Shiny : : CHEAT SHEET

Building an App

A **Shiny** app is a web page (**ui**) connected to a computer running a live R session (**server**).

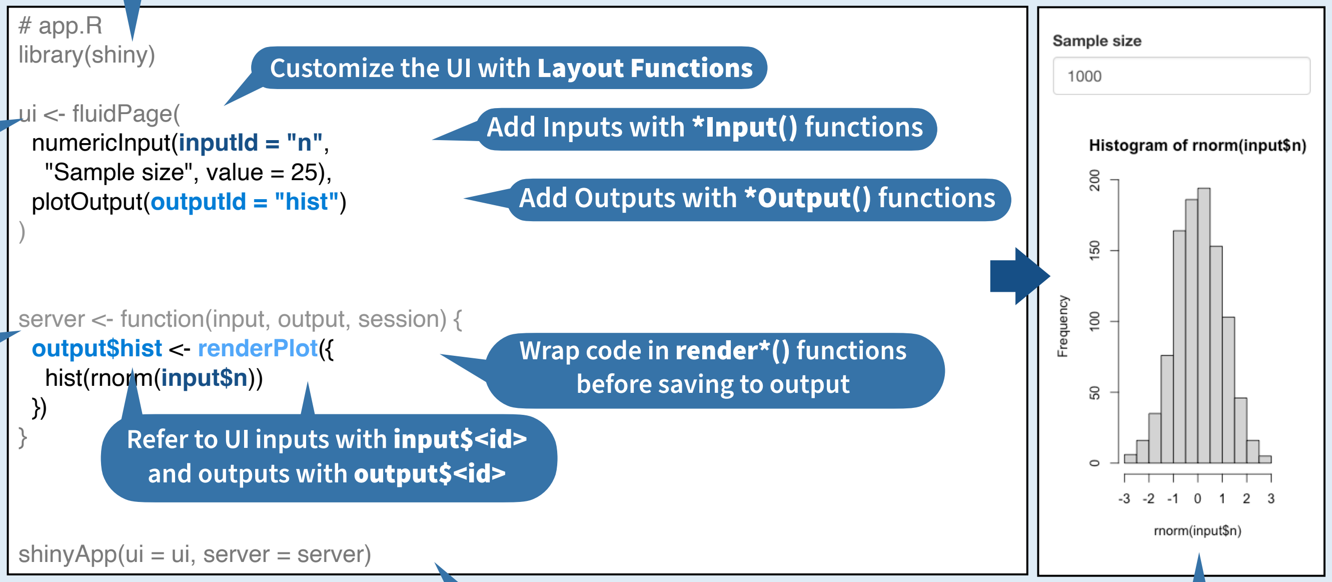
Users can manipulate the UI, which will cause the server to update the UI's displays (by running R code).

Save your template as **app.R**. Keep your app in a directory along with optional extra files.

To generate the template, type shinyapp and press Tab in the RStudio IDE or go to File > New Project > New Directory > Shiny Web Application

In ui nest R functions to build an HTML interface

Tell the server how to render outputs and respond to inputs with R



```r
# app.R
library(shiny)

ui <- fluidPage(
  numericInput(inputId = "n",
    "Sample size", value = 25),
  plotOutput(outputId = "hist")
)

server <- function(input, output, session) {
  output$hist <- renderPlot({
    hist(rnorm(input$n))
  })
}

shinyApp(ui = ui, server = server)
```

Customize the UI with **Layout Functions**
Add Inputs with ***Input() functions**
Add Outputs with ***Output() functions**
Wrap code in **render*() functions** before saving to output
Refer to UI inputs with **input$<id>** and outputs with **output$<id>**
Sample size
1000
Histogram of rnorm(input$n)
Frequency
rnorm(input$n)

Call shinyApp() to combine ui and server into an interactive app!

```
app-name
file  app.R
file  DESCRIPTION
file  README
fo    R/
fo    www/
```

- app-name: The directory name is the app name
- DESCRIPTION, README: (optional) used in showcase mode
- R/: (optional) directory of supplemental .R files that are sourced automatically, must be named "**R**"
- www/: (optional) directory of files to share with web browsers (images, CSS, .js, etc.), must be named "**www**"

Launch apps stored in a directory with **runApp(**<path to directory>**)**.

See annotated examples of Shiny apps by running **runExample(**<example name>**)**. Run **runExample()** with no arguments for a list of example names.

Share

Share your app in three ways:

1. **Host it on shinyapps.io**, a cloud based service from RStudio. To deploy Shiny apps:

   Create a free or professional account at **shinyapps.io**

   Click the Publish icon in RStudio IDE, or run: **rsconnect::deployApp(**"<path to directory>"**)**

2. **Purchase RStudio Connect**, a publishing platform for R and Python. **rstudio.com/products/connect/**

3. **Build your own Shiny Server** **rstudio.com/products/shiny/shiny-server/**

Outputs

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

DT::**renderDataTable(**expr, options, searchDelay, callback, escape, env, quoted, outputArgs**)**	**dataTableOutput(**outputId**)**
renderImage(expr, env, quoted, deleteFile, outputArgs**)**	**imageOutput(**outputId, width, height, click, dblclick, hover, brush, inline**)**
renderPlot(expr, width, height, res, …, alt, env, quoted, execOnResize, outputArgs**)**	**plotOutput(**outputId, width, height, click, dblclick, hover, brush, inline**)**
renderPrint(expr, env, quoted, width, outputArgs**)**	**verbatimTextOutput(**outputId, placeholder**)**
renderTable(expr, striped, hover, bordered, spacing, width, align, rownames, colnames, digits, na, …, env, quoted, outputArgs**)**	**tableOutput(**outputId**)**
renderText(expr, env, quoted, outputArgs, sep**)**	**textOutput(**outputId, container, inline**)**
renderUI(expr, env, quoted, outputArgs**)**	**uiOutput(**outputId, inline, container, …**)** **htmlOutput(**outputId, inline, container, …**)**

These are the core output types. See **htmlwidgets.org** for many more options.

Inputs

Collect values from the user.

Access the current value of an input object with **input$<inputId>**. Input values are **reactive**.

Action	**actionButton(**inputId, label, icon, width, …**)**
Link	**actionLink(**inputId, label, icon, …**)**
✓ Choice 1 ✓ Choice 2 ☐ Choice 3	**checkboxGroupInput(**inputId, label, choices, selected, inline, width, choiceNames, choiceValues**)**
✓ Check me	**checkboxInput(**inputId, label, value, width**)**
	dateInput(inputId, label, value, min, max, format, startview, weekstart, language, width, autoclose, datesdisabled, daysofweekdisabled**)**
	dateRangeInput(inputId, label, start, end, min, max, format, startview, weekstart, language, separator, width, autoclose**)**
Choose File	**fileInput(**inputId, label, multiple, accept, width, buttonLabel, placeholder**)**
1	**numericInput(**inputId, label, value, min, max, step, width**)**
********	**passwordInput(**inputId, label, value, width, placeholder**)**
● Choice A ○ Choice B ○ Choice C	**radioButtons(**inputId, label, choices, selected, inline, width, choiceNames, choiceValues**)**
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, timeFormat, timezone, dragRange**)**
Apply Changes	**submitButton(**text, icon, width**)** (Prevent reactions for entire app)
Enter text	**textInput(**inputId, label, value, width, placeholder**)** Also **textAreaInput()**