = "https://r2resize.obi.obianom.com/m/logo.png", height = "50px"),
        shiny::p("Check out our new amazing product!"),
        bg.color = "#FFEBEE"
      )
    ),
    server = function(input, output) {}
```

```
  )
}
```

`emphasisCard` *Create an Emphasis Card with a Dynamic Border*

Description

Creates a container that visually emphasizes its content with a subtle, dynamic border effect. This card is designed to draw user attention to important information or sections.

Usage

```
emphasisCard(..., bg.color = NULL)
```

Arguments

`...`	The content to be placed inside the emphasis card. Can be any 'shiny::tagList' or HTML content.
`bg.color`	The background color of the content area. Can be a named R color (e.g., "red", "black") or a hexadecimal color code (e.g., "#333333").

Details

The 'empahsisCard' (and its alias 'emphasisCard') provides a unique visual cue to highlight content. The border of the card subtly animates or changes, indicating that the content within is significant or has a special status. This is ideal for calls to action, important notices, or featured content.

Value

A 'shiny::div' element representing the emphasis card with its dynamic border, ready for inclusion in a Shiny UI or R Markdown output.

More examples for r2resize

More examples and demo pages are located at this link - https://r2resize.obi.obianom.com.

Note

This function requires the 'shiny' package for rendering.

See Also

`emphasisCard` (alias), `splitCard`

Other Container Functions: `sizeableCard()`, `splitCard()`, `splitCard2()`, `windowCard()`

Examples

```
if (interactive()) {
  library(shiny)
  # Simple emphasis card with basic text
  shinyApp(
    ui = fluidPage(
      h2("Emphasis Card Example"),
      empahsisCard(
        shiny::h4("Important Announcement!"),
        shiny::p("Please read this crucial message.")
      )
    ),
    server = function(input, output) {}
  )

  # Emphasis card with a custom background color and multiple elements
  shinyApp(
    ui = fluidPage(
      h2("Styled Emphasis Card"),
      empahsisCard(
        shiny::h3("Featured Product"),
       shiny::img(src = "https://r2resize.obi.obianom.com/m/logo.png", height = "50px"),
        shiny::p("Check out our new amazing product!"),
        bg.color = "#FFEBEE"
      )
    ),
    server = function(input, output) {}
  )
}
```

---

flexCard *Flexible card container*

Description

Creates an expandable and flexible card holder, ideal for showcasing images, navigation items, or categorized content in an interactive manner.

Usage

```
flexCard(
  ...,
  height.px = NULL,
  width.px = NULL,
  border.color = "white",
  border.width.px = 1,
  active.panel = 1
)
```

Arguments

...	A list of image or content containers, where each item is a named vector or list specifying properties like 'bg' (background image URL), 'icon' (Font Awesome icon name), 'title', 'subtitle', 'icon.color', and 'text.color'. See examples for the expected structure of these item lists.
height.px	Numeric. The fixed height of the entire 'flexCard' container in pixels. If 'NULL', the height adjusts automatically.
width.px	Numeric. The fixed width of the entire 'flexCard' container in pixels. If 'NULL', the width adjusts automatically.
border.color	Character string. The color of the border for each individual card panel (e.g., "white", "#RRGGBB").
border.width.px	Numeric. The width of the border for each individual card panel in pixels.
active.panel	Numeric. The index (1-based) of the panel that should be initially active (expanded). Use '0' to make all panels initially inactive/collapsed.

Details

The 'flexCard' function generates a visually appealing and interactive set of cards that expand on click. It is particularly useful for dashboards, portfolios, or content sections where space is at a premium but detailed information needs to be accessible. Each card can have its own background image, an icon, a main title, and a subtitle, with customizable colors for icons and text. The function relies on internal CSS and JavaScript to manage the expansion and collapse behavior.

Value

An HTML 'div' element containing multiple flex cards with interactive expansion functionality, suitable for inclusion in Shiny applications or R Markdown documents.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

Ensure that 'shiny' is loaded if using this function within a Shiny application. The 'active.panel' parameter determines which card is initially open; setting it to '0' starts with all cards collapsed.

See Also

elastiCard for hover-elastic cards, shinyExpandImage for elegant image viewing and resizing.

Other Image and Container Resizing Components: elastiCard()

Examples

```
## Not run:
if (interactive()) {
  library(shiny)
  library(r2resize)
  library(htmltools)

  ui <- fluidPage(
    tags$h2("Flexible Cards Example"),
    flexCard(
      item1 = c(
        bg = "https://r2resize.obi.obianom.com/m/image1.jpg",
        icon = "chart-line",
        title = "Market Trends",
        subtitle = "Analysis of current market movements"
      ),
      item2 = c(
        bg = "https://r2resize.obi.obianom.com/m/image2.jpg",
        icon = "flask",
        title = "Research Projects",
        subtitle = "Ongoing studies and experiments",
        icon.color = "blue",
        text.color = "lightgray"
      ),
      item3 = c(
        bg = "https://r2resize.obi.obianom.com/m/image3.jpg",
        icon = "users",
        title = "Team Collaboration",
        subtitle = "Enhancing team productivity",
        icon.color = "green"
      ),
      height.px = 400,
      width.px = 800,
      border.color = "darkgray",
      border.width.px = 2,
      active.panel = 2 # Start with the second panel active
    ),
    tags$br(),
    tags$h2("Flexible Cards with default active panel"),
    flexCard(
      itemA = c(
        title = "Default Panel 1",
        subtitle = "No background image",
        icon = "info-circle"
      ),
      itemB = c(
        title = "Default Panel 2",
        subtitle = "Just text",
        icon = "lightbulb"
      )
    )
  )
```

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

  shinyApp(ui, server)
}

## End(Not run)
```

---

shinyExpandImage *Elegant viewer functionality for images*

Description

Provides an elegant image viewer and resizer for images within Shiny applications.

Usage

```
shinyExpandImage(imageid = c())
```

Arguments

imageid	A character vector containing one or more IDs of HTML 'div' elements that act as containers for images. Each 'div' should contain '<a>' tags, where each '<a>' tag wraps an '<img>' tag. The 'href' attribute of the '<a>' tag should be the source of the high-resolution image to be viewed.

Details

The 'shinyExpandImage' function integrates powerful JavaScript libraries, 'justifiedGallery' and 'lightGallery', to transform static image containers into interactive, user-friendly galleries.

'justifiedGallery' arranges images in a justified grid layout, ensuring aesthetic presentation regardless of image dimensions. It intelligently fills horizontal space, creating a visually appealing and responsive gallery.

'lightGallery' then provides the lightbox functionality. When an image in the gallery is clicked, it expands into a full-screen or modal viewer, offering features such as:

- **Zoom In/Out**: Users can magnify or reduce the image size.
- **Navigation**: Easy transition between multiple images within the same 'imageid' container.
- **Thumbnail Navigation**: A strip of thumbnails at the bottom for quick jumps between images.
- **Download**: Option to download the currently viewed image.
- **Animated Transitions**: Smooth visual effects during image changes.

This function is designed for use in Shiny applications. It takes one or more HTML 'div' element IDs as input, where each 'div' is expected to contain '<a>' tags wrapping '<img>' tags. The 'href' attribute of the '<a>' tag should point to the full-resolution image.

Value

Returns an ‘htmltools::tagList‘ containing an ‘htmltools::htmlDependency‘ for the necessary CSS and JavaScript files (‘imgviewer.css‘, ‘imgviewer.js‘) and multiple ‘htmltools::tags$script‘ elements. These script elements initialize the ‘justifiedGallery‘ and ‘lightGallery‘ functionalities on the specified ‘imageid‘ containers, enabling the interactive image viewer features.

Use case

Use in a shiny application for image(s) that you\'d like to carry a viewer feature on click.

When the image is clicked, it is expanded and toolbars appear to allow the user to in zoom in or out, as well as download the image.

If there are multiple images within the imageid holder, then they are automatically ordered at the bottom for ease of transition.

See Also

flexCard, elastiCard

Examples

```
## Not run:
if (interactive()) {
  library(shiny)
  library(htmltools)
  library(r2resize)

  # Example 1: Simple image gallery with a single container ID
  shinyApp(
    ui = fluidPage(
      h3("Single Image Gallery Example"),
      shinyExpandImage(c("gallery1")), # Initialize viewer for \'gallery1\'
      tags$div(
        id = "gallery1",
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1b.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1b.jpg",
                   alt = "Sample Image 1",
                   style = "width: 150px; height: 100px; object-fit: cover;")
        ),
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1.jpg",
                   alt = "Sample Image 2",
                   style = "width: 150px; height: 100px; object-fit: cover;")
        ),
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1c.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1c.jpg",
                   alt = "Sample Image 3",
```

```
                    style = "width: 150px; height: 100px; object-fit: cover;")
        )
      )
    ),
    server = function(input, output) {}
  )

  # Example 2: Multiple image galleries on the same page
  shinyApp(
    ui = fluidPage(
      h3("Multiple Image Galleries Example"),
      shinyExpandImage(c("myGalleryA", "myGalleryB")), # Initialize for both galleries

      h4("Gallery A: Nature"),
      tags$div(
        id = "myGalleryA",
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1b.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1b.jpg",
                   alt = "Nature Image 1",
                   style = "width: 120px; height: 80px; object-fit: cover;")
        ),
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1c.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1c.jpg",
                   alt = "Nature Image 2",
                   style = "width: 120px; height: 80px; object-fit: cover;")
        )
      ),

      h4("Gallery B: Abstract"),
      tags$div(
        id = "myGalleryB",
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/1.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/1.jpg",
                   alt = "Abstract Image 1",
                   style = "width: 100px; height: 100px; object-fit: cover;")
        ),
        tags$a(
          href = "https://r2resize.obi.obianom.com/m/2.jpg",
          tags$img(src = "https://r2resize.obi.obianom.com/m/2.jpg",
                   alt = "Abstract Image 2",
                   style = "width: 100px; height: 100px; object-fit: cover;")
        )
      )
    ),
    server = function(input, output) {}
  )
}

## End(Not run)
```

`sizeableCard` *Resizable Container Content Holder with Size Controls*

Description

Creates a highly customizable container that holds content and provides a mini toolbar on the right for adjusting the content's display size (small, medium, large).

Usage

```
sizeableCard(..., bg.color = NULL, border.color = NULL)
```

Arguments

`...`	The content to be placed inside the sizeable card. Can be any 'shiny::tagList' or HTML content.
`bg.color`	The background color of the content area. Can be a named R color (e.g., "red", "black") or a hexadecimal color code (e.g., "#333333").
`border.color`	The border color of the container. Can be a named R color or a hexadecimal color code.

Details

The 'sizeableCard' function is designed to present content in a flexible box that users can scale using intuitive "A" (small, medium, large) buttons integrated into a toolbar. This is useful for displaying text, images, or other UI elements where the user might want to adjust their size without altering the entire page layout. It provides a simple, self-contained resizing mechanism.

Value

A 'shiny::div' element representing the sizeable container with a resizing toolbar, ready for inclusion in a Shiny UI or R Markdown output.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

This function requires the 'shiny' package for rendering.

See Also

`splitCard`, `splitCard2`, `windowCard`

Other Container Functions: `empahsisCard()`, `splitCard()`, `splitCard2()`, `windowCard()`

Examples

```
if (interactive()) {
  library(shiny)
  # Simple sizeable card with default settings
  shinyApp(
    ui = fluidPage(
      h2("Basic Sizeable Card"),
      sizeableCard(
        shiny::p("This is some sample text within a sizeable card."),
       shiny::img(src = "https://r2resize.obi.obianom.com/m/image1.jpg", height = "100px"),
        shiny::p("Use the controls on the right to change its size.")
      )
    ),
    server = function(input, output) {}
  )

  # Sizeable card with custom background and border colors
  shinyApp(
    ui = fluidPage(
      h2("Styled Sizeable Card"),
      sizeableCard(
        shiny::h4("My Report Summary"),
        shiny::p("This card contains important information about a project."),
        shiny::em("Adjust the size as needed."),
        bg.color = "#F0F4C3",
        border.color = "#C0CA33"
      )
    ),
    server = function(input, output) {}
  )
}
```

---

splitCard *Resizable split screen container*

Description

Creates a highly customizable and resizable split screen container for arranging UI elements side-by-side or top-and-bottom.

Usage

```
splitCard(
  left,
  right,
  splitter.color = NULL,
  bg.left.color = NULL,
  left.bg.url = NULL,
  right.bg.url = NULL,
```

```
  bg.right.color = NULL,
  border.color = NULL,
  position = c("vertical", "horizontal"),
  text.left.color = "black",
  text.right.color = "black",
  min.height = NULL,
  left.width = NULL
)
```

Arguments

`left`	The content to be displayed in the left (or top, if 'position = "horizontal"') panel. Can be any 'shiny::tagList' or HTML content.
`right`	The content to be displayed in the right (or bottom, if 'position = "horizontal"') panel. Can be any 'shiny::tagList' or HTML content.
`splitter.color`	The color of the draggable splitter line. Can be a named R color (e.g., "red", "black") or a hexadecimal color code (e.g., "#333333").
`bg.left.color`	The background color of the left panel. Can be a named R color or a hexadecimal color code.
`left.bg.url`	An optional URL for a background image for the left panel (e.g., "image1.png" or "https://example.com/image1.png").
`right.bg.url`	An optional URL for a background image for the right panel (e.g., "image1.png" or "https://example.com/image1.png").
`bg.right.color`	The background color of the right panel. Can be a named R color or a hexadecimal color code.
`border.color`	The border color of the entire container. Can be a named R color or a hexadecimal color code.
`position`	The orientation of the splitter. Can be "vertical" (left/right split) or "horizontal" (top/bottom split). Defaults to "vertical".
`text.left.color`	The text color for the content within the left panel.
`text.right.color`	The text color for the content within the right panel.
`min.height`	The minimum height of the entire split container (e.g., "200px", "50vh").
`left.width`	The initial width of the left panel (when 'position = "vertical"') or height of the top panel (when 'position = "horizontal"'). Can be a percentage (e.g., "50%") or a fixed pixel value (e.g., "500px").

Details

The 'splitCard' function provides a dynamic way to present two distinct sections of content within a single, resizable container. Users can drag the splitter to adjust the visible area of each panel, making it ideal for comparisons, dashboards, or any scenario requiring flexible content layout. The 'position' argument allows switching between a left/right split and a top/bottom split, offering versatility in design. It's particularly useful within Shiny applications or R Markdown documents where interactive layouts are desired.

Value

A ‘shiny::div‘ element representing the resizable split screen container, ready for inclusion in a Shiny UI or R Markdown output.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

This function requires the ‘shiny‘ package for rendering and interactive functionality.

See Also

splitCard2, sizeableCard, windowCard

Other Container Functions: empahsisCard(), sizeableCard(), splitCard2(), windowCard()

Examples

```
if (interactive()) {
  library(shiny)
  # Basic vertical split card with default settings
  shinyApp(
    ui = fluidPage(
      h2("Basic Split Card"),
      splitCard(
        shiny::div(h3("Left Panel"), p("Content for the left side.")),
        shiny::div(h3("Right Panel"), p("Content for the right side."))
      )
    ),
    server = function(input, output) {}
  )

  # Horizontal split card with custom colors and minimum height
  shinyApp(
    ui = fluidPage(
      h2("Horizontal Split Card with Custom Styling"),
      splitCard(
        shiny::div(h3("Top Panel (Blue)"), p("Content for the top section.")),
        shiny::div(h3("Bottom Panel (Green)"), p("Content for the bottom section.")),
        bg.left.color = "#E0F2F7",
        bg.right.color = "#E8F5E9",
        splitter.color = "#7CB342",
        position = "horizontal",
        min.height = "300px",
        border.color = "#4CAF50"
      )
    ),
    server = function(input, output) {}
  )
```

```
  # Vertical split card with background images and specific widths
  shinyApp(
    ui = fluidPage(
      h2("Split Card with Background Images"),
      splitCard(
        shiny::div(h3("Image Background Left"), p("Some text over an image.")),
        shiny::div(h3("Image Background Right"), p("More text over another image.")),
        left.bg.url = "https://r2resize.obi.obianom.com/m/image1.jpg",
        right.bg.url = "https://r2resize.obi.obianom.com/m/image2.jpg",
        text.left.color = "white",
        text.right.color = "black",
        left.width = "30%",
        min.height = "450px"
      )
    ),
    server = function(input, output) {}
  )
}
```

---

splitCard2 *Resizable Split Screen Container Version 2 (Fixed Slider)*

Description

Creates a highly customizable and resizable split screen container with a fixed, non-draggable slider position. This version is ideal for presenting two content areas with a pre-defined division.

Usage

```
splitCard2(
  left,
  right,
  bg.left.color = NULL,
  bg.right.color = NULL,
  border.color = NULL,
  text.left.color = "black",
  text.right.color = "black",
  slider.position = charNum1to100
)
```

Arguments

left	The content to be displayed in the left panel. Can be any ‘shiny::tagList‘ or HTML content.
right	The content to be displayed in the right panel. Can be any ‘shiny::tagList‘ or HTML content.
bg.left.color	The background color of the left panel. Can be a named R color (e.g., "red", "black") or a hexadecimal color code (e.g., "#333333").

`bg.right.color` The background color of the right panel. Can be a named R color or a hexadecimal color code.

`border.color` The border color of the entire container. Can be a named R color or a hexadecimal color code.

`text.left.color`
The text color for the content within the left panel.

`text.right.color`
The text color for the content within the right panel.

`slider.position`
The fixed position of the slider as a percentage from 1 to 100 (e.g., "40" for 40% left panel width). Defaults to "80".

Details

Unlike 'splitCard', 'splitCard2' provides a static split where the division between the left and right content areas is set by the 'slider.position' and cannot be interactively adjusted by the user. This makes it suitable for layouts where the proportional display of content is fixed. Common use cases include presenting questions and answers, code alongside output, or two related pieces of information with a predetermined visual hierarchy.

Value

A 'shiny::div' element representing the split screen container style 2, ready for inclusion in a Shiny UI or R Markdown output.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

This function requires the 'shiny' package for rendering.

See Also

splitCard, sizeableCard, windowCard

Other Container Functions: empahsisCard(), sizeableCard(), splitCard(), windowCard()

Examples

```
if (interactive()) {
  library(shiny)
  # Basic split card 2 with a 40\% left panel
  shinyApp(
    ui = fluidPage(
      h2("Basic Fixed Split Card"),
      splitCard2(
        shiny::div(h1("Question:"), p("What is the capital of France?")),
        shiny::div(h1("Answer:"), p("Paris.")),
```

```
      slider.position = "40",
      bg.left.color = "#FFFDE7",
      bg.right.color = "#E8F5E9"
    )
  ),
  server = function(input, output) {}
)

# Split card 2 with custom text and border colors
shinyApp(
  ui = fluidPage(
    h2("Styled Fixed Split Card"),
    splitCard2(
      shiny::div(h4("Left Side"), p("Detailed information here.")),
      shiny::div(h4("Right Side"), p("Corresponding summary or data.")),
      bg.right.color = "white",
      bg.left.color = "#F0F4C3",
      border.color = "#FFC107",
      text.left.color = "darkgreen",
      text.right.color = "darkblue",
      slider.position = "60"
    )
  ),
  server = function(input, output) {}
)
}
```

`windowCard` *Resizable, Moveable, and Expandable Window Card*

Description

Creates an easily expandable, resizable, and moveable window-like container for content, mimicking a desktop window within your Shiny application or HTML output.

Usage

```
windowCard(
  ...,
  title = "Sample title",
  width = "50%",
  bg.color = NULL,
  border.color = NULL,
  header.text.color = NULL,
  body.text.color = NULL
)
```

Arguments

...	The content to be placed inside the window card. Can be any ‘shiny::tagList‘ or HTML content.
title	The title displayed in the header of the window card.
width	The initial width of the window card (e.g., "50%", "600px").
bg.color	The background color of the content area within the window card. Can be a named R color or a hexadecimal color code.
border.color	The border color of the entire window card. Can be a named R color or a hexadecimal color code.
header.text.color	The text color of the title in the header.
body.text.color	The text color of the content within the card’s body.

Details

The ‘windowCard‘ function is a versatile UI component that allows for highly interactive content display. Users can drag the window around the page, resize it from its edges, and expand/collapse its content. This is particularly useful for pop-up information, draggable dashboards, or interactive panels in complex Shiny applications. The window initially appears centered on the screen.

Value

A ‘shiny::div‘ element representing the moveable, resizable, and expandable window card.

Examples for r2resize

More examples and demo pages are located at this link - https://rpkg.net/package/r2resize.

Note

Due to the underlying JavaScript implementation and reliance on specific DOM IDs, only one ‘windowCard‘ should be created per page to ensure proper functionality and avoid conflicts. This function requires the ‘shiny‘ package for rendering and interactive functionality.

See Also

splitCard, splitCard2, sizeableCard

Other Container Functions: empahsisCard(), sizeableCard(), splitCard(), splitCard2()

Examples

```
if (interactive()) {
  library(shiny)
  # Simple window card with default attributes
  shinyApp(
    ui = fluidPage(
      h2("Interactive Window Card"),
```

```
    windowCard(
      shiny::h3("Welcome!"),
      shiny::p("This is a draggable and resizable window."),
      shiny::actionButton("closeBtn", "Close Window")
    )
  ),
  server = function(input, output) {
    observeEvent(input$closeBtn, {
      # Example: How you might handle closing (requires custom JS for actual close)
      showNotification("Window close requested (functionality not built-in)")
    })
  }
)

# Custom styled window card with a plot
shinyApp(
  ui = fluidPage(
    h2("Styled Window Card with Plot"),
    windowCard(
      title = "Dynamic Plot Window",
      width = "600px",
      bg.color = "#E8F5E9",
      border.color = "#4CAF50",
      header.text.color = "white",
      body.text.color = "#333333",
      shiny::plotOutput("myPlot")
    )
  ),
  server = function(input, output) {
    output$myPlot <- shiny::renderPlot({
      hist(rnorm(100), col = "skyblue", border = "white", main = "Random Normal Data")
    })
  }
)
}
```

Index