



A Beginner's Guide to a Multilingual Shiny App

Karl Kennedy

Agenda

- Introduction
- A brief overview of Shiny implementation
- JavaScript and CSS
- Python
- Modularisation
- R Markdown
- Conclusion
- Questions

Introduction



A brief overview of Shiny Implementation

- User Interface
- Server Component
- Shiny Core Files

```
ui <- fluidPage(  
  # Place your UI components  
)  
server <- function(input, output) {  
  # Place your server/application logic  
}
```

JavaScript and CSS

- Communicating between R and JavaScript

```
if(btn1bl == hidebtn1bl){  
  # hideElement is a function from shinyJS package.  
  hideElement(selector = sidebar1)  
  hideElement(selector = sidebar2)  
  
  session$sendCustomMessage("hide", hide(tblid)      )  
}
```

- JavaScript - Receiving messages

```
tags$head(  
  # Listener for show and hide sidebars.  
  tags$script("Shiny.addCustomMessageHandler('hide', function(main){  
    const input = document.getElementsByClassName('col-sm-8 ' + main);  
    input[0].className = 'col-sm-12 ' + main  
  });"),
```

Python

- Configuring Python
- Reticulate package

```
# Display Python version
output$python_version <- renderText({
  rr = reticulate::py_discover_config(use_environment = 'python35_env')

  paste0('Python version: ', rr$version)
})
```

- Python functions

```
def check_file(fn, txt):
    with open(fn) as f:
        x = "false"
        if txt in f.read():
            x = "true"
    return(x)
```

[GitHub - ranikay/shiny-reticulate-app: An end-to-end tutorial creating an R Shiny app that uses the reticulate package with Python 3](#)

Modularisation

- Benefits of Modularisation
- Modularisation example (UI)

```
# Generates the UI download button
downloadButton <- function(id, label) {
  ns <- NS(id)
  tagList(downloadButton(ns("button")))
}
# Inserts the UI download button into the app
downloadButton("download1", "Download")
```

Modularisation Cont.

```
# Modular server logic to download a dataframe to a
PDF file.
downloadServer <- function(id) {
  moduleServer(id, function(input, output, session)
  {
    output$button <- downloadHandler(
      filename = "listing.pdf",
      content = function(f) {
        ...
      }
    )
  }
)

# Inserts the download server logic into the app.
downloadServer("download1")
```

R Markdown

- Purpose of R Markdown
- A Simple R Markdown Example

```
---  
title: "Download Example"  
author: "Karl Kennedy"  
date: "26/5/1999"  
output: pdf_document  
---  
```${r setup, include=FALSE}  
library(knitr)
library(pander)
knitr::opts_chunk$set(echo = FALSE)
```
```

[Dynamic R Markdown Reports with Shiny - Stefan Eng's blog \(stefanengineering.com\)](http://stefanengineering.com)

R Markdown Cont.

```
` `` {r, results='asis' }
```

```
panderOptions('knitr.auto.asis', FALSE)  
for(d in datasets) {  
  pander::pander(d, split.table=120)  
` ``
```

[Dynamic R Markdown Reports with Shiny - Stefan Eng's blog \(stefanengineering.com\)](http://stefanengineering.com)

Conclusion & Questions

