

# Understanding UI

## How to use HTML to make an awesome web app

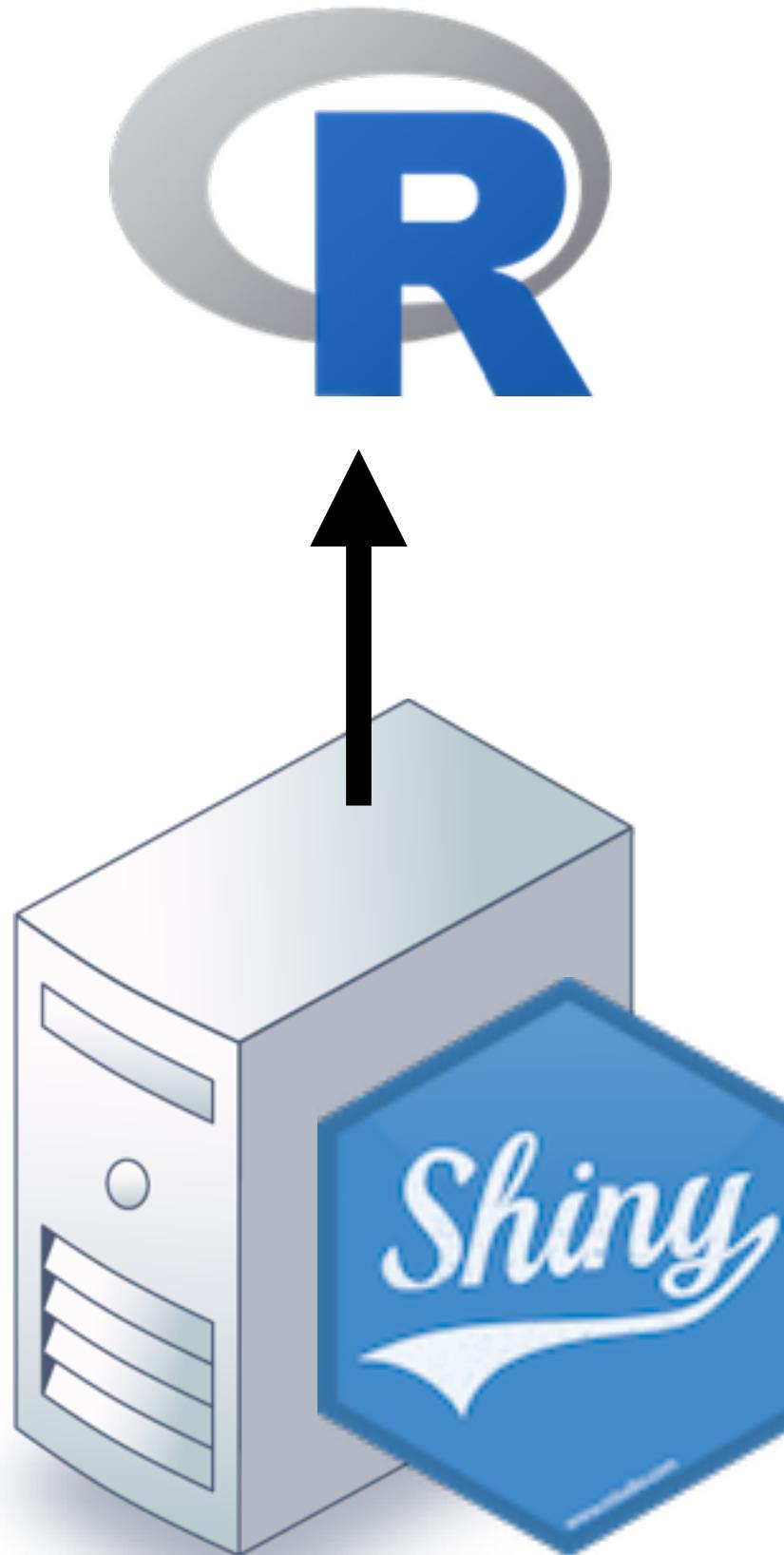


**Garrett Grolemund**  
Data Scientist and Master Instructor  
January 2016  
Email: [garrett@rstudio.com](mailto:garrett@rstudio.com)  
Twitter: @StatGarrett

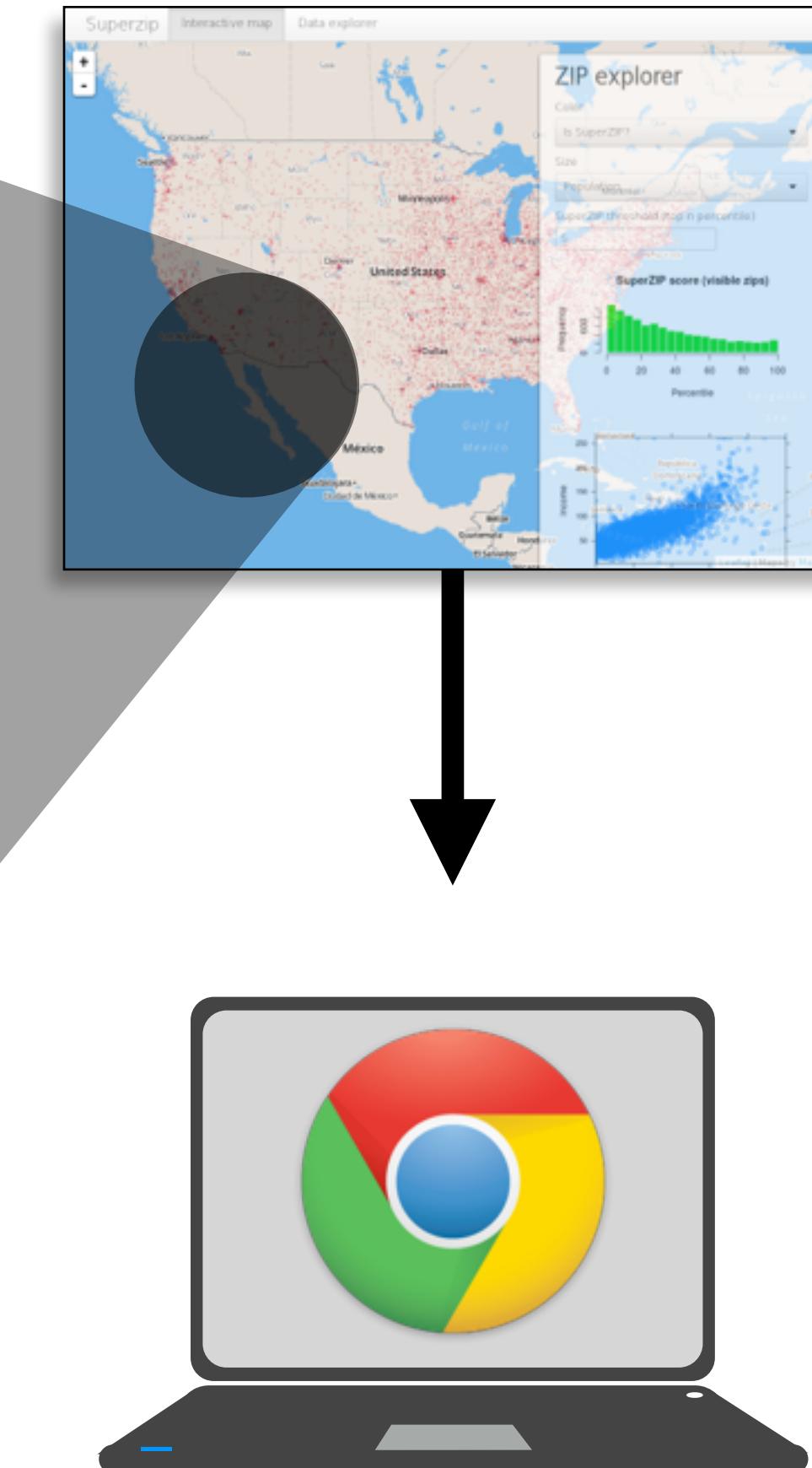
# **UI matters.**

**(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**

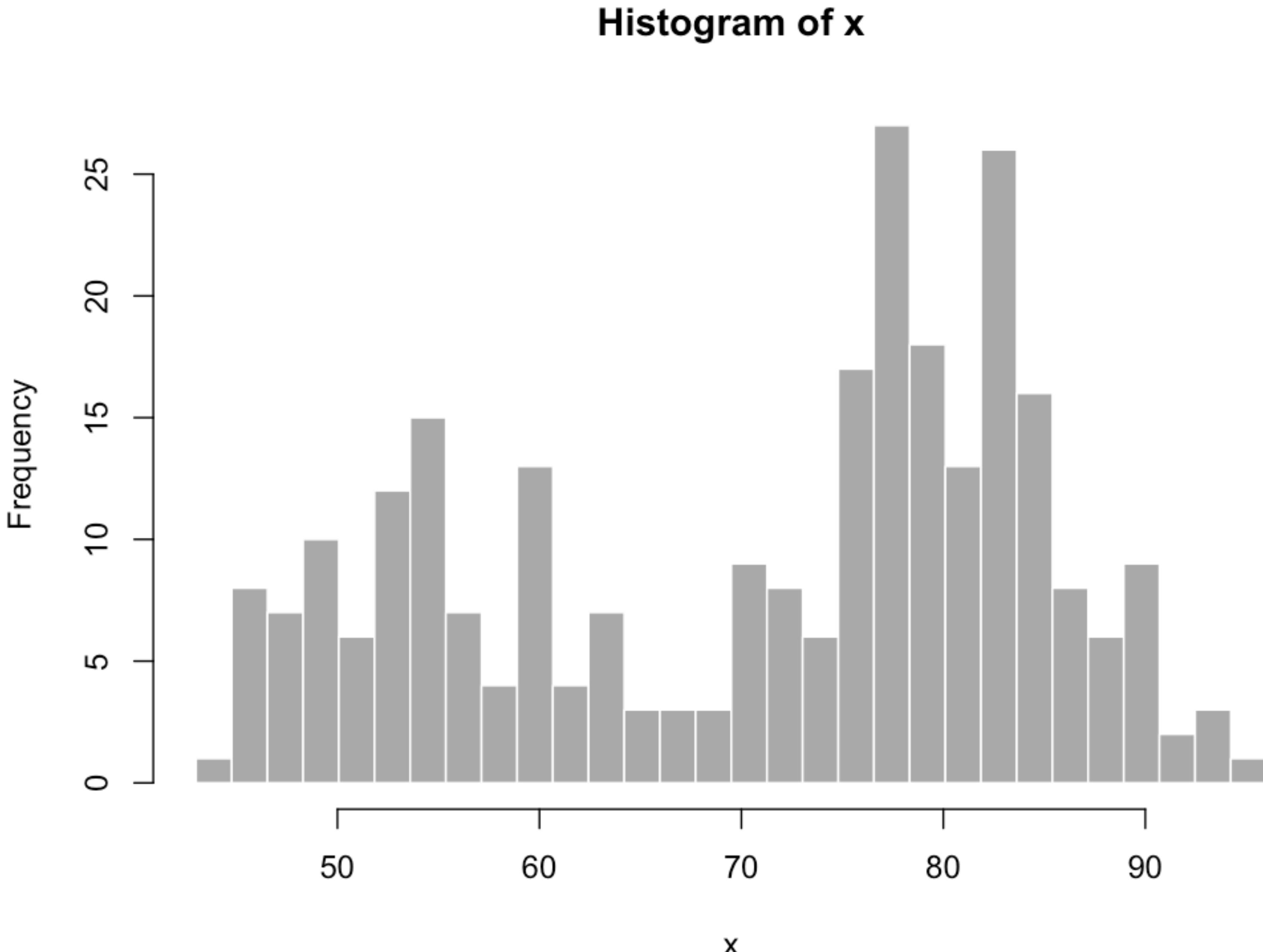
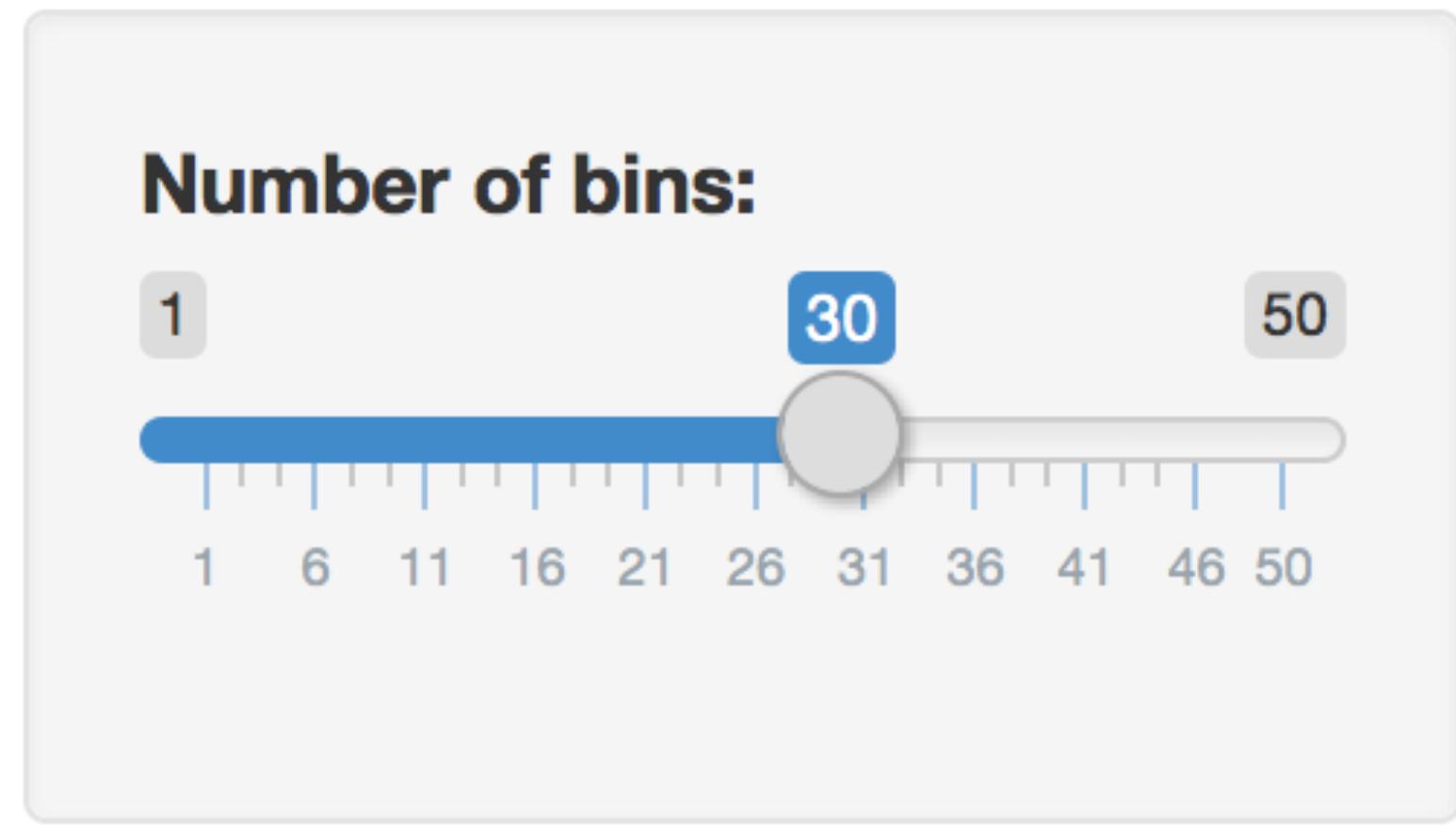
# App template

## The shortest viable shiny app



```
library(shiny)  
  
ui <- fluidPage()  
  
server <- function(input, output) {}  
  
shinyApp(ui = ui, server = server)
```

# Old Faithful Geyser Data



The HTML that builds the user interface for your app

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

```
fluidPage()
```

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

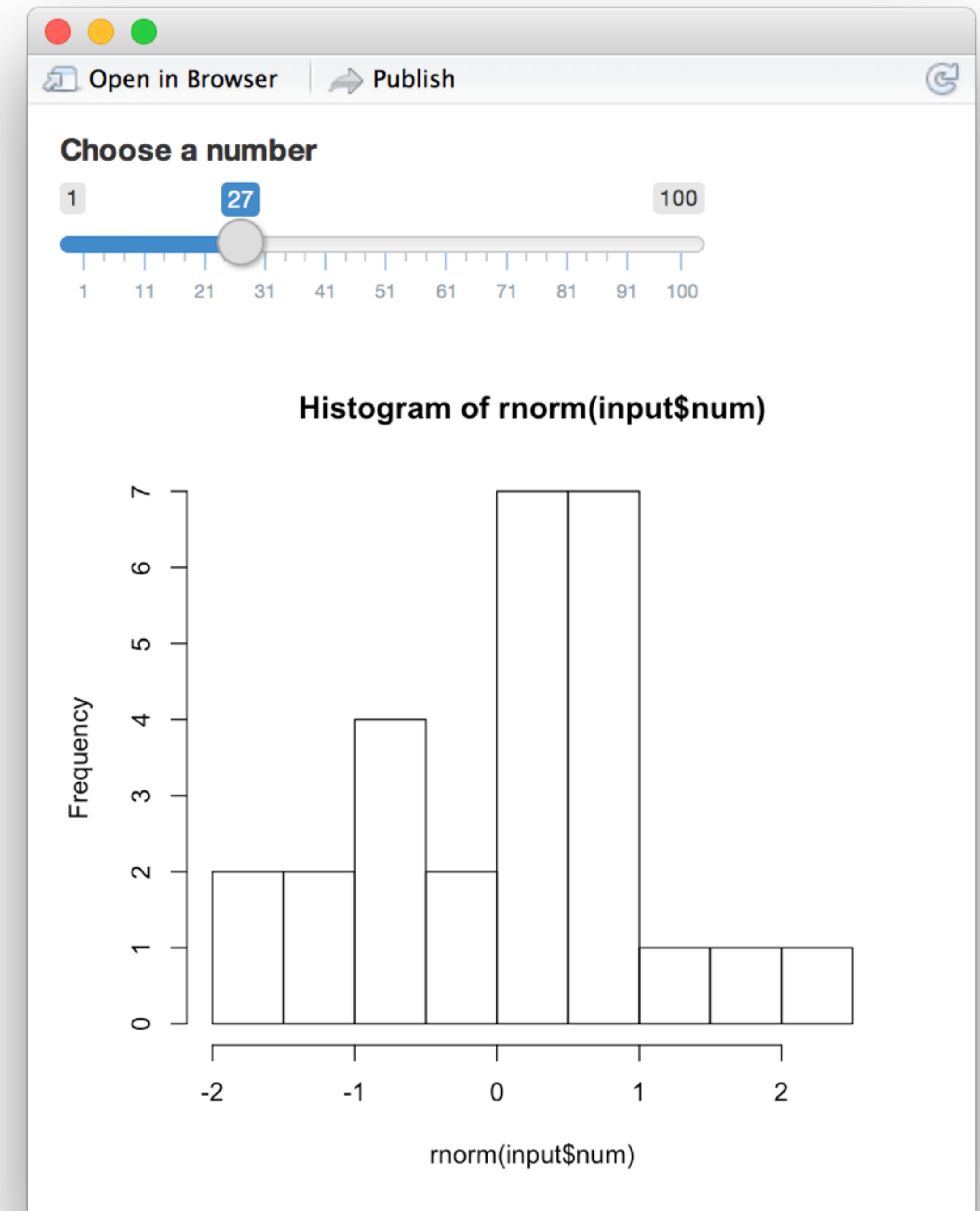
```
sliderInput(inputId = "num",
  label = "Choose a number",
  value = 25, min = 1, max = 100)
```

```
<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-prettyify-separator="," data-keyboard="true"
    data-keyboard-step="1.010101010101"/>
</div>
```

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

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

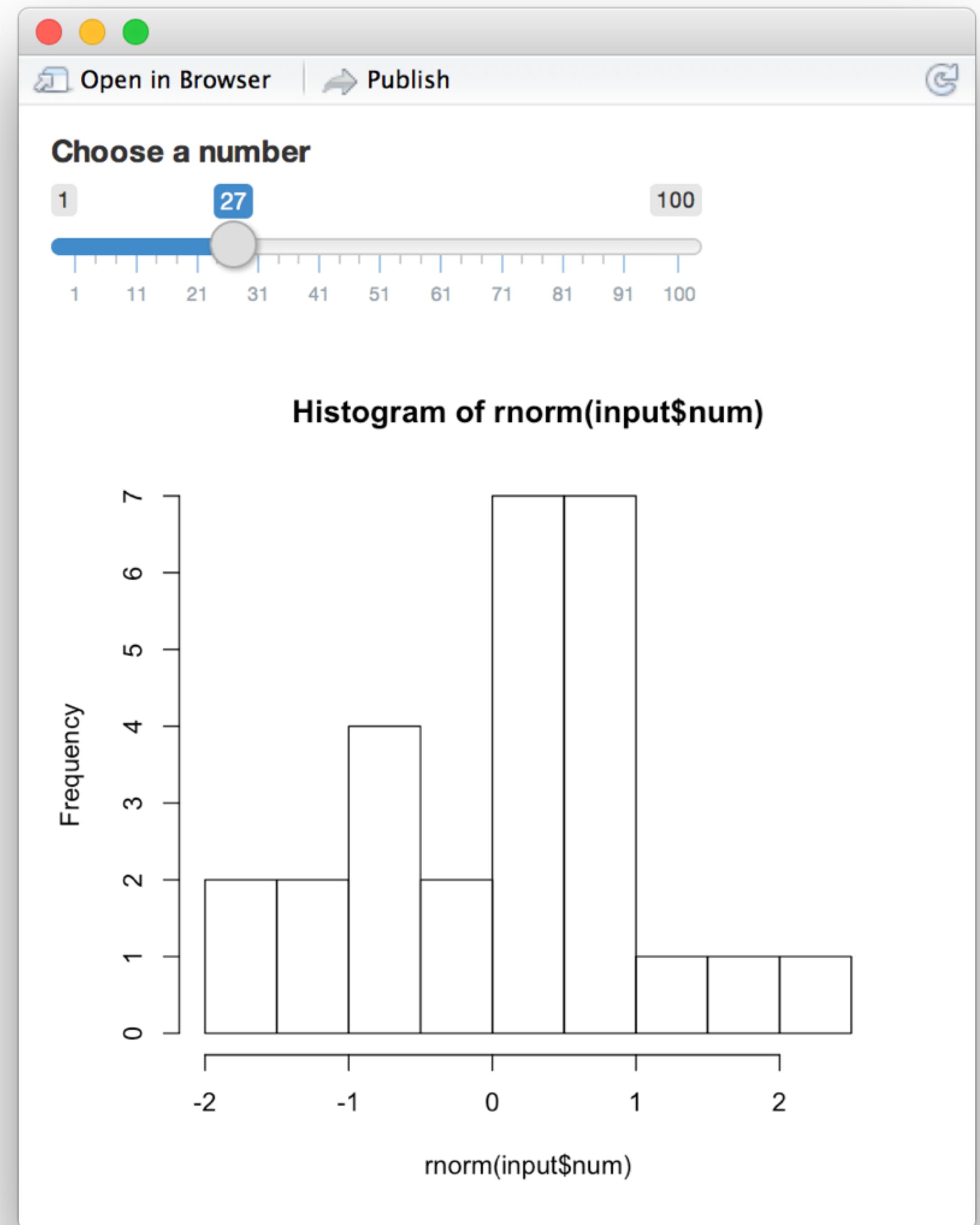
```
ui <- fluidPage(  
  sliderInput(inputId = "num",  
    label = "Choose a number",  
    value = 25, min = 1, max = 100),  
  plotOutput("hist"))
```



```

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-prettyify-
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

Action

Submit

`actionButton()`  
`submitButton()`

## Date range

2014-01-24 to 2014-01-24

`dateRangeInput()`

## Radio buttons

- Choice 1
- Choice 2
- Choice 3

`radioButtons()`

## Single checkbox

Choice A

`checkboxInput()`

## File input

No file chosen

`fileInput()`

## Select box

`selectInput()`

## Checkbox group

- Choice 1
- Choice 2
- Choice 3

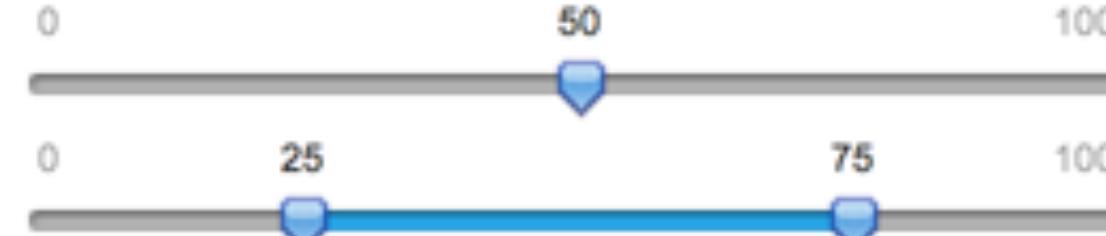
`checkboxGroupInput()`

## Numeric input

1

`numericInput()`

## Sliders



`sliderInput()`

## Date input

2014-01-01

`dateInput()`

## Password Input

.....

`passwordInput()`

## Text input

Enter text...

`textInput()`

# Output functions

Search: <input type="text"/>											
Show 10 entries											
mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	
21	6	160	110	3.9	2.62	16.46	0	1	4	4	
21	6	160	110	3.9	2.875	17.02	0	1	4	4	
22.8	4	108	93	3.85	2.32	18.61	1	1	4	1	
21.4	6	258	110	3.08	3.215	19.44	1	0	3	1	
18.7	8	360	175	3.15	3.44	17.02	0	0	3	2	
18.1	6	225	105	2.76	3.46	20.22	1	0	3	1	
14.3	8	360	245	3.21	3.57	15.84	0	0	3	4	
24.4	4	146.7	62	3.69	3.19	20	1	0	4	2	
22.8	4	140.8	95	3.92	3.15	22.9	1	0	4	2	
19.2	6	167.6	123	3.92	3.44	18.3	1	0	4	4	
mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb	

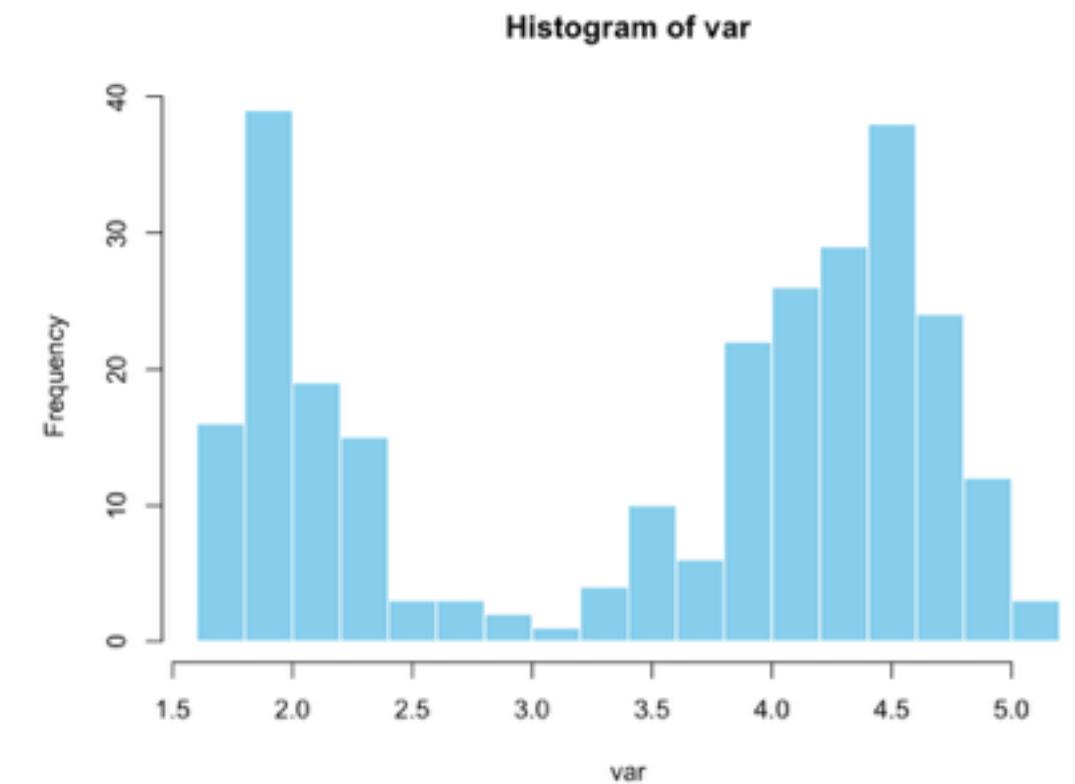
dataTableOutput()



htmlOutput()



imageOutput()



plotOutput()

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

tableOutput()

foo

Choose a dataset:

Number of observations to view:

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

textOutput()

uiOutput()

verbatimTextOutput()

# Panel functions

## **absolutePanel()**

Panel position set rigidly (absolutely), not fluidly

## **headerPanel()**

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

## **navlistPanel()**

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

## **tabsetPanel()**

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

## **conditionalPanel()** **fixedPanel()**

A JavaScript expression determines whether panel is visible or not.

## **inputPanel()**

Panel with grey background, suitable for grouping inputs

## **sidebarPanel()**

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

## **titlePanel()**

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

## **wellPanel()**

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

## **mainPanel()**

Panel for displaying output, used with pageWithSidebar()

## **tabPanel()**

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

## **wellPanel()**

Panel with grey background.

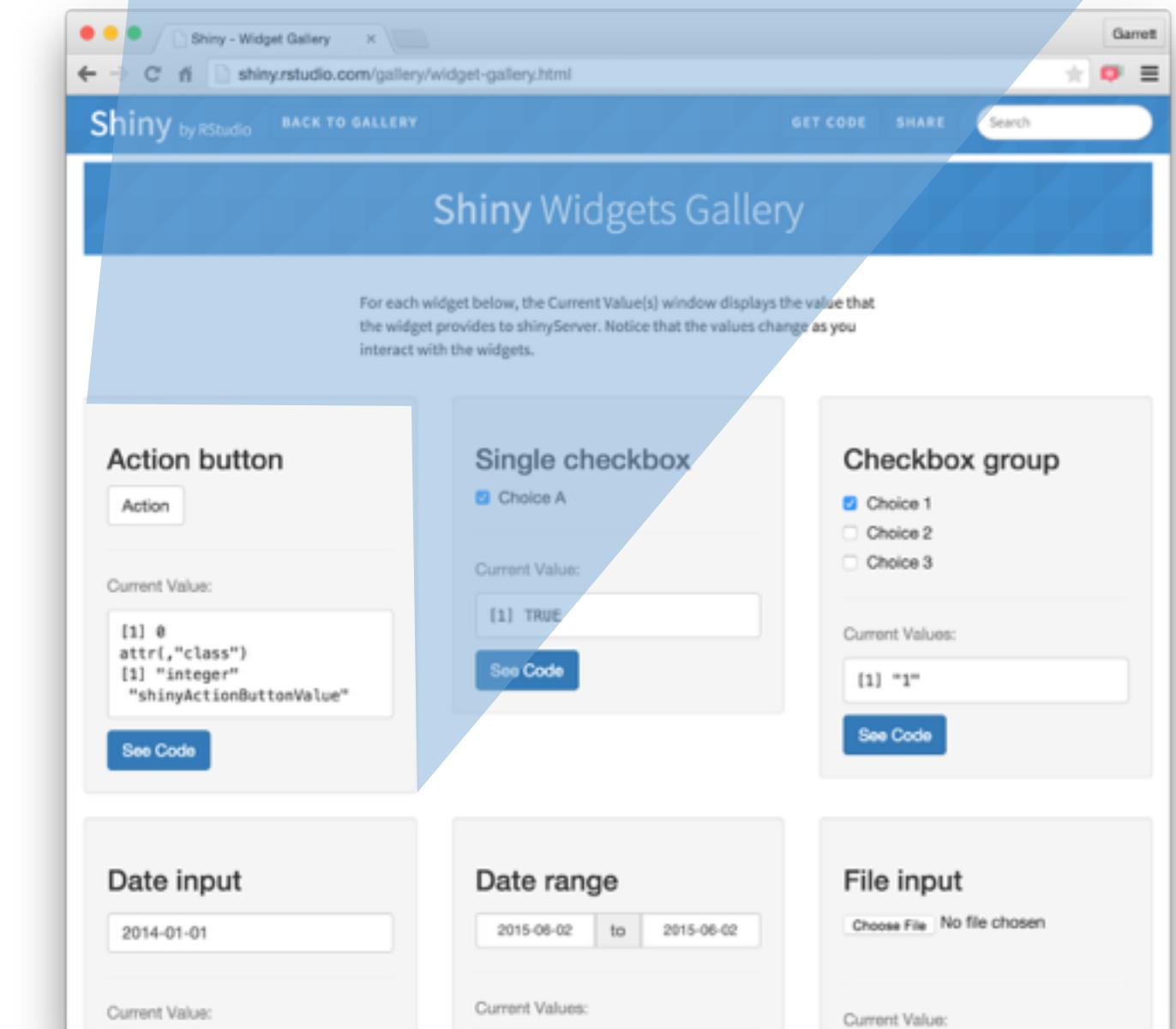
### Action button

Action

Current Value:

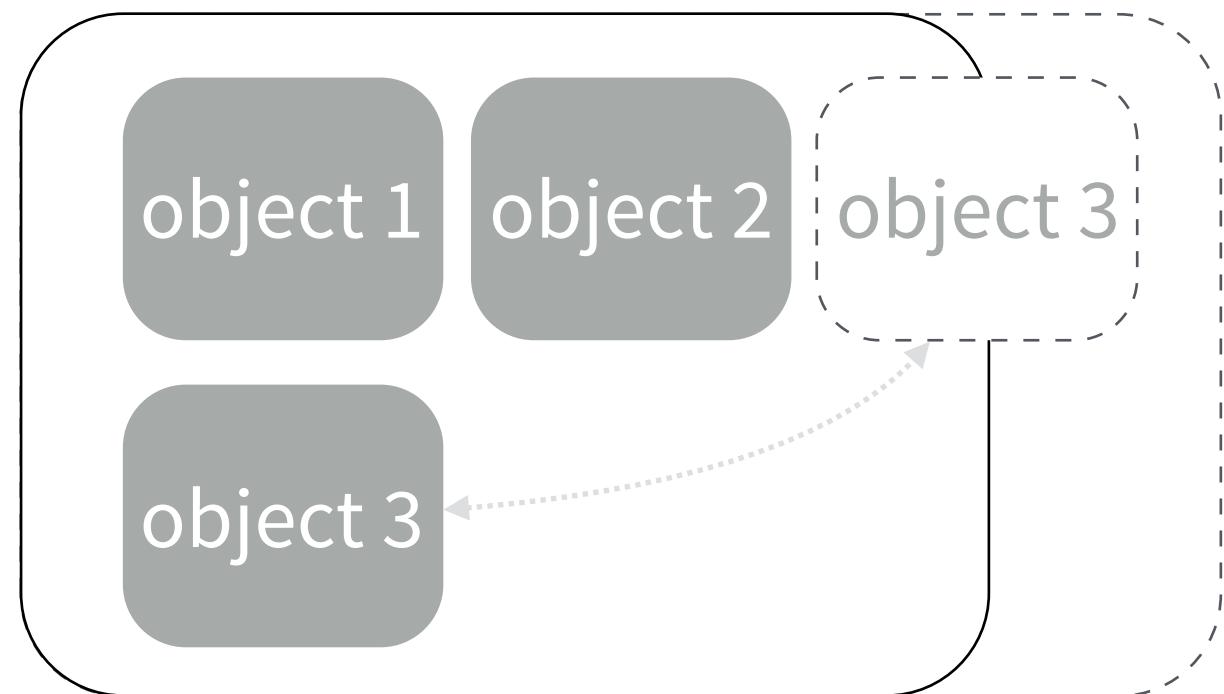
```
[1] 0  
attr("class")  
[1] "integer"  
"shinyActionButtonValue"
```

[See Code](#)

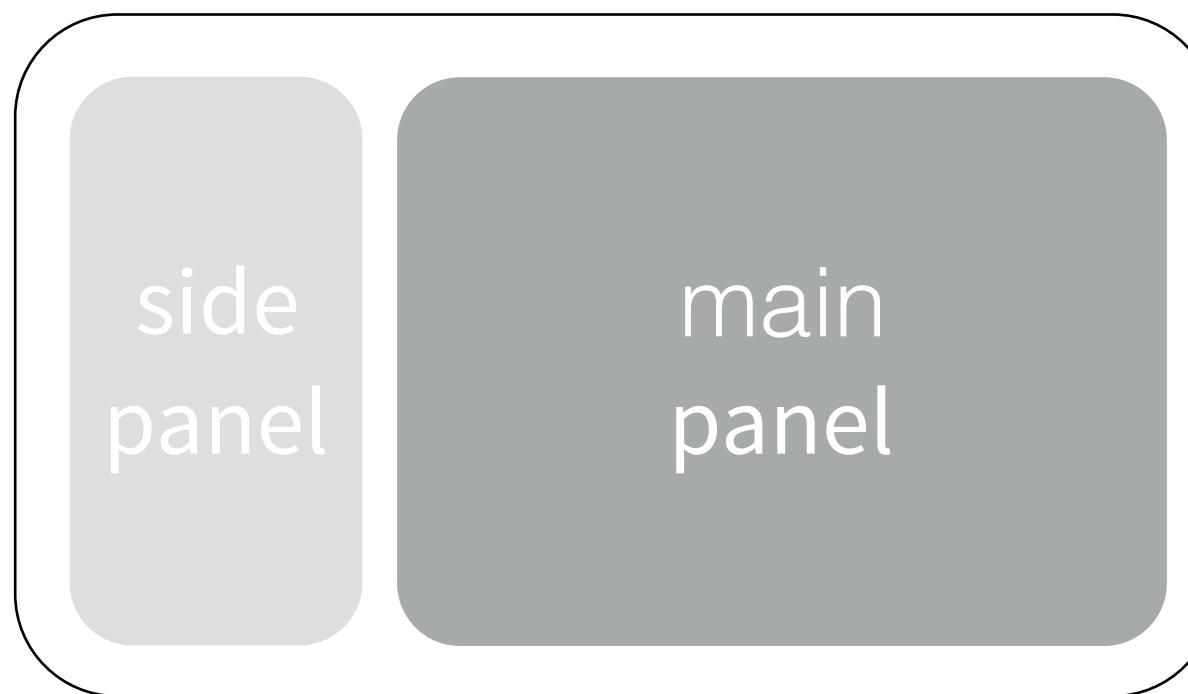


# Layout functions

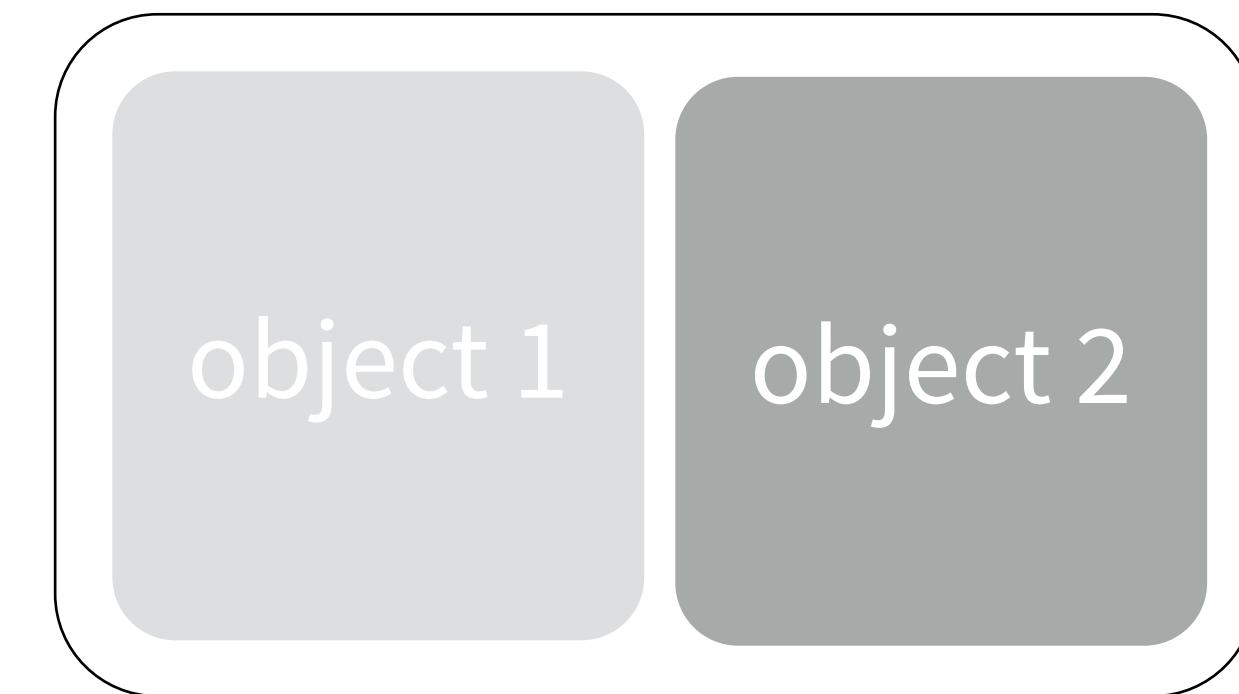
**flowLayout()**



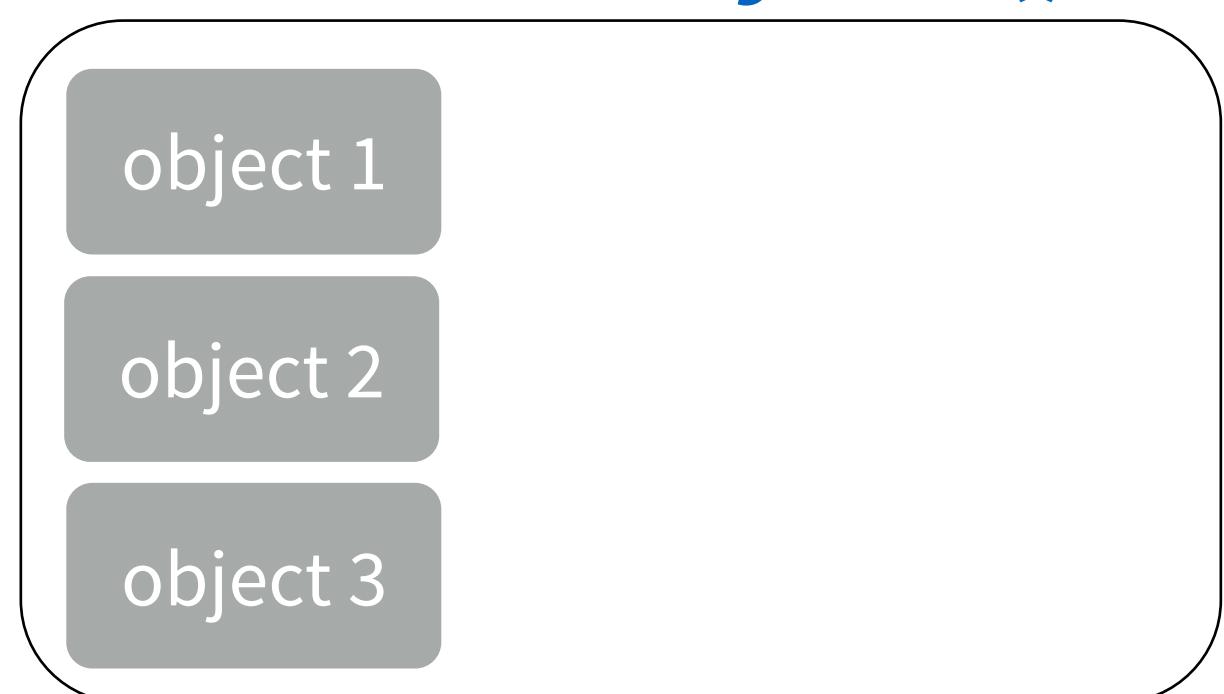
**sidebarLayout()**



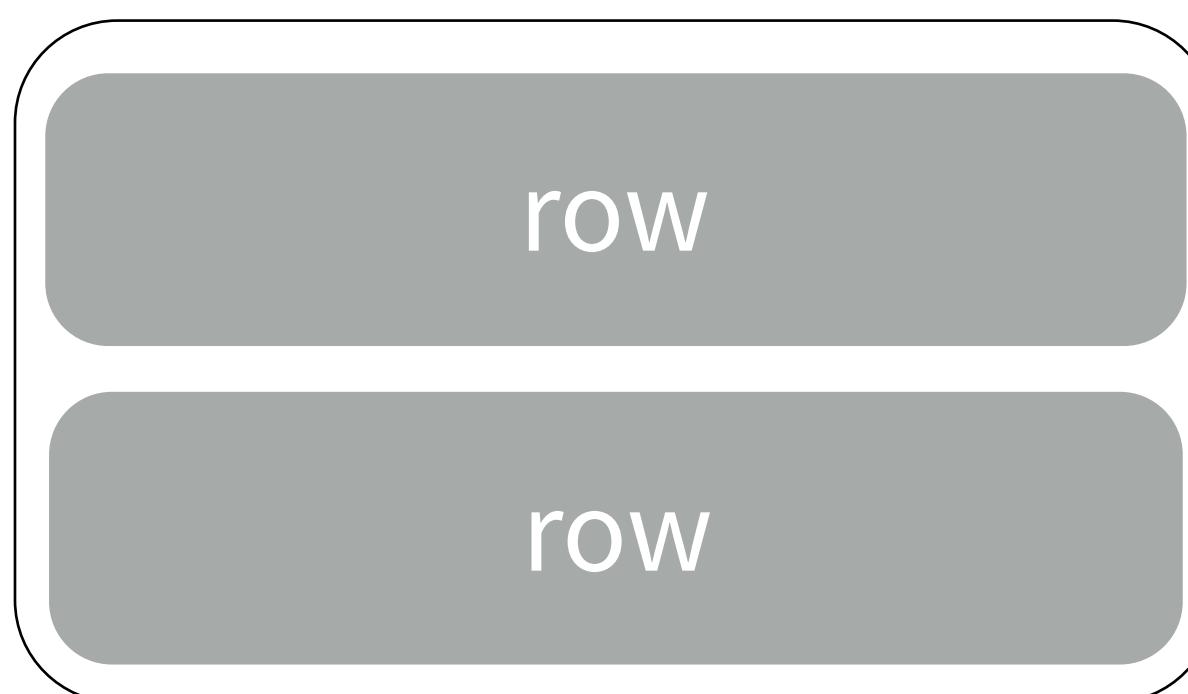
**splitLayout()**



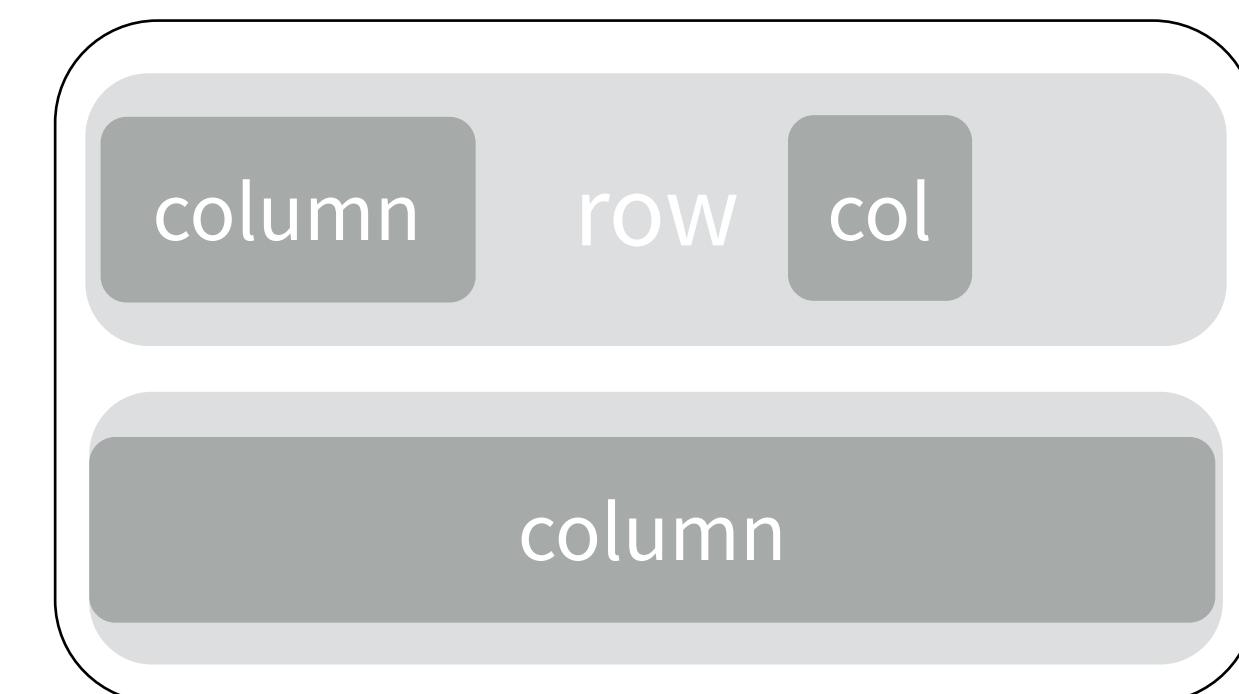
**verticalLayout()**



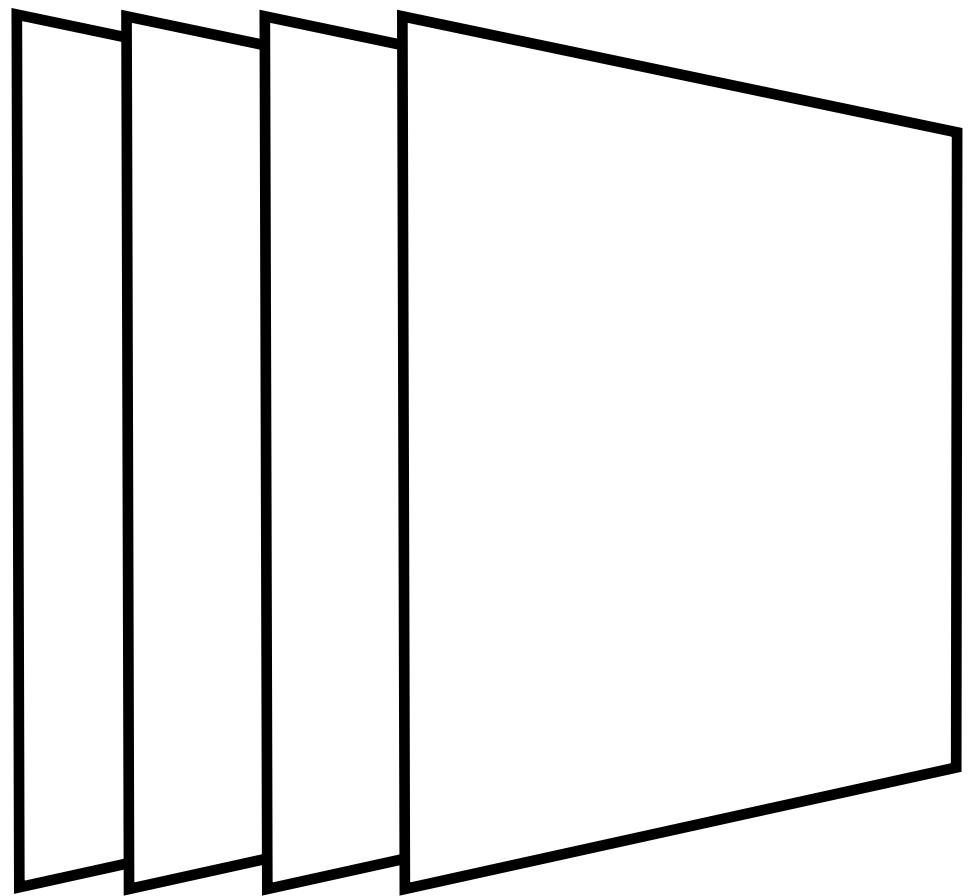
**fluidRow()**



**column()**



# "Layering" functions



`tabPanel()`s

`tabsetPanel()`

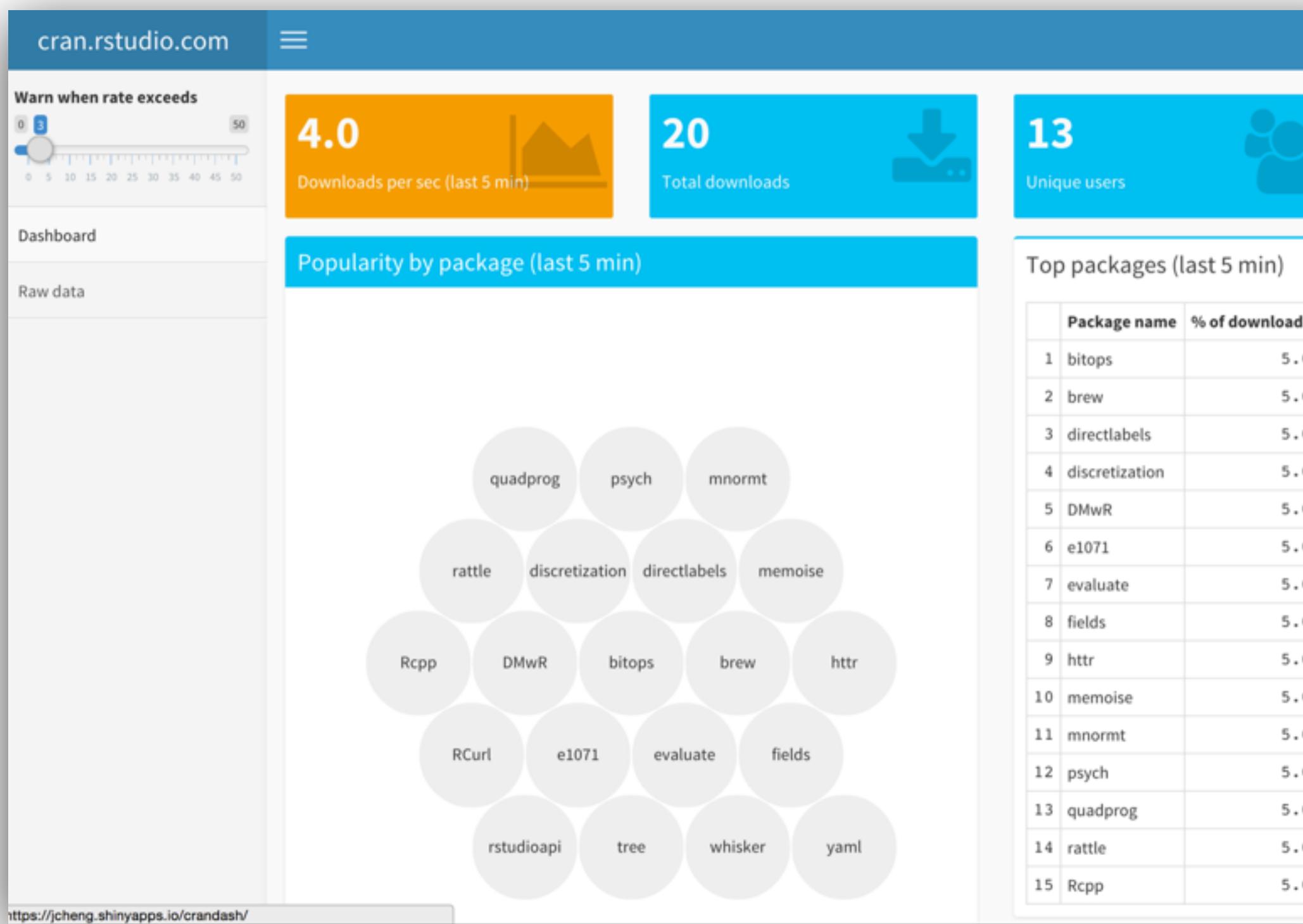
`navlistPanel()`

`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/](http://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)

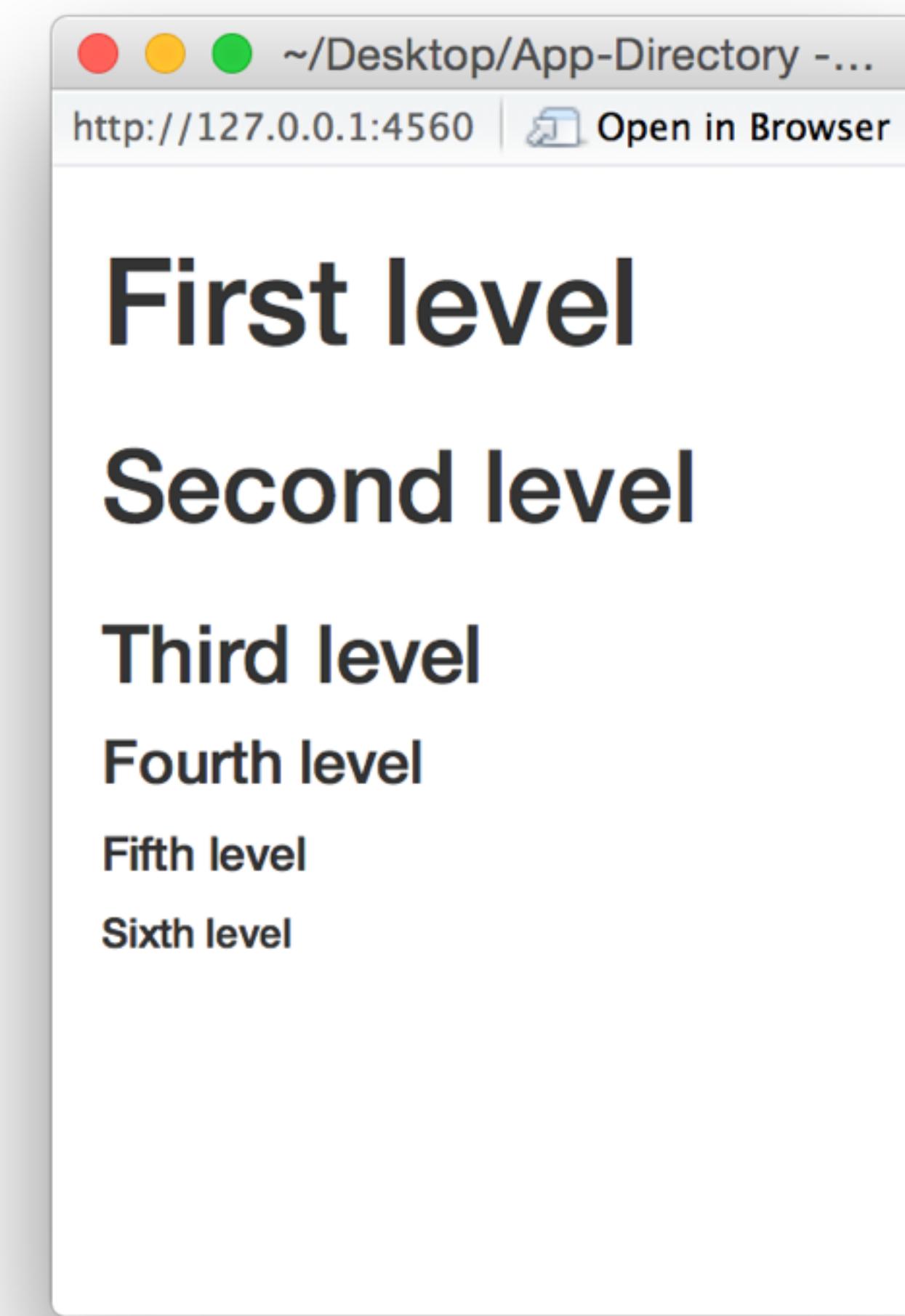
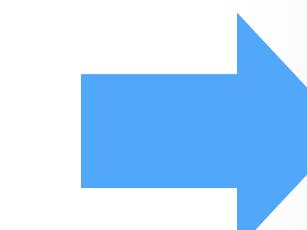
unnamed arguments  
appear inside the tags  
(call tags\$...() to create nested tags)

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

# h1() - h6()

## Headers

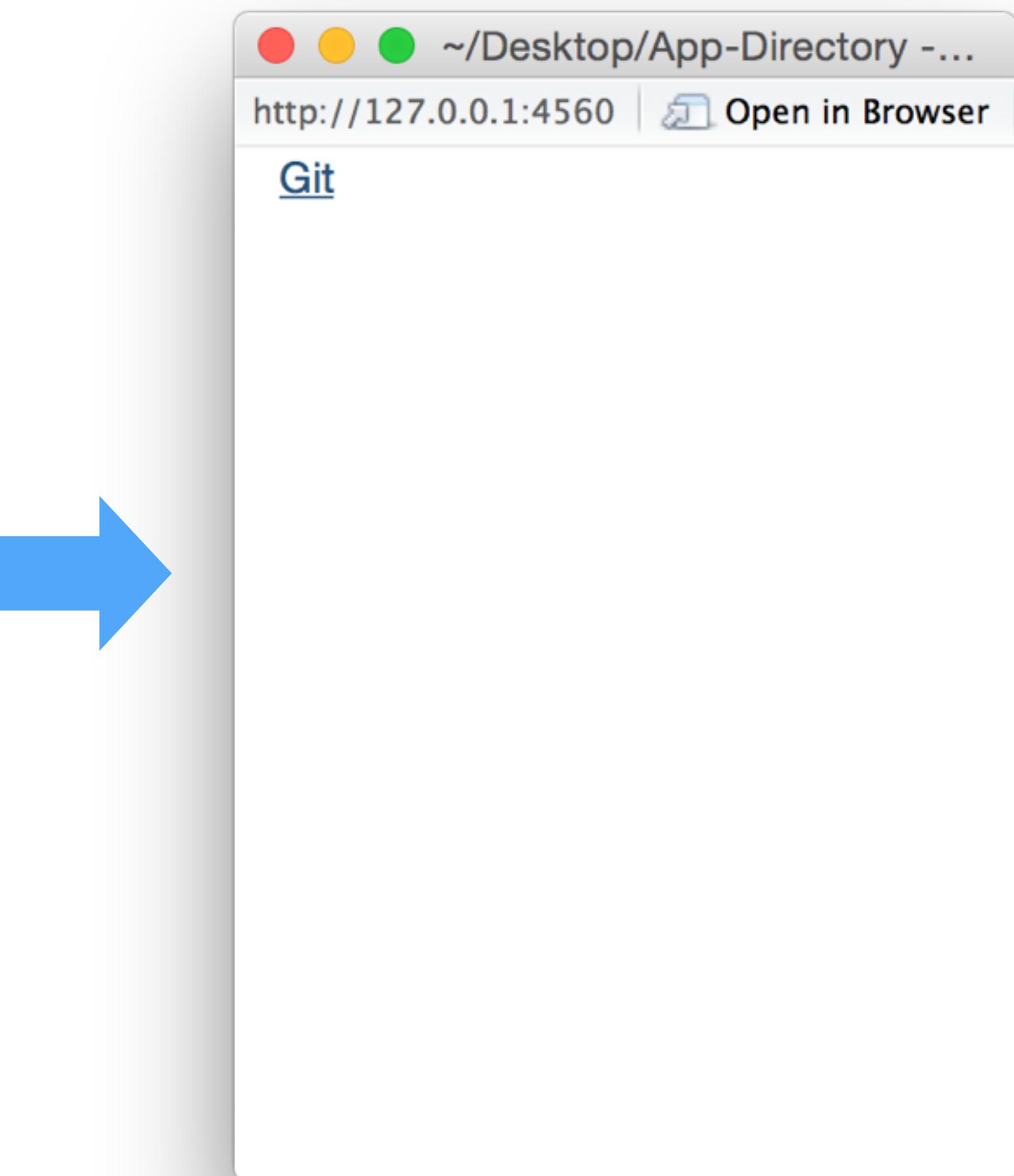
```
fluidPage(  
  tags$h1("First level"),  
  tags$h2("Second level"),  
  tags$h3("Third level"),  
  tags$h4("Fourth level"),  
  tags$h5("Fifth level"),  
  tags$h6("Sixth level"))
```



a()

hyperlink with the href argument

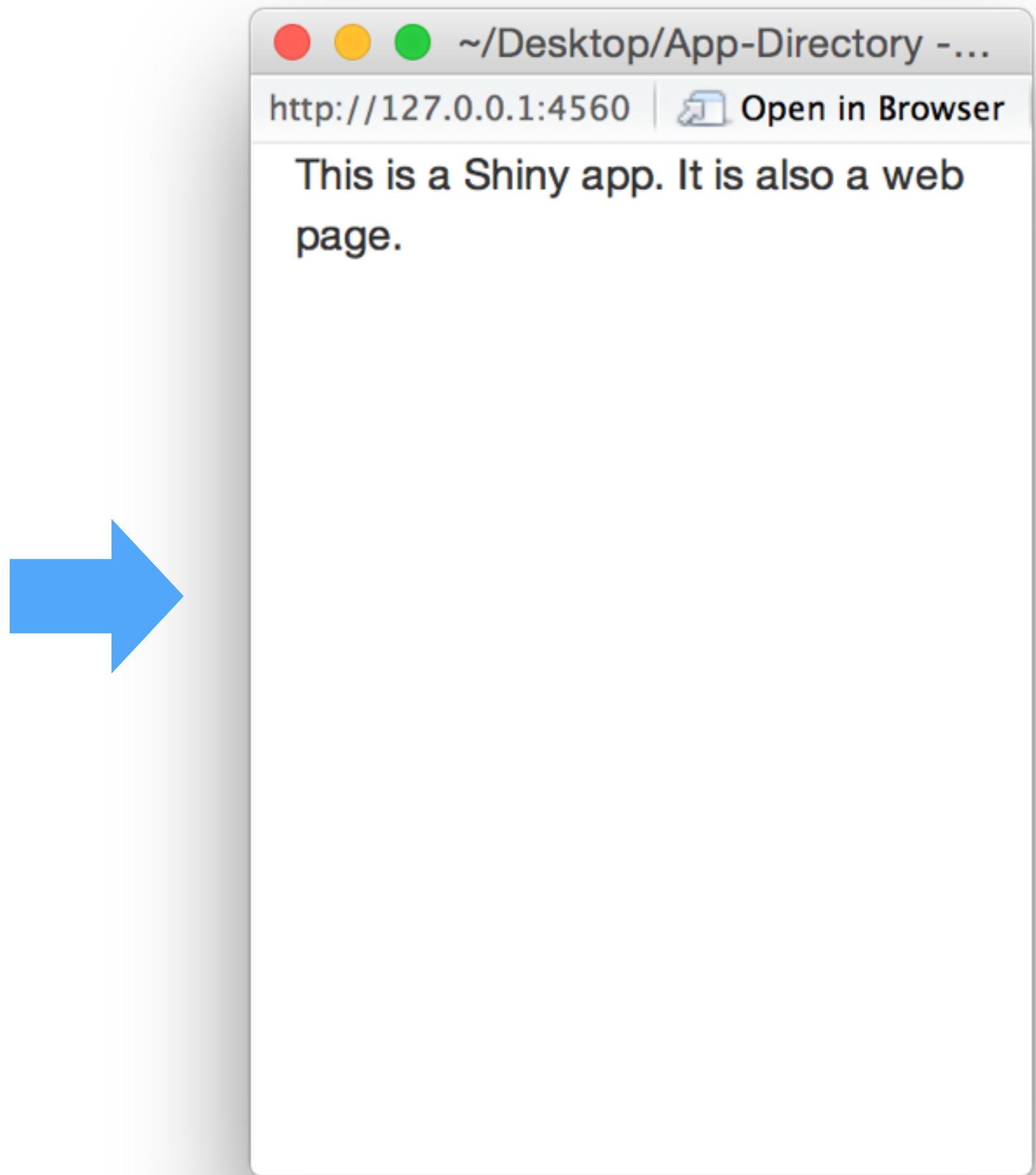
```
fluidPage(  
  tags$a(href= "http://www.git.com",  
         "Git")  
)
```



# text

Character strings do not need a tag.

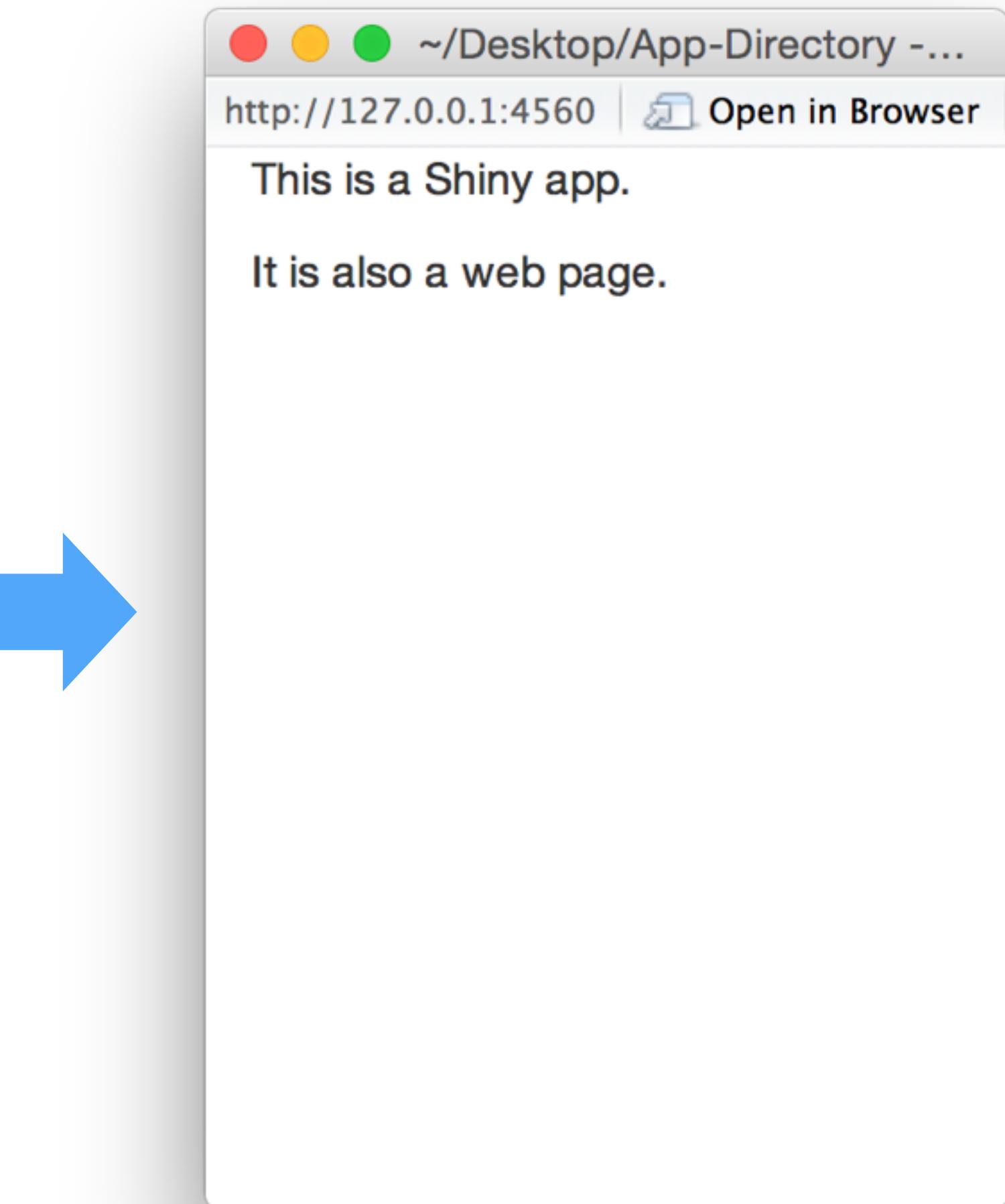
```
fluidPage(  
  "This is a Shiny app.",  
  "It is also a web page."  
)
```



p()

A new paragraph

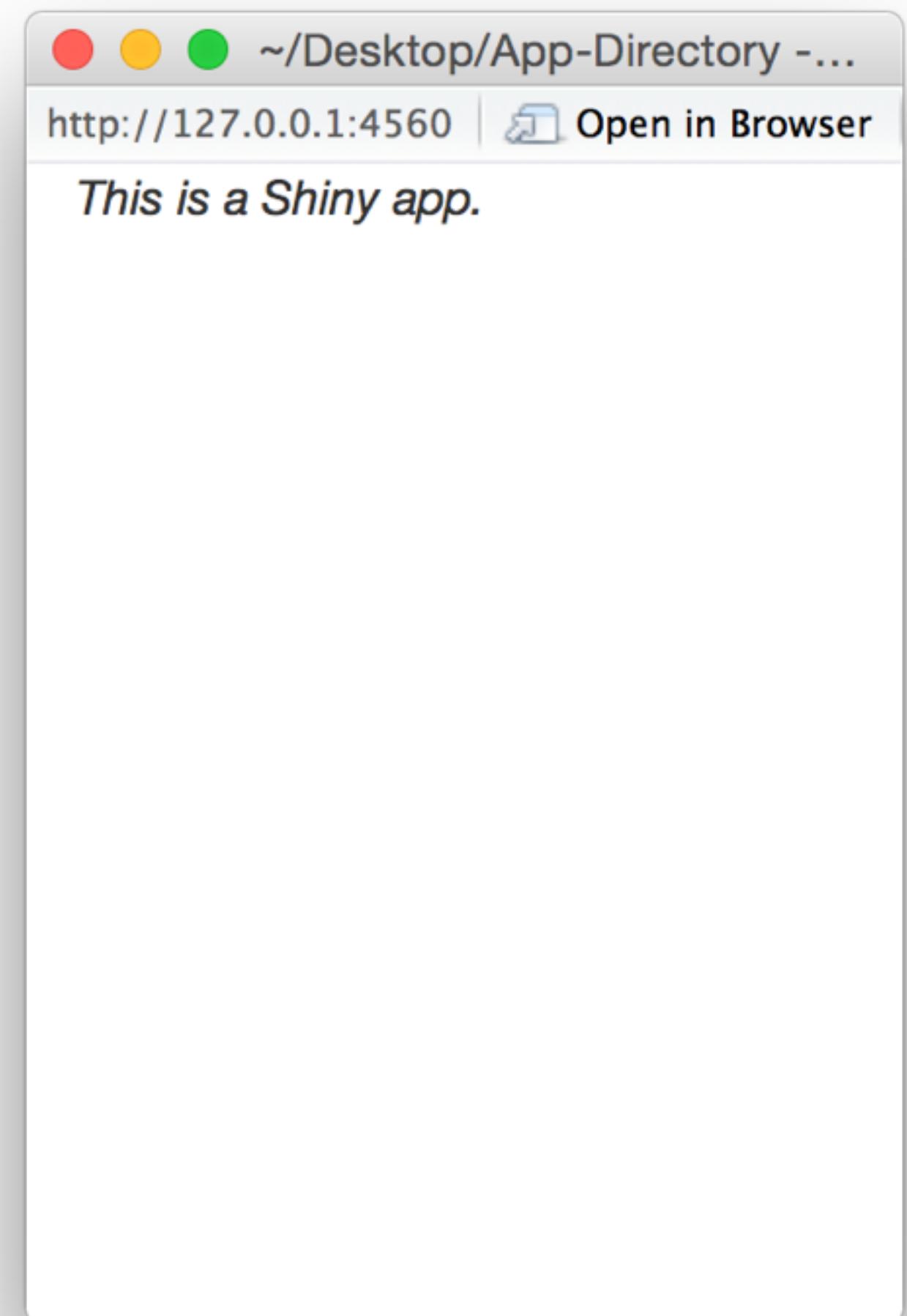
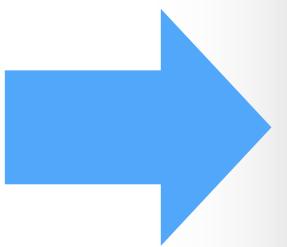
```
fluidPage(  
  tags$p("This is a Shiny app."),  
  tags$p("It is also a web page.")  
)
```



# em()

Emphasized (*italic*) text

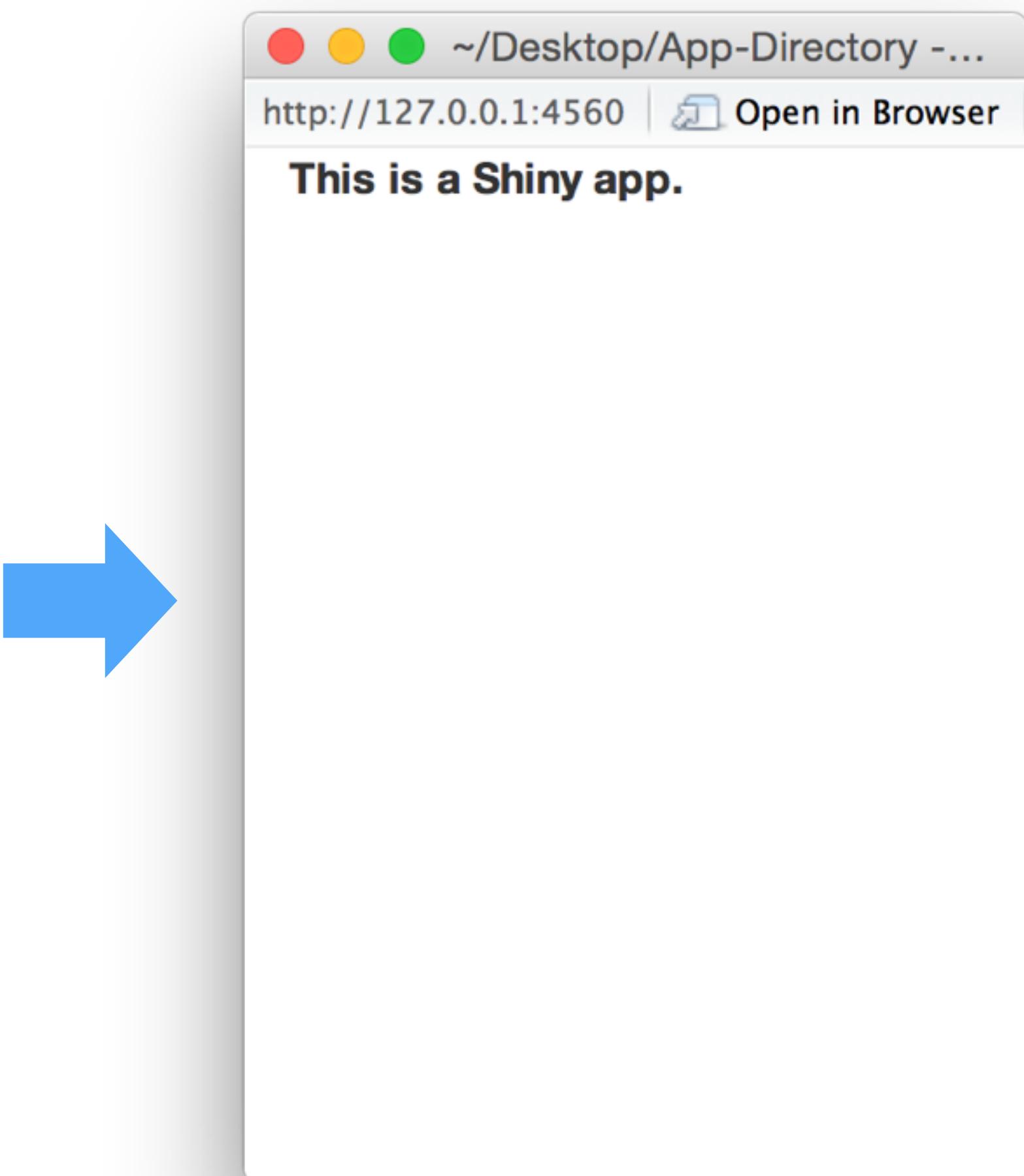
```
fluidPage(  
  tags$em("This is a Shiny app."  
)
```



# strong()

Strong (**bold**) text

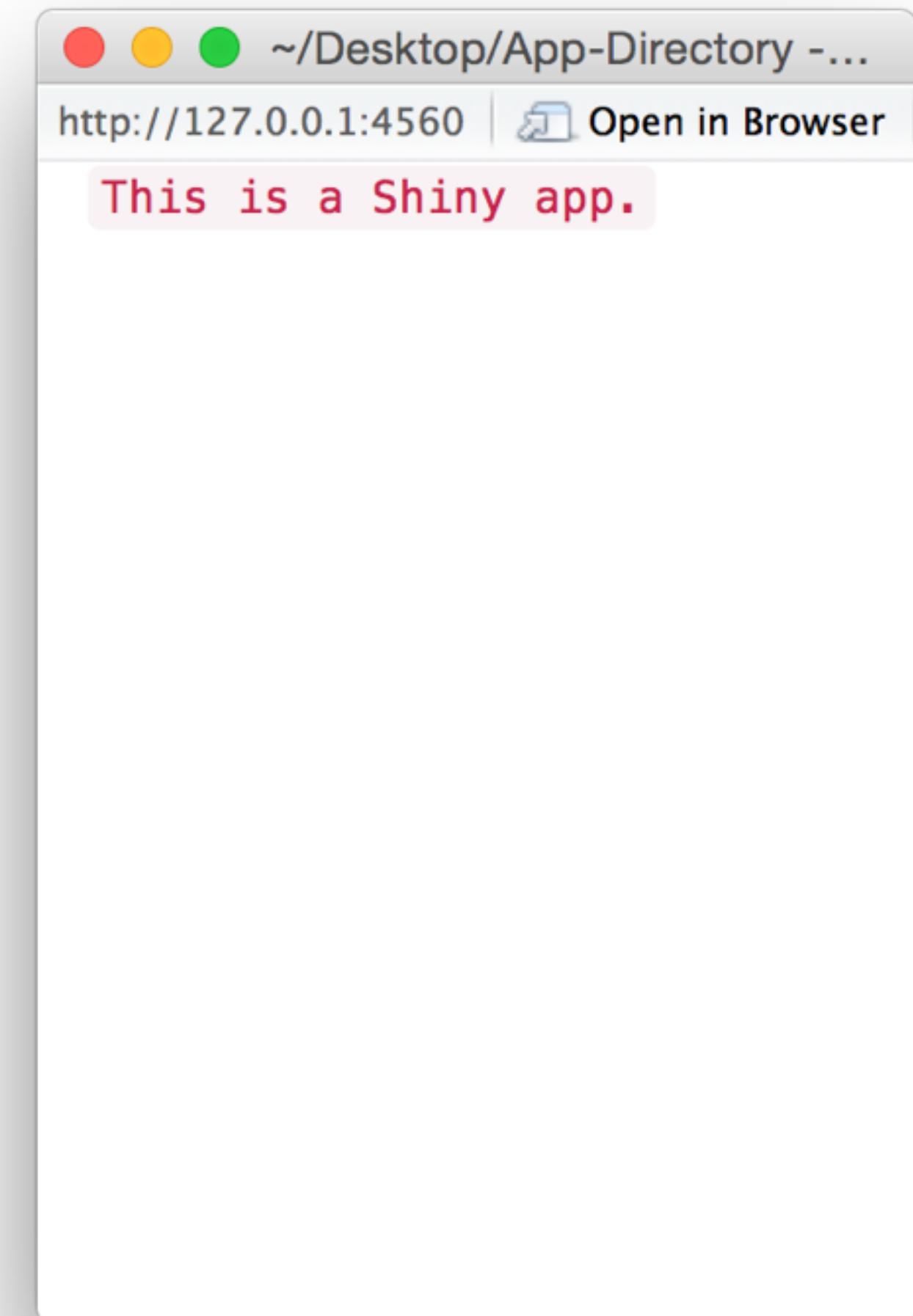
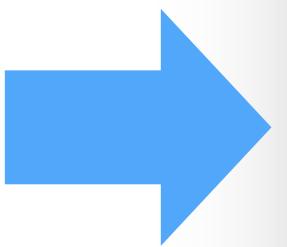
```
fluidPage(  
  tags$strong("This is a Shiny app."  
)
```



# code()

Monospaced text (code)

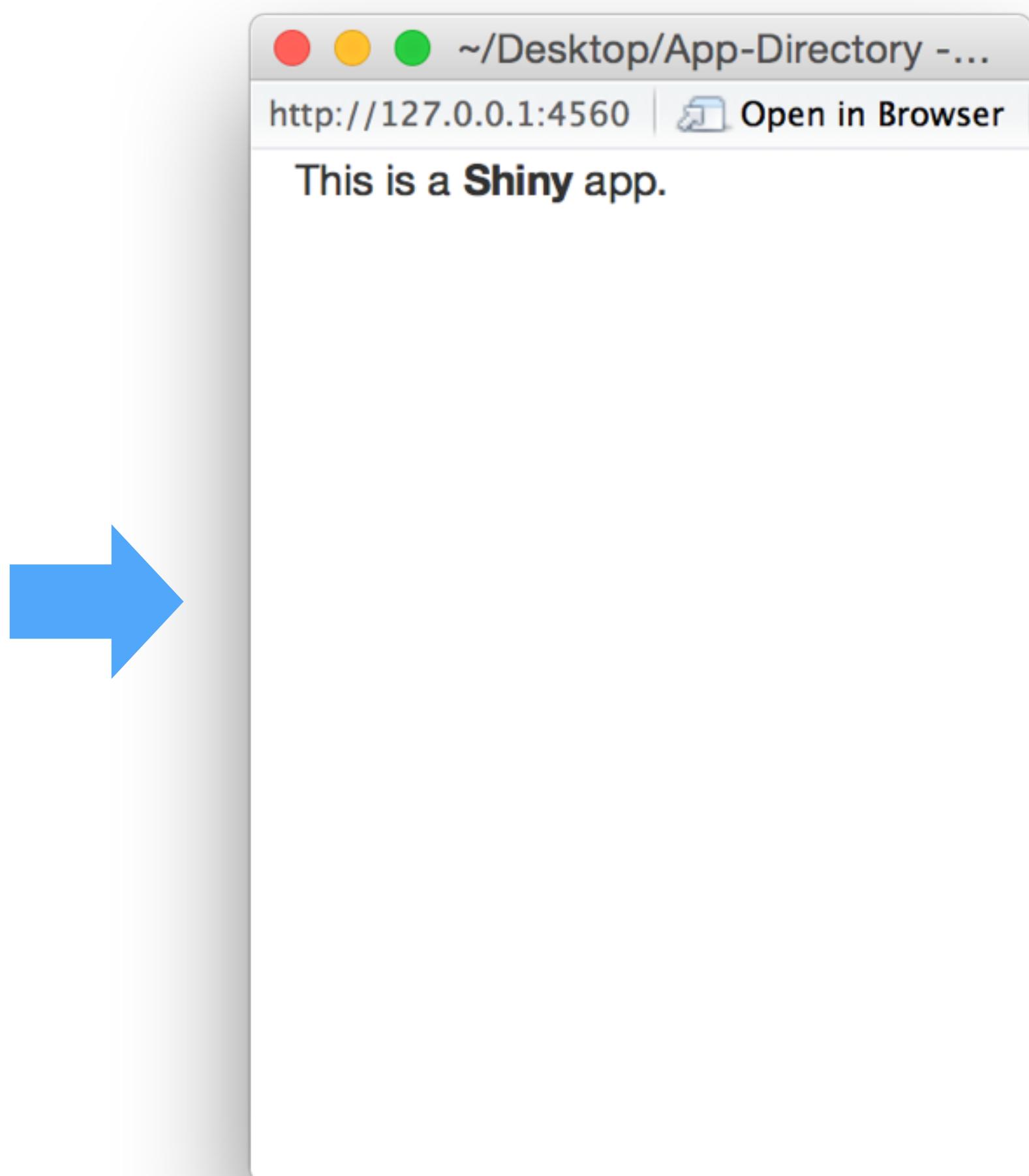
```
fluidPage(  
  tags$code("This is a Shiny app.")  
)
```



# nesting

You can also nest functions inside of others

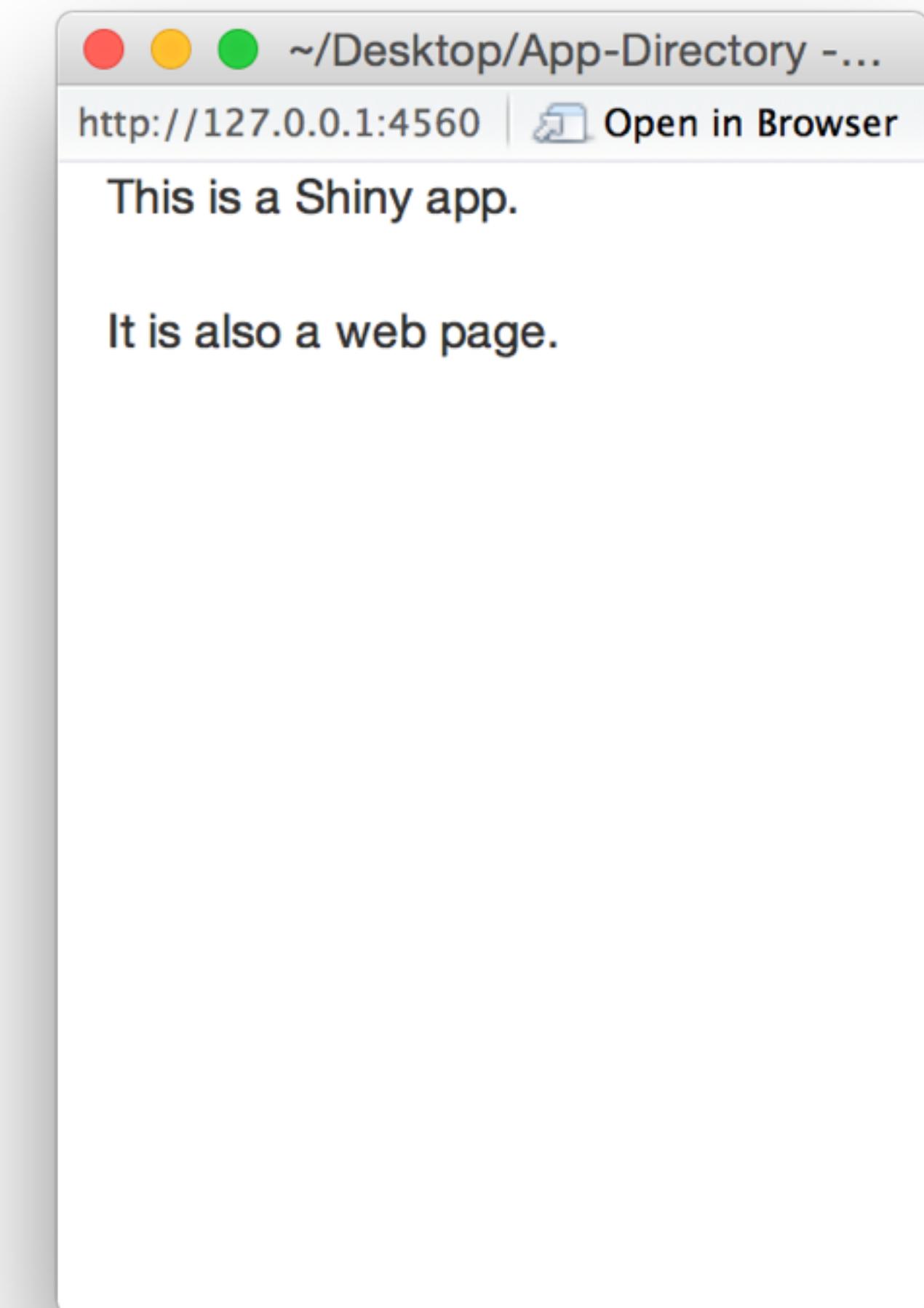
```
fluidPage(  
  tags$p("This is a",  
         tags$strong("Shiny"),  
         "app.")  
)
```



# br()

## A line break

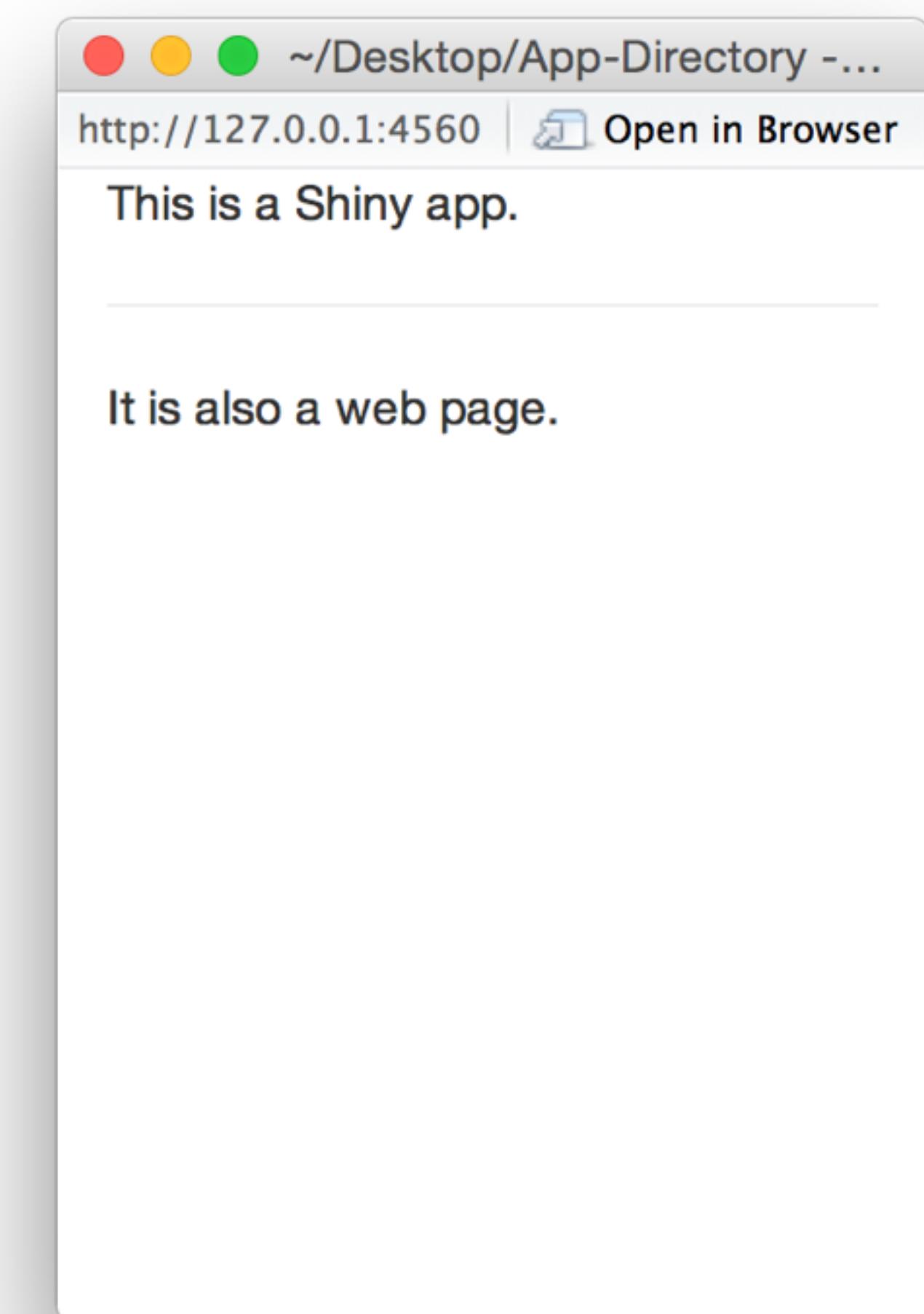
```
fluidPage(  
  "This is a Shiny app.",  
  tags$br(),  
  "It is also a web page."  
)
```



# hr()

## A horizontal rule

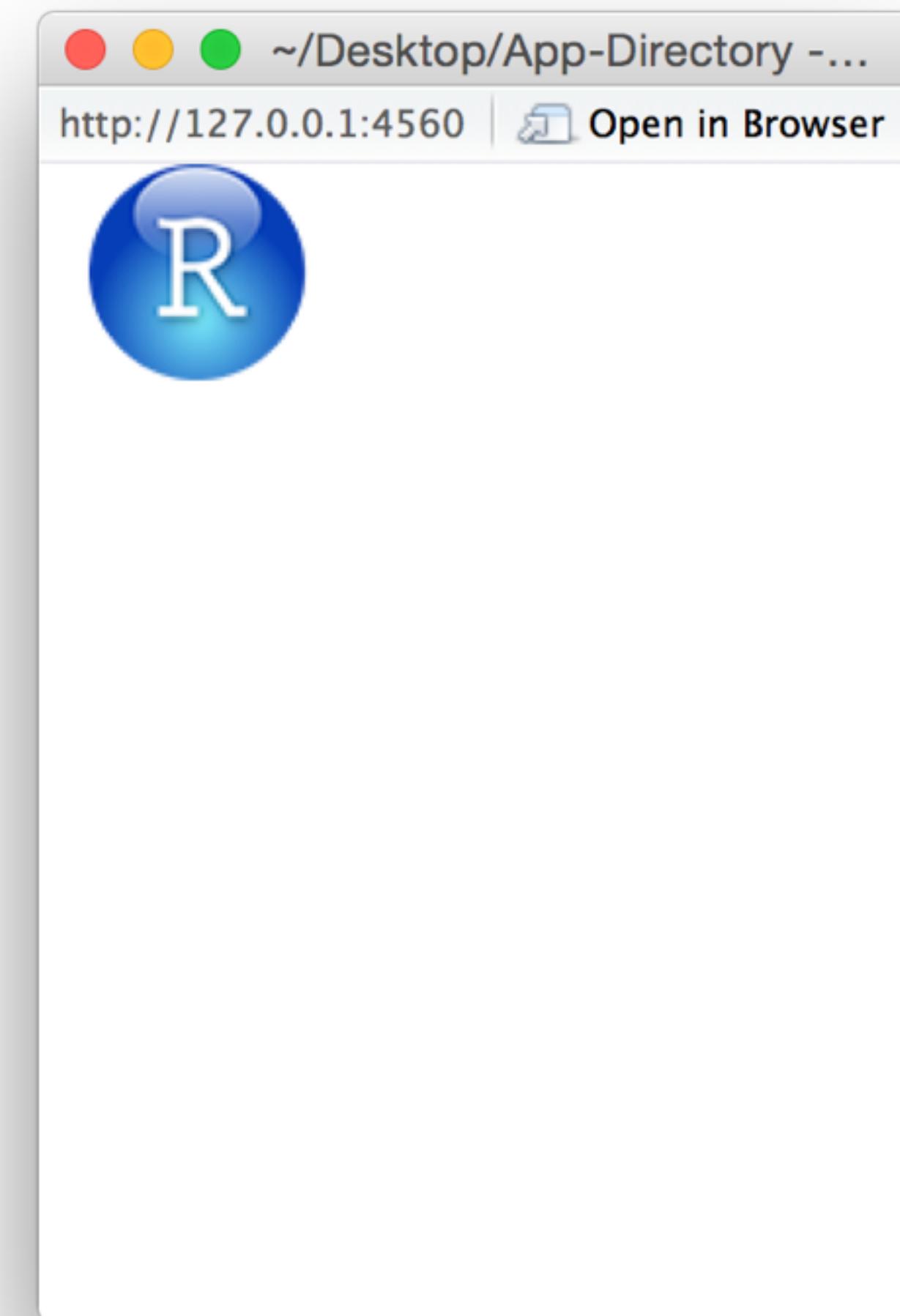
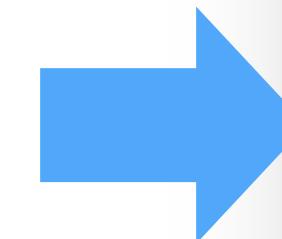
```
fluidPage(  
  "This is a Shiny app.",  
  tags$hr(),  
  "It is also a web page."  
)
```



# img()

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

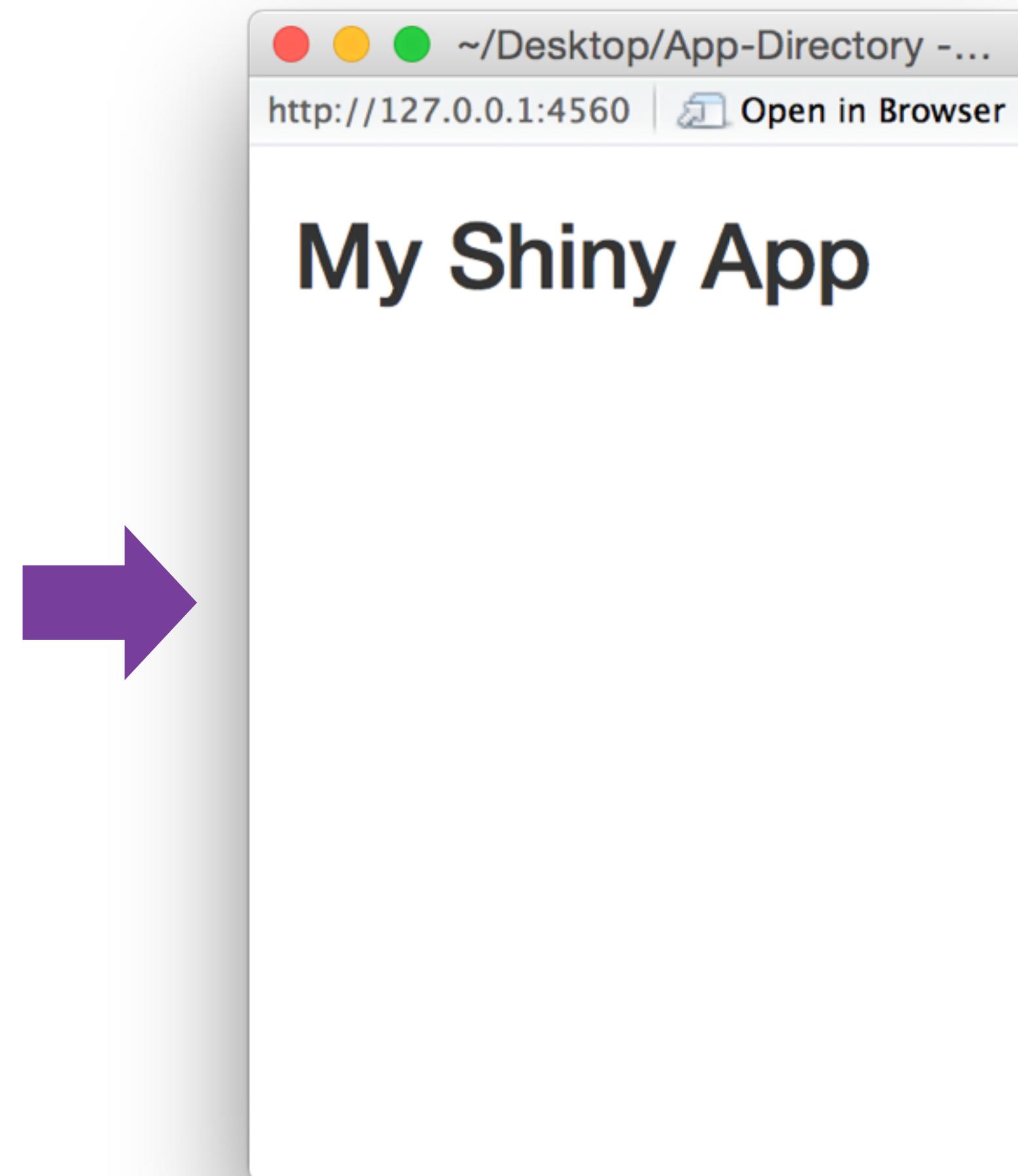
```
fluidPage(  
  tags$img(height = 100,  
            width = 100,  
            src = "http://www.rstudio.com/  
                    images/RStudio.2x.png")  
)
```



# Raw HTML

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

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



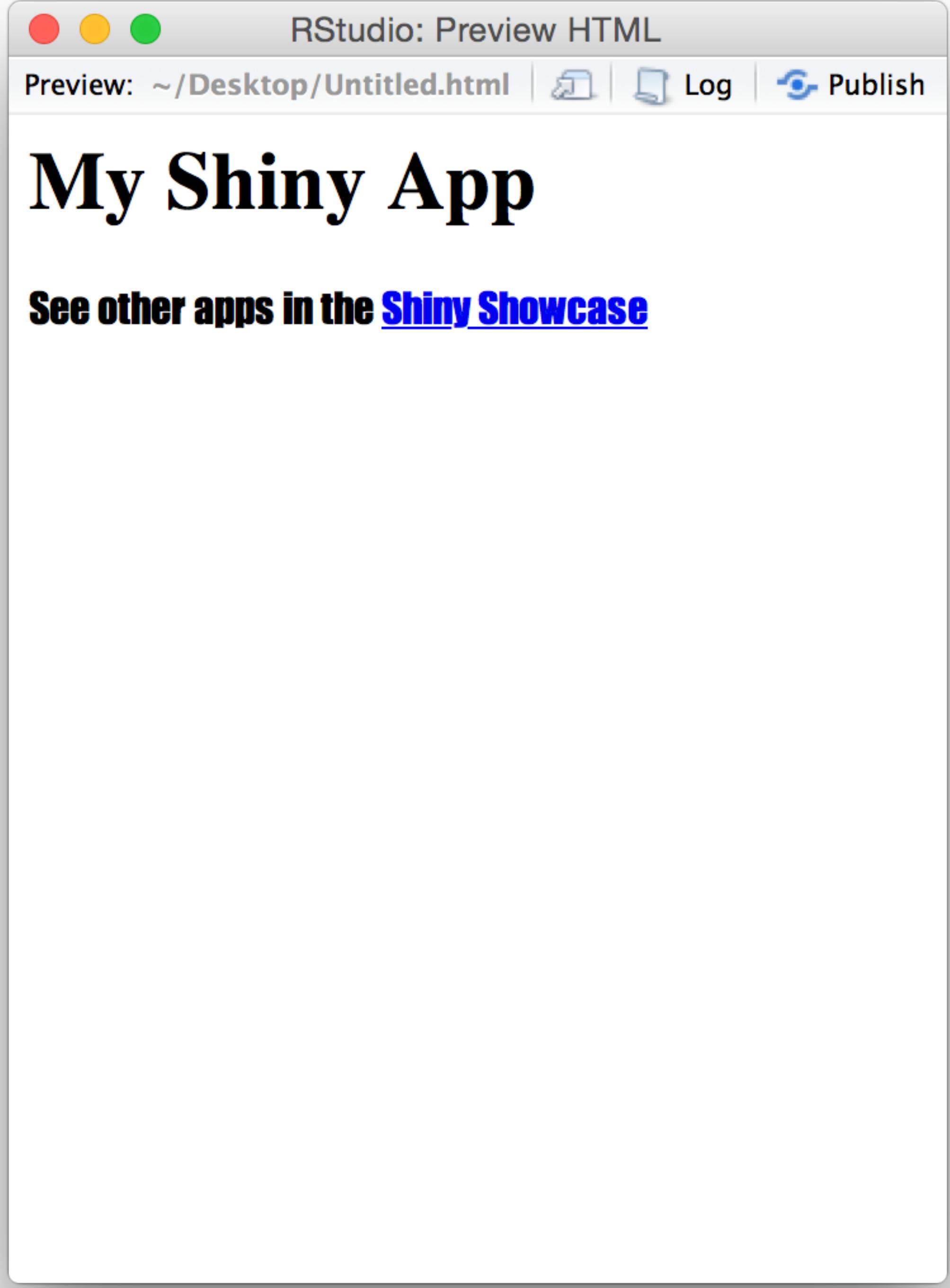
```
<div class="container-fluid">  
  <h1>My Shiny App</h1>  
  <p style="font-family:Impact">  
    See other apps in the  
    <a href="http://www.rstudio.com/  
      products/shiny/shiny-user-  
      showcase/">Shiny Showcase</a>  
  </p>  
</div>
```

RStudio: Preview HTML

Preview: ~/Desktop/Untitled.html | Log | Publish

# My Shiny App

**See other apps in the [Shiny Showcase](#)**



The screenshot shows the RStudio Preview HTML interface. The title bar reads "RStudio: Preview HTML". Below it, the preview path is listed as "Preview: ~/Desktop/Untitled.html" with icons for Log and Publish. The main content area displays the rendered HTML output. It features a large, bold, dark blue header "My Shiny App". Below the header is a paragraph in black font: "See other apps in the [Shiny Showcase](#)". The word "Shiny Showcase" is highlighted in blue, indicating it is a link.

```
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>
```

```
)
```

RStudio: Preview HTML

Preview: ~/Desktop/Untitled.html | Log | Publish

# My Shiny App

See other apps in the [Shiny Showcase](http://www.rstudio.com/products/shiny/shiny-user-showcase/)

```
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>  
)
```

RStudio: Preview HTML

Preview: ~/Desktop/Untitled.html | Log | Publish

# My Shiny App

See other apps in the [Shiny Showcase](#)

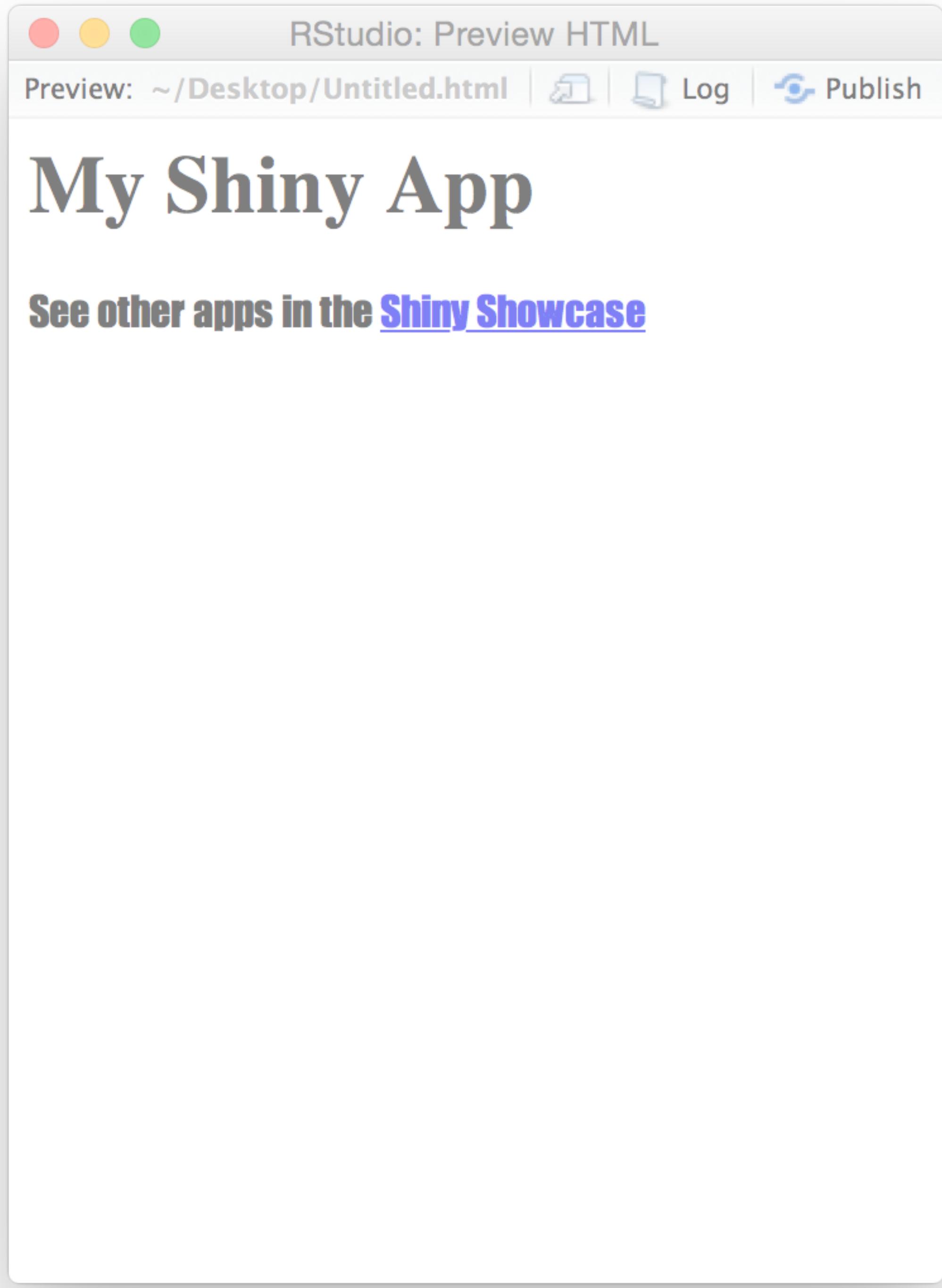
```
fluidPage(  
  h1("My Shiny App"),  
  p(style = "font-family:Impact",  
    "See other apps in the",  
    <a href="http://www.rstudio.com/  
      products/shiny/shiny-user-  
      showcase/">Shiny Showcase</a>  
)  
)
```

RStudio: Preview HTML

Preview: ~/Desktop/Untitled.html | Log | Publish

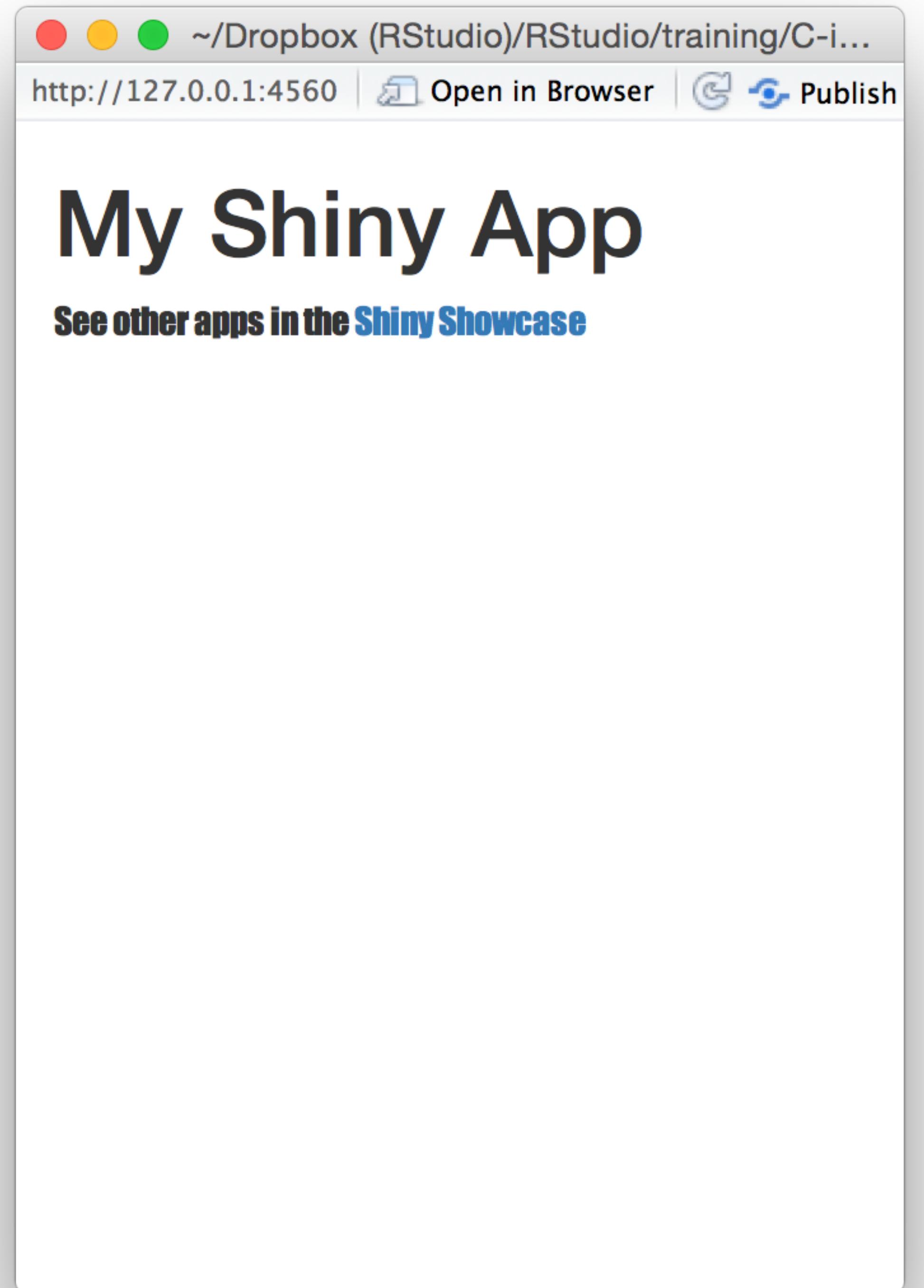
# My Shiny App

See other apps in the [Shiny Showcase](#)

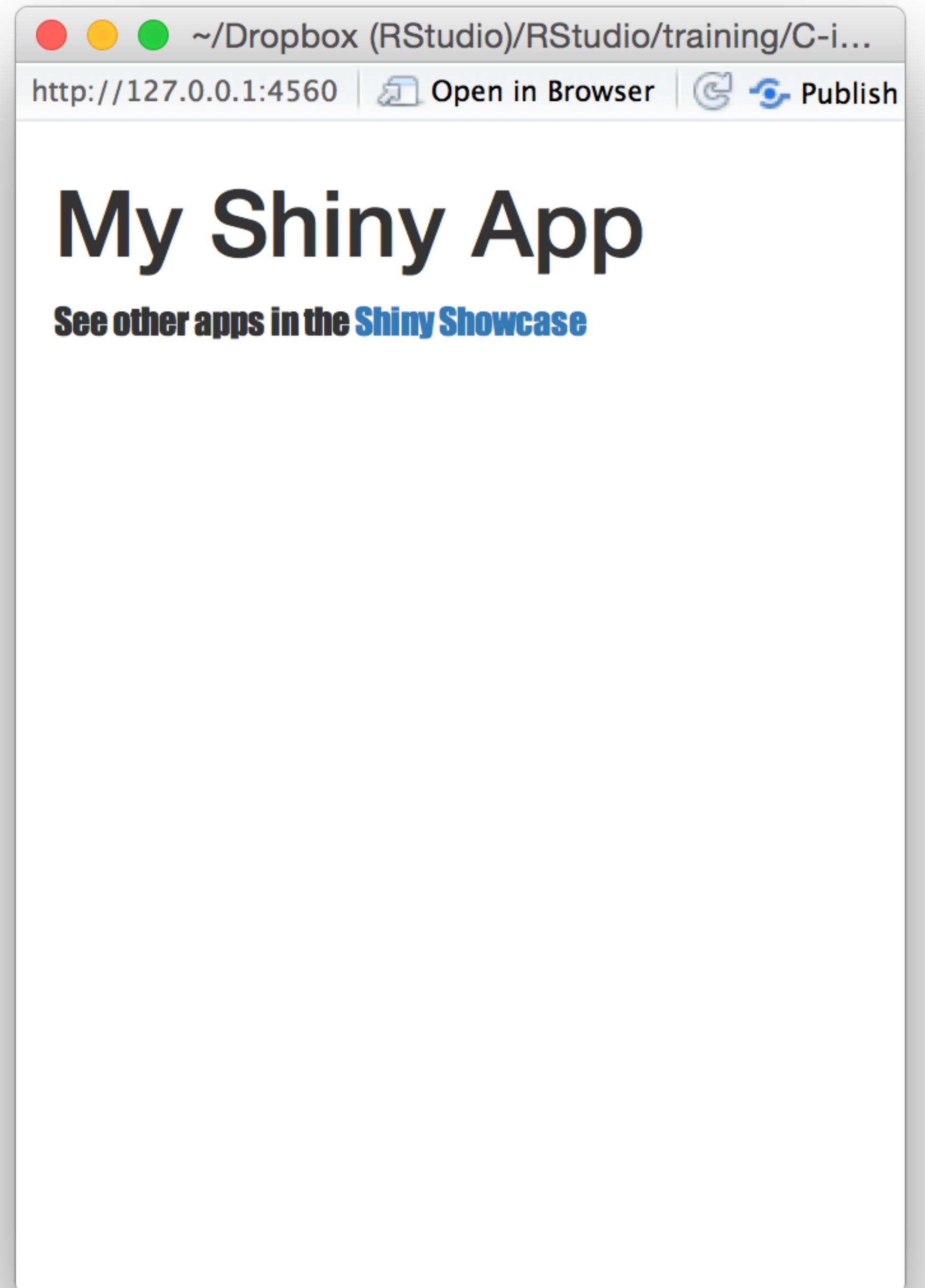


© CC 2016 RStudio, Inc.

```
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>
```

```
)
```

The screenshot shows a Shiny application window in RStudio. The title bar indicates the file path is ~/Dropbox (RStudio)/RStudio/training/C-i... and the URL is http://127.0.0.1:4560. The main content area displays the text "My Shiny App" in a large font, followed by a link "See other apps in the Shiny Showcase". The application is styled with a container-fluid div and an h1 header.

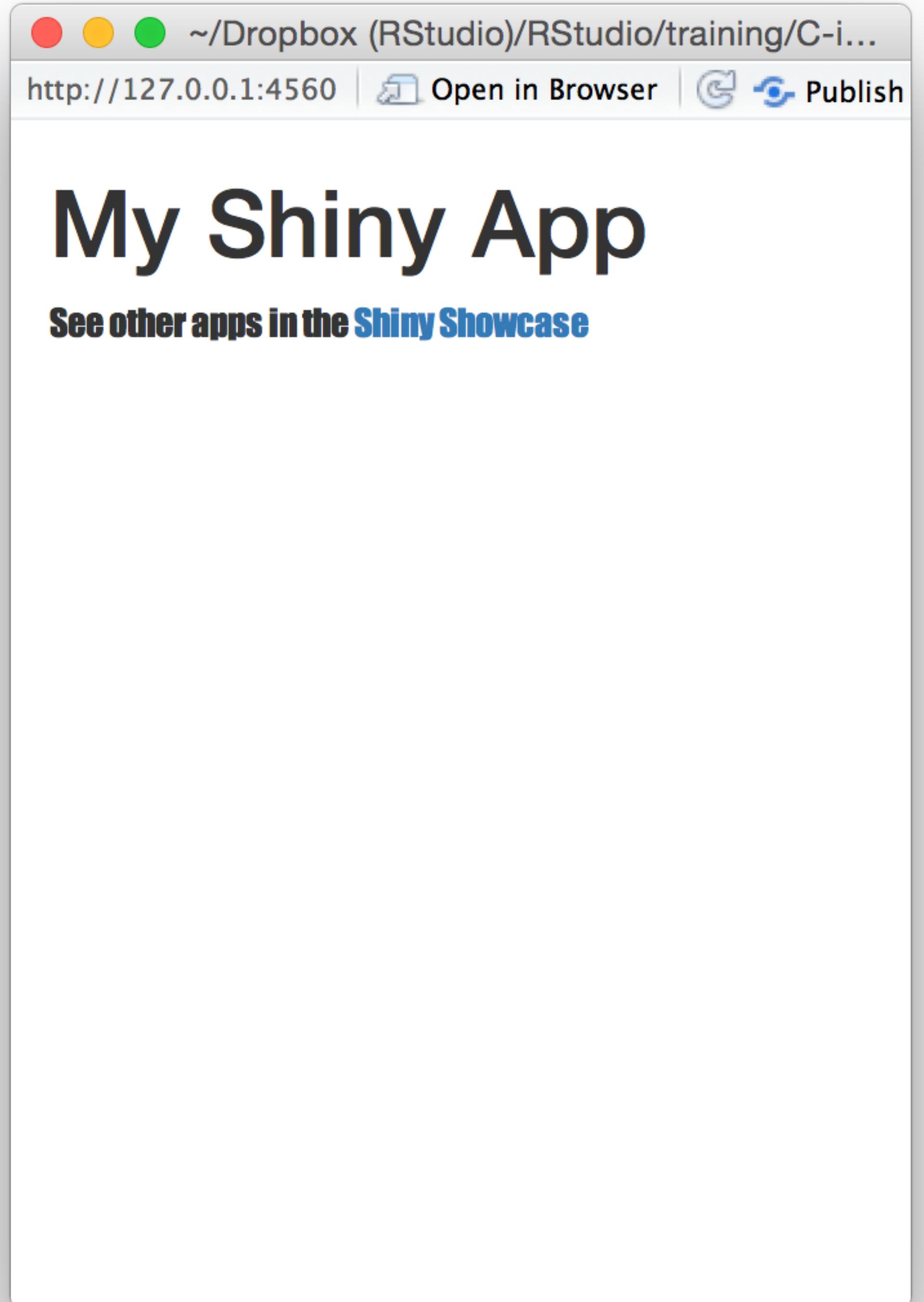
# My Shiny App

See other apps in the [Shiny Showcase](http://www.rstudio.com/products/shiny/shiny-user-showcase/)

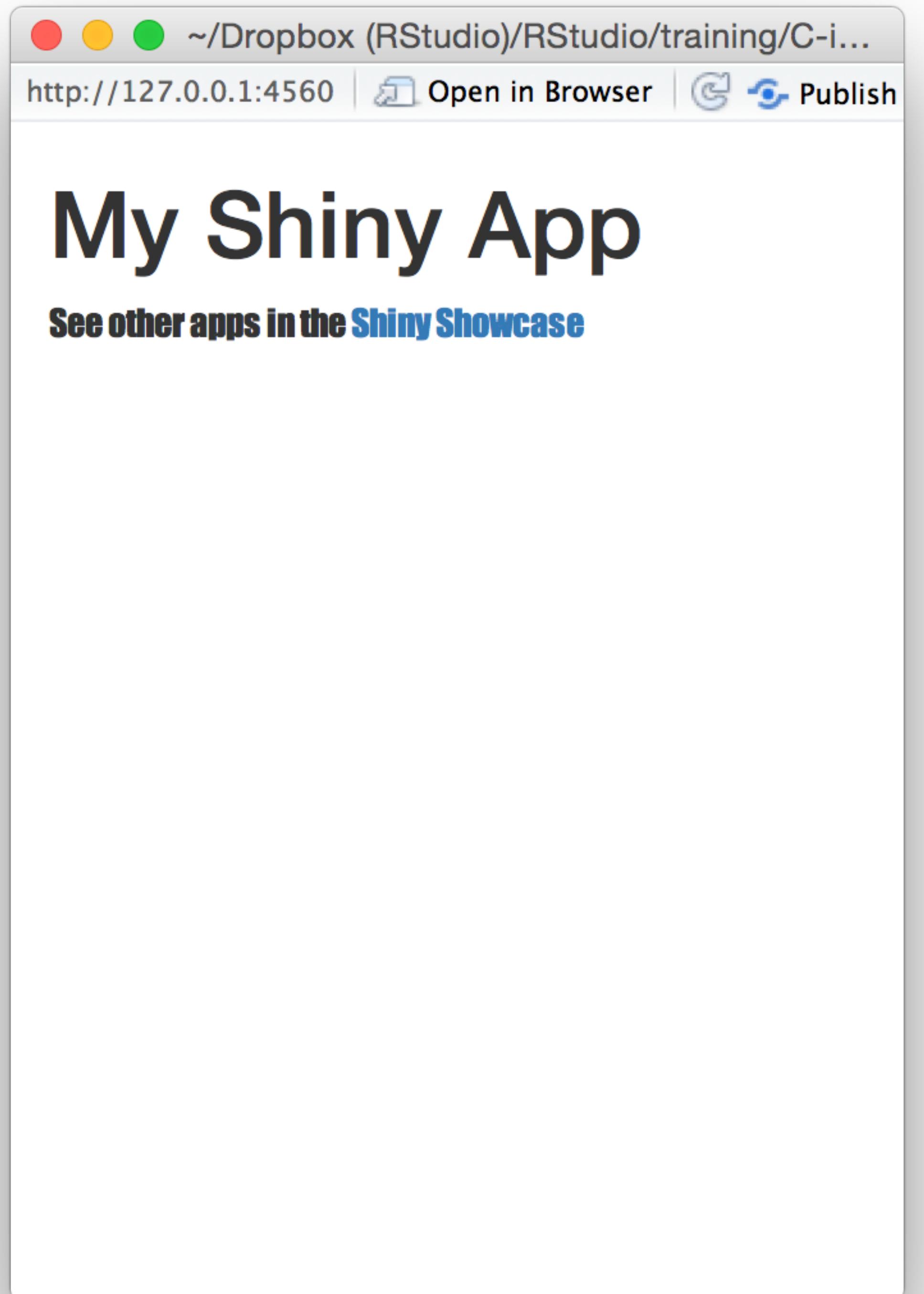
```
fluidPage(
```

```
'<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(  
  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()`



`includeHTML()`



`includeScript()`



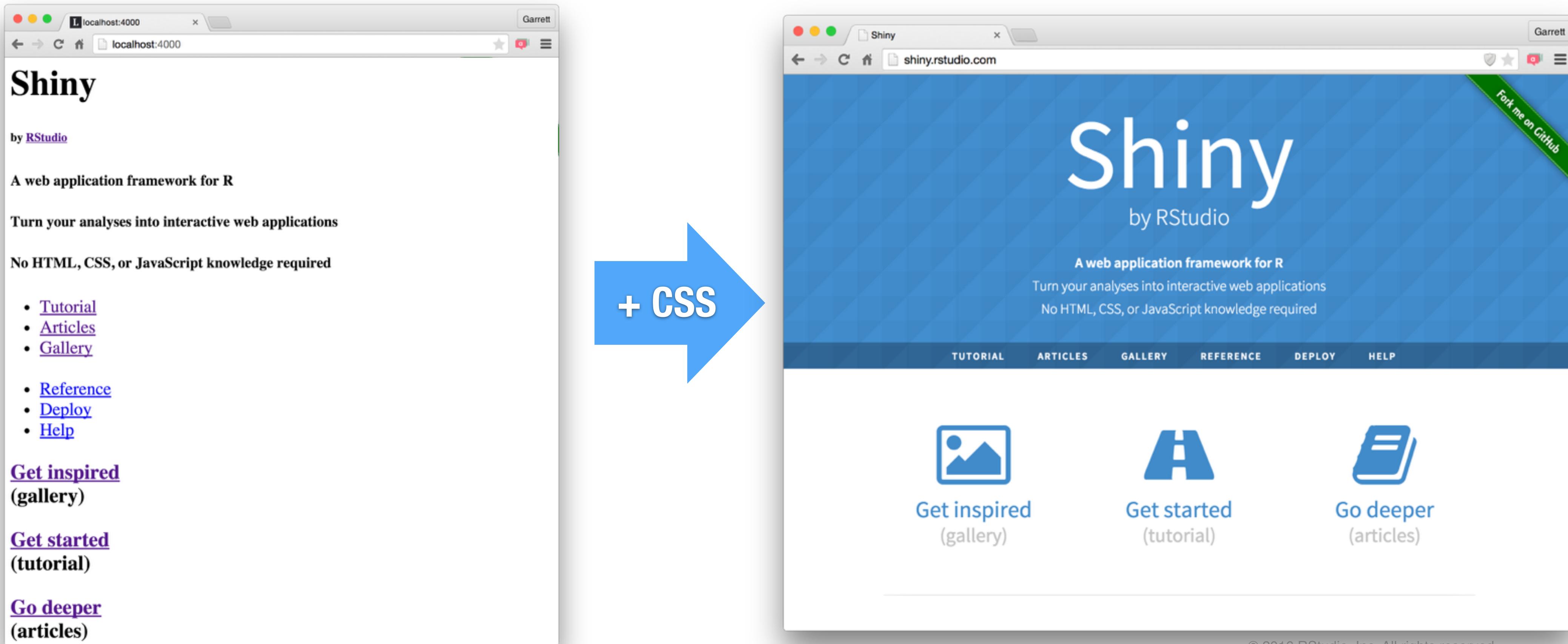
`includeMarkdown()`



`includeText()`

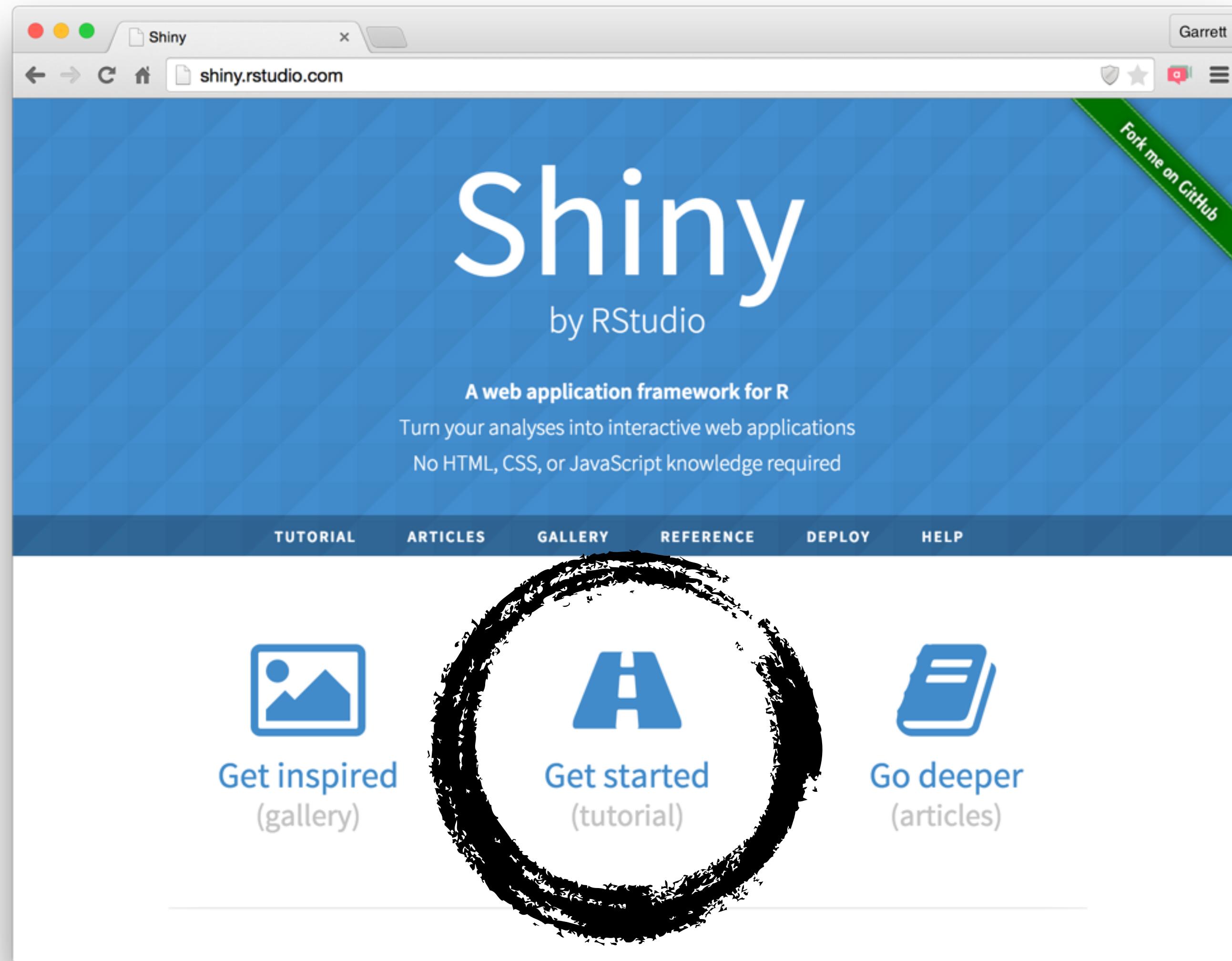
# What is CSS?

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



# The Shiny Development Center

[shiny.rstudio.com](http://shiny.rstudio.com)



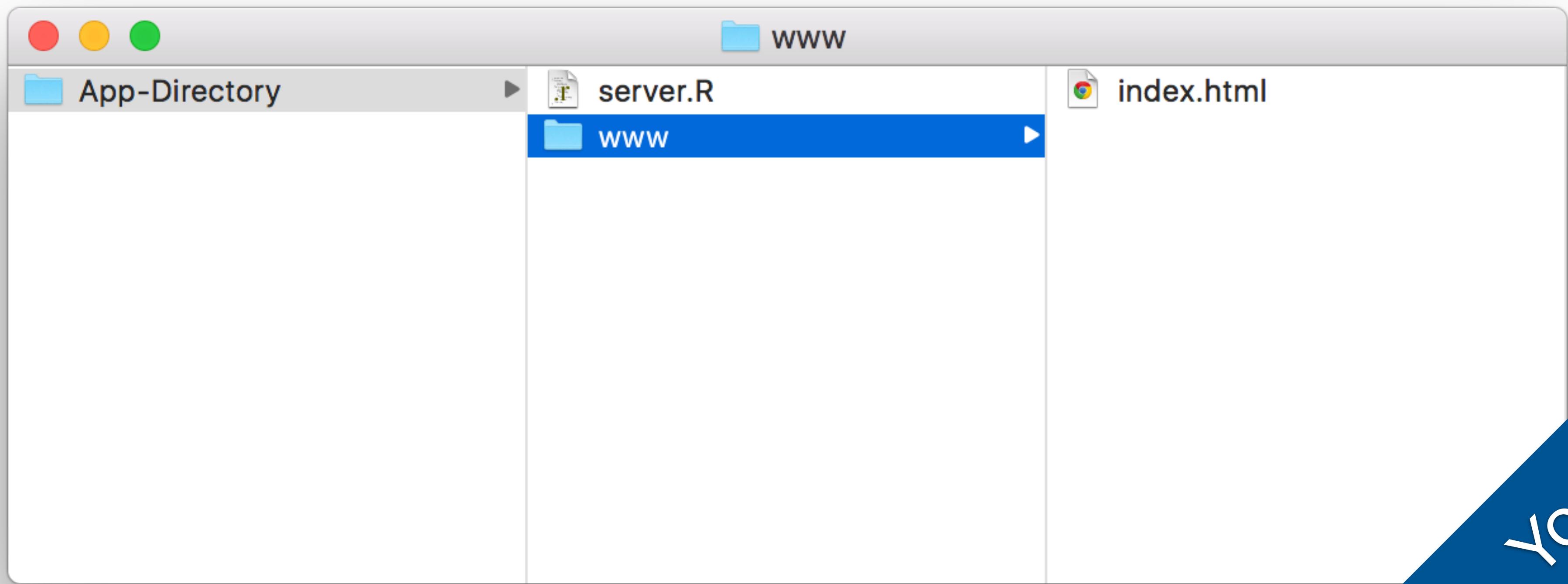


# index.html

Build it with html

# index.html

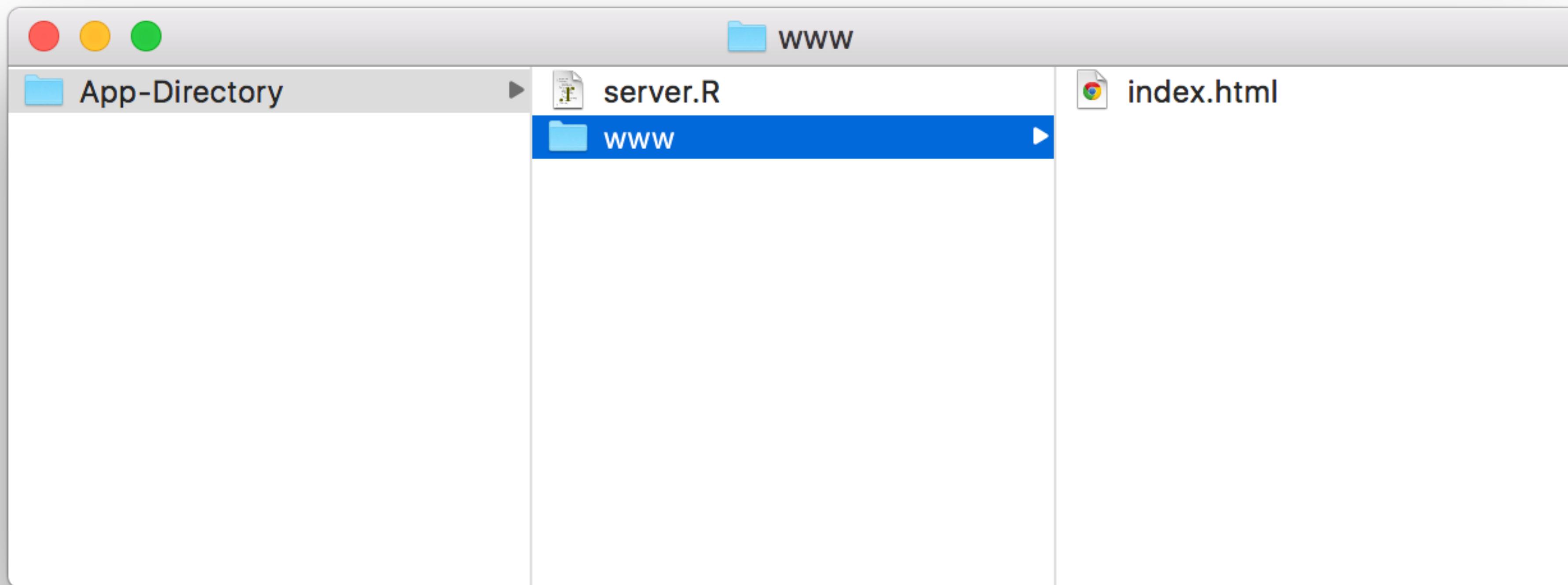
The DOM of your shiny app, written in HTML  
and saved in the www directory.



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

1

```
<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>
```

1

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

index.html

# 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](https://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>
```

# 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()**

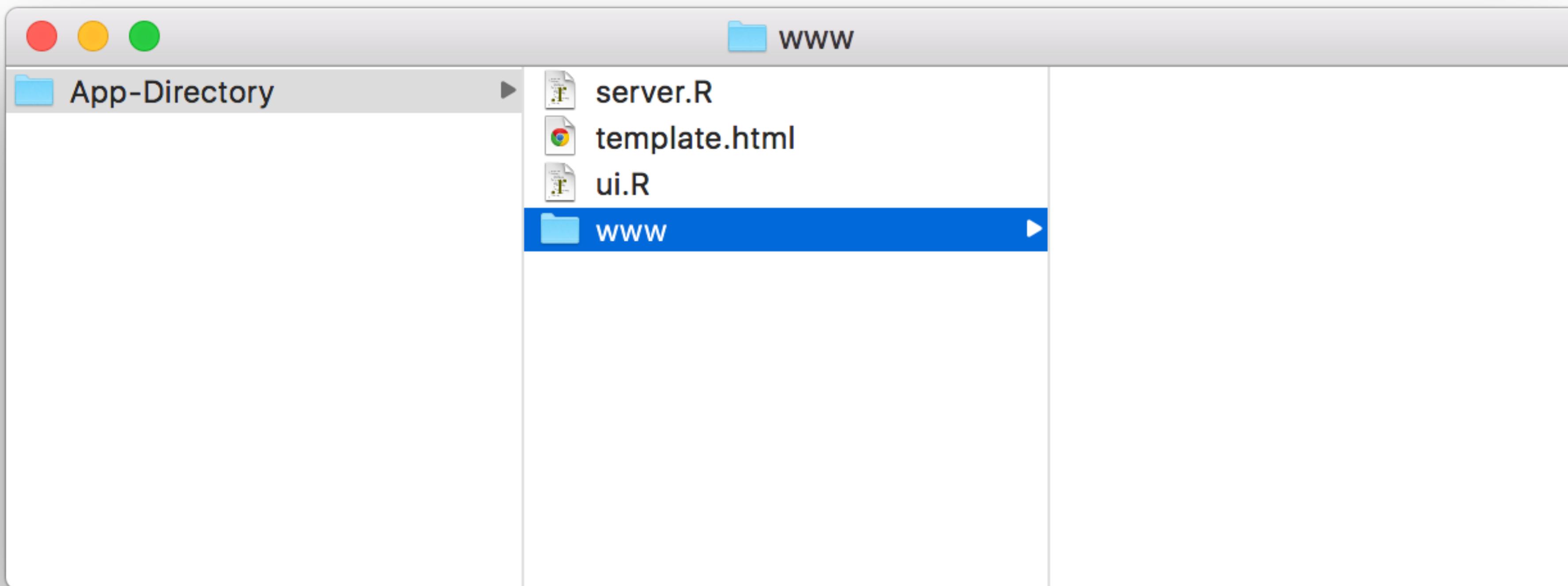
adds required Shiny code to header of a complete page

## **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](http://shiny.rstudio.com/articles/templates.html)

The screenshot shows a web browser window titled "Shiny - HTML Templates". The address bar displays "localhost:4000/articles/templates.html". The page content is titled "HTML Templates" and includes author information: "ADDED: 31 DEC 2015" and "BY: WINSTON CHANG". A paragraph of text discusses the use of HTML templates in Shiny applications, mentioning the fluidPage(), div(), and htmltools packages. The footer of the page includes the RStudio logo and copyright information.

Garrett

localhost:4000/articles/templates.html

Shiny by RStudio

Search

OVERVIEW TUTORIAL ARTICLES GALLERY REFERENCE DEPLOY HELP

## HTML Templates

ADDED: 31 DEC 2015  
BY: WINSTON CHANG

In most cases, the best way to create a Shiny application's user interface is to build it with R code, using functions like `fluidPage()`, `div()`, and so on. Sometimes, though, you may want to integrate Shiny with existing HTML, and starting with Shiny 0.13 (and `htmltools` 0.3), this can be done with the HTML templates. Templates can be used to generate complete web pages, and they can also be used to generate the HTML for components that are included in a Shiny app.

Complete web pages

© 2016 RStudio, Inc. All rights reserved.

**Thank you**  
Learn more at  
[shiny.rstudio.com](https://shiny.rstudio.com)