RStudio
250 Northern Ave, Boston, MA 02210
Phone: 844-448-1212
Email: info@rstudio.com
Web: http://www.rstudio.com

Understanding UI

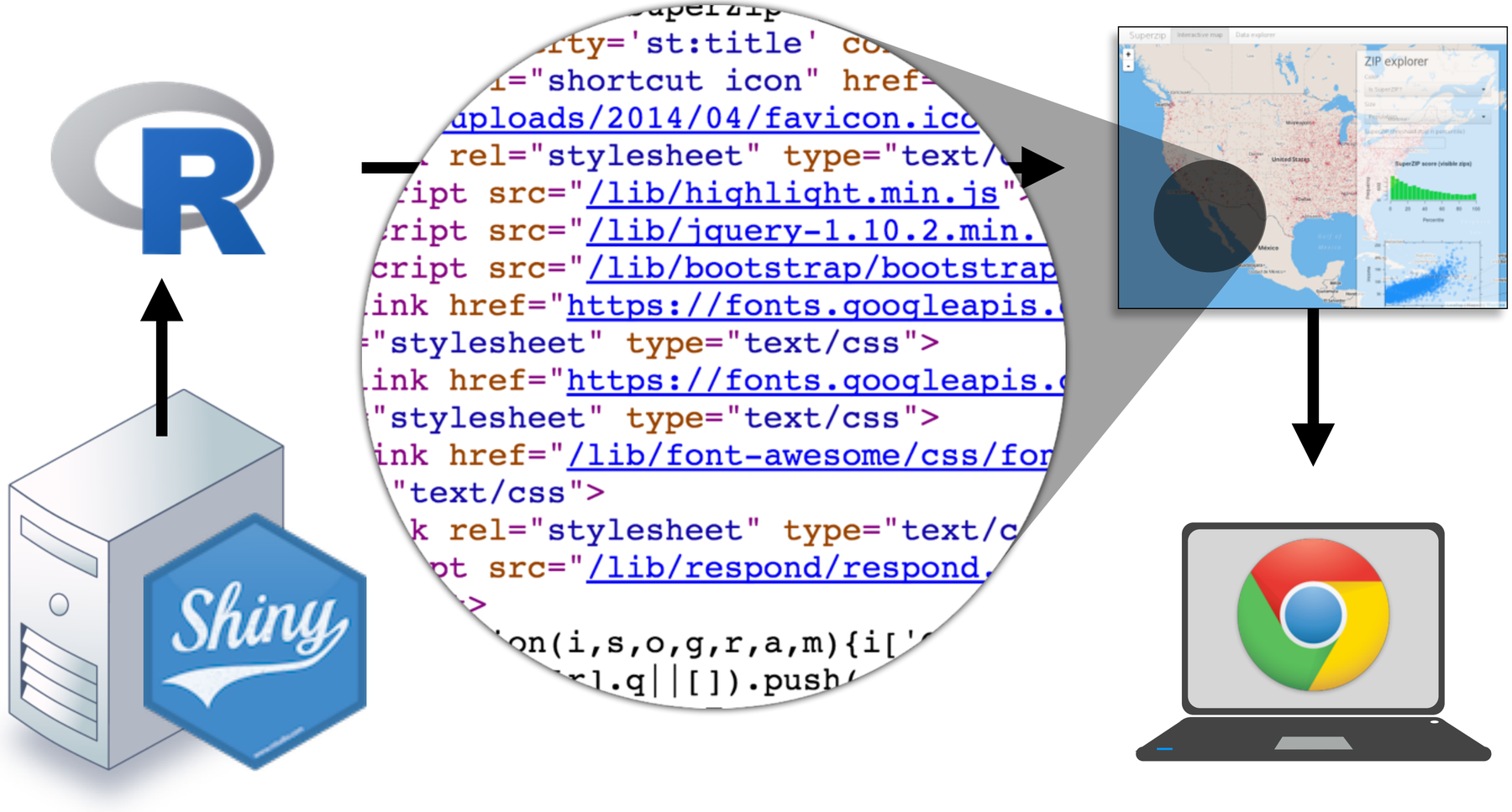
How to use HTML to make an awesome web app

Garrett Grolemund
Data Scientist and Master Instructor
January 2016
Email: garrett@rstudio.com
Twitter: @StatGarrett

UI matters.

(And we have a new way to make it.)

Every Shiny app is delivered as a web document



Shiny Server (Pro)

A Web browser

To Build the UI

1. Use R functions to write HTML and Shiny components (**ui.R**)

2. Use HTML to write HTML and Shiny components (**index.html**)

3. Use HTML to write HTML components and R to write Shiny components (**htmlTemplates**)

ui.R

Build it with R

App template

The shortest viable shiny app

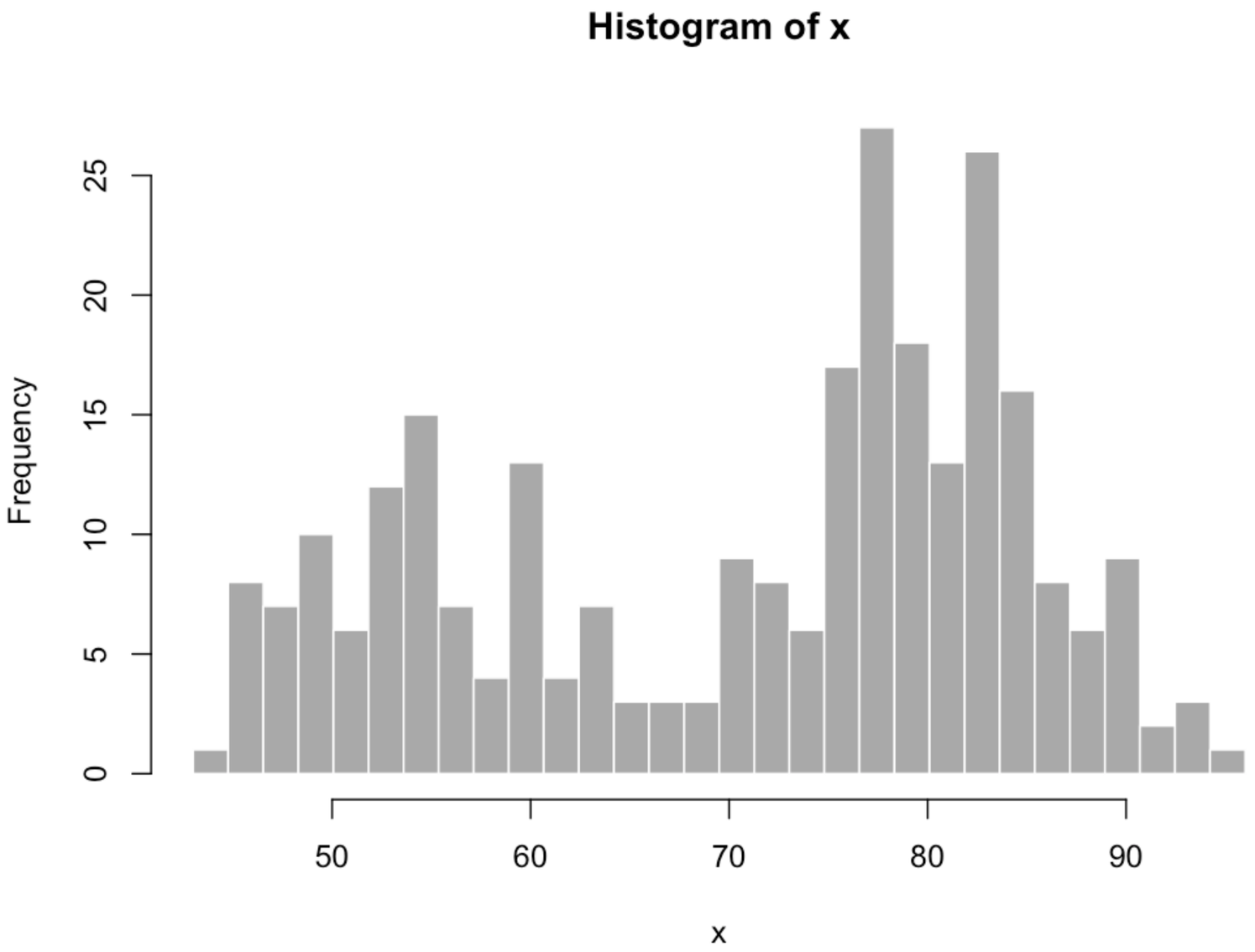

```
library(shiny)

ui <- fluidPage()

server <- function(input, output) {}

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

Old Faithful Geyser Data

Number of bins:

1 30 50

1 6 11 16 21 26 31 36 41 46 50



The HTML that builds the user interface for your app

```
ui <- fluidPage()
```

```
fluidPage()
```

```
<div class="container-fluid"></div>
```

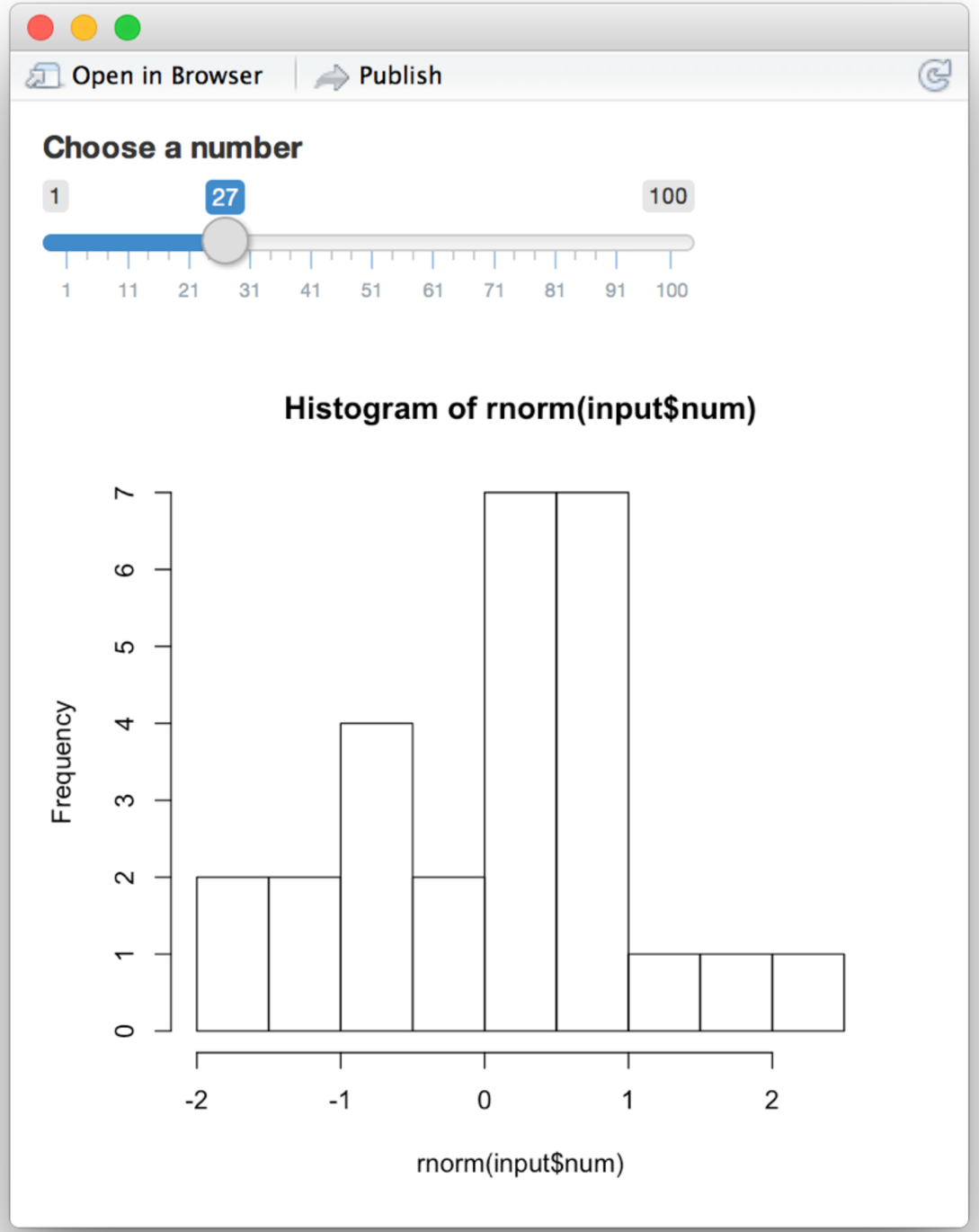
```
sliderInput(inputId = "num",
  label = "Choose a number",
  value = 25, min = 1, max = 100)
```

```
<div class="form-group shiny-input-container">
  <label class="control-label" for="num">Choose a number</label>
  <input class="js-range-slider" id="num" data-min="1" data-max="100"
     data-from="25" data-step="1" data-grid="true" data-grid-num="9.9"
     data-grid-snap="false" data-prettify-separator="," data-keyboard="true"
     data-keyboard-step="1.01010101010101"/>
</div>
```

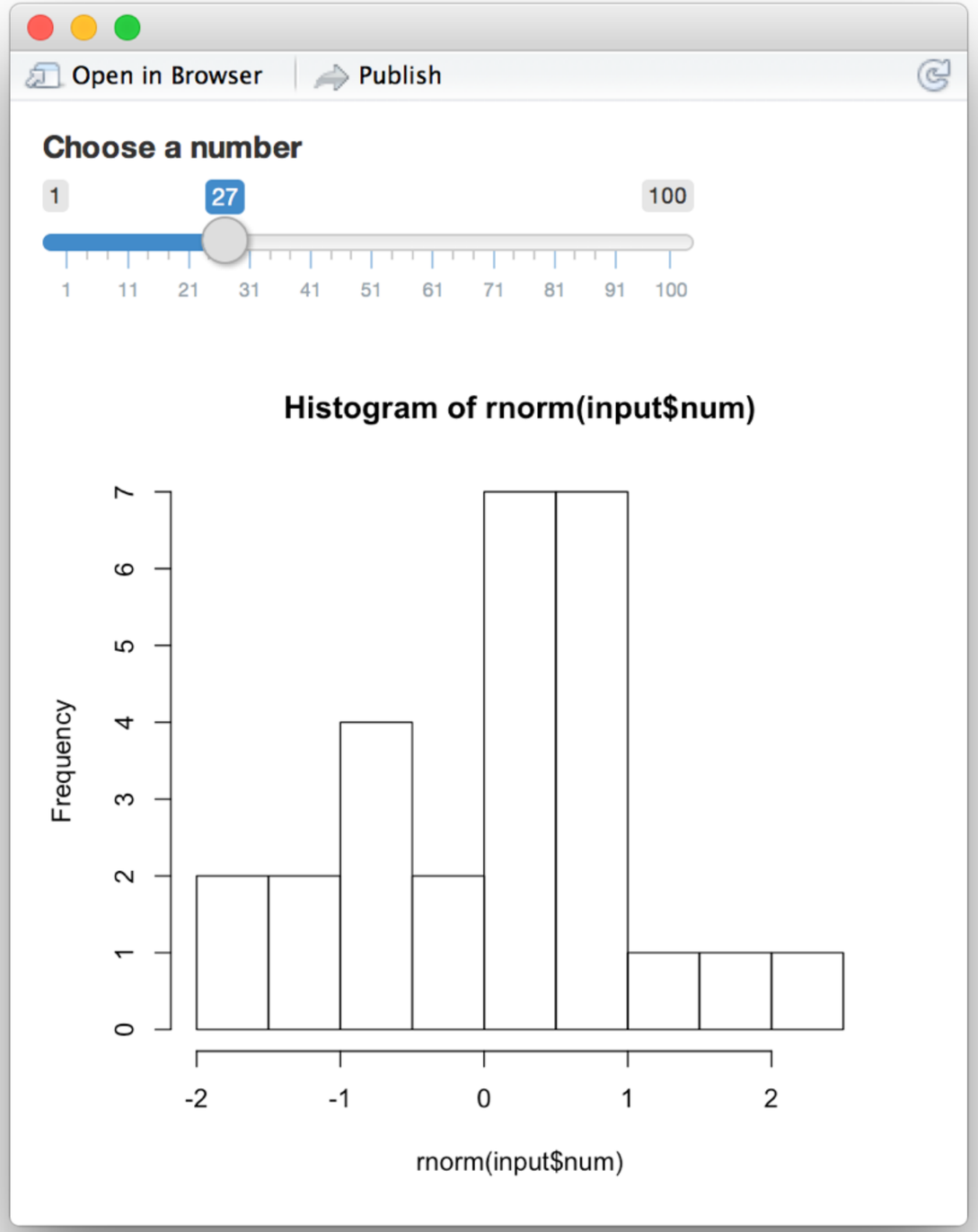
```
plotOutput("hist")
```

```
<div id="hist" class="shiny-plot-output" style="width: 100% ; height: 400px"></div>
```

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



```
ui <-
<div class="container-fluid">
  <div class="form-group shiny-input-
container">
    <label class="control-label"
for="num">Choose a number</label>
    <input class="js-range-slider"
id="num" data-min="1" data-max="100"
data-from="25" data-step="1" data-
grid="true" data-grid-num="9.9" data-
grid-snap="false" data-prettify-
separator="," data-keyboard="true" data-
keyboard-step="1.01010101010101" data-
drag-interval="true" data-data-
type="number"/>
  </div>
  <div id="hist" class="shiny-plot-
output" style="width: 100% ; height:
400px"></div>
</div>
```



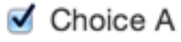
The "ui.R" system

Input functions

Buttons

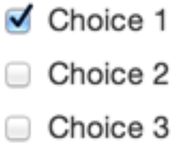
`actionButton()`
`submitButton()`

Single checkbox



`checkboxInput()`

Checkbox group



`checkboxGroupInput()`

Date input

`dateInput()`

Date range

`dateRangeInput()`

File input

`fileInput()`

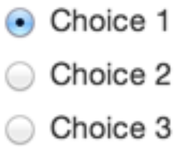
Numeric input

`numericInput()`

Password Input

`passwordInput()`

Radio buttons

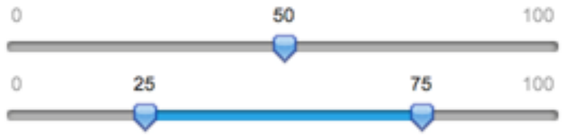


`radioButtons()`

Select box

`selectInput()`

Sliders



`sliderInput()`

Text input

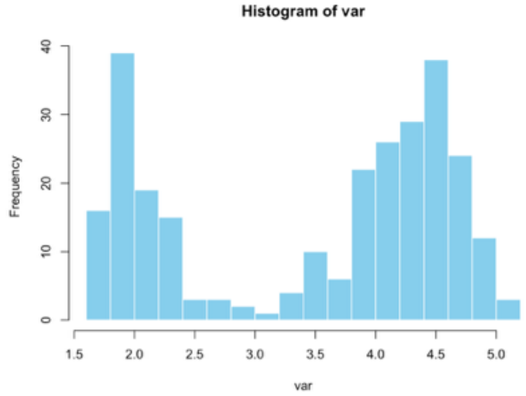
`textInput()`

Output functions

dataTableOutput()

htmlOutput()

imageOutput()



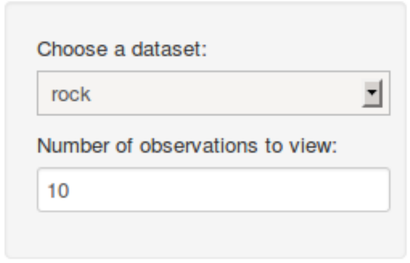
plotOutput()

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.10	3.50	1.40	0.20	setosa
2	4.90	3.00	1.40	0.20	setosa
3	4.70	3.20	1.30	0.20	setosa
4	4.60	3.10	1.50	0.20	setosa
5	5.00	3.60	1.40	0.20	setosa
6	5.40	3.90	1.70	0.40	setosa

tableOutput()

foo

textOutput()



uiOutput()

```
      area             peri            shape              perm
 Min.   : 1016   Min.   : 308.6   Min.   :0.09033   Min.   :   6.30
 1st Qu.: 5305   1st Qu.:1414.9   1st Qu.:0.16226   1st Qu.:  76.45
 Median : 7487   Median :2536.2   Median :0.19886   Median : 130.50
 Mean   : 7188   Mean   :2682.2   Mean   :0.21811   Mean   : 415.45
 3rd Qu.: 8870   3rd Qu.:3989.5   3rd Qu.:0.26267   3rd Qu.: 777.50
 Max.   :12212   Max.   :4864.2   Max.   :0.46413   Max.   :1300.00
```

verbatimTextOutput()

Panel functions

absolutePanel()

Panel position set rigidly (absolutely), not fluidly

conditionalPanel()

A JavaScript expression determines whether panel is visible or not.

fixedPanel()

Panel is fixed to browser window and does not scroll with the page

headerPanel()

Panel for the app's title, used with pageWithSidebar()

inputPanel()

Panel with grey background, suitable for grouping inputs

mainPanel()

Panel for displaying output, used with pageWithSidebar()

navlistPanel()

Panel for displaying multiple stacked tabPanels(). Uses sidebar navigation

sidebarPanel()

Panel for displaying a sidebar of inputs, used with pageWithSidebar()

tabPanel()

Stackable panel. Used with navlistPanel() and tabsetPanel()

tabsetPanel()

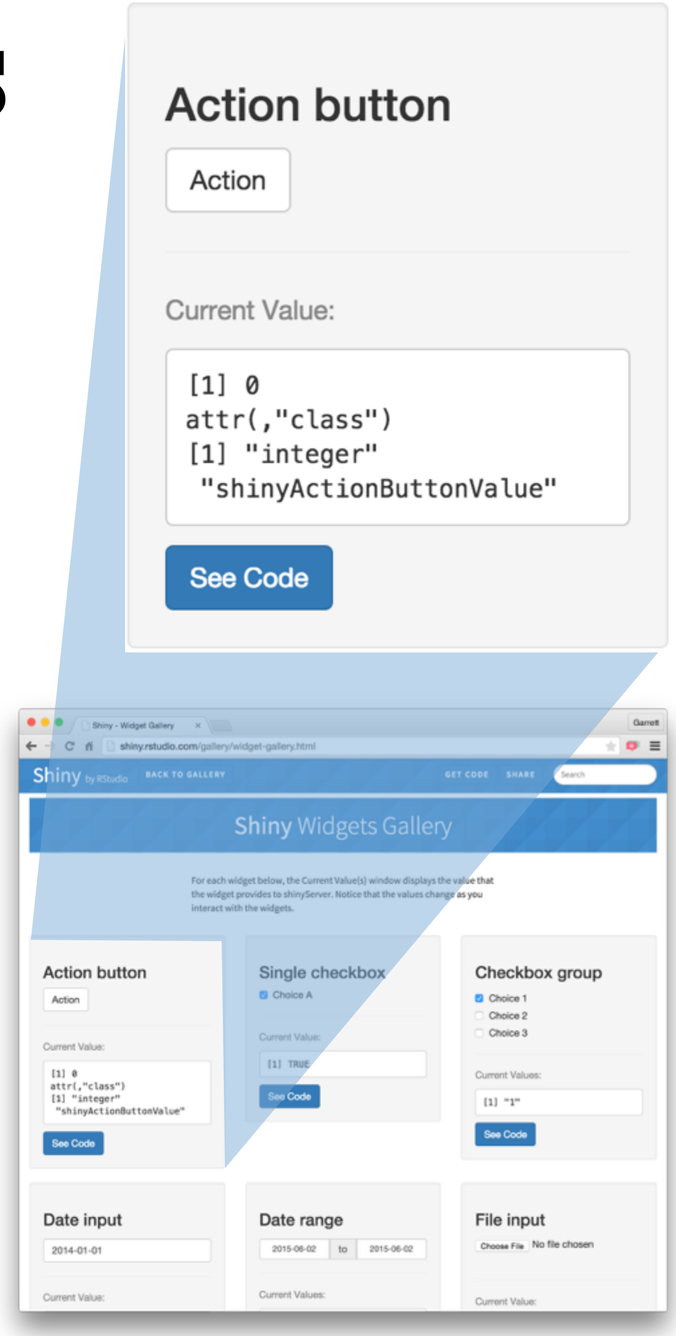
Panel for displaying multiple stacked tabPanels(). Uses tab navigation

titlePanel()

Panel for the app's title, used with pageWithSidebar()

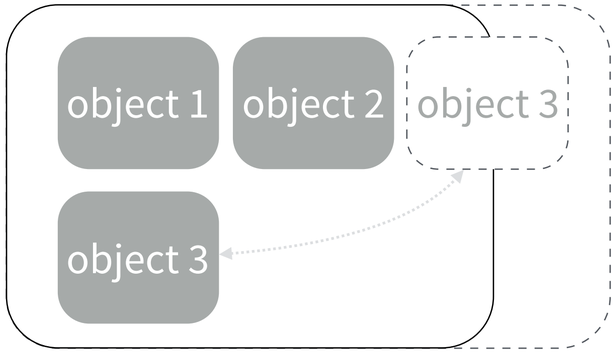
wellPanel()

Panel with grey background.

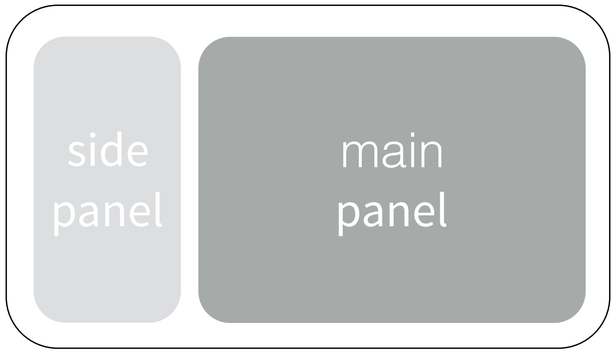


Layout functions

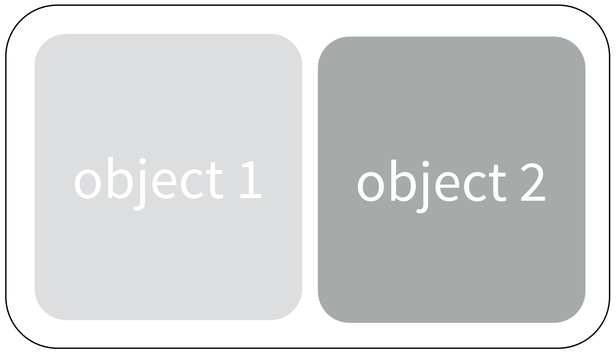
flowLayout()



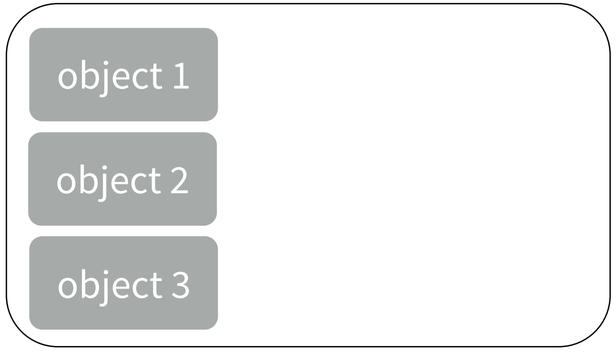
sidebarLayout()



splitLayout()

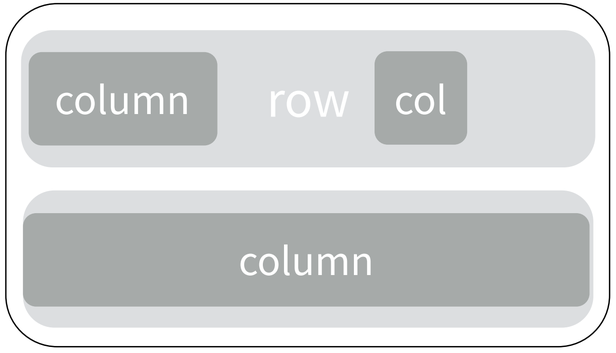


verticalLayout()

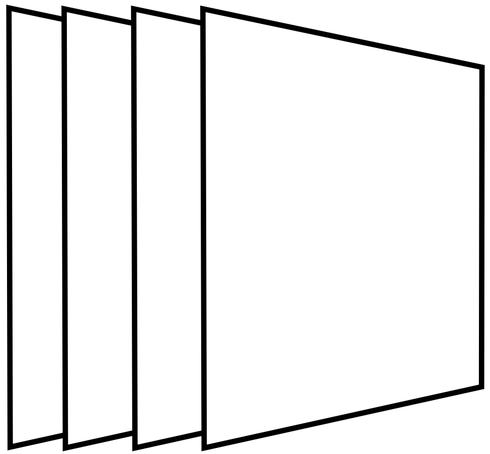


fluidRow()

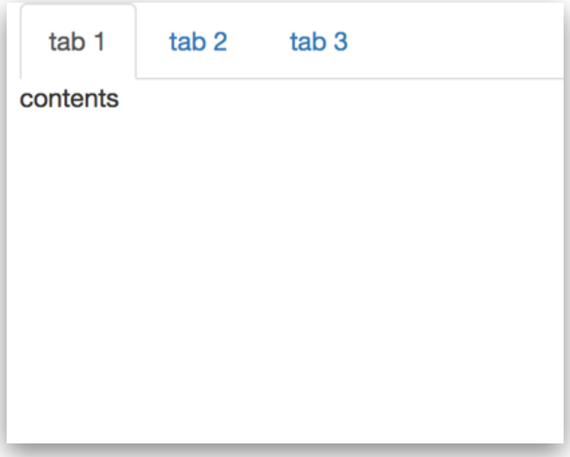
column()



"Layering" functions

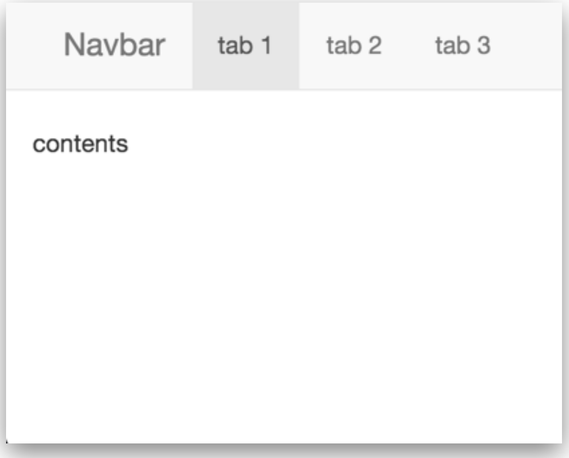


tabPanel()s



tabsetPanel()

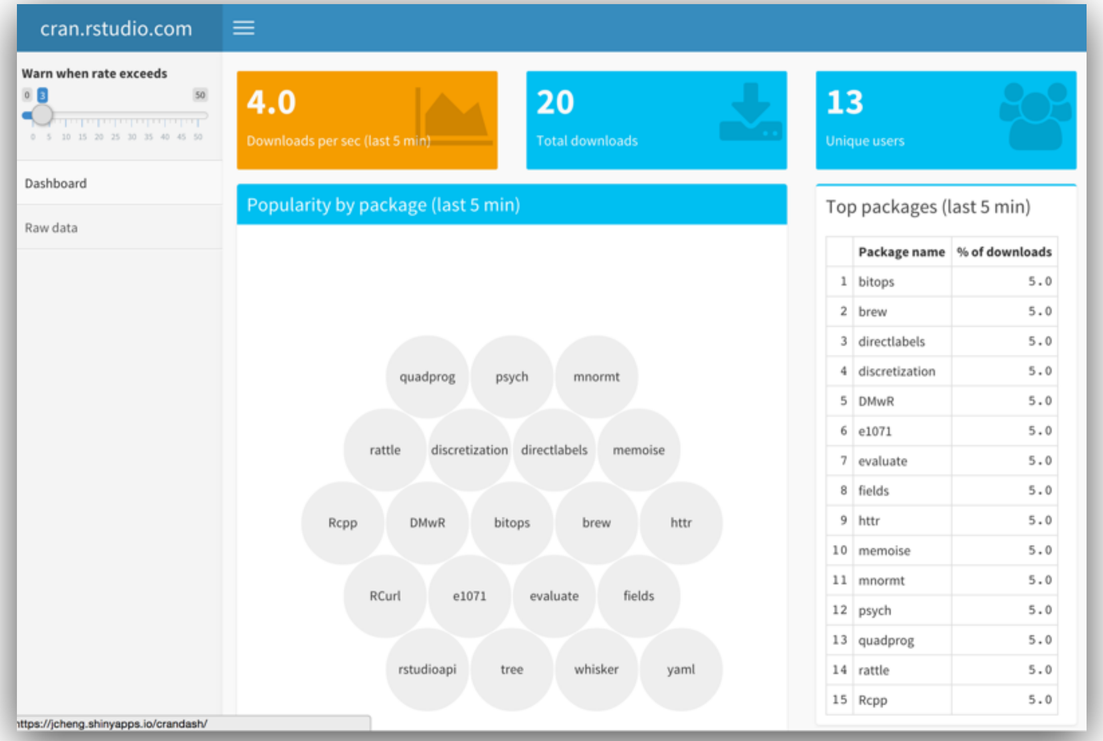
navlistPanel()



navbarPage()

shinydashboard

http://rstudio.github.io/shinydashboard/



A package of layout functions for building administrative dashboards with Shiny

Dynamic Dashboards with Shiny Webinar:

www.rstudio.com/resources/webinars/

What if you want to add your own HTML?

tags

Each element of the tags list is a function that recreates an html tag.

```
names(tags)
##  [1] "a"          "abbr"       "address"    "area"
##  [5] "article"    "aside"      "audio"      "b"
##  [9] "base"       "bdi"        "bdo"        "blockquote"
## [13] "body"       "br"         "button"     "canvas"
## [17] "caption"    "cite"       "code"       "col"
## [21] "colgroup"   "command"    "data"       "datalist"
## [25] "dd"         "del"        "details"    "dfn"
## [29] "div"        "dl"         "dt"         "em"
## [33] "embed"      "eventsource" "fieldset"  "figcaption"
## [37] "figure"     "footer"     "form"       "h1"
## [41] "h2"         "h3"         "h4"         "h5"
```

Shiny HTML tag functions

Shiny provides R functions to recreate HTML tags.

`tags$h1()` ⟷ `<h1></h1>`

`tags$a()` ⟷ `<a></a>`

tags syntax

```
tags$a(href = "www.rstudio.com", "RStudio")
```

the list named tags

the function/tag name (followed by parentheses)

named arguments appear as tag attributes (set boolean attributes to NA)

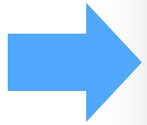
unnamed arguments appear inside the tags (call tags$…() to create nested tags)

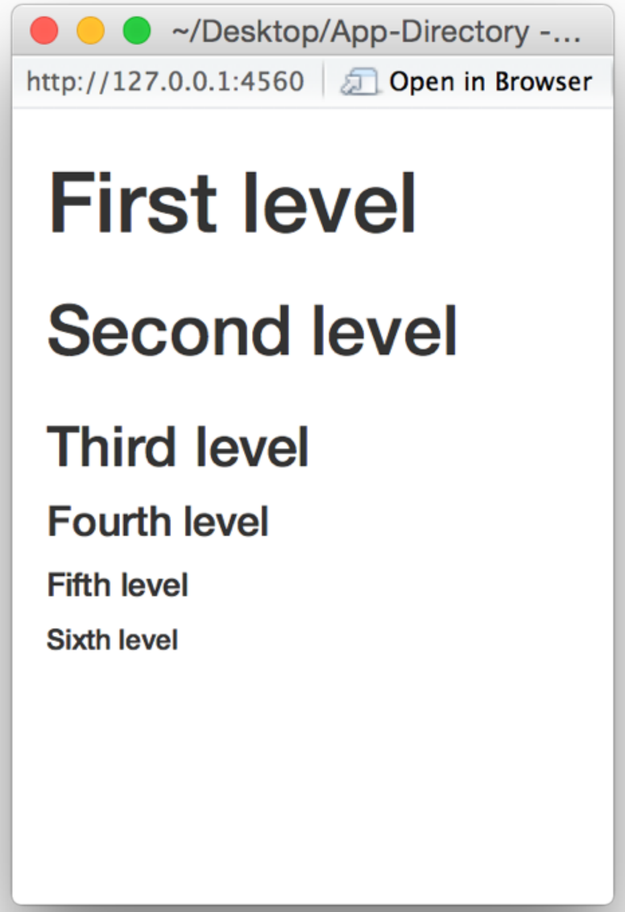
```
<a href="www.rstudio.com">RStudio</a>
```

h1() - h6()

Headers

```
fluidPage(
  tags$h1("First level"),
  tags$h2("Second level"),
  tags$h3("Third level"),
  tags$h4("Fourth level"),
  tags$h5("Fifth level"),
  tags$h6("Sixth level")
)
```

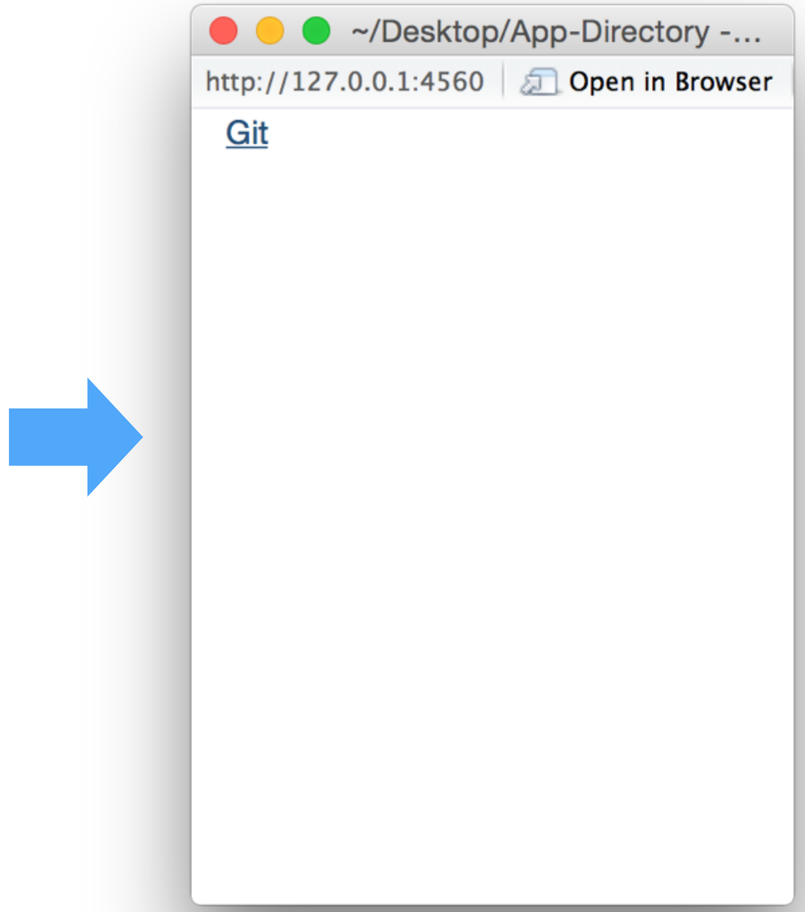




a()

hyperlink with the href argument

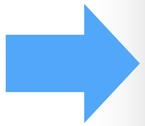
```
fluidPage(
  tags$a(href= "http://www.git.com",
    "Git")
)
```

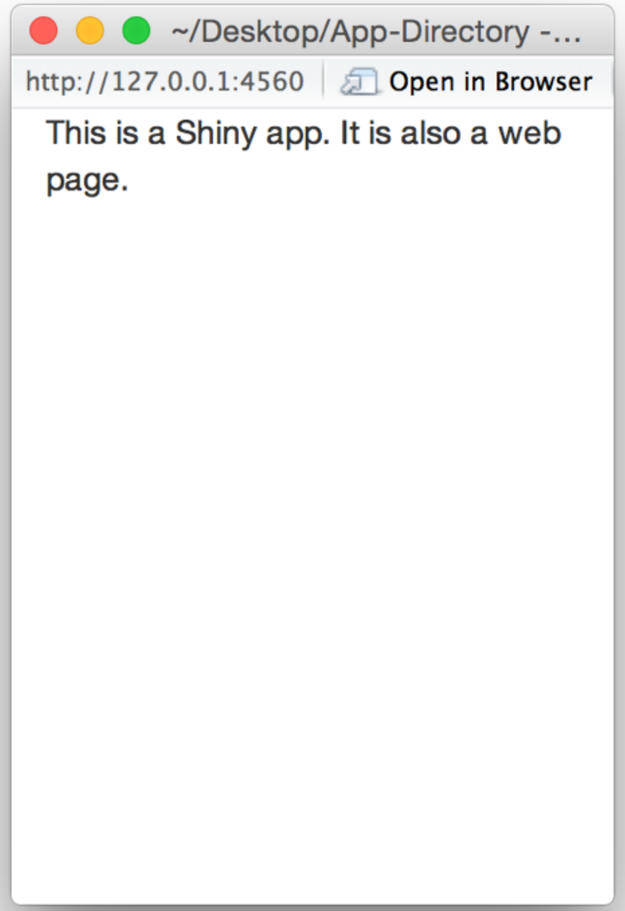


text

Character strings do not need a tag.

```
fluidPage(
 "This is a Shiny app.",
 "It is also a web page."
)
```

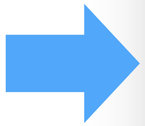


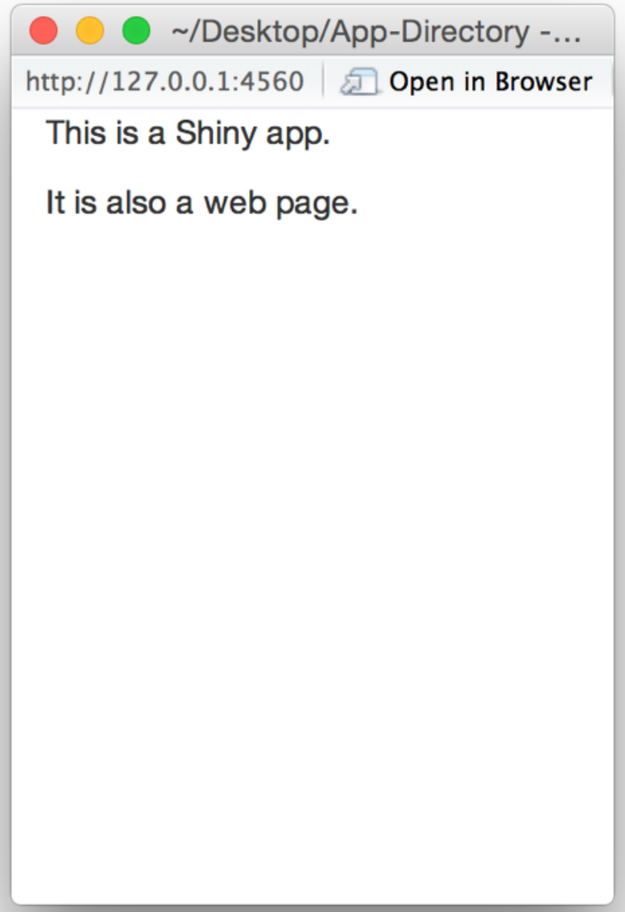


p()

A new paragraph

```
fluidPage(
 tags$p("This is a Shiny app."),
 tags$p("It is also a web page.")
)
```

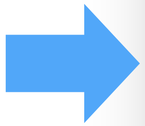


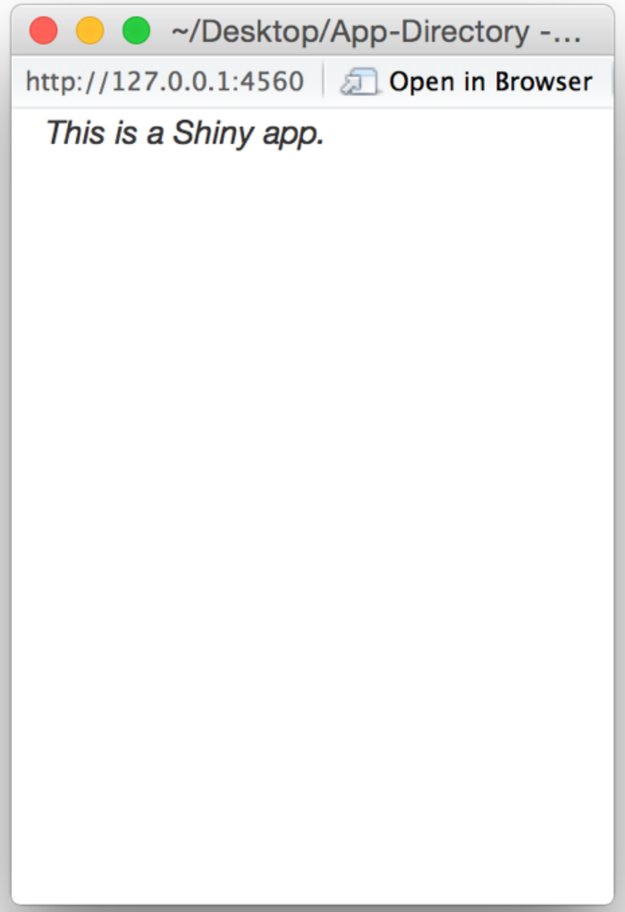


em()

Emphasized (*italic*) text

```
fluidPage(
  tags$em("This is a Shiny app.")
)
```

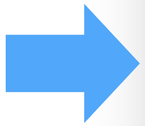


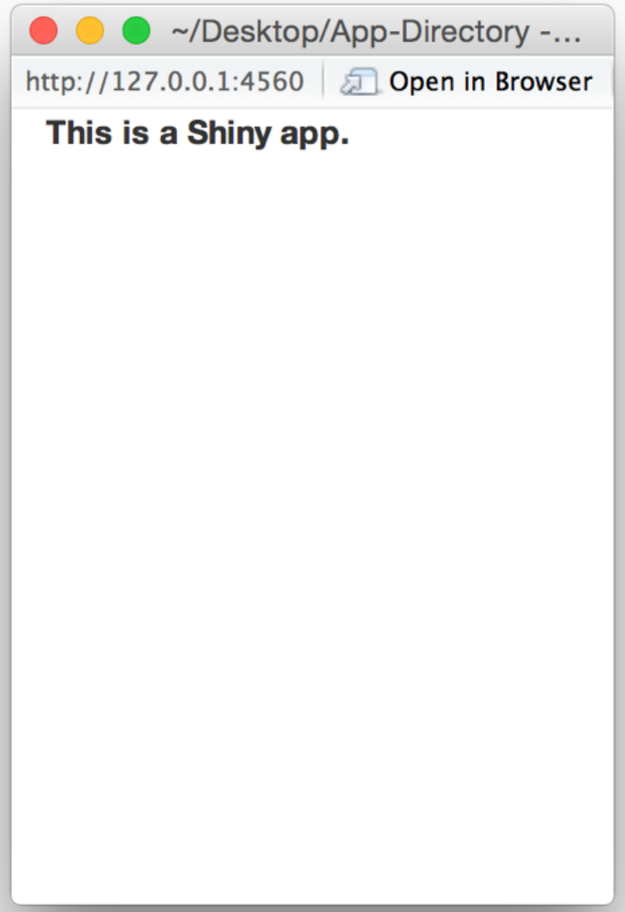


strong()

Strong (**bold**) text

```
fluidPage(
  tags$strong("This is a Shiny app.")
)
```

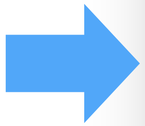


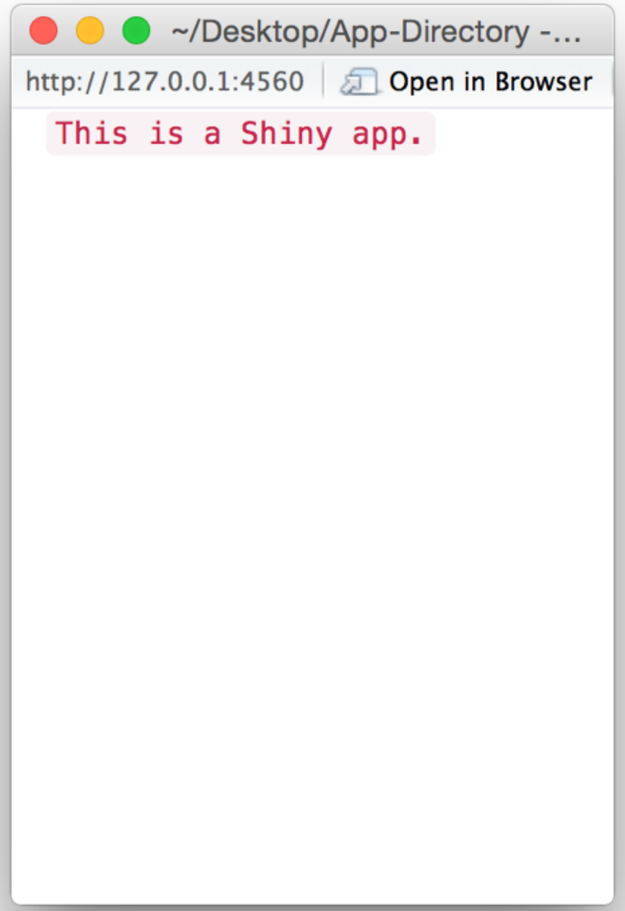


code()

Monospaced text (code)

```
fluidPage(
  tags$code("This is a Shiny app.")
)
```

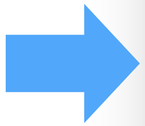


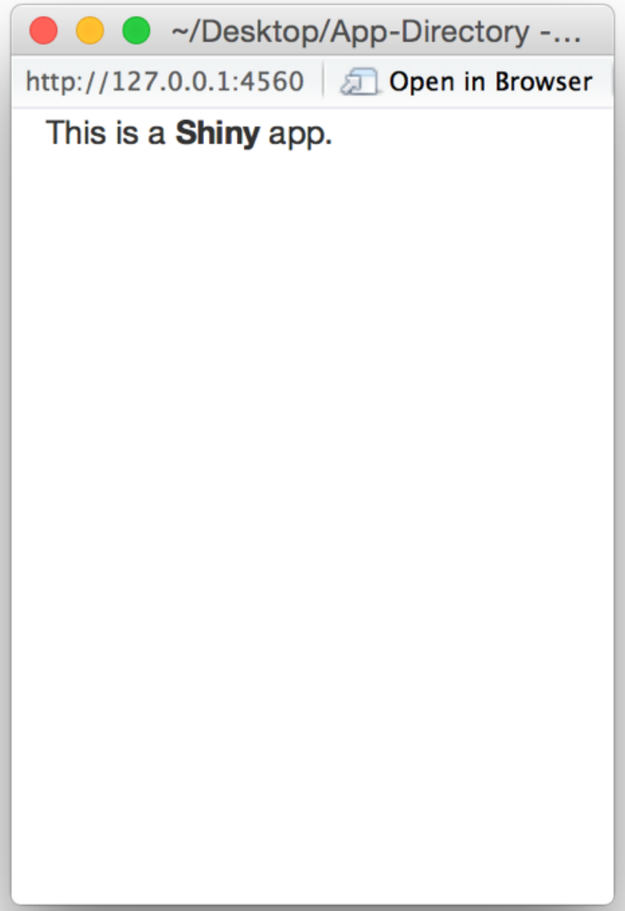


nesting

You can also nest functions inside of others

```
fluidPage(
  tags$p("This is a",
    tags$strong("Shiny"),
    "app.")
)
```

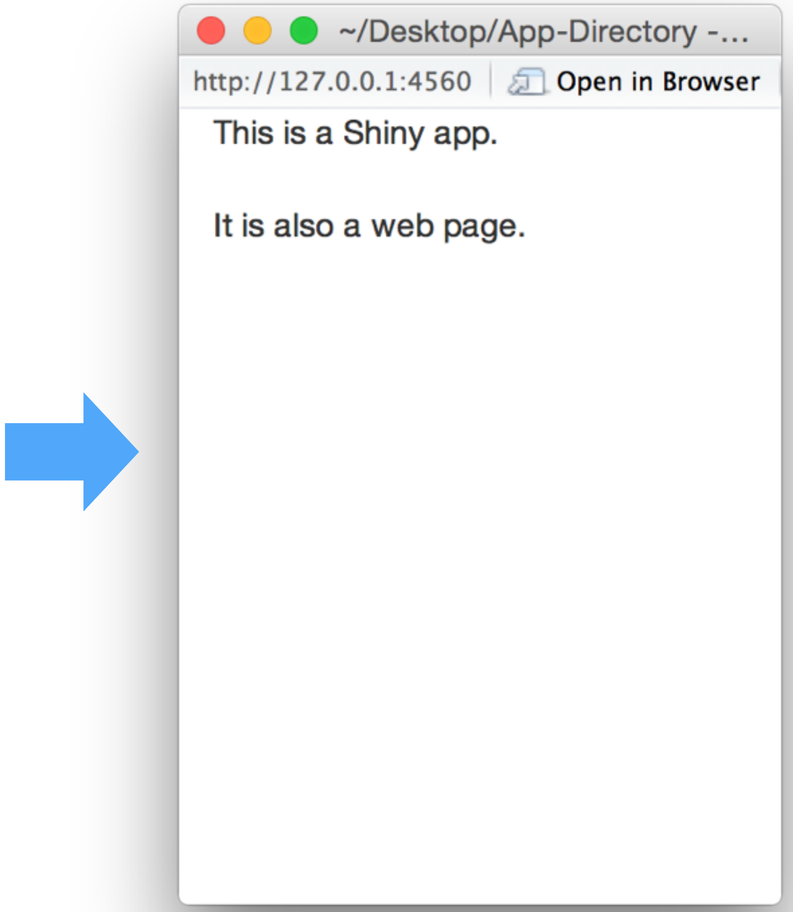




br()

A line break

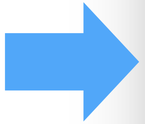
```
fluidPage(
  "This is a Shiny app.",
  tags$br(),
  "It is also a web page."
)
```

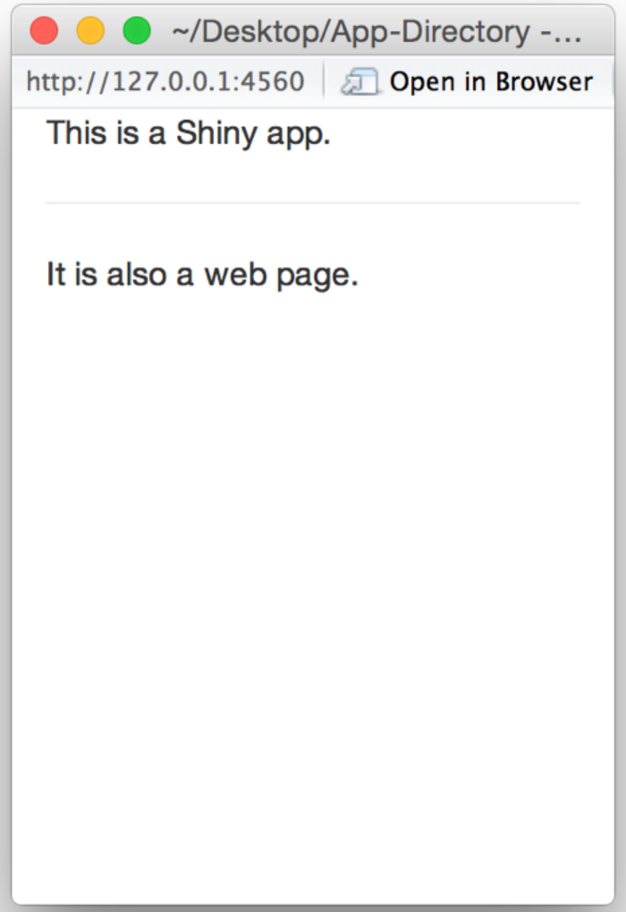


hr()

A horizontal rule

```
fluidPage(
  "This is a Shiny app.",
  tags$hr(),
  "It is also a web page."
)
```

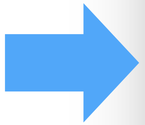




img()

Add an image. Use **src** argument to point to the image URL.

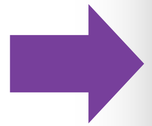
```
fluidPage(
  tags$img(height = 100,
           width = 100,
           src = "http://www.rstudio.com/
                  images/RStudio.2x.png")
)
```

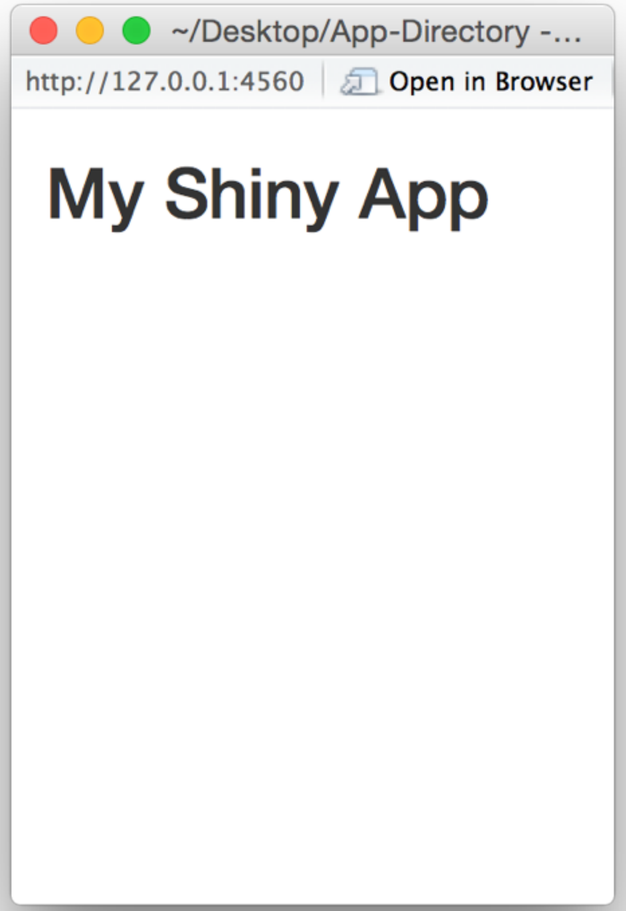


Raw HTML

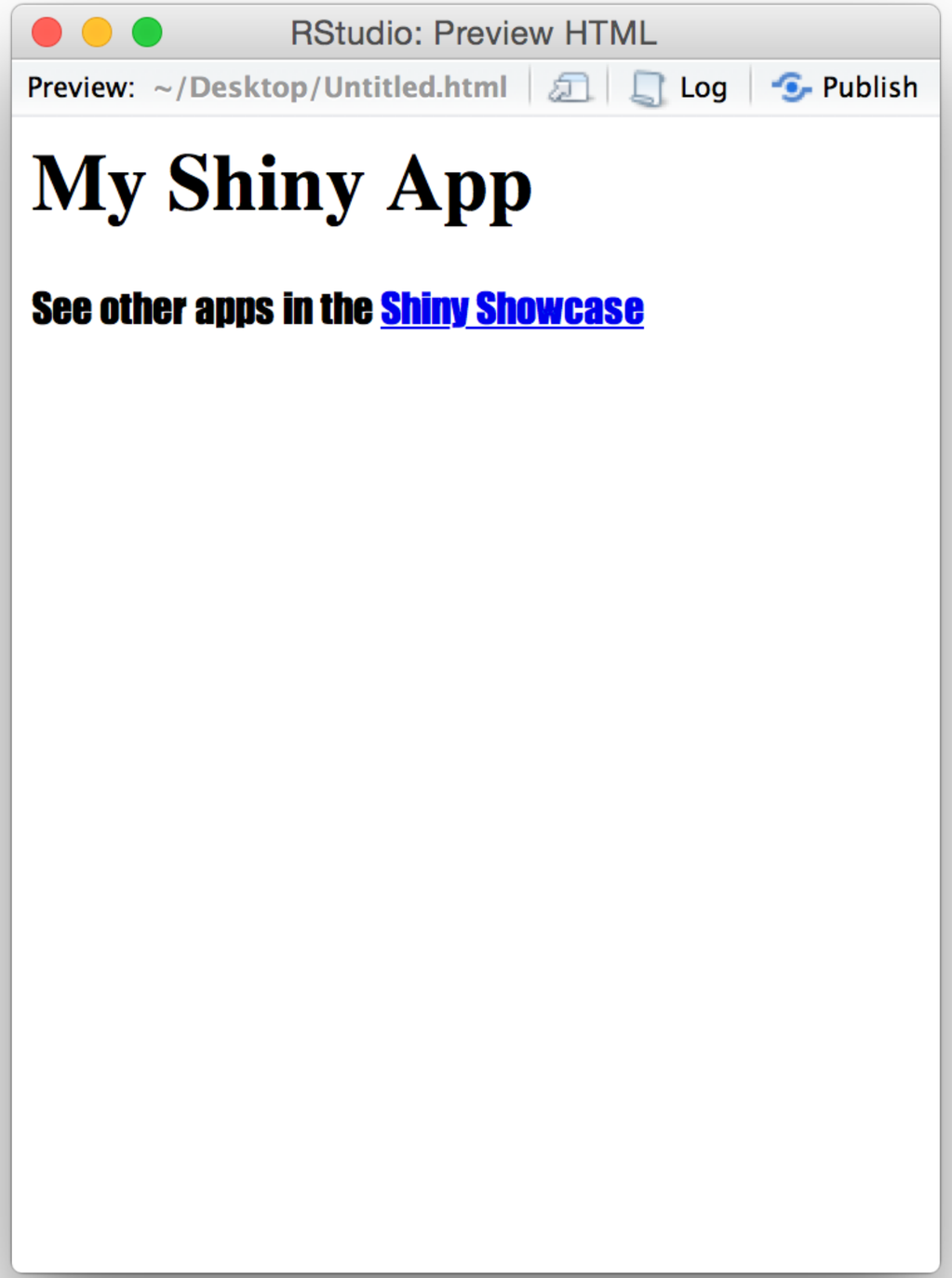
Use HTML() to pass a character string as raw HTML

```
fluidPage(
  HTML("<h1>My Shiny App</h1>")
)
```

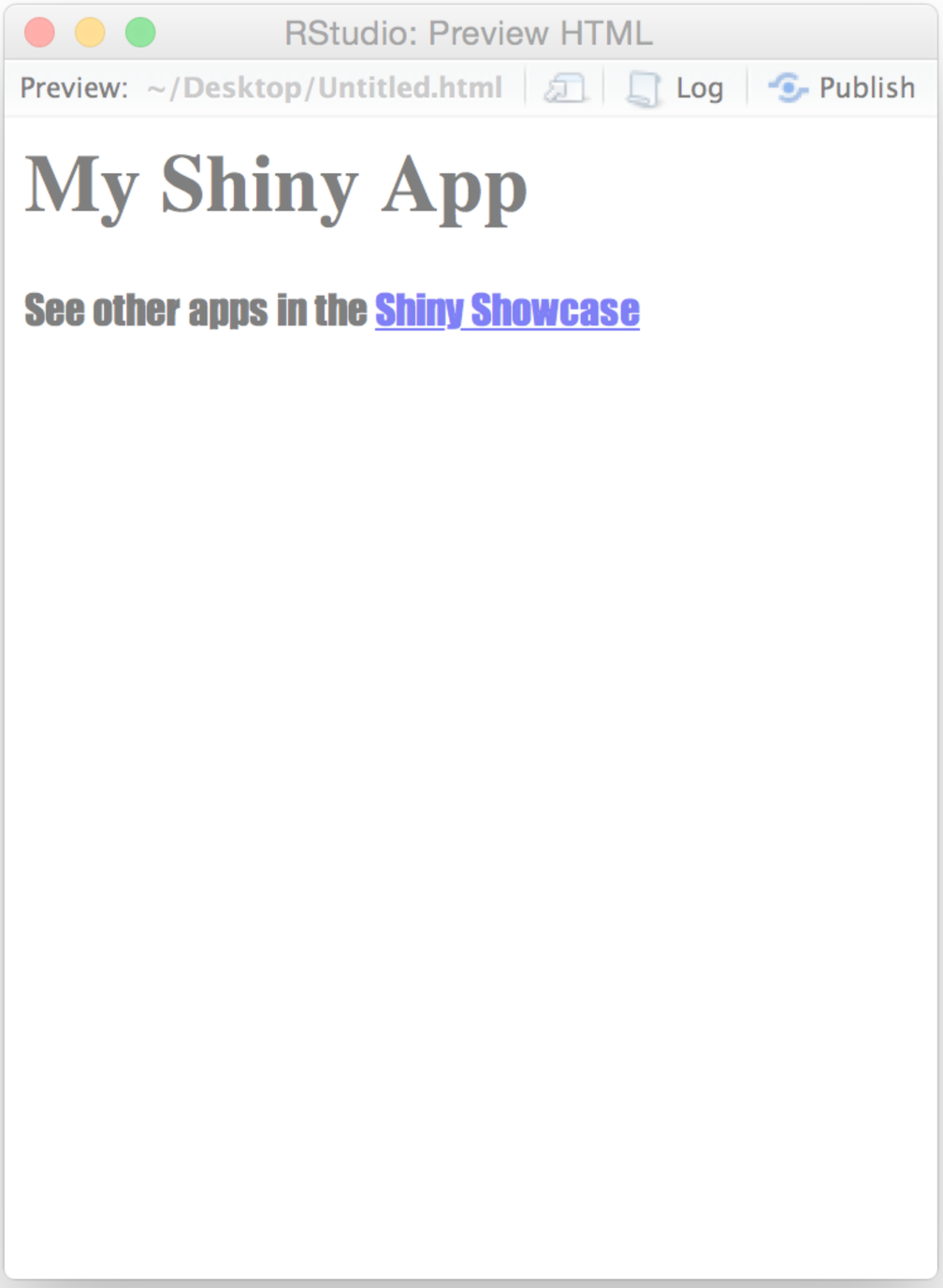




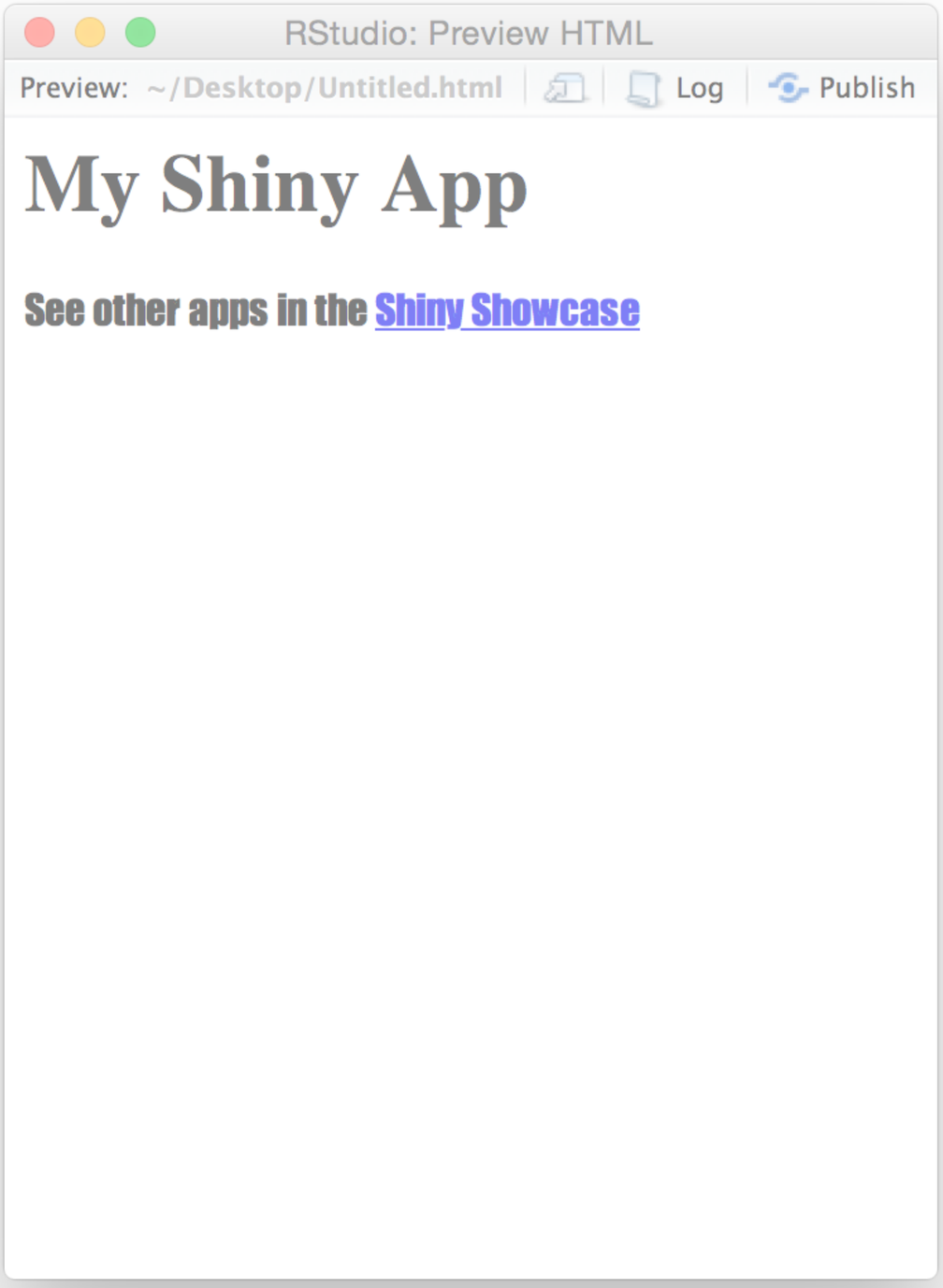
```
<div class="container-fluid">
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
</div>
```



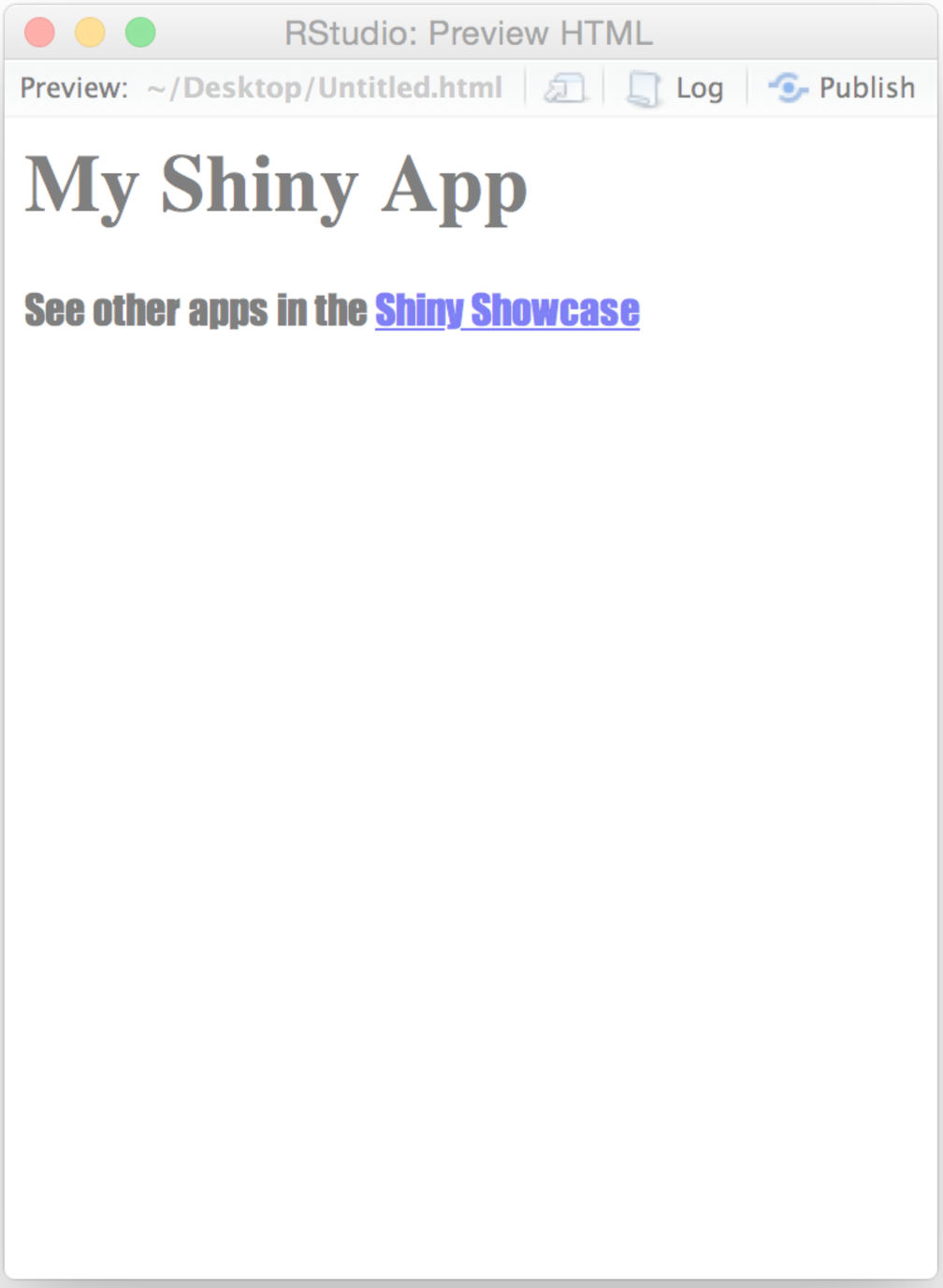
```
fluidPage(
  <h1>My Shiny App</h1>
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



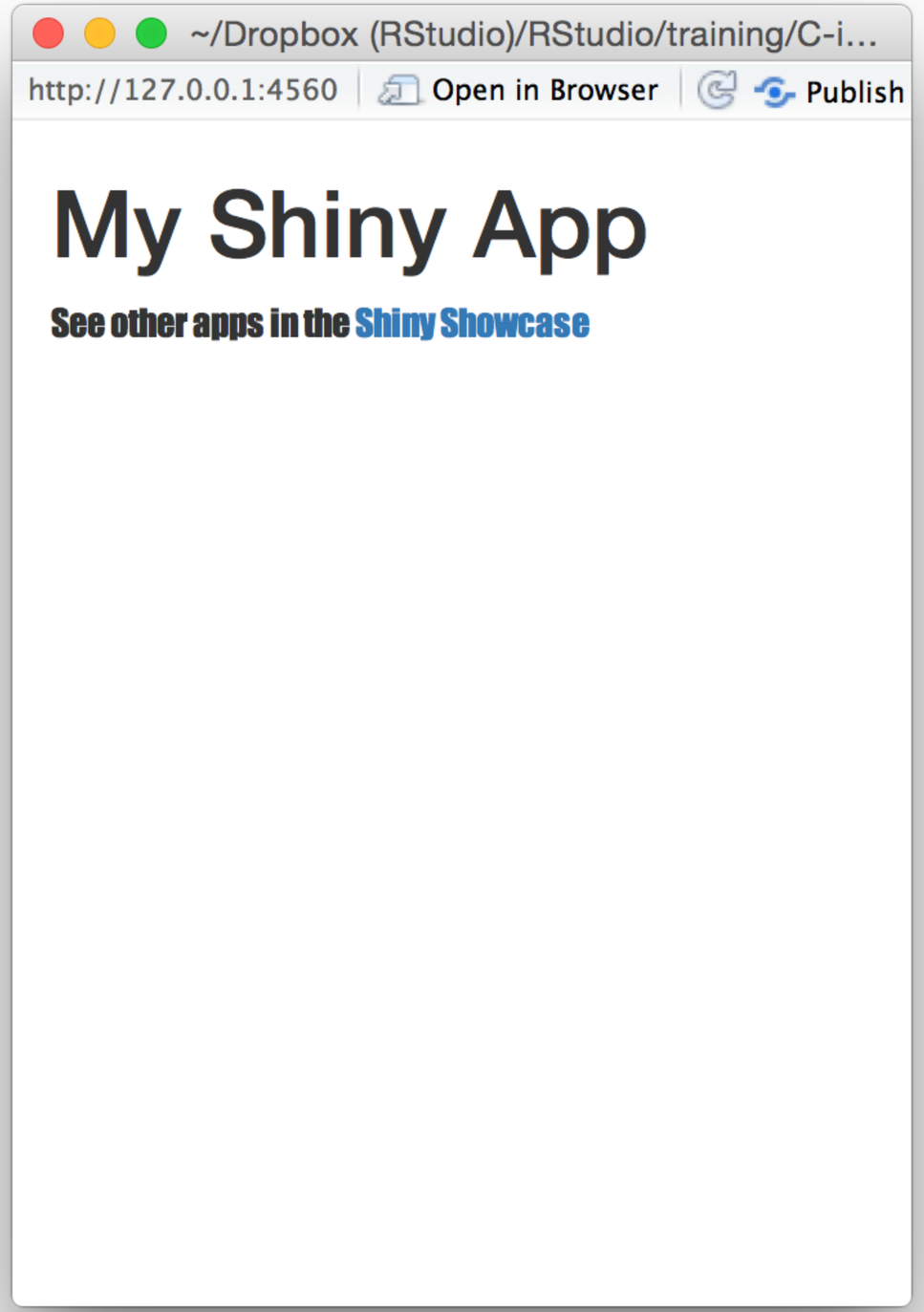
```
fluidPage(
  h1("My Shiny App"),
  <p style="font-family:Impact">
    See other apps in the
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  </p>
)
```



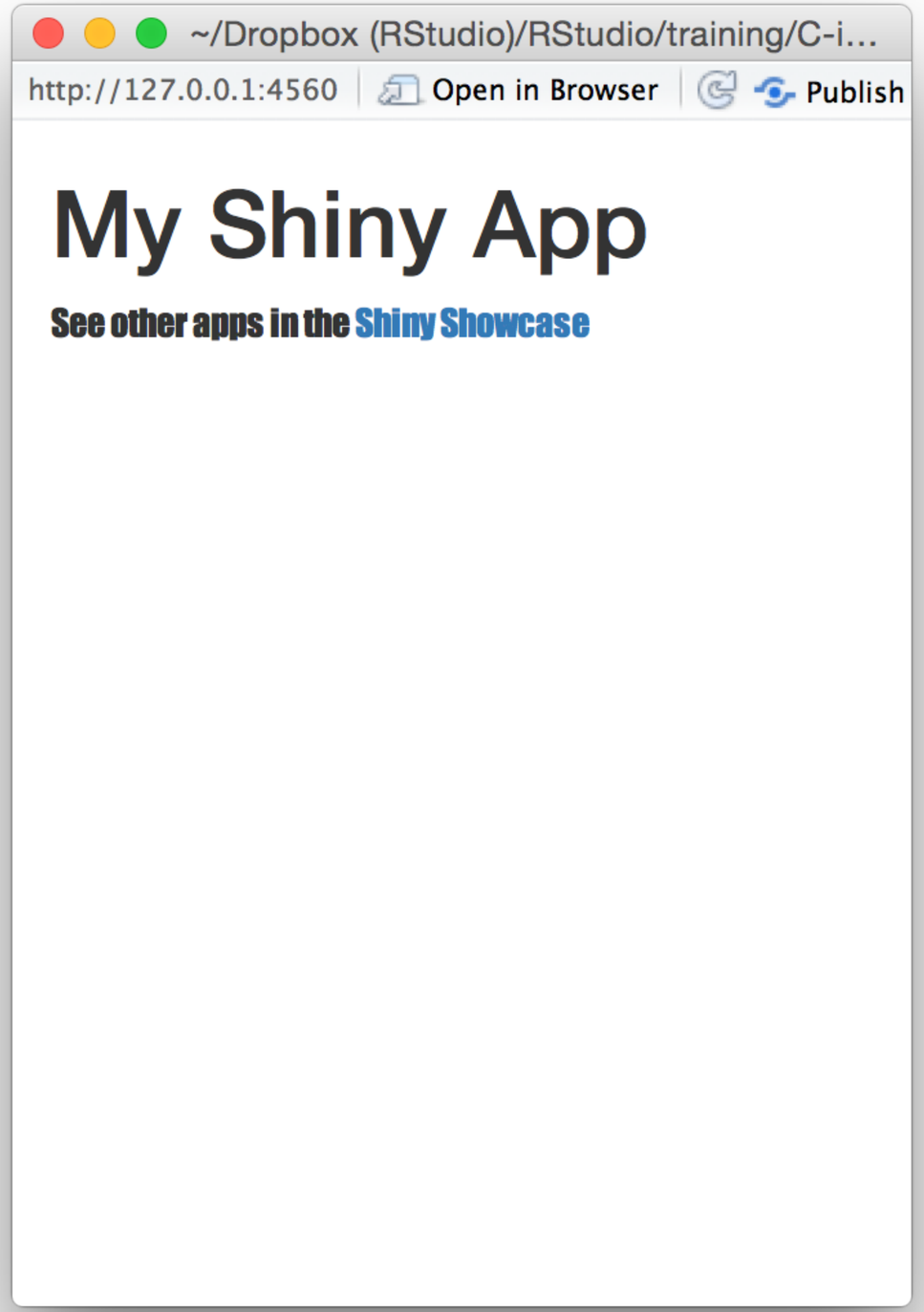
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    <a href="http://www.rstudio.com/
       products/shiny/shiny-user-
       showcase/">Shiny Showcase</a>
  )
)
```



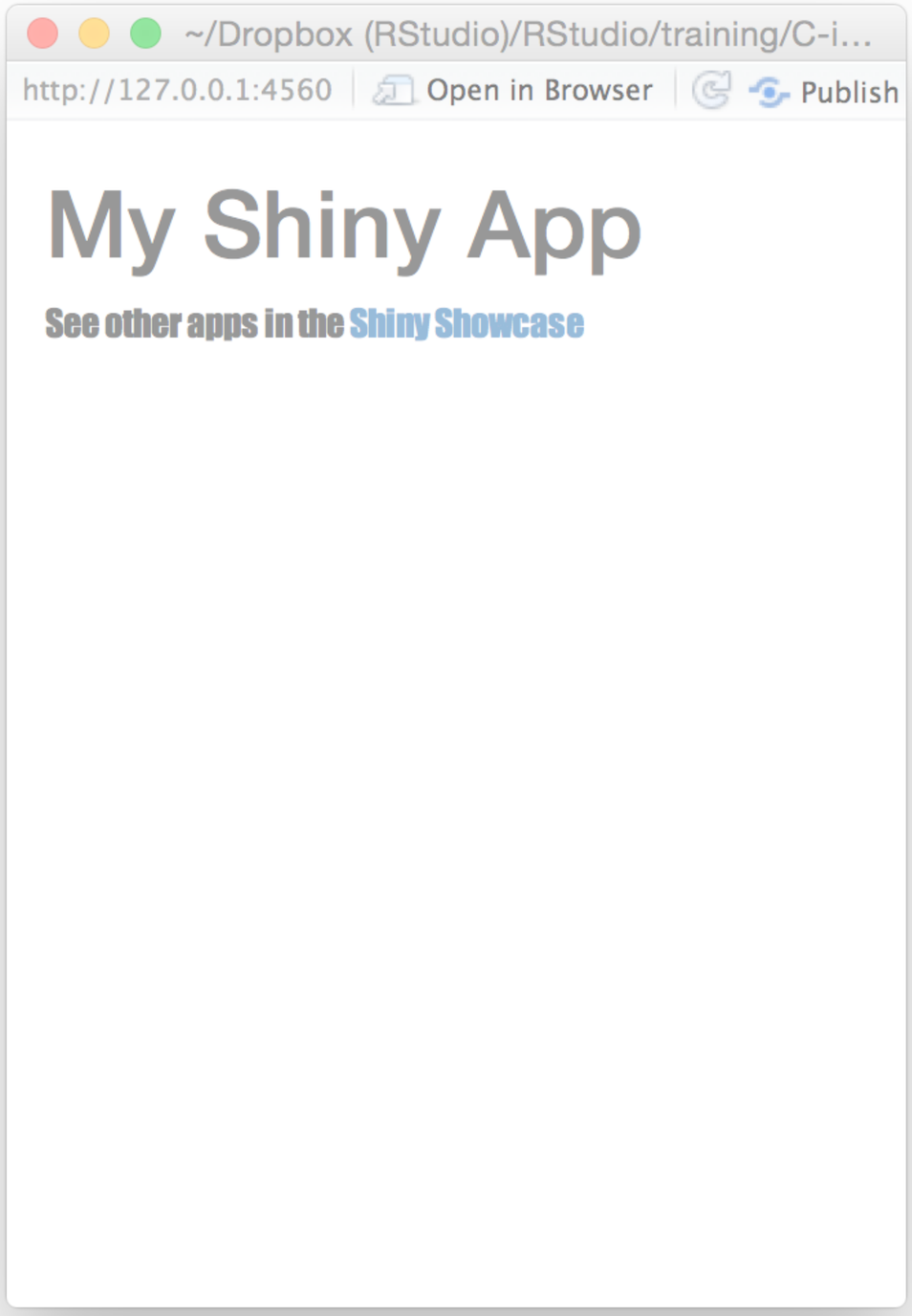
```
fluidPage(
  h1("My Shiny App"),
  p(style = "font-family:Impact",
    "See other apps in the",
    a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



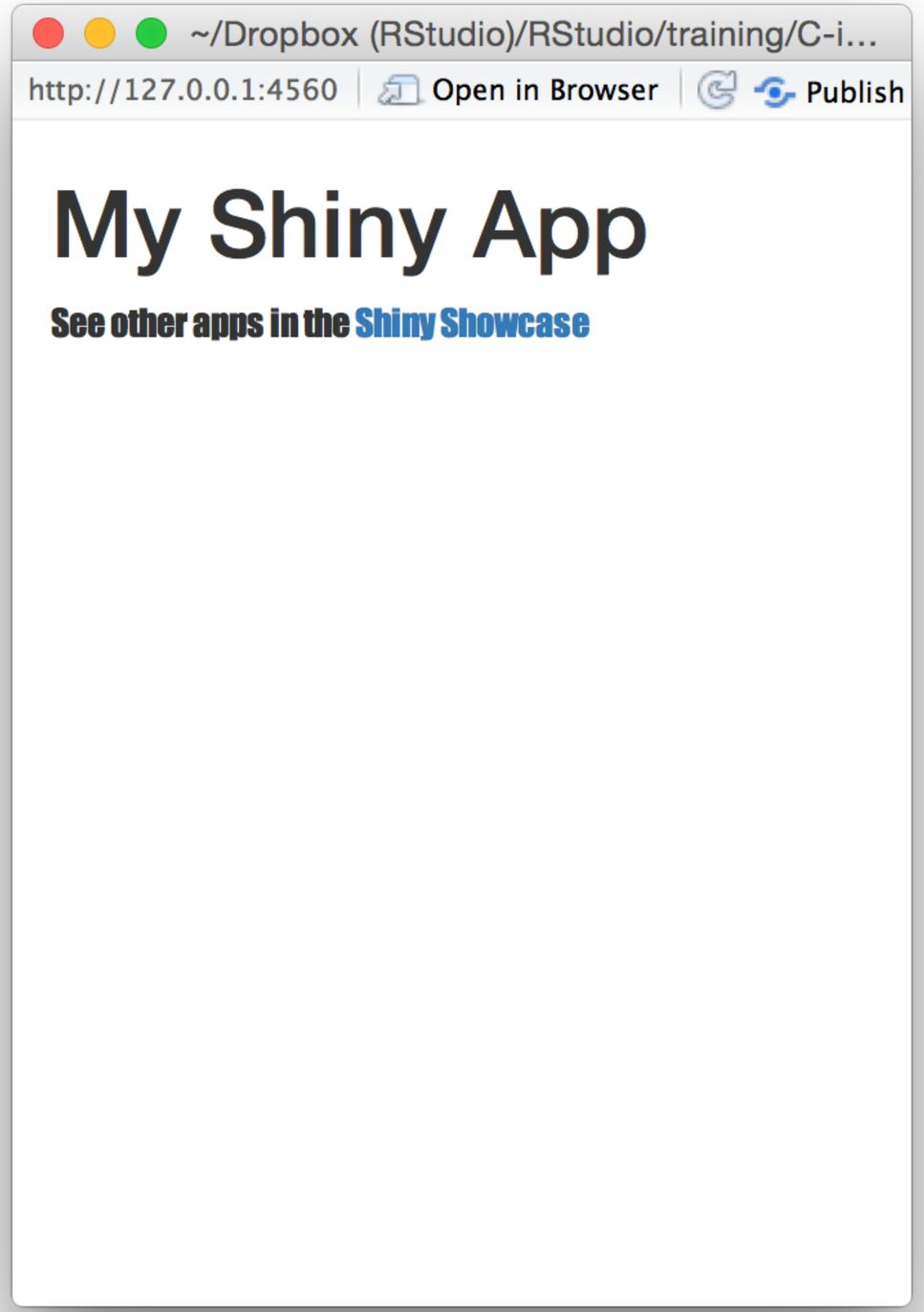
```
fluidPage(
  tags$h1("My Shiny App"),
  tags$p(style = "font-family:Impact",
    "See other apps in the",
    tags$a("Shiny Showcase",
      href = "http://www.rstudio.com/
      products/shiny/shiny-user-showcase/")
  )
)
```



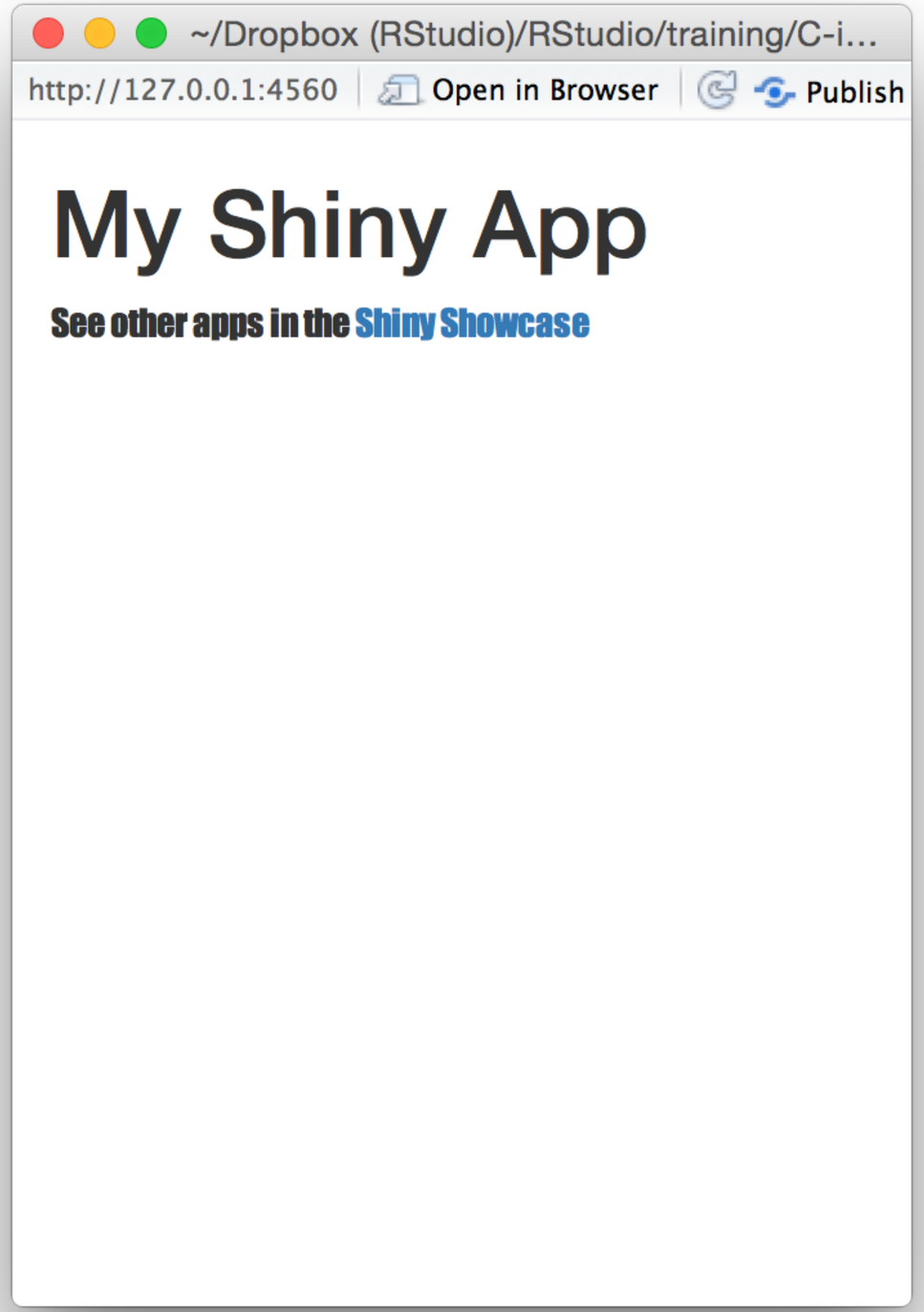
```
fluidPage(

    <div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>

)
```



```
fluidPage(

    '<div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>'

)
```



```
fluidPage(
  HTML(
    '<div class="container-fluid">
      <h1>My Shiny App</h1>
      <p style="font-family:Impact">
        See other apps in the
        <a href="http://www.rstudio.com/
          products/shiny/shiny-user-
          showcase/">Shiny Showcase</a>
      </p>
    </div>'
  )
)
```



Include from file

Pass filepath relative to app directory.

includeCSS()

includeMarkdown()

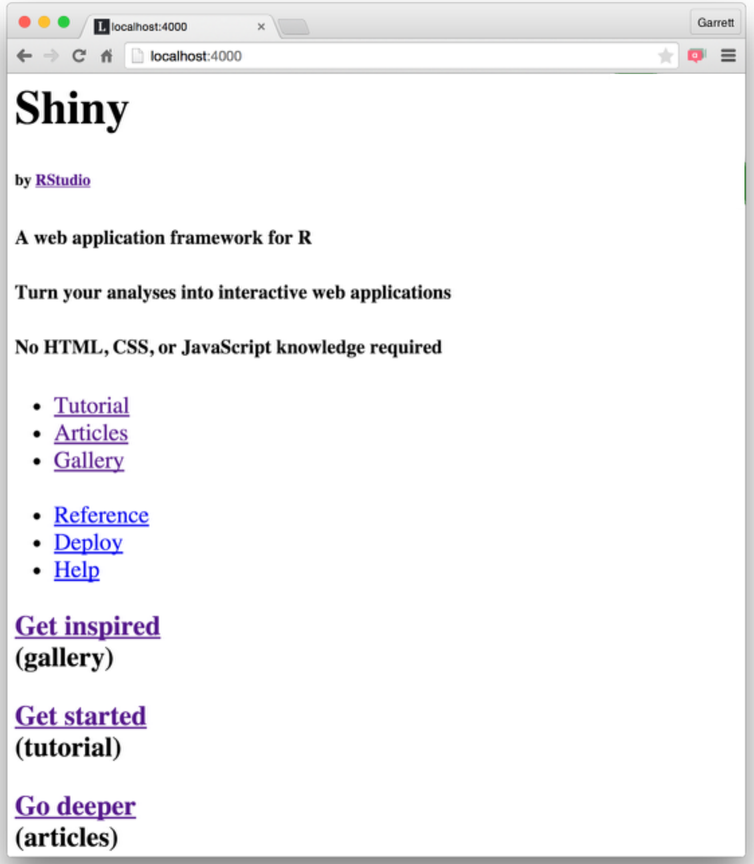
includeHTML()

includeText()

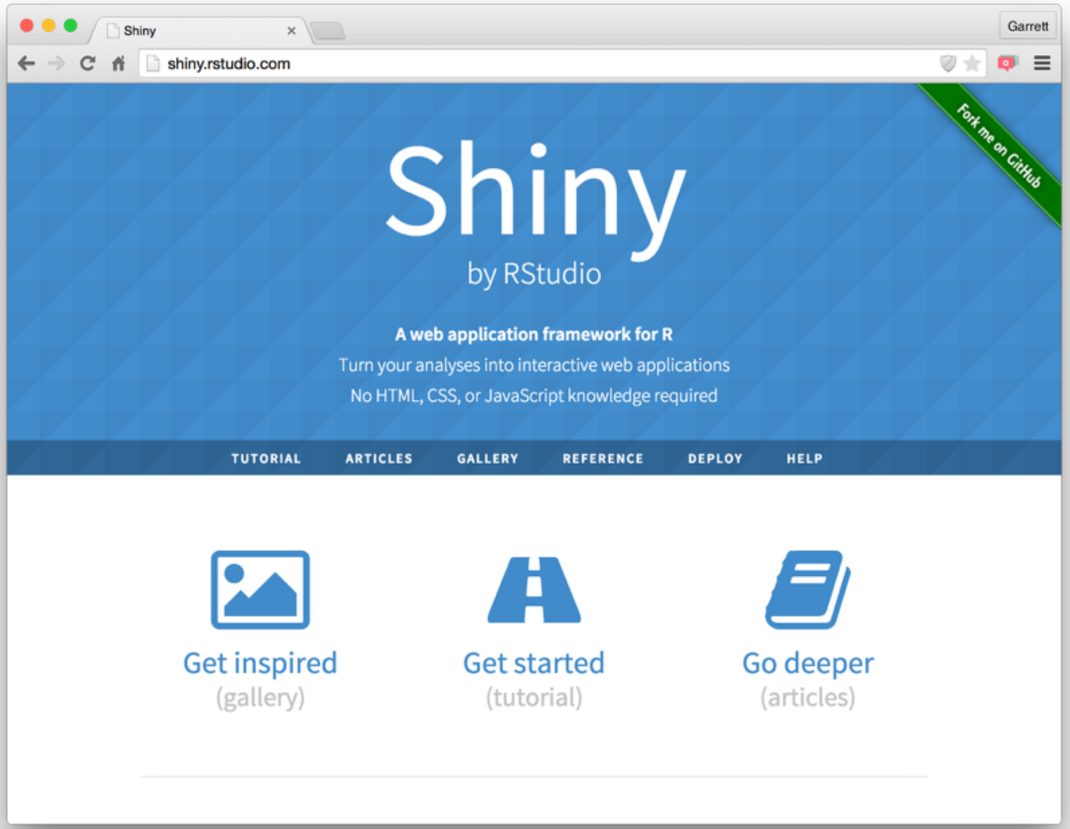
includeScript()

What is CSS?

Cascading Style Sheets (CSS) are a framework for customizing the appearance of elements in a web page.

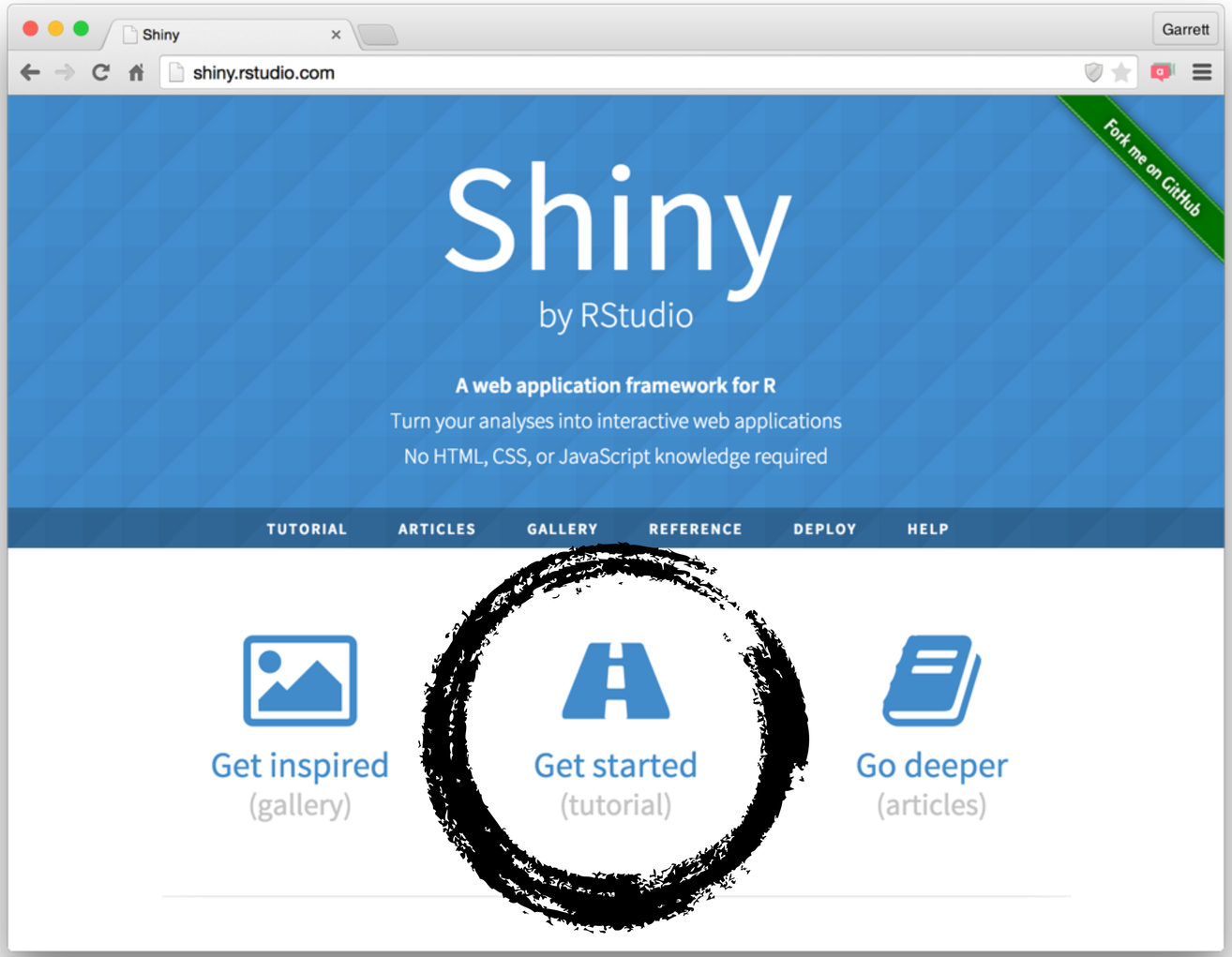


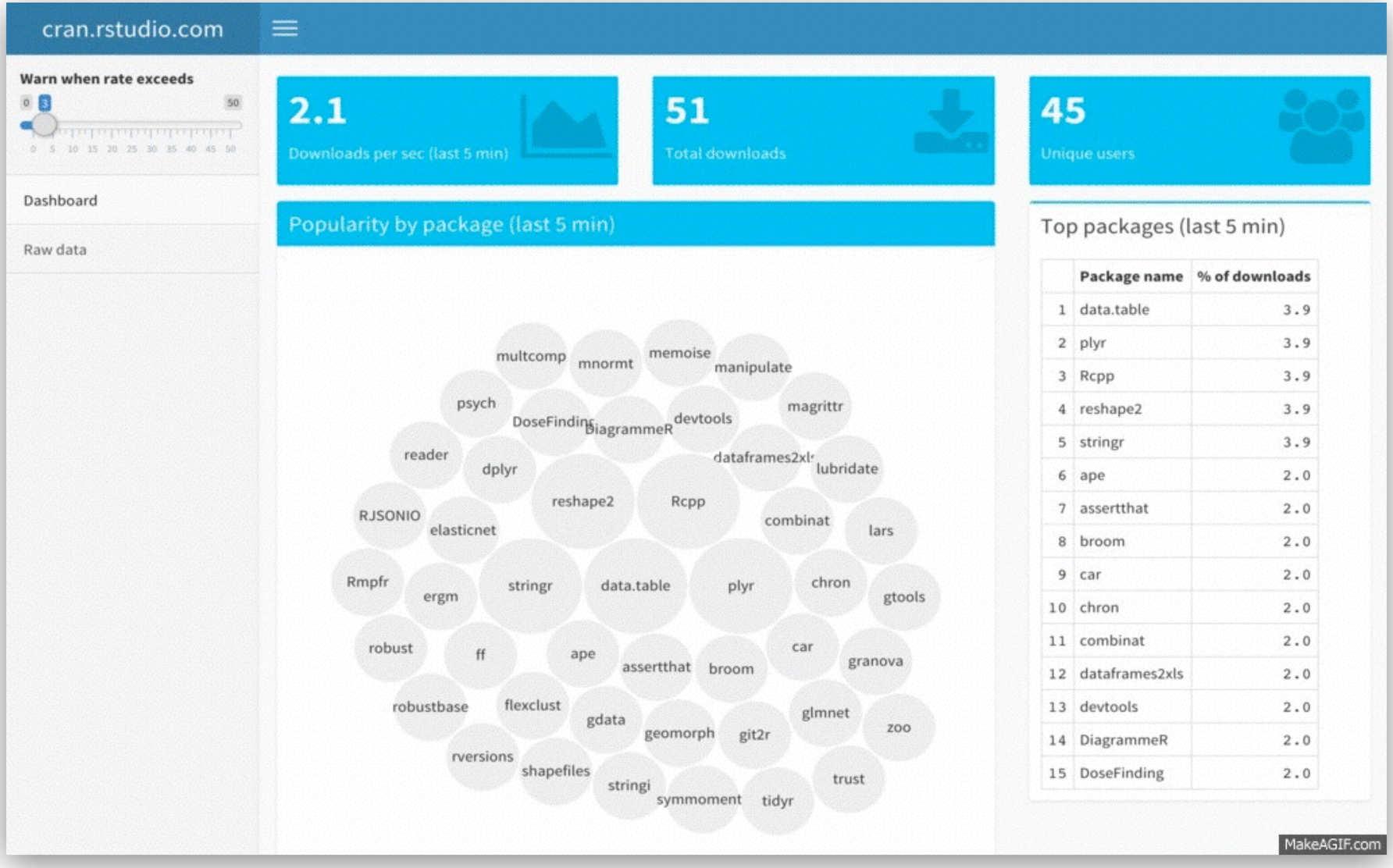
+ CSS



The Shiny Development Center

shiny.rstudio.com



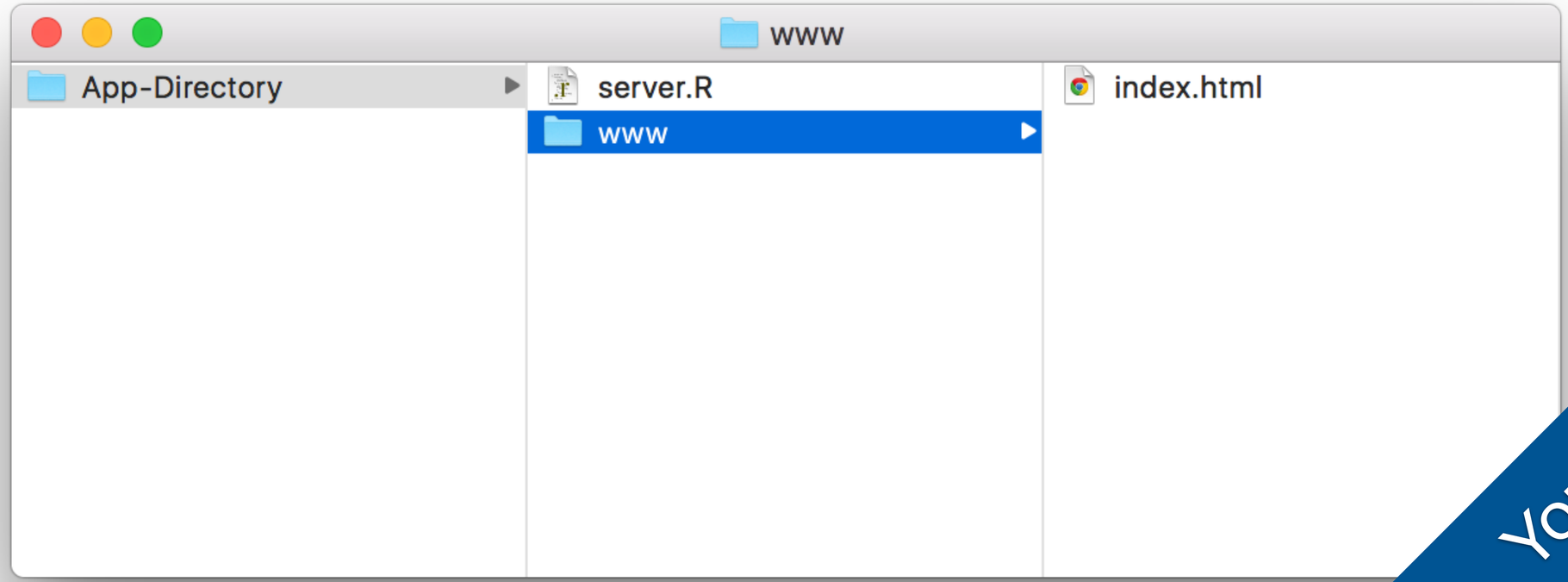


index.html

Build it with html

index.html

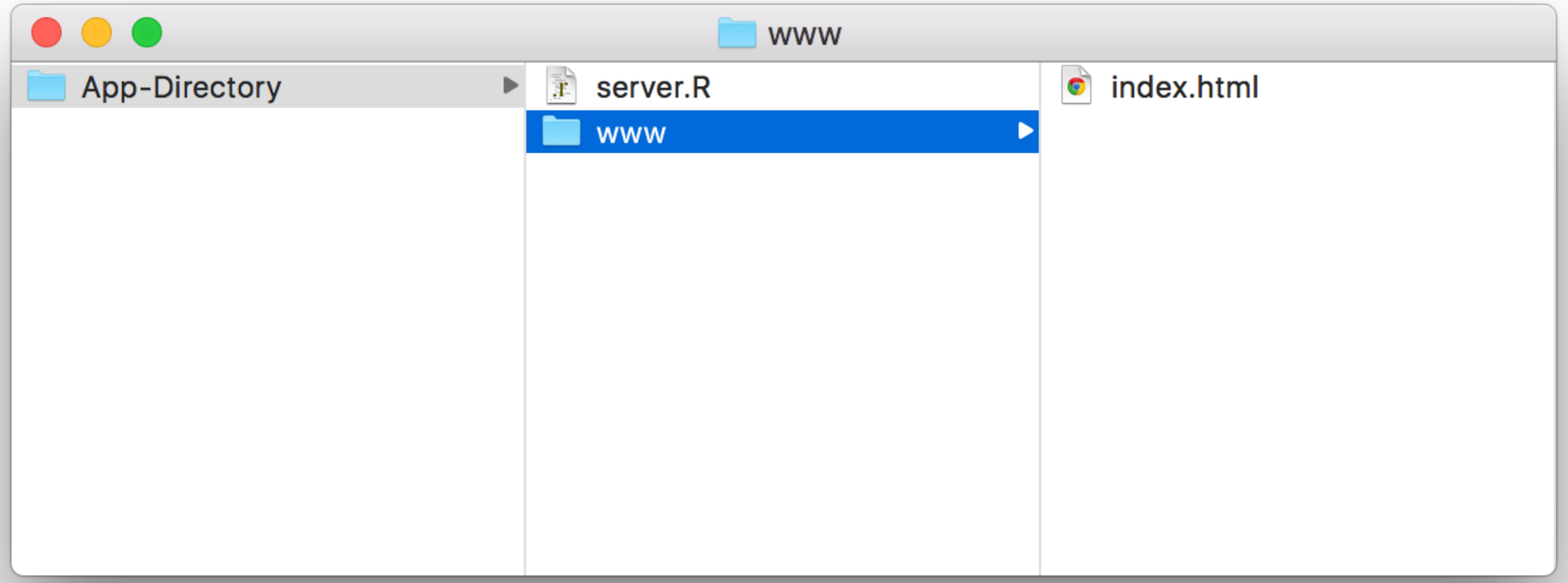
The DOM of your shiny app, written in HTML and saved in the www directory.



You do not write a ui.R

www

To share a file with your user's browser, put it in a subdirectory named **www**.



Adding Shiny elements

```
output$plot <-
  renderPlot({
    dist <- input$dist
    n <- input$n
    hist(data())
  })
```

server.R

❶

```
<div id="plot" class="shiny-plot-output"
    style="width: 100%; height: 400px"></div>


<select name="dist">
  <option value="norm">Normal</option>
  <option value="unif">Uniform</option>
  <option value="lnorm">Log-normal</option>
  <option value="exp">Exponential</option>
</select>
```

index.html

❶ Match output slot to id attribute and CSS class to output type.

Adding Shiny elements

```
output$plot <-
  renderPlot({
    dist <- input$dist
    n <- input$n
    hist(data())
  })
```

server.R

1

```
<div id="plot" class="shiny-plot-output"
    style="width: 100%; height: 400px"></div>
```

2

```
<select name="dist">
  <option value="norm">Normal</option>
  <option value="unif">Uniform</option>
  <option value="lnorm">Log-normal</option>
  <option value="exp">Exponential</option>
</select>
```

index.html

1

Match output slot to id attribute and CSS class to output type.

2

Gather input with a form element. Match the name attribute to the input slot.

HTML UI

shiny.rstudio.com/articles/html-ui.html

Build your entire UI from HTML

```
<html>

<head>
  <script src="shared/jquery.js" type="text/javascript"></script>
  <script src="shared/shiny.js" type="text/javascript"></script>
  <link rel="stylesheet" type="text/css" href="shared/shiny.css"/>
</head>

<body>
  <h1>HTML UI</h1>

  <p>
    <label>Distribution type:</label><br />
```

htmlTemplates

Build it with both

Basic idea

1. Write a template in HTML

```
<!DOCTYPE html>
<html>
  <head>



  </head>
  <body>
    <h1>A Bivariate Normal Distribution</h1>
    <div>



    </div>
  </body>
</html>
```

Basic idea

1. Write a template in HTML
2. Insert R code between {{ }}
3. Call with htmlTemplate() in ui
4. Add CSS, scripts, and other web dependencies just as you would for a web page

```
<!DOCTYPE html>
<html>
  <head>
    {{ headContent() }}
    {{ bootstrapLib() }}
  </head>
  <body>
    <h1>A Bivariate Normal Distribution</h1>
    <div>
      {{ plot }}
      {{ slider }}
    </div>
  </body>
</html>
```

2

Basic idea

Code will be evaluated when template is processed.

You can pass a block of R code (only last item in block will be inserted into page)

Or you can pass a parameter set by htmlTemplate()

```
<!DOCTYPE html>
<html>
  <head>
    {{ headContent() }}
    {{ bootstrapLib() }}
  </head>
  <body>
    <h1>A Bivariate Normal Distribution</h1>
    <div>
      {{ plot }}
      {{ slider }}
   </div>
  </body>
</html>
```

Components

Use a HTML template to create a complete page, or just a component to insert into the UI

app.R

```
ui <- bootstrapPage(
  h2("HTML template example"),
  htmlTemplate("component.html",
    name = "component1")
)
server <- function(input, output) {}
shinyApp(ui, server)
```

component.html

```
<!-- component.html -->
<div>
  This is an HTML template
  named <code>{{ name }}</code>.
</div>
```

headContent()

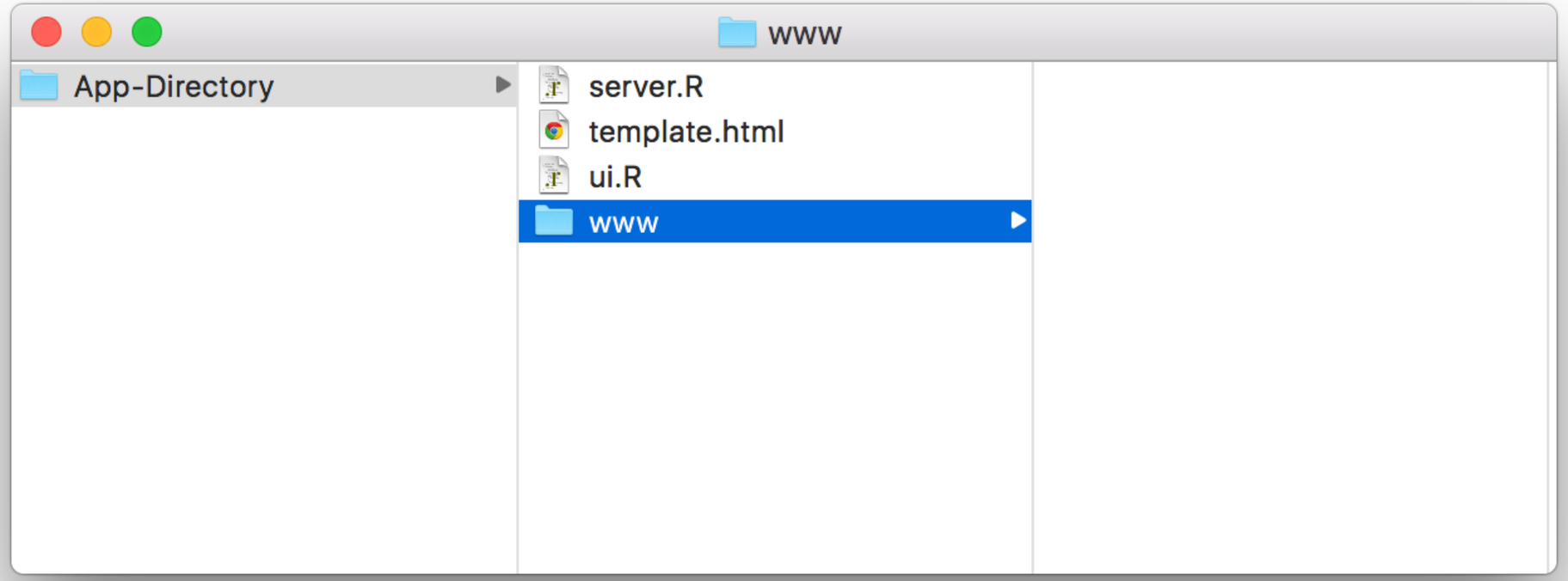
adds required Shiny code to header of a complete page

bootstrapLib()

adds bootstrap CSS, which is required by many Shiny objects (normally loaded by fluidPage(), etc.)

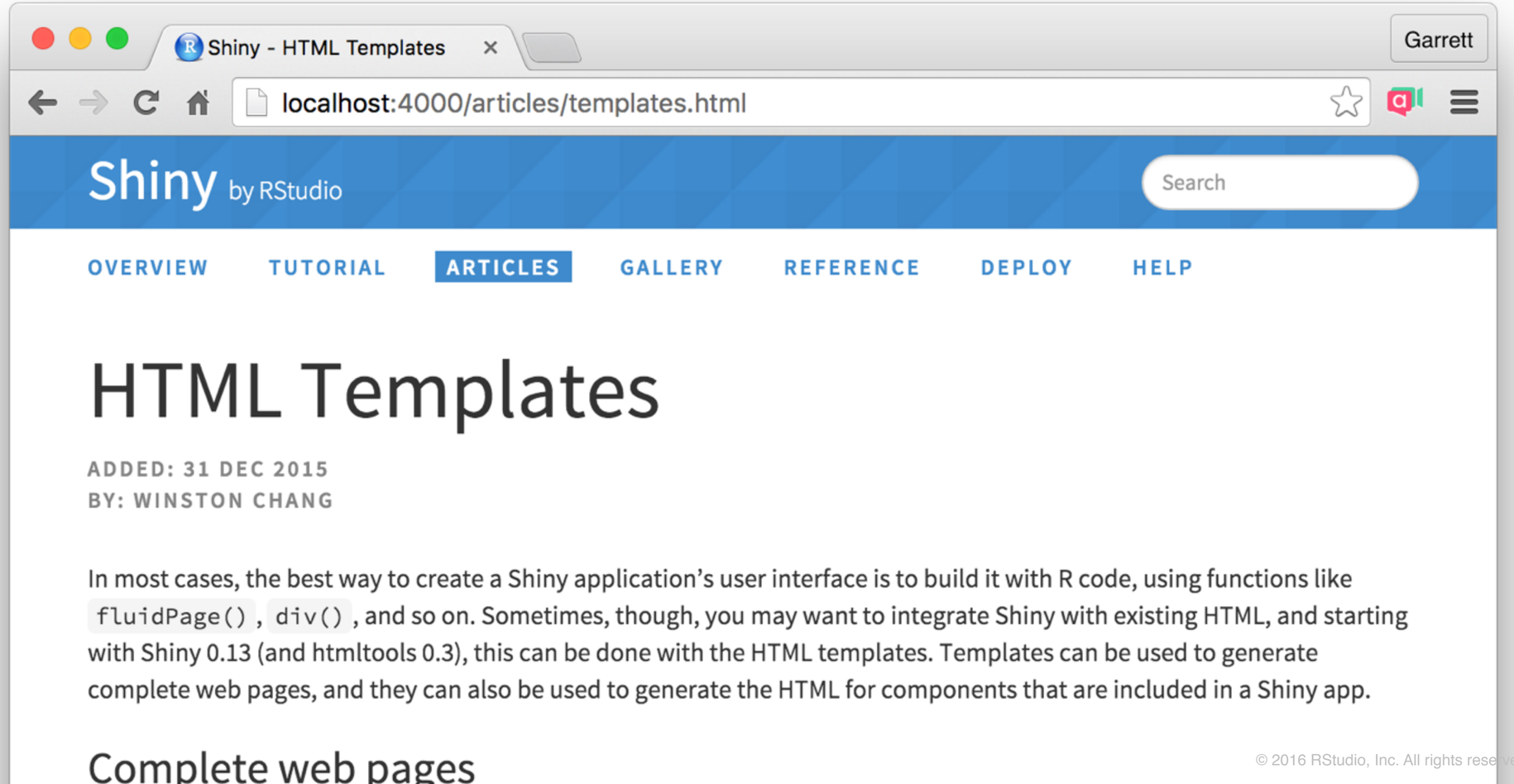
File structure

Place template.html beside app.R, server.R, and ui.R in app directory.



htmlTemplates

shiny.rstudio.com/articles/templates.html



Thank you

Learn more at

shiny.rstudio.com