{bootstraplib} package basically provides tools for theming shiny and rmarkdown from R via Bootstrap (3 or 4) Sass. If you’re not aware of [Bootstrap](https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)), it’s one of the most popular (open-source) css framework available on the web.

While {bootstraplib} has got a lot of things, this post is going to be about one thing that I love the most which is “Live Theme Preview” where we can simply edit the values of CSS properties and see the change in Real time. This changes are then printed on your RStudio Console as R code which can be used in your Shiny app / RMarkdown and Tada you’ve just upgraded your boring Shiny app / RMarkdown to a charming Shiny app.

**Package:**

library(bootstraplib)

**Steps to enable Real Time / Live Bootstrap themer for your Shiny app**

Considering that you’ve got an existing shiny app.

1. Load the library with library(bootstraplib)
2. Add the below code to initialize a bootstrap theme in your ui.R

bs\_theme\_new(bootswatch = "sketchy")

1. Add the theme preview function inside your Shiny Server function of server.R. Note this is only required as long as you need the Live theme Customization feature and Run the app (as usually you run it)

bootstraplib::bs\_themer()

1. While you’re customizing the theme, the corresponding R code gets printed on your Console which you can copy and add it in your ui.R just below where you’ve created the bootstrap theme.

bs\_theme\_new(bootswatch = "sketchy")

bs\_theme\_add\_variables(`body-bg` = "#2CEC11")

bs\_theme\_add\_variables(`body-bg` = "#2CEC11", `body-color` = "#292123")

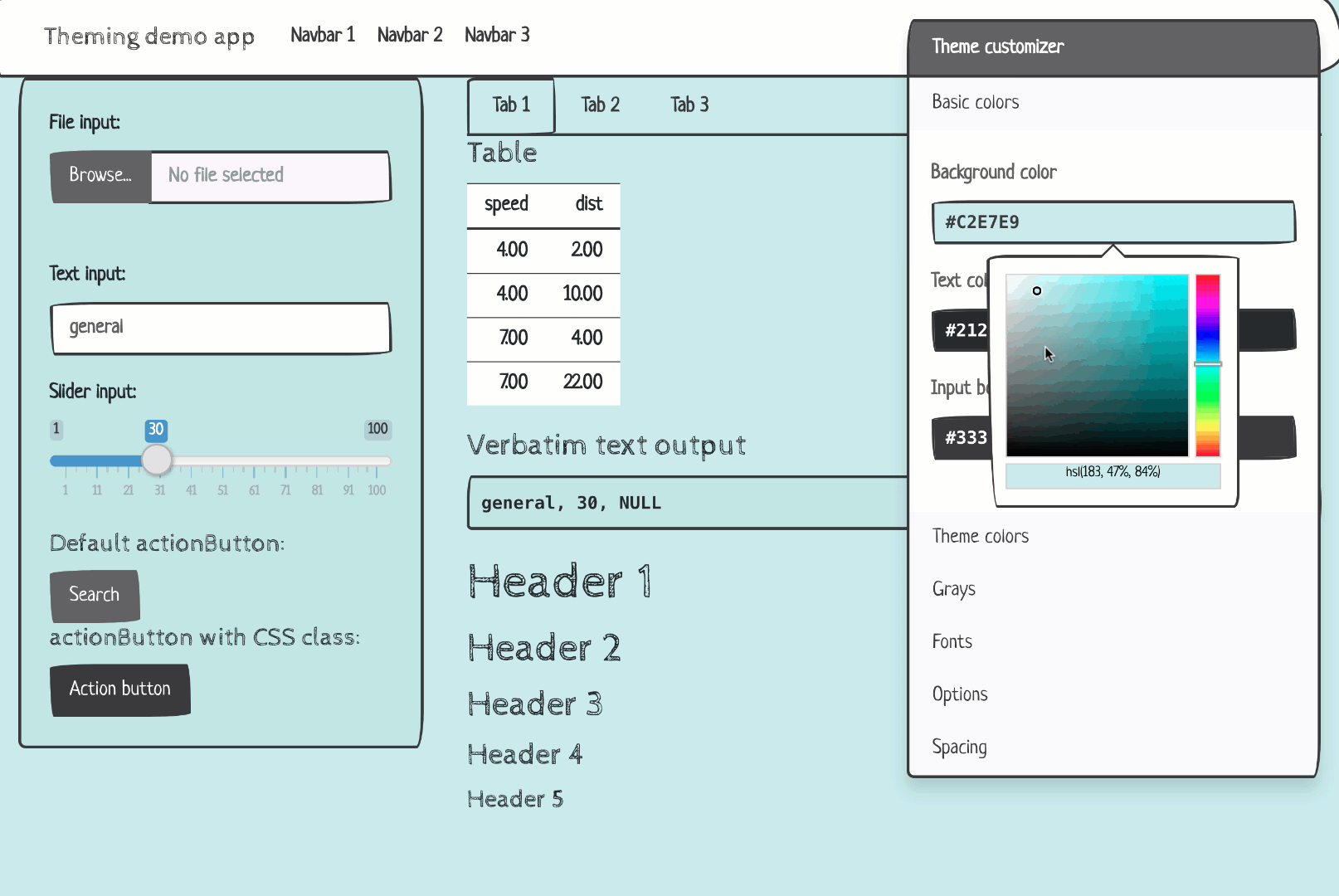
bs\_theme\_add\_variables(`body-bg` = "#2CEC11", `body-color` = "#F61D49")

1. Don’t forget to remove or comment out the live preview function you had added in the Server.R’s Shiny server function (Step 3)

#bootstraplib::bs\_themer()

**Sample Screenshot of Live Theme Editing**

**Bootstraplib’s Documentation Demo**



This not only can help you in beautifying the looks of your Shiny app / RMarkdown but also can open up multiple arenas to create Organziation-theme specific reports from RMarkdown and dashboards from Shiny, Thus making Organiztions embrace reproducible reports/apps than the never-die Powerpoint.

**Sample Shiny Code with Bootstraplib Live Theme Editor**

**ui.R**

#

# This is the user-interface definition of a Shiny web application. You can

# run the application by clicking 'Run App' above.

#

# Find out more about building applications with Shiny here:

#

library(shiny)

bs\_theme\_new(bootswatch = "sketchy")

bs\_theme\_add\_variables(`body-bg` = "#2CEC11")

bs\_theme\_add\_variables(`body-bg` = "#2CEC11", `body-color` = "#292123")

bs\_theme\_add\_variables(`body-bg` = "#2CEC11", `body-color` = "#F61D49")

# Define UI for application that draws a histogram

shinyUI(fluidPage(

bootstrap(),

# Application title

titlePanel("Old Faithful Geyser Data"),

# Sidebar with a slider input for number of bins

sidebarLayout(

sidebarPanel(

sliderInput("bins",

"Number of bins:",

min = 1,

max = 50,

value = 30)

),

# Show a plot of the generated distribution

mainPanel(

plotOutput("distPlot")

)

)

))

**server.R**

#

# This is the server logic of a Shiny web application. You can run the

# application by clicking 'Run App' above.

#

# Find out more about building applications with Shiny here:

#

library(shiny)

library(bootstraplib)

# Define server logic required to draw a histogram

shinyServer(function(input, output) {

# uncomment this when you want live previewing / editing of this theme

#bootstraplib::bs\_themer()

output$distPlot <- renderPlot({

# generate bins based on input$bins from ui.R

x <- faithful[, 2]

bins <- seq(min(x), max(x), length.out = input$bins + 1)

# draw the histogram with the specified number of bins

hist(x, breaks = bins, col = 'darkgray', border = 'white')

})

})