Package ‘drawer’

August 19, 2022

Title An Interactive HTML Image Editing Tool

Version 0.2.0.1

Date 2022-08-19

Description An interactive image editing tool that can be added as part of the HTML in Shiny, R markdown or any type of HTML document. Often times, plots, photos are embedded in the web application/file. 'drawer' can take screenshots of these image-like elements, or any part of the HTML document and send to an image editing space called 'canvas' to allow users immediately edit the screenshot(s) within the same document. Users can quickly combine, compare different screenshots, upload their own images and maybe make a scientific figure.

Depends R (>= 4.0.0)

Imports htmltools, magrittr, glue, bsplus, utils, shiny, stringr

Suggests testthat, ggplot2

License GPL (>= 3)

Encoding UTF-8

BugReports `https://github.com/lz100/drawer/issues`

URL `https://github.com/lz100/drawer`

RoxygenNote 7.2.1

Config/testthat/edition 3

NeedsCompilation no

Author Le Zhang [aut, cre]

Maintainer Le Zhang <`lezhang100@gmail.com`>

Repository CRAN

Date/Publication 2022-08-19 18:30:06 UTC

R topics documented:

`canvas` *drawer main canvas workbench compinent*

Description

use this function on Shiny UI or R markdown to create the image editing area.

Usage

```
canvas(
  canvasID,
  title = "drawer",
  height = "100vh",
  width = "100%",
  logo_src = "drawer/img/drawer.png",
  log_link = "https://github.com/lz100/drawer",
  on_start = TRUE,
  rmarkdown = FALSE
)
```

Arguments

`canvasID`	string, an unique HTML ID
`title`	string, title of the canvas
`height`	string, css value of initial height of the canvas, like "100vh" for full height current window, "50vh" for half.
`width`	string, css value of initial width of the canvas
`logo_src`	string, link of an image you want to display as logo on the top left
`log_link`	string, a link, when the logo is clicked, where should it jump to
`on_start`	`TRUE` or a CSS selector. See details
`rmarkdown`	bool, are you using inside R markdown? If yes, drawer will copy all image icons that required by the canvas to current directory to `./drawer/img/...`

Details

outside Shiny or Rmarkdown:

If you are not working in Shiny or R markdown, you need to add the required full "Bootstrap3" javascript and CSS + latest "jquery" dependencies by yourself.

height and width:

There are two options for canvas height and width:

- dynamic CSS units like "100vh" (viewpoint height), "100vw" (view point width), or "100%" for both. This kind of units adapt to all kinds of user screen settings.

- fixed unit, `px` (pixels). This does not change across users, but fixes the `on_start` problem (read below).

height:

- `height`, css style `vh` is safer than `%` is not safe, unless the parent has some defined height, "%" will work. Otherwise, if the parent height is "auto" or not defined, and you choose "100%", canvas will still have 0 height.

Width usually does not have this problem. As long as an element is displayed, it has some width.

`on_start`:

This argument specify if you want to initiate the canvas when the document is loaded. If `TRUE`, then when the document loading is done, start the canvas. The problem is if you set the height to be "vh" (view height) units and if the canvas is hidden, like in a different tab and not displayed on start, the view height is 0, because it is hidden on another tab (display property is `none`), so it will cause the canvas cannot be initiated properly.

The solution is to bind the initiation with a clicking event, like on a tab or a button. For example, make a button on the second tab and bind `on_start` to that button: `on_start = "#buttonID"`. Then when users click on that button, canvas initiate. Remember this is a Jquery CSS selector, which means you need to append "#" in front your button ID.

If you want to do it automatically, like clicking on a certain tab, some CSS knowledge may required. For example, in Shiny, you can use shiny::tabsetPanel to create a tab panel.

```
tabsetPanel(id = "tabs",
    tabPanel("Tab A", value = "A", ...),
    tabPanel("Tab B", value = "B", ...),
    ...
)
```

Then, bind to it `canvas(on_start = '#tabs li a[data-value="B"]', ...)`. This means we are selecting the element with ID "tabs", which is the main tabsetPanel ID, then a list item (`li`) which is the tab titles you see on UI, and finally, the link jump to tab B, (`a[data-value="B"]`). See examples for a real case.

Another way to fix it is by given the `height` and `width` a fixed pixel unit:

```
canvas(
    canvasID = "canvas_f",
    height = "900px",
    width = "1500px"
)
```

Upload your own image to canvas:

You can drag your own images to the canvas. Support major image formats, like "jpg", "png", "svg", "gif", "webp", "bmp". Moving images like "gif", "webp" will be animated on left side preview, but will not move on canvas.

Value

a HTML component to be added to a Shiny app or document

Examples

```
# basic usage
if(interactive()){
  library(shiny)

  ui <- fluidPage(
    h3("Try to drag pictures locally to canvas"),
    canvas("canvas_a")
  )

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

  }

  shinyApp(ui, server)
}

# multiple canvas on a page
if(interactive()){
  library(shiny)

  ui <- fluidPage(
    h3("multiple canvas on a page"),
    p("They are independent"),
    p("Dragging from one canvas to another is not supported currently"),
    column(6, canvas("canvas_left")),
    column(6, canvas("canvas_right"))
  )

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

  }

  shinyApp(ui, server)
}

# with capture buttons buttons
if(interactive()){
  library(shiny)
  library(ggplot2)
  ui <- fluidPage(
    fluidRow(
      id = "new_row",
      column(
        6,
        h3("this is a title"),
        column(6, tags$label("plot 1"), plotOutput("plot_1")),
        column(6, tags$label("plot 2"), plotOutput("plot_2")),
      ),
      column(
        6,
        h2("To canvas buttons"),
        h4("pure button with `toCanvasBtn`"),
        toCanvasBtn(
          dom = "plot_1",
          label = "capture plot 1",
          canvasID = "canvas_b"
        ), br(),
        toCanvasBtn(
          dom = "capture_button",
          label = "capture this button itself",
          canvasID = "canvas_b",
          id = "capture_button"
        ), br(),
        toCanvasBtn(
          dom = "#new_row .col-sm-6:first-of-type",
          label = "complex selector to select left column",
          canvasID = "canvas_b",
          isID = FALSE
        ), br(),
        h4("button text input for any part of document with `toCanvasBtn`"),
        toCanvasTextBtn(
          label = "try #plot_2 to for plot 2 or other selector",
          canvasID = "canvas_b",
          text_value = "#plot_2"
        )
      )
    ),
    canvas("canvas_b")
  )

  server <- function(input, output, session) {
    output$plot_1 <- renderPlot({
      ggplot(mpg, aes(cty, hwy)) +
        geom_count(col="tomato3", show.legend=F)
    })
    output$plot_2 <- renderPlot({
      ggplot(mpg, aes(cty, hwy)) +
        geom_point()
    })
  }

  shinyApp(ui, server)
}

# start canvas as hidden, initiate later in tab panels
if(interactive()){
  library(shiny)

  ui <- fluidPage(
    tabsetPanel(
      id = "tabs",
      tabPanel(
        "Home page",
        value = "tab_1",
        h4("Content on home page ...."),
        p("Canvas is hidden on start, go to other tabs")
      ),
      tabPanel(
        "Canvas C",
        value = "tab_2",
        markdown(
          '
          # canvas hidden on start
          In this example, you will see if the canvas is hidden,
          not on the first tab in a `tabsetPanel`, or other similar
          UI where you do not see canvas on start. Then, the canvas
          cannot be initiate properly using the default height value (100vh).
          Using the dynamic computed CSS height like "100%", or "100vh" with "hidden"
          (display = none) elements give the height of `0` on start.
          So, you **should not see the canvas** on this tab, but a broken
          structure and no canvas grid.

          To fix it, either give it a fixed `height` and `width` pixel unit, like

          - `height = "800px"`, `width = "1500px"`

          or bind the initiation event to a click of a button, the tab title or
          any other element you specify with the `on_start` argument. See the
          example code and watch how we do it in "canvas D-F".
          '
        ),
        canvas(canvasID = "canvas_c")
      ),
      tabPanel(
        "Canvas D",
        value = "tab_3",
        h4("Initiate canvas by a button"),
        actionButton("start_canvas", "Start Canvas C"),
        canvas(
          canvasID = "canvas_d",
          on_start = "#start_canvas"
        )
      ),
      tabPanel(
        "Canvas E",
        value = "tab_4",
        h4("Initiate canvas by clicking tab title"),
        p("Canvas initiate when first time users come to this tab"),
        canvas(
          canvasID = "canvas_e",
          on_start = "#tabs li a[data-value='tab_4']"
        )
      ),
      tabPanel(
        "Canvas F",
        value = "tab_5",
        h4("Initiate canvas with fixed height and width"),
        canvas(
          canvasID = "canvas_f",
          height = "800px",
          width = "1500px"
        )
      )
    )
  )

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

  }

  shinyApp(ui, server)
}
```

`toCanvasBtn` *Screenshot HTML elements to canvas button*

Description

A bootstrap button that allows users to take a screenshot of specified HTML element (usually an image) and send it to the drawer canvas for editing. In addition, you can download it as "png" or "jpg" by opening up the dropdown menu.

Usage

```
toCanvasBtn(
  dom,
  canvasID,
  isID = TRUE,
  id = "",
  label = "To Canvas",
  color_class = "primary"
)
```

Arguments

`dom`	a HTML DOM selector, mostly common is to select the element by ID: e.g. a plot with ID "plot1", to select, use `dom = "plot1"` to select the plot if `isID = TRUE`. If `isID = FALSE`, use `dom = "#plot1"` Other complex selector is supported. First turn `isID = FALSE`, then try things like `dom = ".btn i"` selects an icon inside an element with "btn" class. If more than one element is matched, only the first one will be screenshoted.
`canvasID`	string, the ID of canvas. Unlike `dom`, you should not add "#" for `canvasID` even if `isID=FALSE`
`isID`	bool, `TRUE` if you want to select the `dom` by HTML ID, `FALSE` if the selector is other than ID.

`id`	string, ID for this button, optional
`label`	label of this button, optional
`color_class`	bootstrap button color class suffix, usually one of 'default', 'primary', 'info', 'success', 'warning', 'danger'

Details

This component will not work unless a drawer canvas has been loaded on current document.

Value

a button

Examples

```
# see example of "canvas", `?canvas`
```

`toCanvasTextBtn` *Screenshot HTML elements to canvas text-button*

Description

Unlike toCanvasBtn only screenshot a defined element, this function can take screenshot of any element you specify in the text box and sent to canvas by using Jquery selector format.

Usage

```
toCanvasTextBtn(
  canvasID,
  label = "",
  text_value = "",
  placeholder = "type a selector",
  tooltip = "Screenshot any element to drawer canvas",
  placement = "bottom",
  btn_label = "To Canvas",
  color_class = "primary",
  style = ""
)
```

Arguments

`canvasID`	string, the ID of canvas.
`label`	string, label of the whole group, on the top
`text_value`	string, nitial value of the text input
`placeholder`	string, placeholder text of the text input
`tooltip`	a tooltip of the group

`placement`	where should the tooltip go?
`btn_label`	text on the button
`color_class`	bootstrap button color class suffix, usually one of 'default', 'primary', 'info', 'success', 'warning', 'danger'
`style`	additional CSS style of the group, like "width: 50%"

Details

This component will not work unless a drawer canvas has been loaded on current document.

Input selector:

The selector uses Jquery selector.

- If you are not familiar with it, just remember the mostly commonly used is the element ID, which is the `inputID`, `ID` arguments in most Shiny components. Jquery selector is almost the same for ID, but requires you to add "#" in front, "#element-ID".
- If you have no idea about shiny or HTML selector, right click on the element, and click inspect, you should see the document HTML code in the inspector and the element you want should be highlighted. Find the attribute of "*id*", that's what you need. Again, append "#" in front of that value. Some elements do not have an "id", in this case, you need some advanced selectors. Learn about them by Google "CSS selector".

Value

a shiny input group

Examples

```
# see the example of "canvas", `?canvas`
```

Index