Outstanding User Interfaces with Shiny

David Granjon

2020-08-31

To my son,

without whom I should have finished this book two years earlier

Contents

Prerequisites

- Be familiar with Shiny[1]
- Basic knowledge in HTML and JavaScript is a plus but not mandatory

Disclaimer

This book is not an HTML/Javascript/CSS course! Instead, it provides a *survival kit* to be able to customize Shiny. I am sure however that readers will want to explore more about these topics.

Is this book for me?

You should read this book if you answer yes to the following questions:

- Do you want to know how to develop outstanding shiny apps?
- Have you ever wondered how to develop new input widgets?

Related content

See the RStudio Cloud[2] dedicated project. This book will serve as a reference for the e-Rum2020 workshop[3] about "Advanced User Interfaces for Shiny Developers", provided by Novartis[4]. The detailed agenda is available here[5].

[1] `https://mastering-shiny.org`
[2] `https://rstudio.cloud`
[3] `https://2020.erum.io/program/workshops/`
[4] `https://www.novartis.com`
[5] `https://github.com/Novartis/Advanced-User-Interfaces-for-Shiny-Developers`

Acknowledgements

- I am very grateful to Douglas Robinson for proof reading the book and fixing many typos it contained.
- A special thanks to my friends John and Victor for contributing to RinteRface. By contribution, I am not meaning only code contribution but also support in any form.
- Thanks to the eRum organizers for giving me the opportunity to present this work at the virtual e-Rum2020[6] conference
- RinteRface and this book won't exist without the amazing R community. Thanks for their valuable feedback.

Packages

```
library(shiny)
library(shinydashboard)
library(cascadess)
library(htmltools)
library(purrr)
library(magrittr)
library(ggplot2)
library(thematic)
library(fresh)
library(testthat)
library(jstools)
library(scales)
library(dplyr)
library(apexcharter)
library(shinyWidgets)
```

[6] https://2020.erum.io

Chapter 1

Introduction

There are various Shiny focused resources introducing basic[1] as well as advanced topics such as modules[2] and Javascript/R[3] interactions, however, handling advanced user interfaces design was never an emphasis. Clients often desire custom templates, yet this generally exceeds core features of Shiny (not out of the box).

Generally, R App developers lack a significant background in web development and often find this requirement overwhelming. It was this sentiment that motivated writing this book, namely to provide readers the necessary knowledge to extend Shiny's layout, input widgets and output elements. This project officially started at the end of 2018 but was stopped when Joe Cheng revealed the upcoming Mastering Shiny Book[4]. Fortunately, the later, does not cover a lot about the customization of Shiny user interfaces. Besides, this book may constitute a good complement to the work in progress Engineering Production-Grade Shiny Apps[5] by the ThinkR team, where the link between Shiny and CSS/JavaScript is covered.

This book is organized into five parts.

- We first go through the basics of HTML, JavaScript and jQuery
- Part 2 contains chapters dedicated to the partially hidden features of Shiny, yet so fun. We dedicate an entire chapter to describe how inputs work in detail and how to add new inputs to the system
- In part 3, we dive into the `{htmltools}` package, providing functions to create and manipulate shiny tags as well as manage dependencies
- Part 4 focuses on the development of a new template for Shiny by demonstrating examples from the `{tablerDash}`, `{bs4Dash}` and

[1] `https://shiny.rstudio.com/tutorial/`
[2] `https://shiny.rstudio.com/articles/#modules`
[3] `https://js4shiny.com`
[4] `https://mastering-shiny.org`
[5] `https://engineering-shiny.org`

`{shinyMobile}` packages. These, and more may be explored further as part of the RinteRface project.
- Part 5 present some tools of the R community, like `{fresh}`, to beautify apps with only few lines of code

1.1 About the code

This book has a side package containing all the necessary material to run the code without having to reload each previous snippet.

```
remotes::install_github("DivadNojnarg/outstanding-shiny-ui-code
")
```

It covers Chapters **??**, **??** and the whole Practice section.

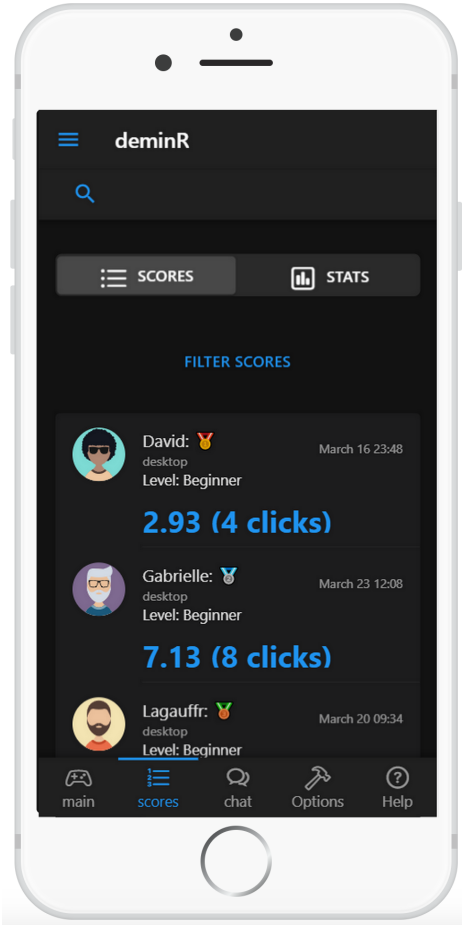
There is another good reason for this package: provide a robust method to bundle JavaScript/CSS code along side any shiny app. Indeed, it is quite tempting to proceed as below:

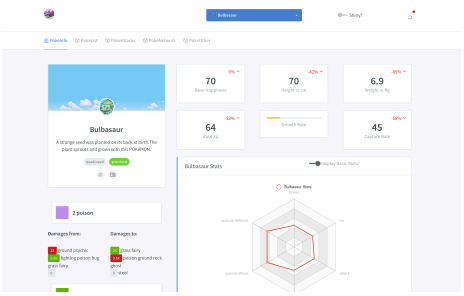
```
ui <- fluidPage(
  tags$script(
    "$(function() {
      Shiny.addCustomMessageHandler('hello', function(message) {     alert(message);
      });
    });
    "
  ),
  actionButton("go", "Send")
)

server <- function(input, output, session) {
  observeEvent(input$go, {
    session$sendCustomMessage("hello", message = "plop")
  })
}

shinyApp(ui, server)
```

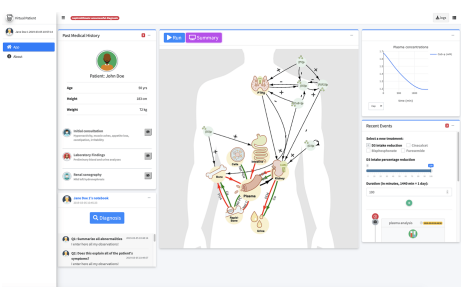
It is fine if the app purpose is a simple demonstration. In our case, since we aim at providing reusable template elements, we need a better approach, that will be described later.

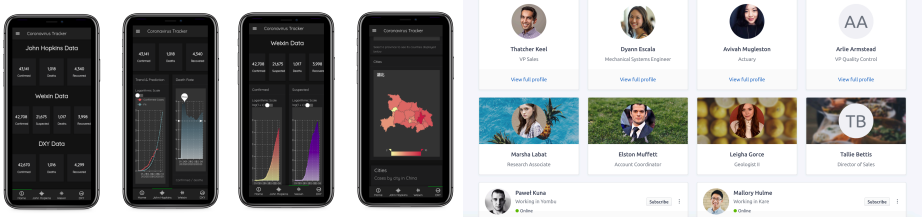
1.2 Preliminary exercices

Before starting with technical details, we propose to play a little game. Among all the images shown, what are the ones corresponding to shiny apps? Images are numbered from A to F (left to right, top to bottom).








Coronavirus Tracker
John Hopkins Data
Welsin Data
DXY Data
tabler
Snippets
Marija Lebarree
Civil Engineer
Egon Peetz
Research Nurse
Kellie Skingley
Teacher
Christabel Charlwood
Tax Accountant
Thatcher Keel
VP Sales
Dyann Escala
Mechanical Systems Engineer
Avivah Mugleston
Actuary
Arlie Armstead
VP Quality Control
AA
View full profile
Marsha Labat
Research Associate
Elston Muffett
Account Coordinator
Leigha Gorce
Geologist II
Tallie Bettis
Director of Sales
TB
Pawel Kuna
Working in Yombu
Online
Subscribe
Mallory Hulme
Working in Kare
Online
Subscribe

Survival Kit

This part will give you basis in HTML, JavaScript to get started...

Chapter 2

HTML

This chapter provides a short introduction to the HTML language. As a quick example, open up RStudio and perform the following:

- Load shiny with `library(shiny)`
- Execute `p("Hello World")`

Notice the output format is an example of an HTML tag!

2.1 HTML Basics

HTML (Hypertext Markup Language) is derived from SGML (Standard Generalized markup Language). An HTML file contains tags that may be divided into 2 categories:

- paired-tags: the text is inserted between the opening and the closing tag
- closing-tags

```
<!-- paired-tags -->
<p></p>
<div></div>

<!-- self-closing tags -->
<iframe/>
<img/>
<input/>
<br/>
```

Tags may be divided into 3 categories, based on their role:

- structure tags: they constitute the skeleton of the HTML page (`<title></title>`, `<head></head>`, `<body></body>`)
- control tags: script, inputs and buttons (and more). Their role is to include external resources, provide interactivity with the user
- formatting tags: to control the size, font of the wrapped text

Finally, we distinguish block and inline elements:

- block elements may contain other tags and take the full width (block or inline). `<div></div>` is the most commonly used block element. All elements of a block are printed on top of each others
- inline elements (for instance `<span></span>`, `<a></a>`) are printed on the same line. They can not contain block tags but may contain other nested inline tags. In practice, we often see `<a><span></span></a>`
- inline-block elements allow to insert block element in an inline

Consider the following example. This is clearly a bad use of HTML conventions since an inline tag can not host block elements.

```
<span>
  <div><p>Hello World</p></div>
  <div></div>
</span>
```

Importantly, `<div>` and `<span>` don't have any semantic meaning, contrary to `<header>` and `<footer>`, which allow to structure the HTML page.

2.2 Tag attributes

Attributes are text elements allowing to specify some properties of the tag. For instance for a link tag (`<a></a>`), we actually expect more than just the tag itself: a target url and how to open the new page … In all previous examples, tags don't have any attributes. Yet, there exist a large range of attributes and we will only see 2 of them for now (the reason is that these are the most commonly used in CSS and JavaScript):

- class: may be shared between multiple tags
- id: each must be unique