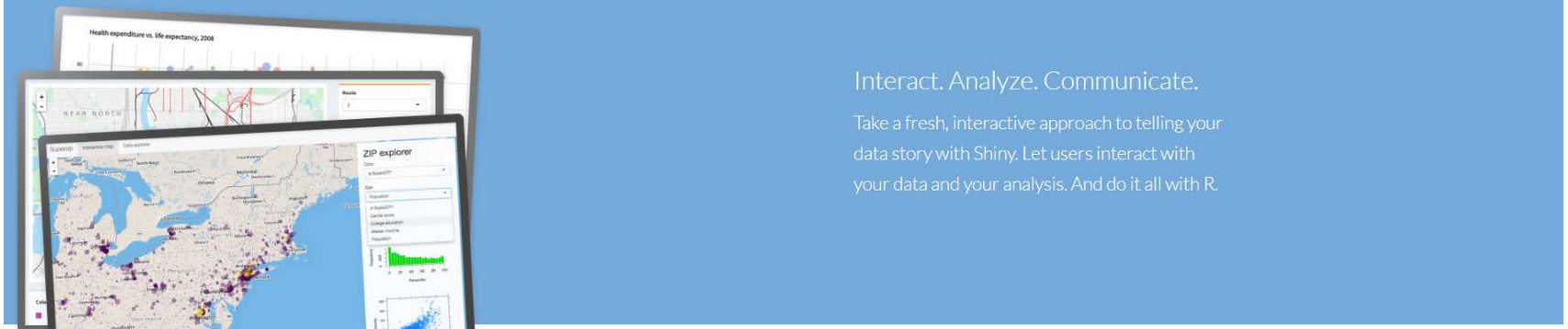
R 기반 웹 프로그래밍
-Rshiny Package-

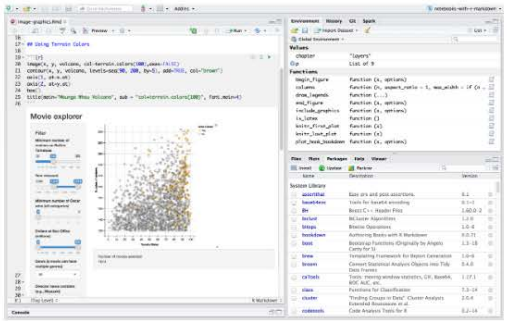
한국환경정책·평가연구원 (KEI)
부연구위원 진대용

R Shiny

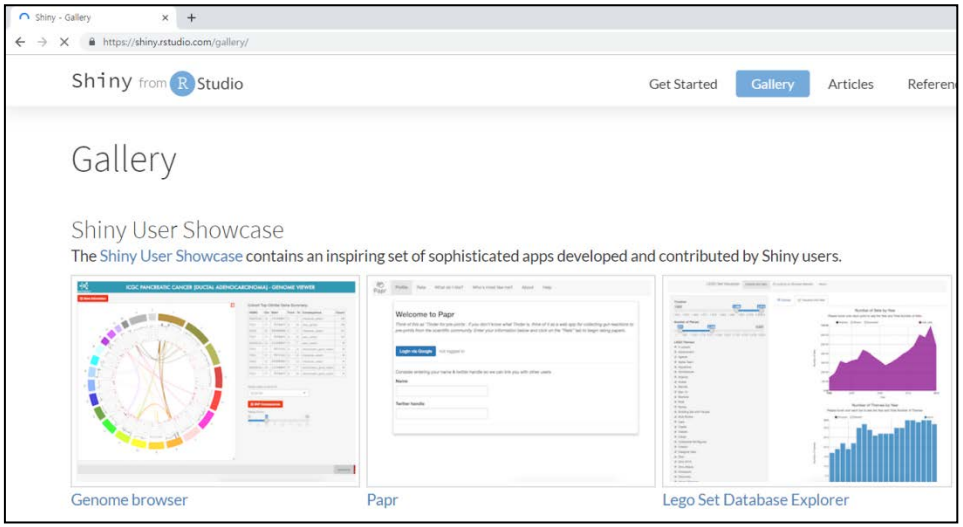
- R에 기반한 웹 프로그래밍 패키지

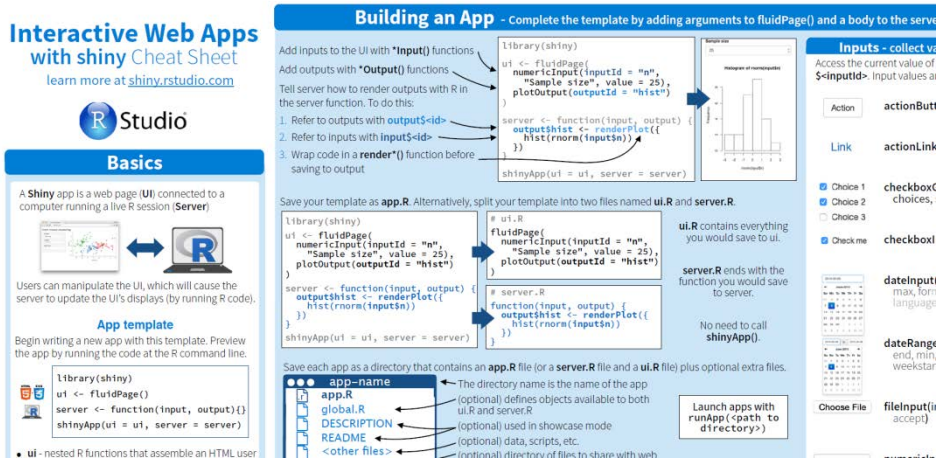


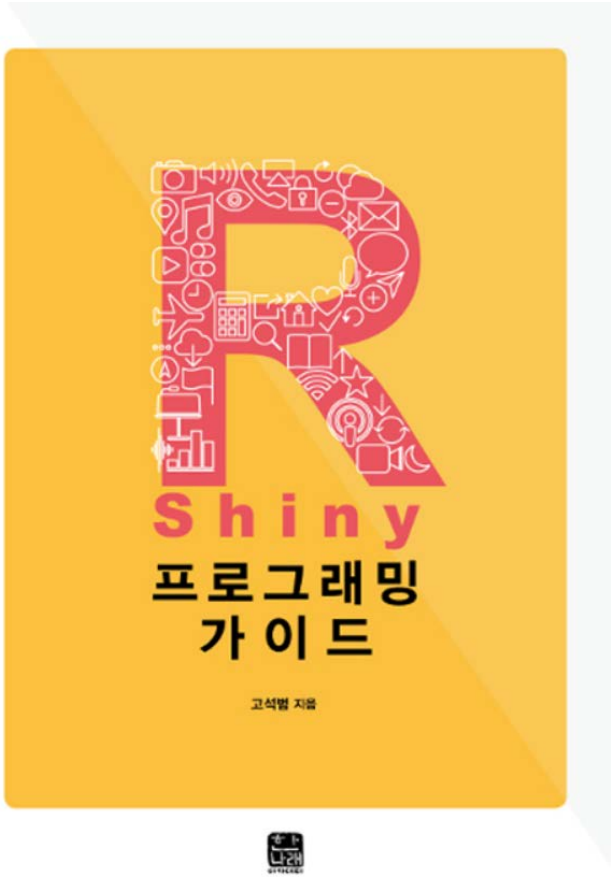
Shiny is an R package that makes it easy to build interactive web apps straight from R. You can host standalone apps on a webpage or embed them in R Markdown documents or build dashboards. You can also extend your Shiny apps with CSS themes, htmlwidgets, and JavaScript actions.



참고자료


Shiny - Gallery
https://shiny.rstudio.com/gallery/
Shiny from RStudio
Get Started
Gallery
Articles
Gallery
Shiny User Showcase
The Shiny User Showcase contains an inspiring set of sophisticated apps developed and contributed by Shiny users.
Genome browser
Papr
Lego Set Database Explorer


Interactive Web Apps with shiny Cheat Sheet
learn more at shiny.rstudio.com
RStudio
Basics
A Shiny app is a web page (UI) connected to a computer running a live R session (Server)
Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).
App template
Begin writing a new app with this template. Preview the app by running the code at the R command line.
library(shiny)
ui <- fluidPage()
server <- function(input, output){}
shinyApp(ui = ui, server = server)
Building an App - Complete the template by adding arguments to fluidPage() and a body to the serve
Add inputs to the UI with *Input() functions
Add outputs with *Output() functions
Tell server how to render outputs with R in the server function. To do this:
1. Refer to outputs with output$<id>
2. Refer to inputs with input$<id>
3. Wrap code in a render*() function before saving to output
Save your template as app.R. Alternatively, split your template into two files named ui.R and server.R.
ui.R contains everything you would save to ui.
server.R ends with the function you would save to server.
No need to call shinyApp().
Save each app as a directory that contains an app.R file (or a server.R file and a ui.R file) plus optional extra files.
app-name
app.R
global.R
DESCRIPTION
README
<other files>
The directory name is the name of the app
(optional) defines objects available to both ui.R and server.R
(optional) used in showcase mode
(optional) data, scripts, etc.
Launch apps with runApp(<path to directory>)
Inputs - collect va
Action
actionButt
Link
actionLink
Choice 1
Choice 2
Choice 3
checkboxG
choices,
Check me
checkboxI
dateInput
dateRange
end, min,
weekstar
Choose File
fileInput(
accept)


Shiny
프로그래밍
가이드
고석범 지음

Shiny 기본 뼈대

```
library(shiny)
ui <- fluidPage{

  # 입출력 위젯

}
```

ui.R

선호하는 프로그래밍 언어는?

- C
- 파이썬
- R

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

shinyApp(ui,server)
```

server.R

첫 웹 프로그램 : Hello World

ui.R* server.R

```
1 library(shiny)
2 ui <- fluidPage(
3
4   # 입출력 위젯
5   titlePanel('Hello World')
6 )
```

ui.R* server.R*

```
1 server <- function(input,output,session)
2 {
3   # 서버 코드
4 }
5
6 shinyApp(ui,server)
```

입력 위젯함수

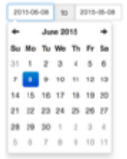
Action — **actionButton**(inputId, label, icon, …)

Link — **actionLink**(inputId, label, icon, …)

Choice 1
Choice 2
Choice 3 — **checkboxGroupInput**(inputId, label, choices, selected, inline)

Check me — **checkboxInput**(inputId, label, value)

dateInput(inputId, label, value, min, max, format, startview, weekstart, language)



dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator)

Choose File — **fileInput**(inputId, label, multiple, accept)

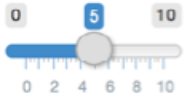
1 — **numericInput**(inputId, label, value, min, max, step)

1 — **numericInput**(inputId, label, value, min, max, step)

••••••••• — **passwordInput**(inputId, label, value)

Choice A
Choice B
Choice C — **radioButtons**(inputId, label, choices, selected, inline)

Choice 1
Choice 1
Choice 2 — **selectInput**(inputId, label, choices, selected, multiple, selectize, width, size) (also **selectizeInput()**)



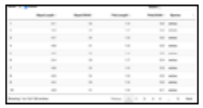
sliderInput(inputId, label, min, max, value, step, round, format, locale, ticks, animate, width, sep, pre, post)

Apply Changes — **submitButton**(text, icon)
(Prevents reactions across entire app)

Enter text — **textInput**(inputId, label, value)

출력 위젯함수

Outputs - render*() and *Output() functions work together to add R output to the UI



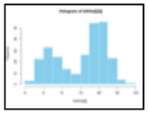
DT::**renderDataTable**(expr, options, callback, escape, env, quoted)

works with

dataTableOutput(outputId, icon, …)

renderImage(expr, env, quoted, deleteFile)

imageOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

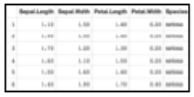


renderPlot(expr, width, height, res, …, env, quoted, func)

plotOutput(outputId, width, height, click, dblclick, hover, hoverDelay, hoverDelayType, brush, clickId, hoverId, inline)

renderPrint(expr, env, quoted, func, width)

verbatimTextOutput(outputId)



renderTable(expr,…, env, quoted, func)

tableOutput(outputId)

foo

renderText(expr, env, quoted, func)

textOutput(outputId, container, inline)

renderUI(expr, env, quoted, func)

uiOutput(outputId, inline, container, …)
& **htmlOutput**(outputId, inline, container, …)

TextInput widget

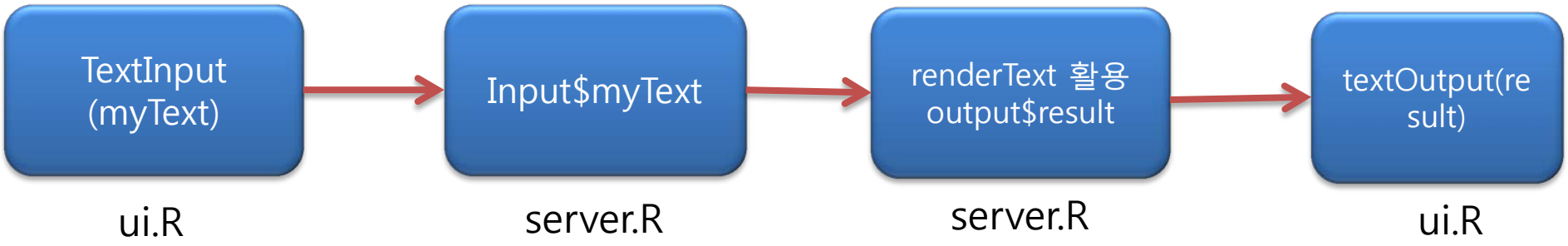
ui.R server.R

```r
library(shiny)
ui <- fluidPage(
  # 입출력 위젯
  titlePanel('Hello World'),
  textInput("myText","텍스트를 입력하세요"),
  textOutput("result")
)
```

ui.R server.R

```r
server <- function(input,output,session)
{
  # 서버 코드
  output$result <- renderText({
    input$myText
  })
}

shinyApp(ui,server)
```

TextInput widget 출력과정



반응성 맥락

- 반응성 맥락
  - 여러 객체(변수)들이 엮여 있는것
  - 보통의 방법으로는 객체에 접근 불가

NumericInput widget

server.R* ui.R

```
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderText({
    input$myNum
  })
}

shinyApp(ui,server)
```

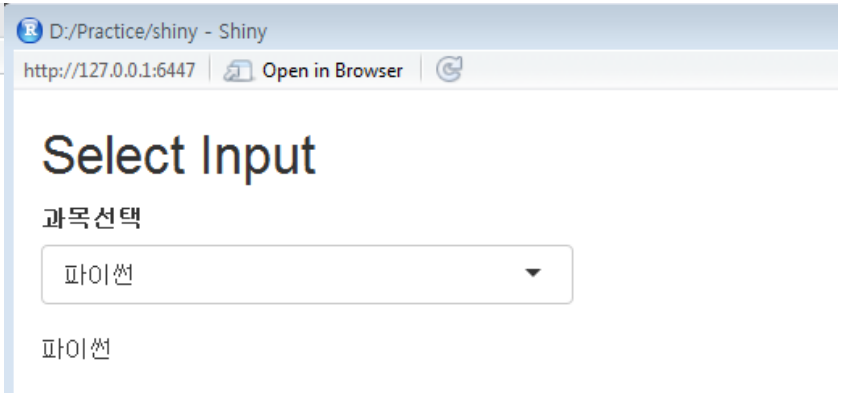
server.R* ui.R

```
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(
  # 입출력 위젯
  titlePanel("Numeric Input"),
  numericInput("myNum","숫자를 입력하세요:",10),
  textOutput("result")
)

```

SelectInput widget

ui.R* server.R*

```
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderText({
    input$sel
  })
}

shinyApp(ui,server)
```



ui.R* server.R* Run App

```
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(

  # 입출력 위젯
  titlePanel("Select Input"),
  selectInput("sel","과목선택",
              choices = list('프로그래밍' = c('C++','파이썬','R'),
                             '수학' = c('미적','통계','벡터'))),
  textOutput("result"))
```

Switch 구문

- switch(2,"red","green","blue")
- switch("color", "color" = "red", "shape" = "square", "length" = 5)

https://www.datamentor.io/r-programming/switch-function/

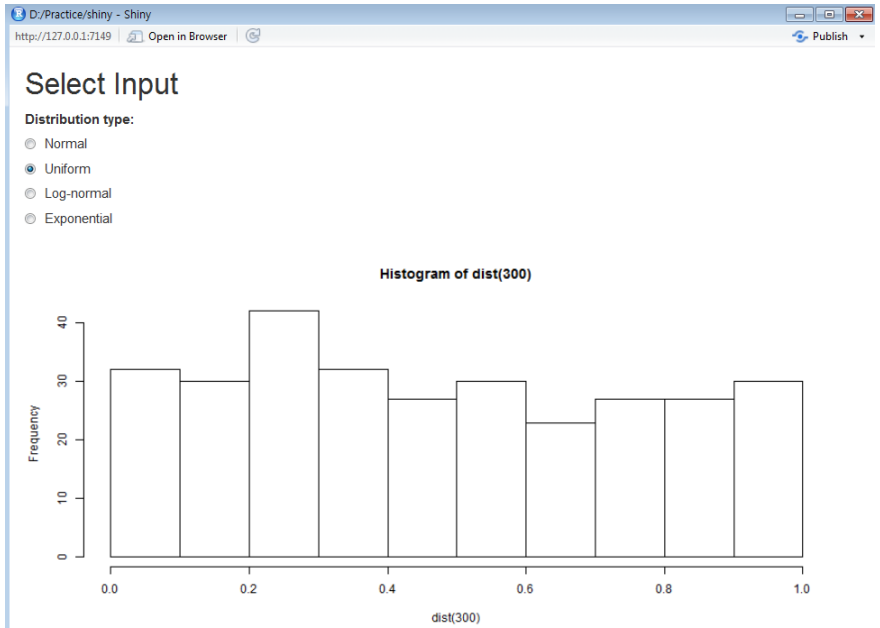
RadioButtons widget

server.R × ui.R* ×

```r
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(

  # 입출력 위젯
  titlePanel("Select Input"),
  radioButtons("dist", "Distribution type:",
               c("Normal" = "norm",
                 "Uniform" = "unif",
                 "Log-normal" = "lnorm",
                 "Exponential" = "exp")),
  plotOutput("result"))
```

server.R* × ui.R* ×

```r
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderPlot({
      print(input$dist)
      dist <- switch(input$dist,
                     norm = rnorm,
                     unif = runif,
                     lnorm = rlnorm,
                     exp = rexp,
                     rnorm)
      hist(dist(300))
  })
}
shinyApp(ui,server)
```



Checkbox widget

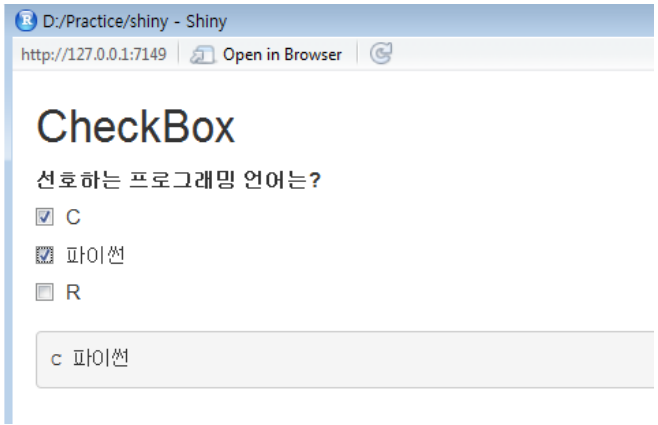
server.R × ui.R* ×

Run App

```
1  library(shiny)
2
3  setwd("D:/Practice/shiny")
4  ui <- fluidPage(
5
6    # 입출력 위젯
7    titlePanel("Select Input"),
8
9    checkboxGroupInput("subj", "선호하는 프로그래밍 언어는?",
10                      c("C", '파이썬','R')),
11   verbatimTextOutput("result")
12 )
```

server.R × ui.R* ×

```
1  setwd("D:/Practice/shiny")
2  server <- function(input,output,session)
3  {
4    output$result <- renderText({
5      input$subj
6    })
7  }
8  shinyApp(ui,server)
```



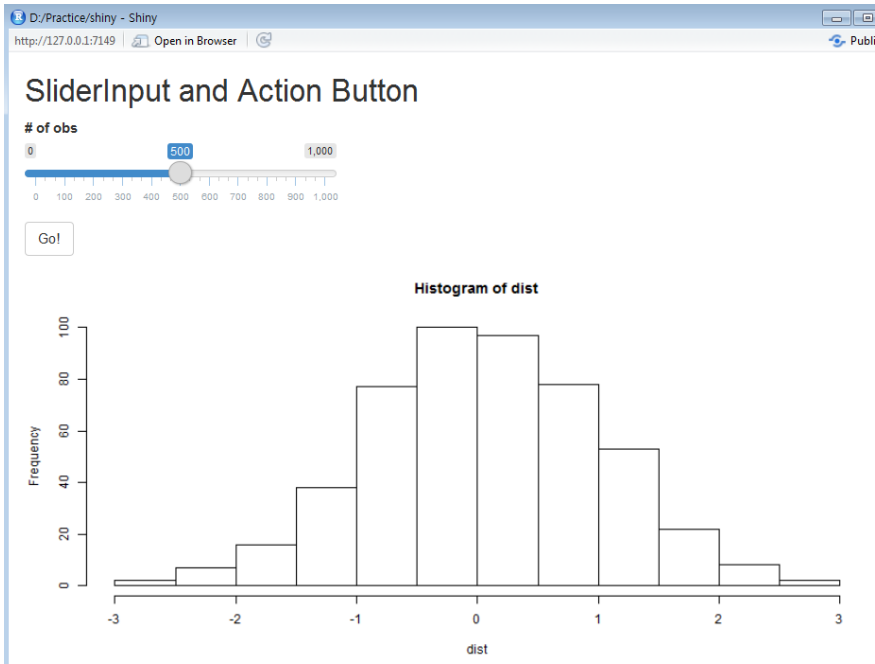
ActionButton Widget

ui.R server.R

```r
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(

  # 입출력 위젯
  titlePanel("SliderInput and Action Button"),

  sliderInput('si','# of obs', 0, 1000, 500),
  actionButton('goBtn', 'Go!'),
  plotOutput('result')
)
```

ui.R server.R

```r
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderPlot({
    # goBtn이 바뀔경우 재실행
    input$goBtn
    print(input$goBtn)

    dist <- isolate(rnorm(input$si))
    hist(dist)
  })
}
shinyApp(ui,server)
```



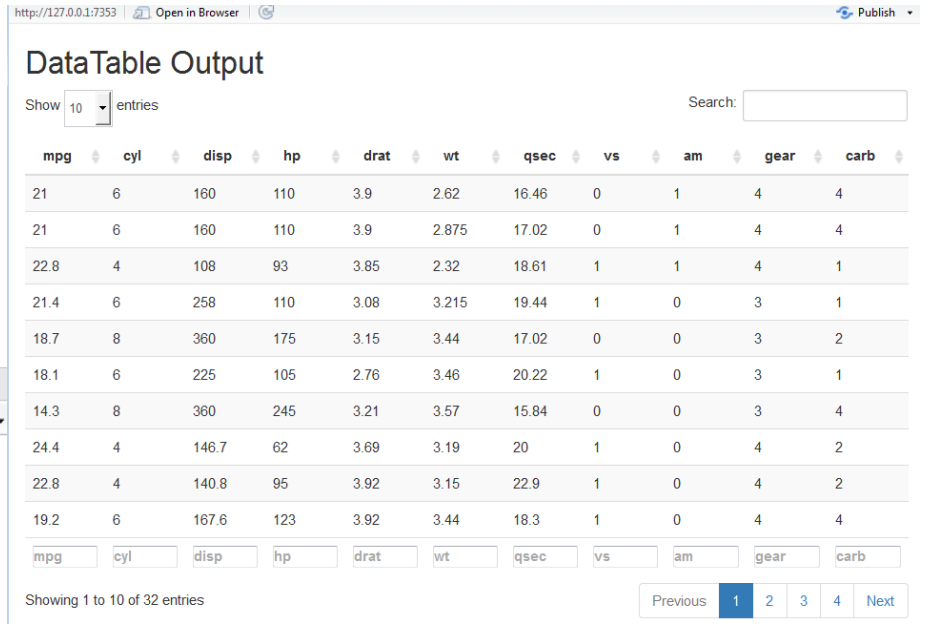
DataTableOutput

ui.R* server.R*

```r
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(

  # 입출력 위젯
  titlePanel("DataTable Output"),
  dataTableOutput("result")
)
```

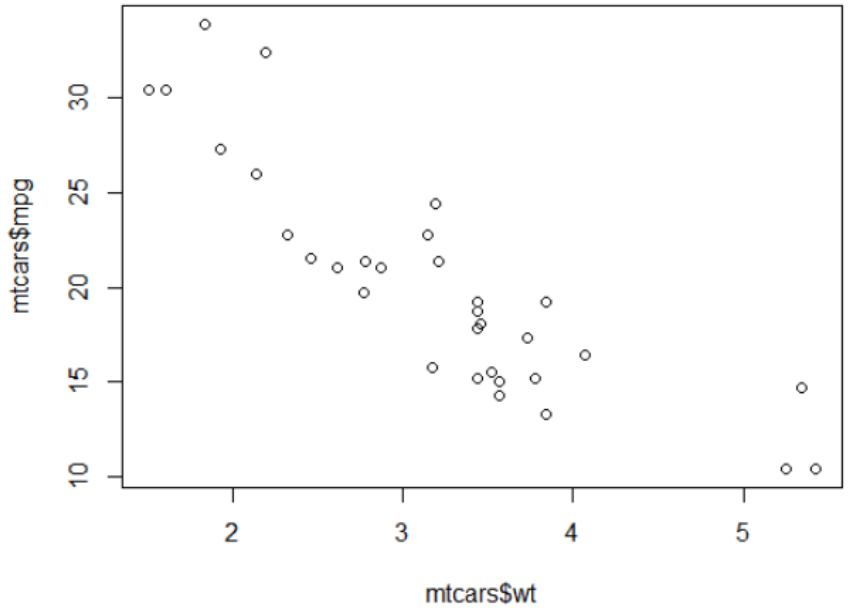
ui.R* server.R* Run App

```r
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderDataTable(
    {result <- mtcars
     result
     },
    options = list(pageLength=10))
}

shinyApp(ui,server)
```



PlotOutput

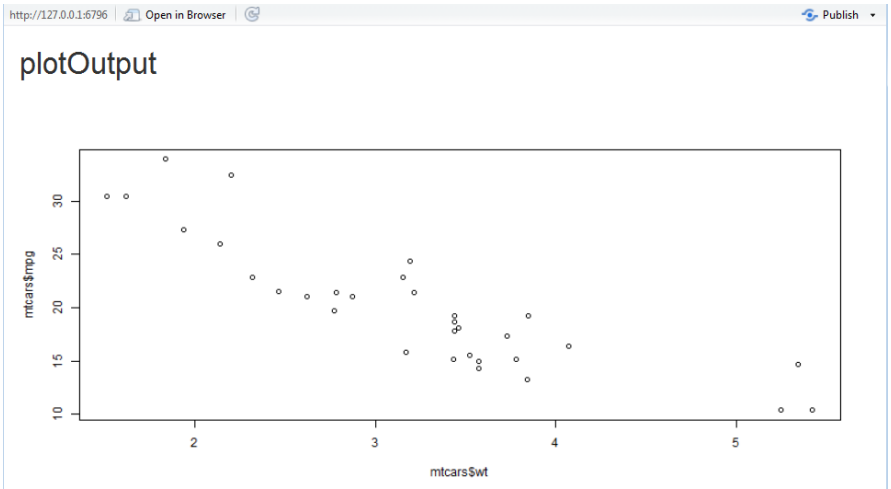
```
plot(mtcars$wt, mtcars$mpg)
```



PlotOutput

server.R | ui.R — Run App

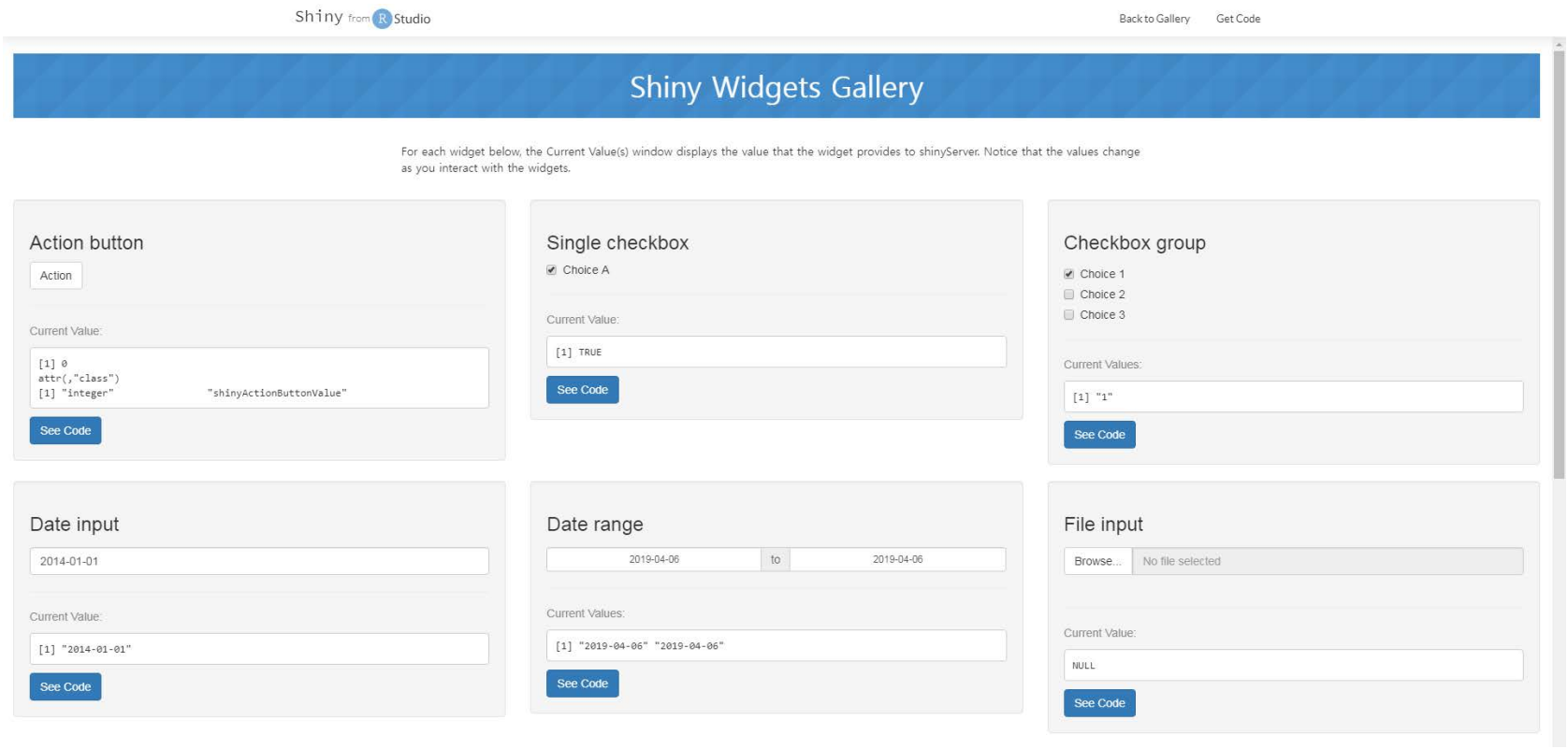
```r
setwd("D:/Practice/shiny")
server <- function(input,output,session)
{
  output$result <- renderPlot({
    plot(mtcars$wt, mtcars$mpg)
  })
}

shinyApp(ui,server)
```



server.R | ui.R — Ru

```r
library(shiny)

setwd("D:/Practice/shiny")
ui <- fluidPage(

  # 입출력 위젯
  titlePanel("plotOutput"),
  plotOutput("result")
)
```

Widget Gallery



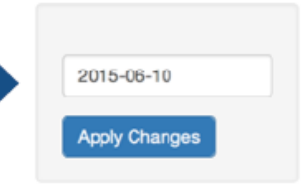
https://shiny.rstudio.com/gallery/widget-gallery.html

레이아웃 (Layout)

Layouts

Combine multiple elements into a "single element" that has its own properties with a panel function, e.g.

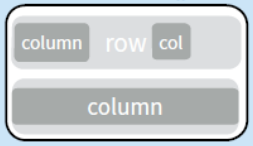
```
wellPanel(
  dateInput("a", ""),
  submitButton()
)
```



absolutePanel()	inputPanel()	tabPanel()
conditionalPanel()	mainPanel()	tabsetPanel()
fixedPanel()	navlistPanel()	titlePanel()
headerPanel()	sidebarPanel()	wellPanel()

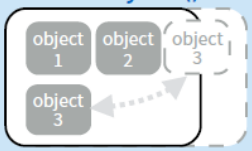
Organize panels and elements into a layout with a layout function. Add elements as arguments of the layout functions.

fluidRow()




```
ui <- fluidPage(
  fluidRow(column(width = 4),
    column(width = 2, offset = 3)),
  fluidRow(column(width = 12))
)
```

flowLayout()

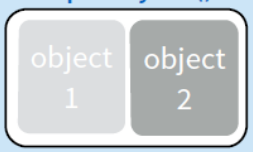



```
ui <- fluidPage(
  flowLayout( # object 1,
              # object 2,
              # object 3
  )
)
```

sidebarLayout()

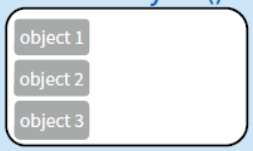

```
ui <- fluidPage(
  sidebarLayout(
    sidebarPanel(),
    mainPanel()
  )
)
```

splitLayout()




```
ui <- fluidPage(
  splitLayout( # object 1,
               # object 2
  )
)
```

verticalLayout()

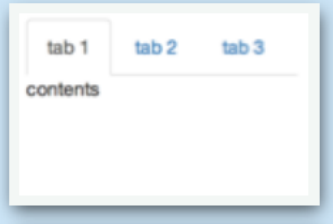



```
ui <- fluidPage(
  verticalLayout( # object 1,
                  # object 2,
                  # object 3
  )
)
```

Layer tabPanels

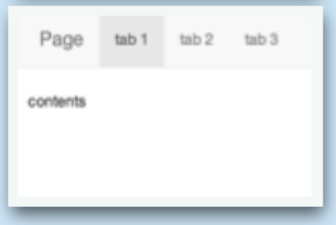
Layer tabPanels on top of each other, and navigate between them, with:

```
ui <- fluidPage( tabsetPanel(
    tabPanel("tab 1", "contents"),
    tabPanel("tab 2", "contents"),
    tabPanel("tab 3", "contents")))
```




```
ui <- fluidPage( navlistPanel(
    tabPanel("tab 1", "contents"),
    tabPanel("tab 2", "contents"),
    tabPanel("tab 3", "contents")))
```


```
ui <- navbarPage(title = "Page",
    tabPanel("tab 1", "contents"),
    tabPanel("tab 2", "contents"),
    tabPanel("tab 3", "contents"))
```



Reactive()

ui.R* server.R*

```
1  library(shiny)
2  ui <- fluidPage(
3    textInput("a", ""),
4    textInput("b", ""),
5    textOutput("z")
6  )
```

ui.R* server.R*

```
1  server <- function(input,output)
2  {
3    re <- reactive({
4      paste(input$a,input$b)
5    })
6
7    output$z <- renderText({
8      re()
9    })
10 }
11
12 shinyApp(ui, server)
```

ObserveEvent()

server.R* ui.R*

```
1 library(shiny)
2 ui <- fluidPage(
3   textInput("a","입력"),
4   actionButton("go", "GO!"),
5   textOutput("z")
6 )
```

server.R* ui.R*

```
1 server <- function(input,output){
2     observeEvent(input$go,{
3       output$z <- renderText({
4         input$a
5       })
6     })
7 }
8
9 shinyApp(ui, server)
```

Example 확인

- runExample()

그 외 알아야 할 내용

- isolate() 함수
- observer() 함수
- eventReactive() 함수
- reactiveValues() 함수
- reactiveVal() 함수
- 테마
- ....