Reportes Interactivos

Juan C. Correa

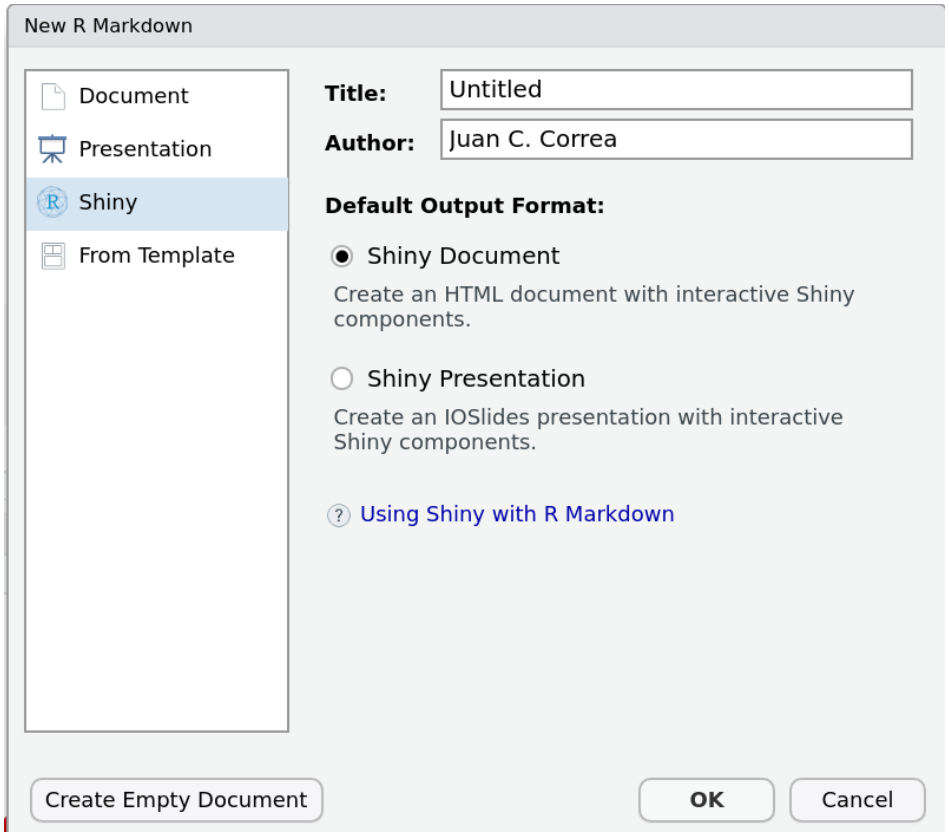
Material de uso exclusivo para
INGENIO PANTALEON, S.A.
Diagonal 6, 10-31, Zona 10

Ciudad de Guatemala

Agenda

Breve Introducción

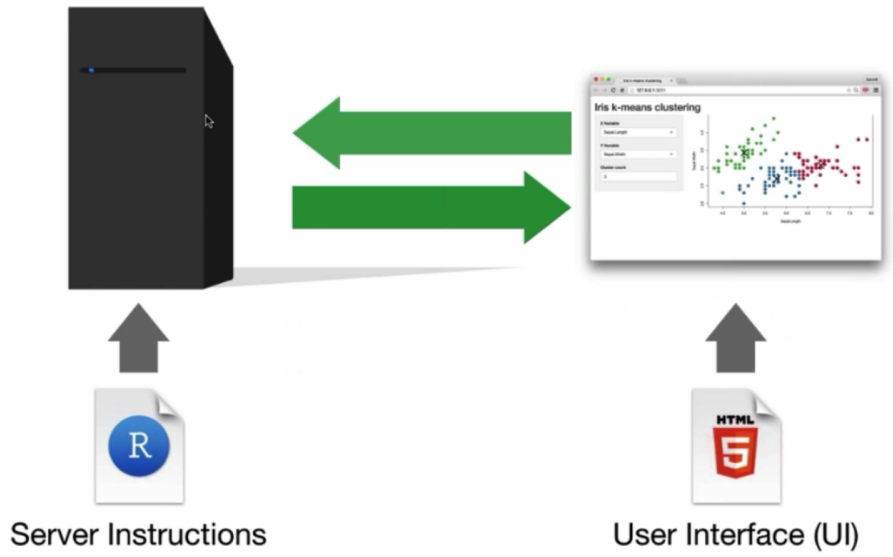
En RStudio, el usuario puede generar reportes ejecutivos dinámicos a través de la herramienta Shiny



https://vimeo.com/rstudioinc/review/131218530/212d8a5a7a/

Breve Introducción

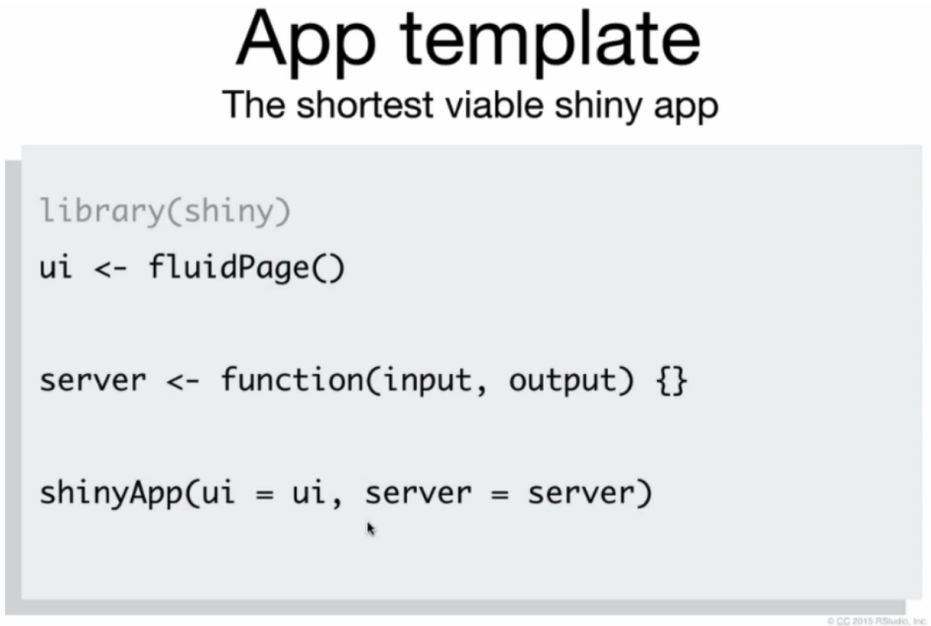
Lo primero que debemos entender para hacer un reporte interactivo son los componentes que se requieren para desarrollarlo.



Componente 1: Interfaz de Usuario (User Interface)
Componente 2: Instrucciones del Servidor (Server Instructions).

Breve Introducción

Los componentes se integran a través de una plantilla shiny que siempre tiene la misma estructura.



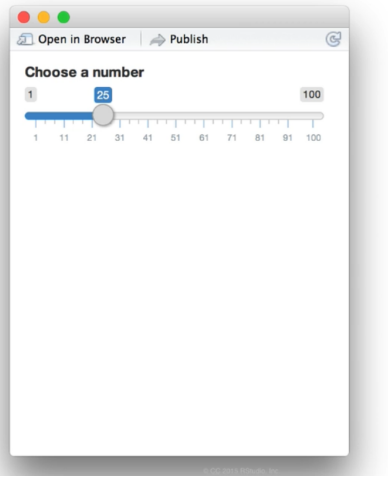
Breve Introducción

Los reportes interactivos se diseñan alrededor de Inputs y Outputs

Inputs & Outputs

Los Inputs son elementos del reporte que el usuario final puede manipular a su antojo para mirar cómo cambian los outputs en función de estas simples manipulaciones

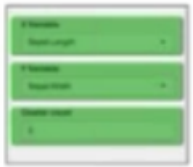
```
library(shiny)
ui <- fluidPage(
  sliderInput(inputId = "num",
    label = "Choose a number",
    value = 25, min = 1, max = 100)
)

server <- function(input, output) {}

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



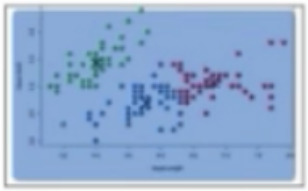
Recap

Begin each app with the template

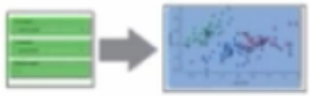
Add elements as arguments to **fluidPage()**



Create reactive inputs with an ***Input()** function



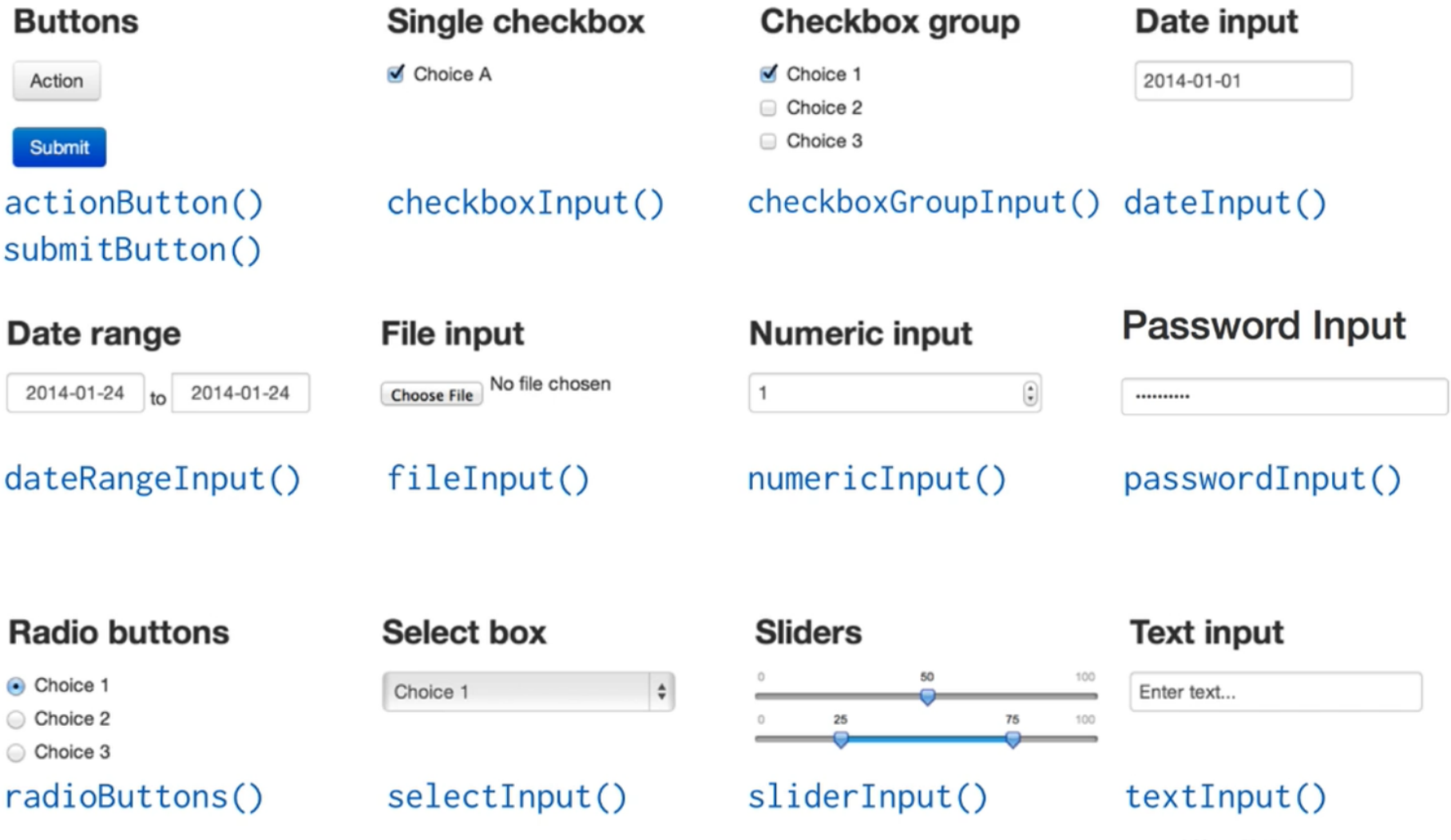
Display reactive results with an ***Output()** function



Use the server function to assemble inputs into outputs

Inputs & Outputs

Estas son las sintaxis para los inputs.



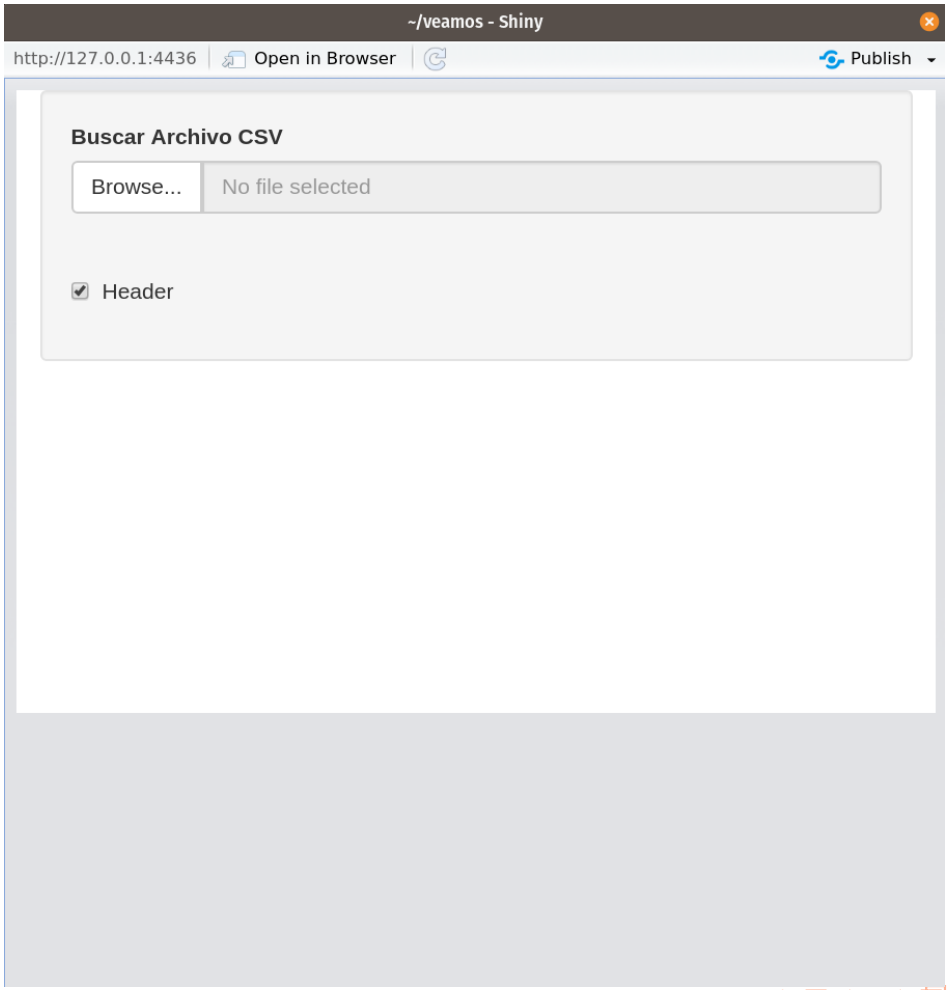
Inputs & Outputs

Veamos un ejemplo

```
1  library(shiny)
2  ui <- fluidPage(sidebarLayout(sidebarPanel(
3              fileInput("file1", "Buscar Archivo CSV", accept = ".csv"),
4              checkboxInput("header", "Header", TRUE)),
5          mainPanel(tableOutput("contents"))))
6
7  server <- function(input, output) {
8      output$contents <- renderTable({
9          file <- input$file1
10         ext <- tools::file_ext(file$datapath)
11         req(file)
12         validate(need(ext == "csv", "Please upload a csv file"))
13         read.csv(file$datapath, header = input$header)
14     })
15 }
16
17 shinyApp(ui, server)
```

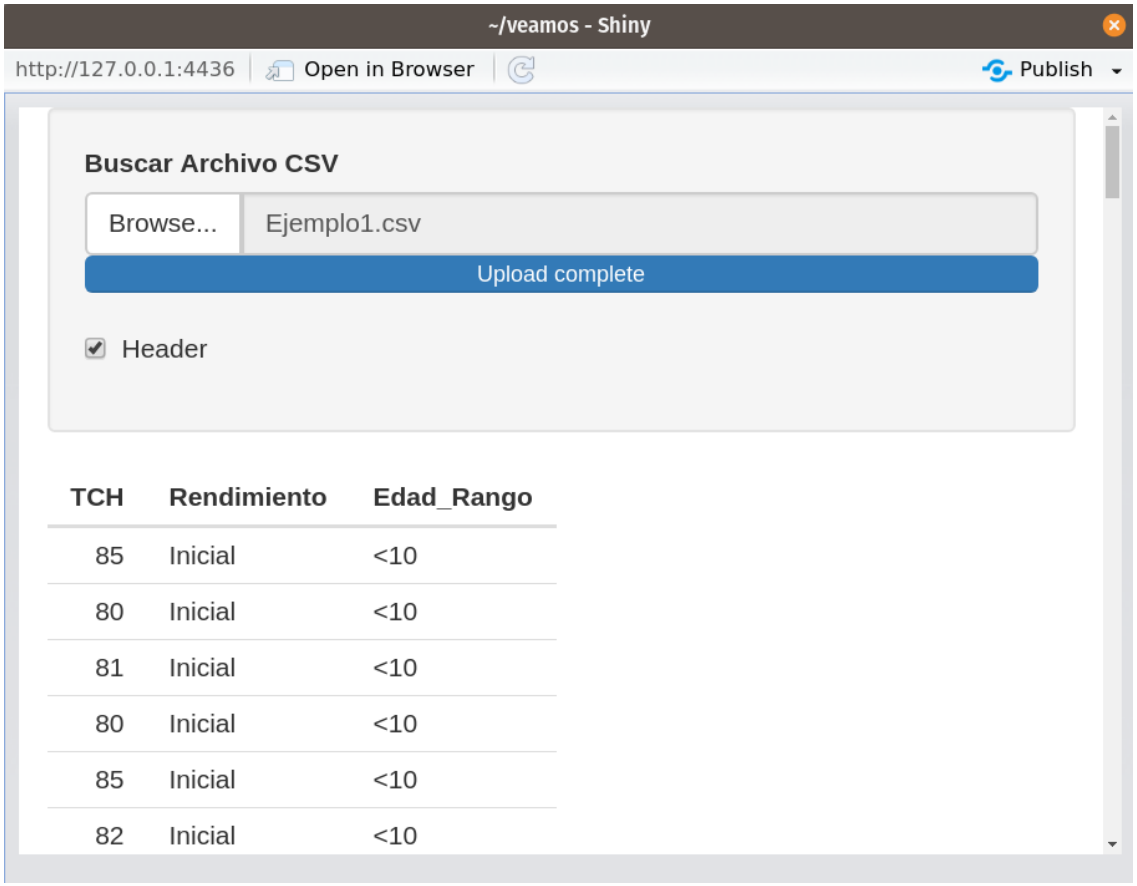
Inputs & Outputs

Al hacer clic en el botón Browse, se nos va a abrir una ventana para que busquemos el archivo específico que queremos mostrar en la aplicación Shiny.



Inputs & Outputs

Este es el resultado que deberíamos ver



Inputs & Outputs

También existe una familia de funciones para Outputs

Function	Inserts
`dataTableOutput()`	an interactive table
`htmlOutput()`	raw HTML
`imageOutput()`	image
`plotOutput()`	plot
`tableOutput()`	table
`textOutput()`	text
`uiOutput()`	a Shiny UI element
`verbatimTextOutput()`	text

Inputs & Outputs

*Output()

To display output, add it to `fluidPage()` with an `*Output()` function

```
plotOutput(outputId = "hist")
```

- `plotOutput`: the type of output to display
- `outputId = "hist"`: name to give to the output object

Server

Server

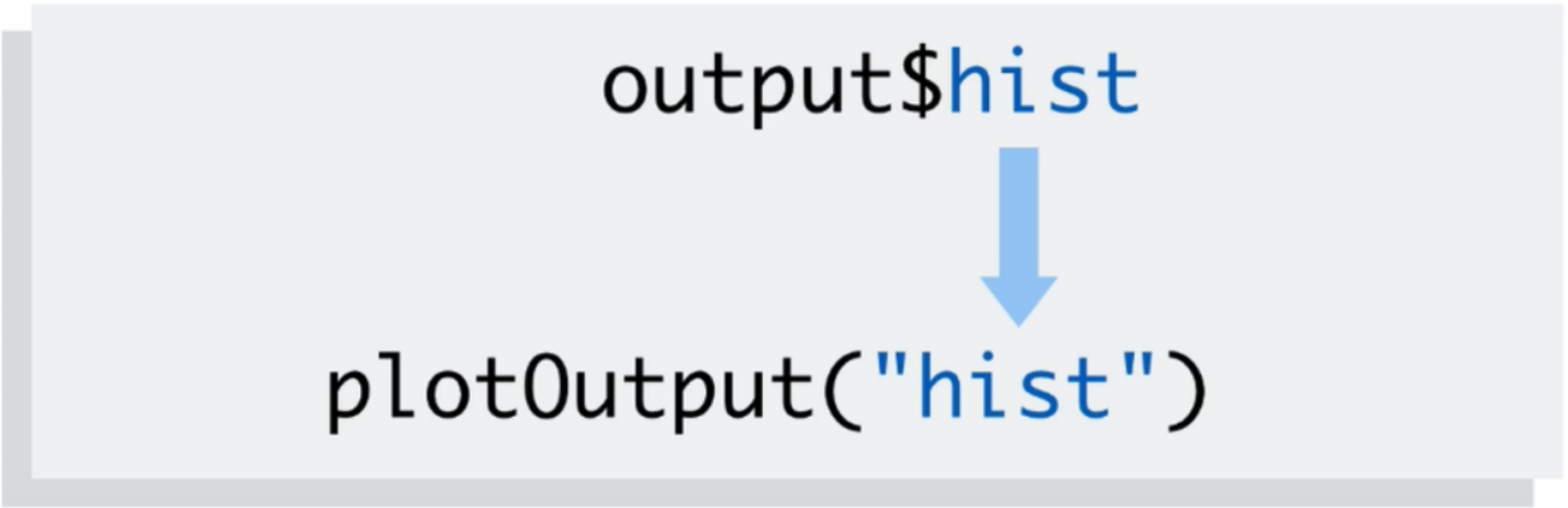
Hay tres reglas para escribir las funciones que el servidor usará para desarrollar la aplicación Shiny

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



}
```

- Regla 1: El output debe guardarse con `output$`
- Regla 2: Para generar el output debe especificarse la forma cómo va a mostrarse y esto se hace con las funciones `render*()`
- Regla 3: Use valores de Input con `input$`

Server

Regla 1



Server

Regla 2

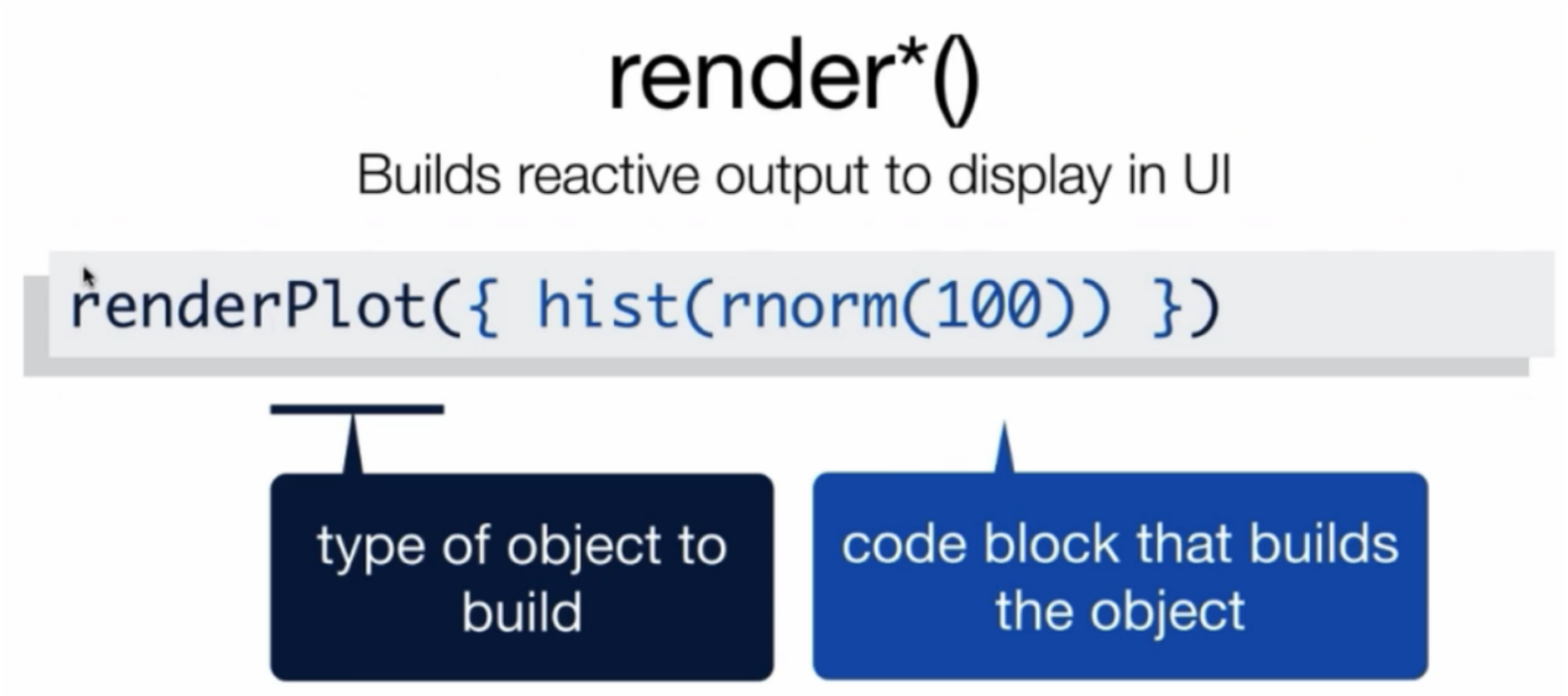
```
server <- function(input, output) {
  output$hist <- renderPlot({

  })
}
```

Regla 2

Use the **render*()** function that creates the type of output you wish to make.

function	creates
`renderDataTable()`	An interactive table (from a data frame, matrix, or other table-like structure)
`renderImage()`	An image (saved as a link to a source file)
`renderPlot()`	A plot
`renderPrint()`	A code block of printed output
`renderTable()`	A table (from a data frame, matrix, or other table-like structure)
`renderText()`	A character string
`renderUI()`	a Shiny UI element

Regla 2


render*()
Builds reactive output to display in UI
renderPlot({ hist(rnorm(100)) })
type of object to build
code block that builds the object

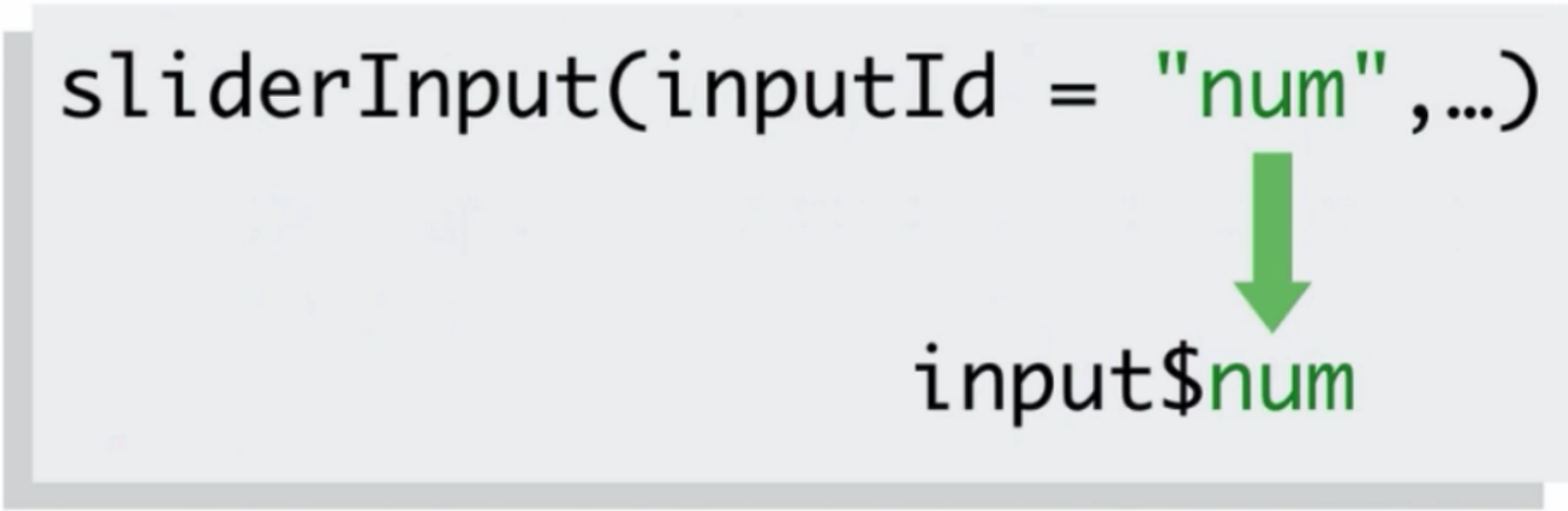
Server

Regla 2

```
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(100))
  })
}
```

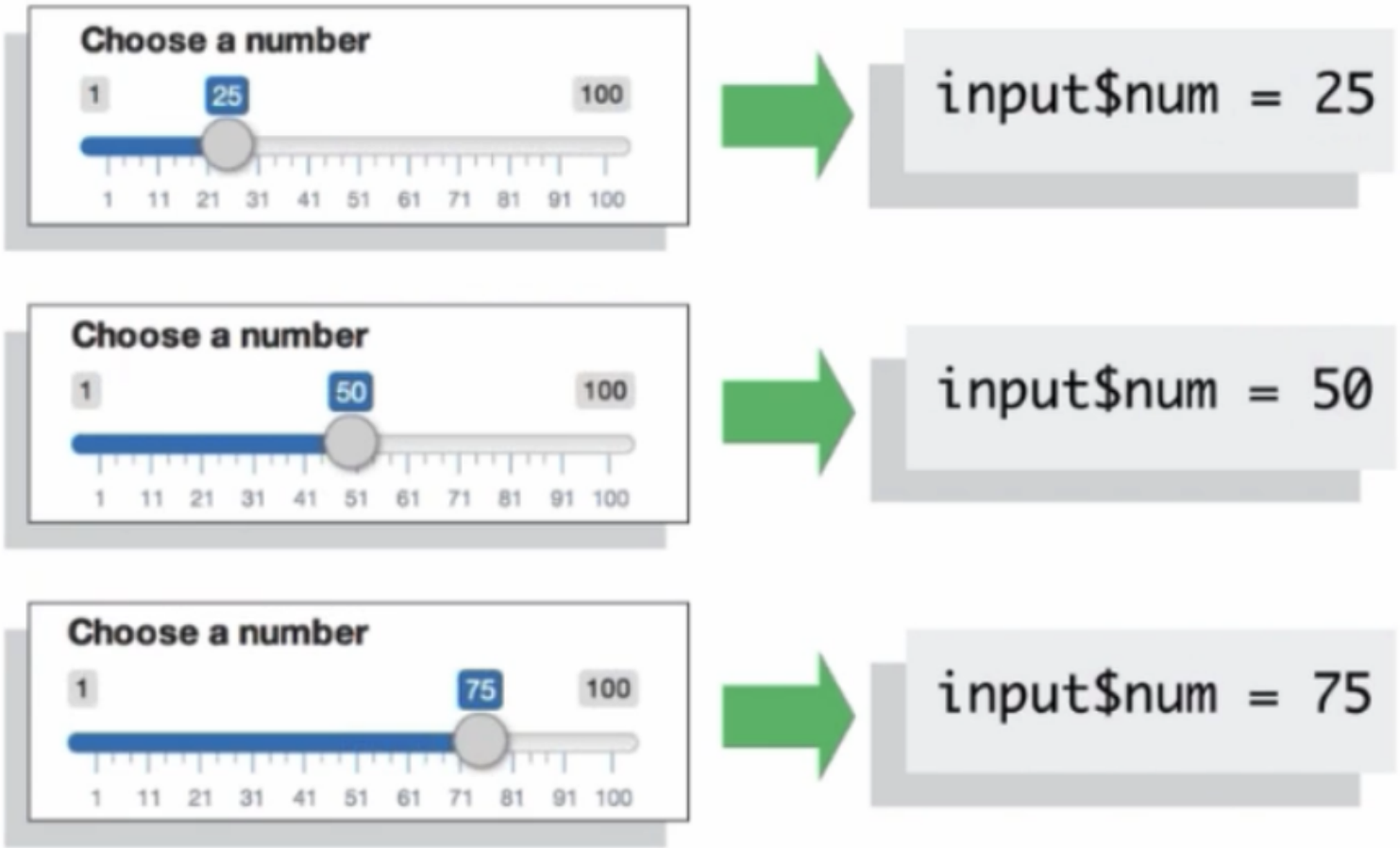
Server

Regla 3



Server

Regla 3



Server

Regla 3

```
server <- function(input, output) {
  output$hist <- renderPlot({
    hist(rnorm(input$num))
  })
}
```

Recap: Server

Use the server function to assemble inputs into outputs. Follow 3 rules:

`output$hist <-` 1. Save the output that you build to **output$**

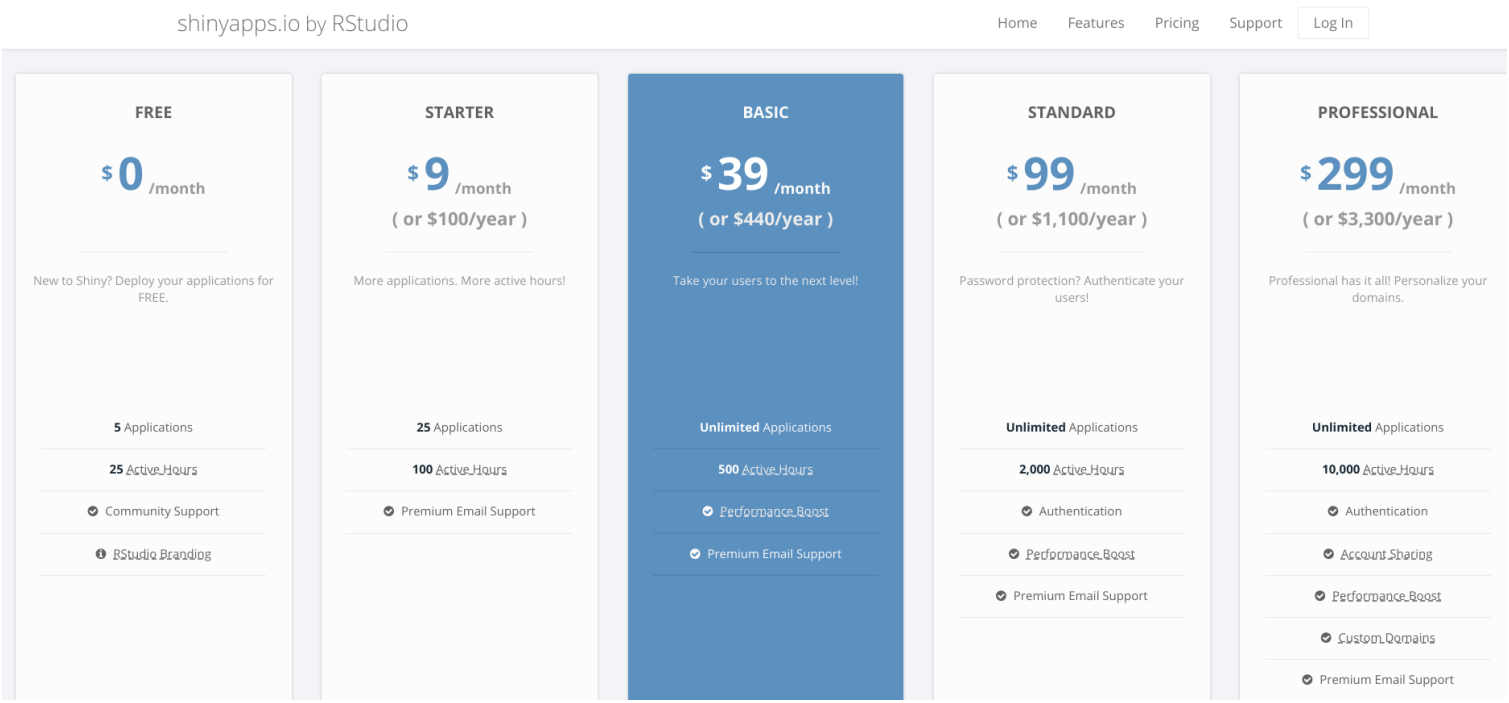
```
renderPlot({
  hist(rnorm(input$num))
})
```

2. Build the output with a **render*()** function

`input$num` 3. Access input values with **input$**

Create reactivity by using **Inputs** to build **rendered Outputs**

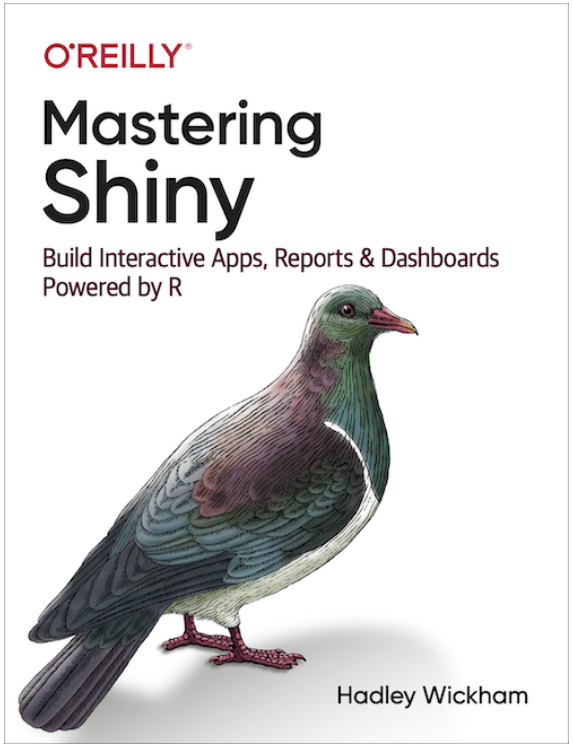
Más Recursos



Para desarrollar aplicaciones Shiny accesibles por Internet, hay distintos planes de afiliación.

Más Recursos

Probablemente la mejor referencia sobre Shiny es el libro de Hadley Wickham.



https://mastering-shiny.org/index.html