Introduction Shiny mainly from:

Bayes 2024/2025
Dr. Denise Welsch
Hochschule Koblenz MIT

R Shiny

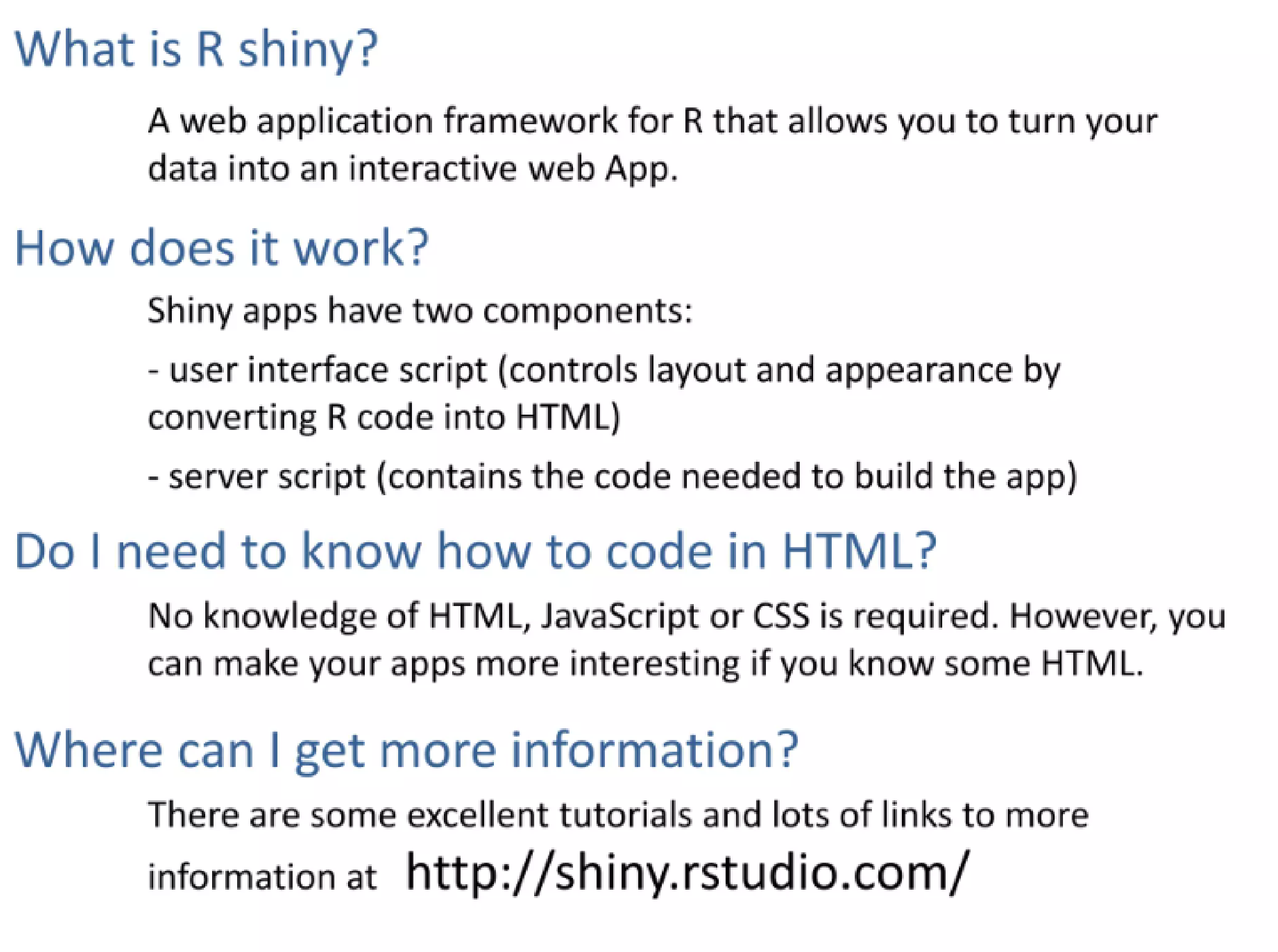
An Introduction to Making an Interactive Web Application

Chloe Thomas

ScHARR, University of Sheffield

2015

Shiny Intro



What is R shiny?

A web application framework for R that allows you to turn your data into an interactive web App.

How does it work?

Shiny apps have two components:

- user interface script (controls layout and appearance by converting R code into HTML)

- server script (contains the code needed to build the app)

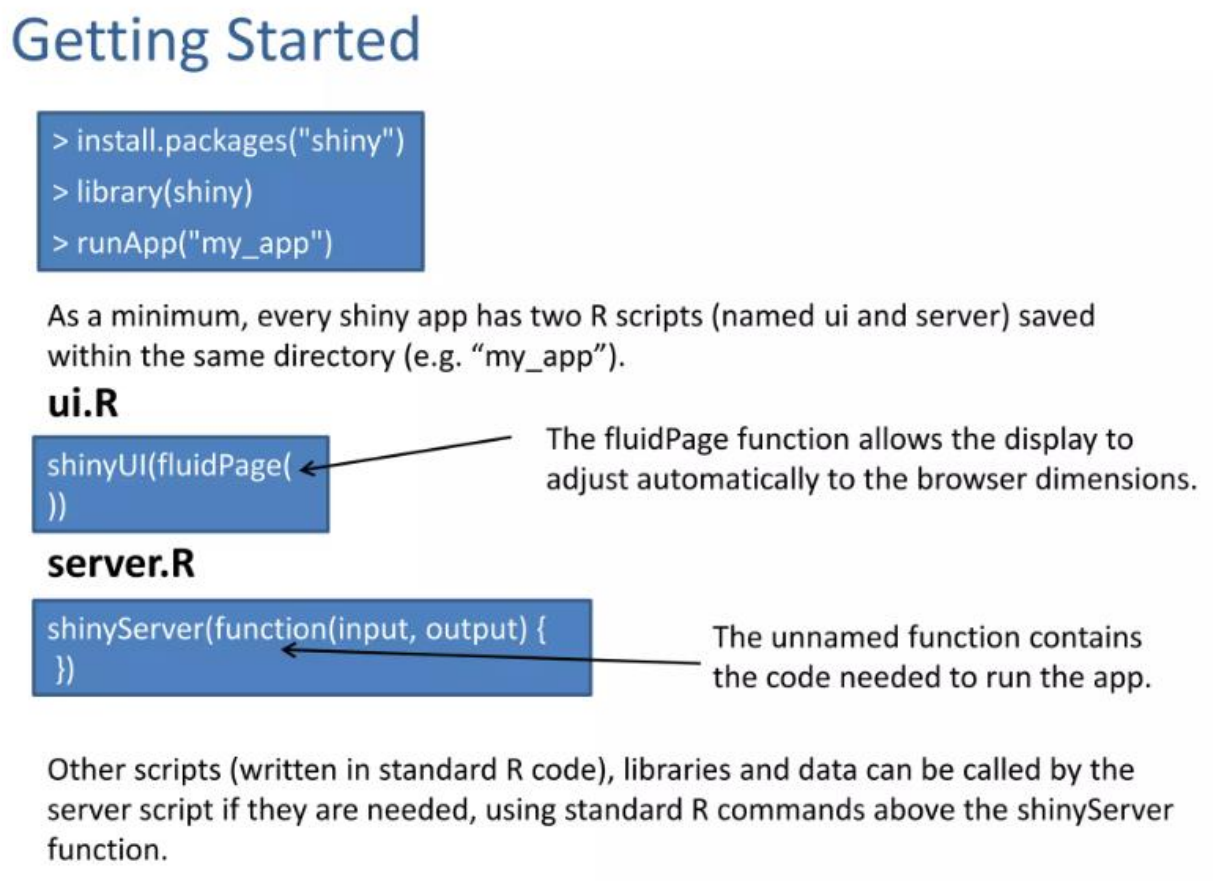
Do I need to know how to code in HTML?

No knowledge of HTML, JavaScript or CSS is required. However, you can make your apps more interesting if you know some HTML.

Where can I get more information?

There are some excellent tutorials and lots of links to more information at http://shiny.rstudio.com/

Shiny Intro



Getting Started

```
> install.packages("shiny")
> library(shiny)
> runApp("my_app")
```

As a minimum, every shiny app has two R scripts (named ui and server) saved within the same directory (e.g. "my_app").

ui.R

```
shinyUI(fluidPage(
))
```

The fluidPage function allows the display to adjust automatically to the browser dimensions.

server.R

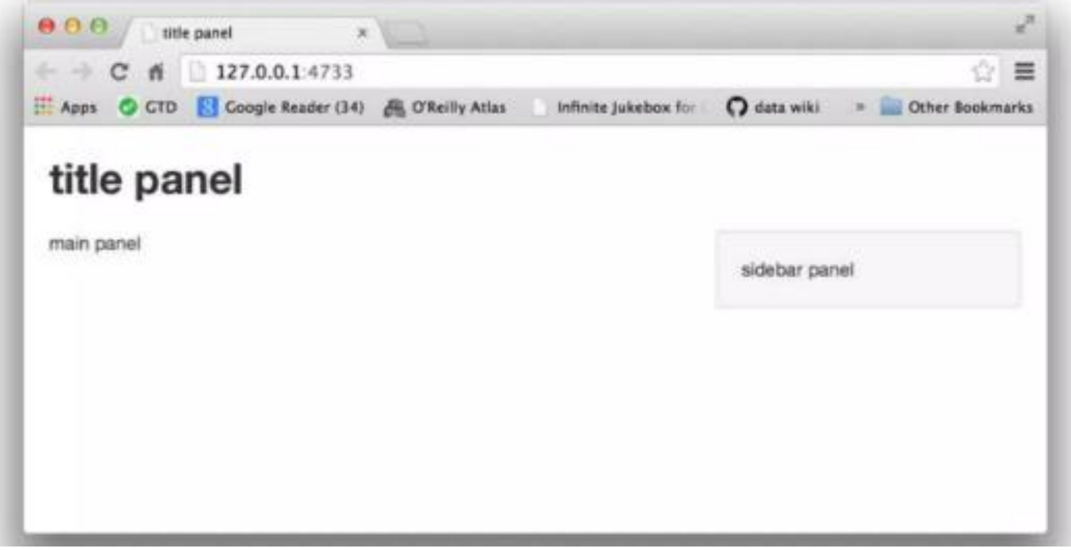
```
shinyServer(function(input, output) {
 })
```

The unnamed function contains the code needed to run the app.

Other scripts (written in standard R code), libraries and data can be called by the server script if they are needed, using standard R commands above the shinyServer function.

The User Interface (ui.R)

```
# ui.R

shinyUI(fluidPage(
        titlePanel("title panel"),

        sidebarLayout(position = "right",
                sidebarPanel( "sidebar panel"),
                mainPanel("main panel")
        )
))
```



title panel

main panel

sidebar panel

A series of nested shiny functions control the layout of the content.

sidebarLayout is a function with two compulsory arguments, both themselves functions (sidebarPanel and mainPanel). An optional argument controls the position of the panels.

Text is written as a character string inside quotation marks.

Shiny User Interface Functions

absolutePanel **(fixedPanel)**	Panel with absolute positioning
bootstrapPage **(basicPage)**	Create a Bootstrap page
column	Create a column within a UI definition
conditionalPanel	Conditional Panel
fixedPage **(fixedRow)**	Create a page with a fixed layout
fluidPage **(fluidRow)**	Create a page with fluid layout
headerPanel	Create a header panel
helpText	Create a help text element
icon	Create an icon
mainPanel	Create a main panel
navbarPage **(navbarMenu)**	Create a page with a top level navigation bar
navlistPanel	Create a navigation list panel
pageWithSidebar	Create a page with a sidebar
sidebarLayout	Layout a sidebar and main area
sidebarPanel	Create a sidebar panel
tabPanel	Create a tab panel
tabsetPanel	Create a tabset panel
titlePanel	Create a panel containing an application title.
inputPanel	Input panel
flowLayout	Flow layout
splitLayout	Split layout
verticalLayout	Lay out UI elements vertically
wellPanel	Create a well panel
withMathJax	Load the MathJax library and typeset math expressions

Formatting Text

To make things look a bit more interesting there are lots of shiny commands that can alter the text colour, style and size. Alternatively, if you are familiar with HTML, you can write HTML code directly inside HTML("").

shiny function	HTML5 equivalent	Creates
p	<p>	A paragraph of text
h1	<h1>	A first level header
h2	<h2>	A second level header
h3	<h3>	A third level header
h4	<h4>	A fourth level header
h5	<h5>	A fifth level header
h6	<h6>	A sixth level header
a	<a>	A hyper link
br	
	A line break (e.g. a blank line)
div	<div>	A division of text with a uniform style
span	<span>	An in-line division of text with a uniform style
pre	<pre>	Text 'as is' in a fixed width font
code	<code>	A formatted block of code
img	<img>	An image
strong	<strong>	Bold text
em	<em>	Italicized text
HTML		Directly passes a character string as HTML code

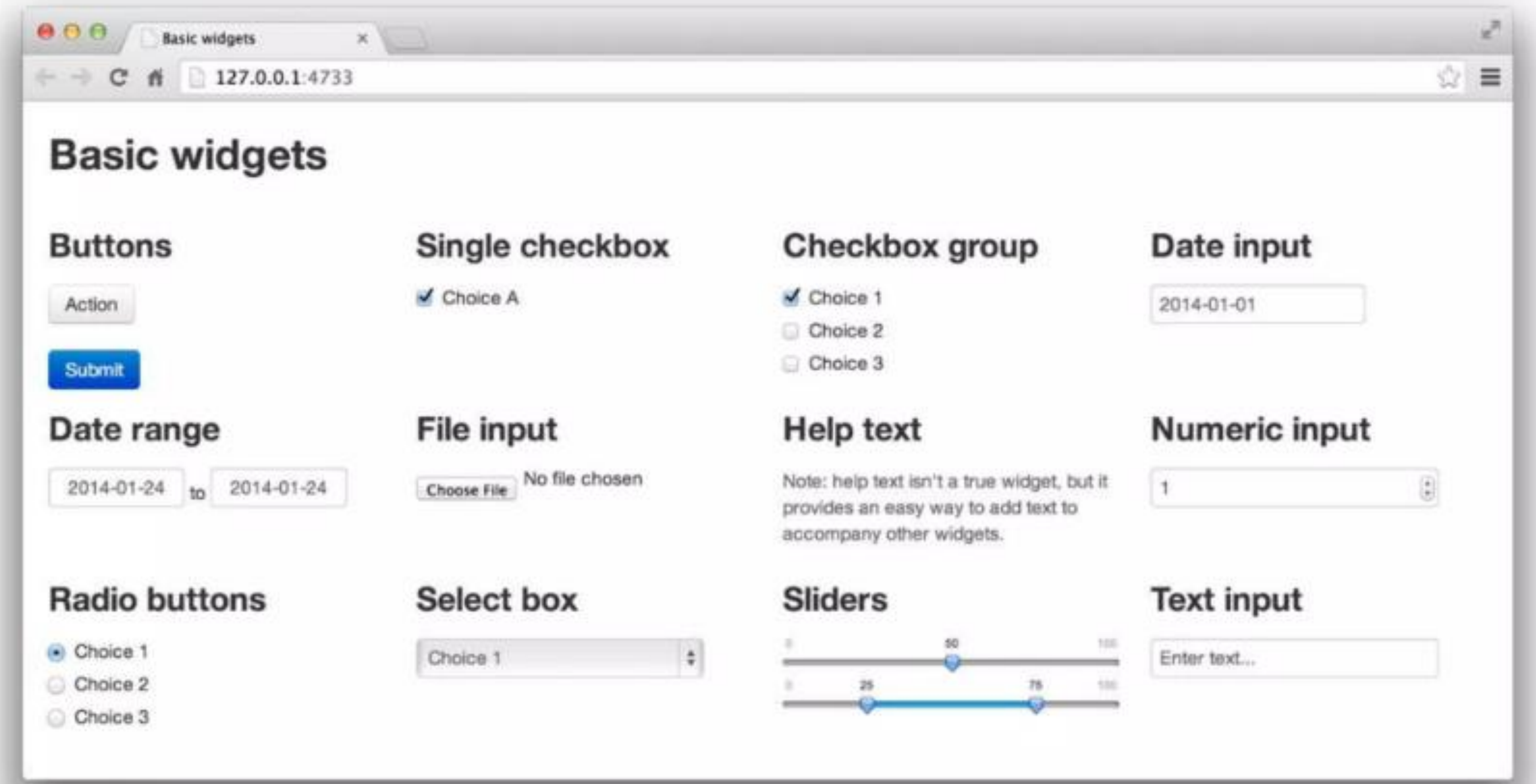
Adding Images

```
# ui.R

shinyUI(fluidPage(
          titlePanel("My Shiny App"),

          sidebarLayout(
                    sidebarPanel(),
                    mainPanel(
                    img(src = “my_image.png", height = 400, width = 400)
                    )
          )
))
```

The image file must be placed inside a folder named www within the same directory as the ui and server scripts. Any file placed in the www folder will be shared with the web browser.

Adding Widgets

Widgets are interactive web elements. A series of inbuild shiny functions allows widgets to be easily added to webpages. Each function has several arguments. It must have at least a name (used to access its value, but not visible) and a label (visible on the page). Both are character strings. Other arguments are widget specific.



Note: help text isn't a true widget, but it provides an easy way to add text to accompany other widgets.

Adding Widgets

```
# ui.R

shinyUI(fluidPage(
        titlePanel("Basic widgets"),

        fluidRow(
                column(3,
                        h3("Buttons"),
                        actionButton("action", label = "Action"),
                        br(),
                        br(),
                        submitButton("Submit")),

                column(3,
                        h3("Single checkbox"),
                        checkboxInput("checkbox", label = "Choice A", value = TRUE)),

                column(3,
                        checkboxGroupInput("checkGroup",
                        label = h3("Checkbox group"),
                        choices = list("Choice 1" = 1, "Choice 2" = 2, "Choice 3" = 3),
                        selected = 1)),

                column(3,
                        dateInput("date",
                        label = h3("Date input"),
                        value = "2014-01-01"))
        )
))
```

Action button

Submit button

Single checkbox (whether pre-selected)

Checkbox group (list choices and which pre-selected)

Date input

Adding Reactive Output

Widgets allow users to input their choices. Reactive output is a response to those choices.
Programming it into an app is a two-step process:
- Add an R object to the user interface to indicate where it should be displayed.
- Add instructions on how to build the object in the server

```
# ui.R

shinyUI(fluidPage(
  titlePanel("My Shiny App"),

    sidebarLayout(
      sidebarPanel(
        sliderInput(“value",
        label = “Value of interest:",
        min = 0, max = 100, value = 50)
      ),

    mainPanel(
      textOutput("text1")
    )
  )
))
```

Output function	**creates**
htmlOutput	raw HTML
imageOutput	image
plotOutput	plot
tableOutput	table
textOutput	text

Output functions have one argument, which is a name (character string). In this case “text1”. This is used in the server to identify the output.

Adding Reactive Output

The unnamed function inside shinyServer contains all the code that needs to be updated when a user accesses the app. All R output objects used in the ui need to be defined in server using the prefix output$ followed by the name of the object. e.g. output$text1

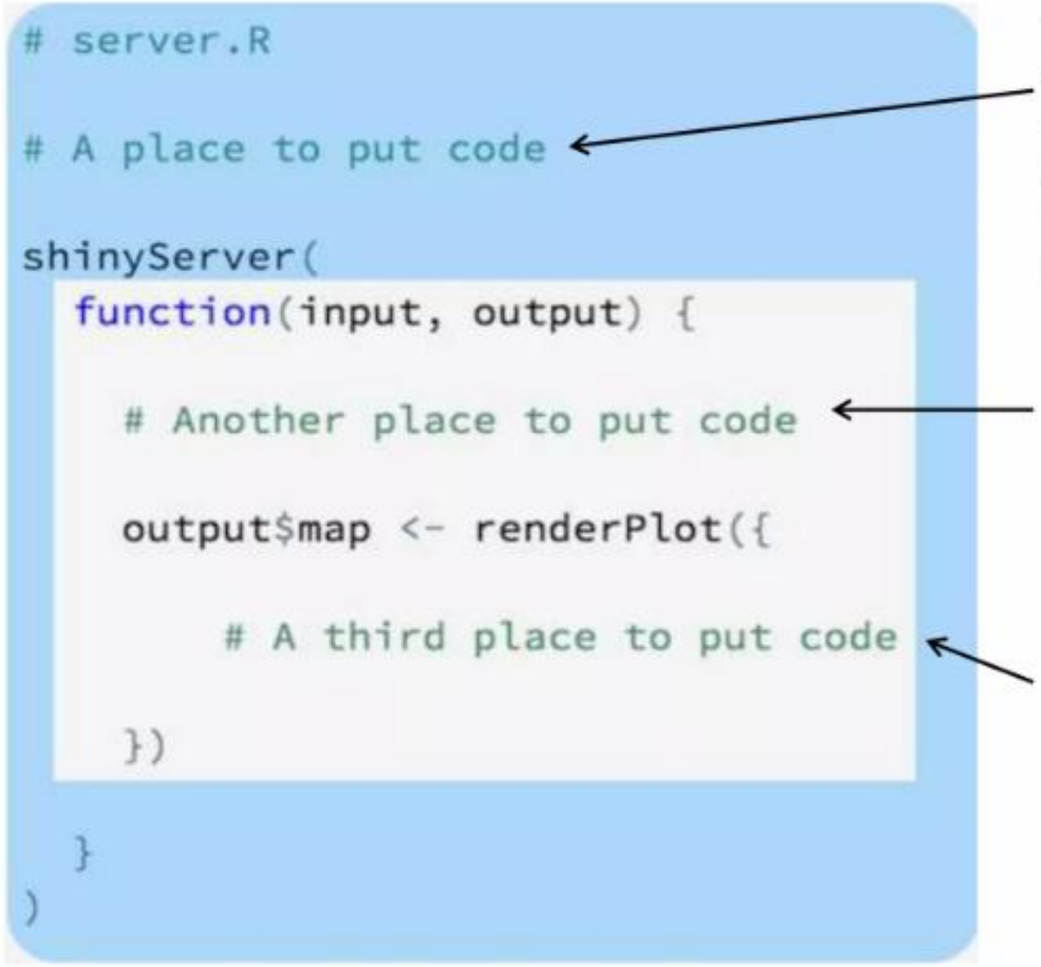
The element should be defined using one of the shiny render* functions – this should be chosen to reflect the type of output. Each render function takes a single argument (R expression) surrounded by braces.

```
# server.R

shinyServer(function(input, output) {

     output$text1 <- renderText({
       "You have selected" , input$value)
     })
  }
)
```

render function	creates
renderImage	images (saved as a link to a source file)
renderPlot	plots
renderPrint	any printed output
renderTable	data frame, matrix, other table like structures
renderText	character strings
renderUI	a Shiny tag object or HTML

The Server

The server is where all the code is located for execution of the app. How frequently code is run depends upon where it is placed within the server script.



```
# server.R

# A place to put code

shinyServer(
  function(input, output) {

    # Another place to put code

    output$map <- renderPlot({

        # A third place to put code

    })

  }
)
```

Code placed here will be run once when the app is launched. shinyServer will be executed and the unnamed function saved. Good place to source other scripts, read data sets and load libraries.

Code placed here will run once each time a new user visits the app. The unnamed function will run once and save a distinct set of reactive objects for each user. Good place to put objects that record a user's session.

Code placed here will run every time the user changes the widget that this particular output depends upon. Important to not put unnecessary code here as it will slow down the entire app.

Server.R

The server function will be called when each client (web browser) first loads the Shiny application's page. It must take an `input` and an `output` parameter. Any return value will be ignored. It also takes an optional `session` parameter, which is used when greater control is needed.

```r
# A very simple Shiny app that takes a message from the user
# and outputs an uppercase version of it.
shinyServer(function(input, output, session) {
  output$uppercase <- renderText({
    toupper(input$message)
  })
})
```