A while ago researching on creating a splash screen for a Shiny application. My gut feel was that there will readily be a package available for this activity. I was surprised to see that not much information is available based on a 10 minute Google search. It also discovered the waiter package which is very cool.

In this article, I present a solution using Shiny modules and the shinyjs package. I am not sure the solution is scalable for a very complex application, but should work for simple applications. The gif file used in the demo application was downloaded from giphy.

The key ideas to do this are very simple

* create a module for the UI and server which forms your main application
* create another module to display an image as a splash screen
* display the image for a few seconds and then hide it
* the main application UI should be hidden by default and should be shown a few miliseconds after the splash screen has been hidden

The last two tasks are accomplished using functions from the shinyjs package.

For the demo application, I took the default code created by RStudio when you create a Shiny project. This is a simple visualization of the *faithful* dataset, where the number of bins in the histogram are controlled by a slider input. The code for this, with some modifications, are defined in the module *mainapp*.

mainapp\_ui <- function(id) {

ns <- NS(id)

fluidPage(

# Application title

hidden(div(id = ns("app\_title"),

titlePanel("Old Faithful Geyser Data"))),

# Application UI elements

hidden(

fluidRow(id = ns("app\_slider\_plot"),

column(

4,

sliderInput(ns("bins"),

"Number of bins:",

min = 1,

max = 50,

value = 30)

),

column(

8,

plotOutput(ns("distPlot"))

)

)

)

)

}

mainapp\_server <- function(input, output, session) {

delay(ms = 3500, show("app\_title"))

delay(ms = 3800, show("app\_slider\_plot"))

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')

})

}

I have converted the *sidebarLayout* into a single *fluidRow* divided into two columns. Note that the row has been provided an id of *app\_slider\_plot* and is hidden by default. The same is true of the *titlePanel* – as an id cannot be defined for a *titlePanel*, this has been wrapped in a HTML *div*. In the server function for this module, we use the *show* function from the shinyjs package to display the title and the application UI elements, but only after a delay of around three and a half seconds. As we see below, this is because the initial splash screen is shown for three seconds.

splash\_ui <- function(id) {

ns <- NS(id)

div(id = ns("splash\_screen"), img(src = "giphy.gif"),

style = "text-align:center; padding-top:250px;")

}

splash\_server <- function(input, output, session) {

hide("splash\_screen", anim = TRUE, animType = "fade", time = 3)

}

The module for the splash screen is really simple. It just loads a GIF image which is animated to provide the appearance that the app is being loaded. Creative people will also include some kind of logo and branding as part of this image. The code in the server function ensures that the image is hidden after three seconds. This action is also animated to make it slightly cooler.

The UI and server for the full Shiny application is really simple.

# Define UI for application that draws a histogram

ui <- fluidPage(

useShinyjs(),

fluidRow(splash\_ui("splash\_module")),

fluidRow(mainapp\_ui("mainapp\_module"))

)

# Define server logic required to draw a histogram

server <- function(input, output) {

ss <- callModule(splash\_server, "splash\_module")

ma <- callModule(mainapp\_server, "mainapp\_module")

}

The *useShinyjs* function is required to enable shinyjs. Other than that, it simply calls the modules for the main application and the splash screen.