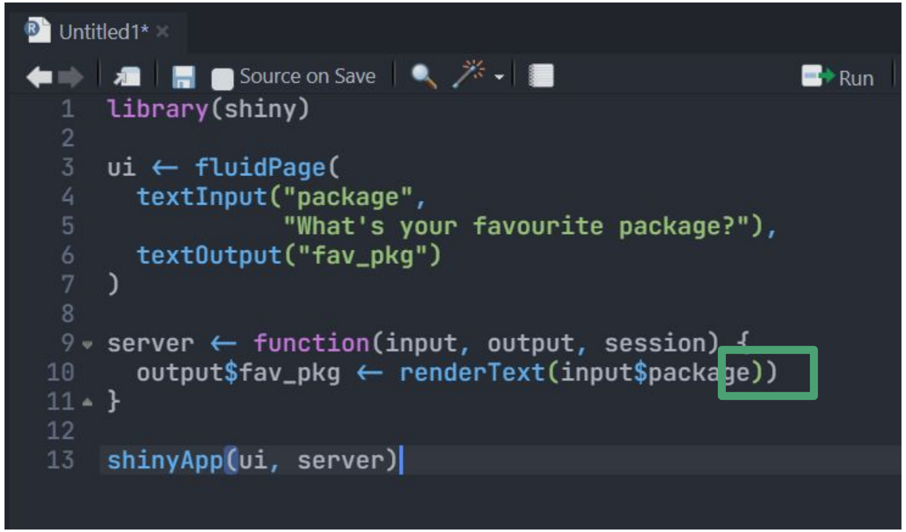
Debugging Shiny Apps

Tan Ho
ML Engineer, Zelus Analytics
ShinyConf 2023

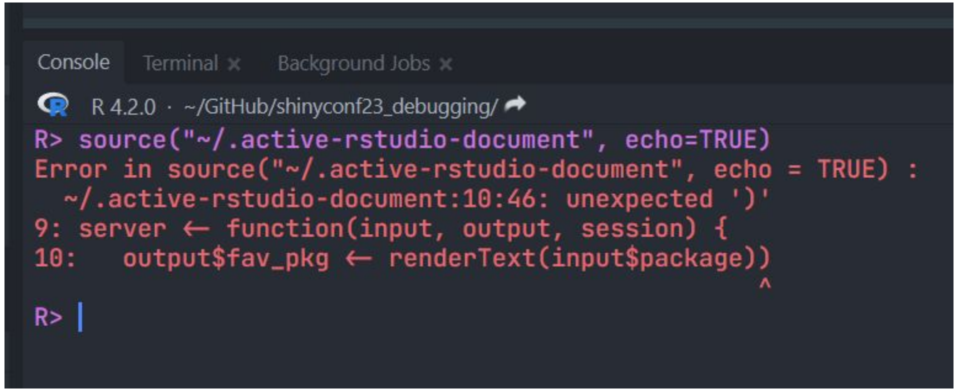
tanho.ca/debugging-shiny

Pop quiz!

What does this error mean?

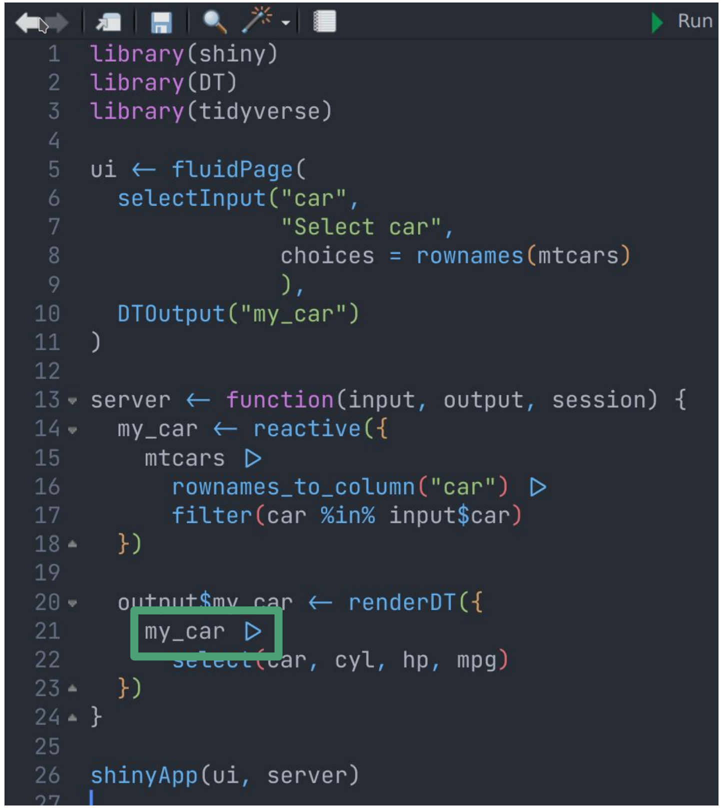


```r
library(shiny)

ui <- fluidPage(
  textInput("package",
            "What's your favourite package?"),
  textOutput("fav_pkg")
)

server <- function(input, output, session) {
  output$fav_pkg <- renderText(input$package))
}

shinyApp(ui, server)
```

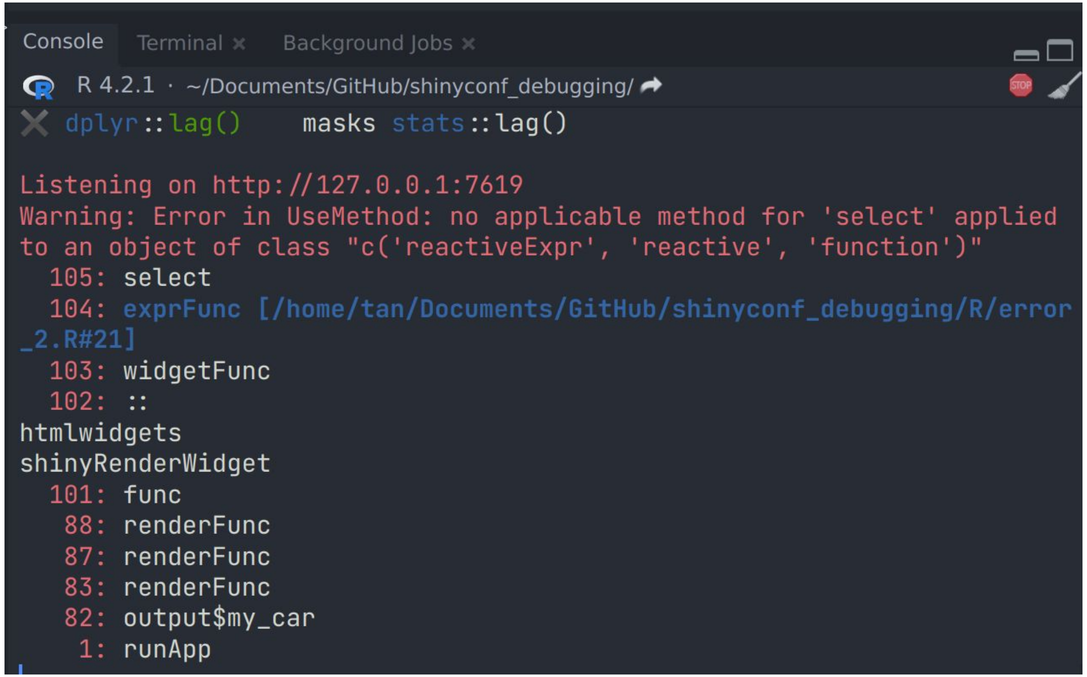


```
R> source("~/.active-rstudio-document", echo=TRUE)
Error in source("~/.active-rstudio-document", echo = TRUE) :
  ~/.active-rstudio-document:10:46: unexpected ')'
9: server <- function(input, output, session) {
10:    output$fav_pkg <- renderText(input$package))
                                                ^

R>
```

What does this error mean, part 2

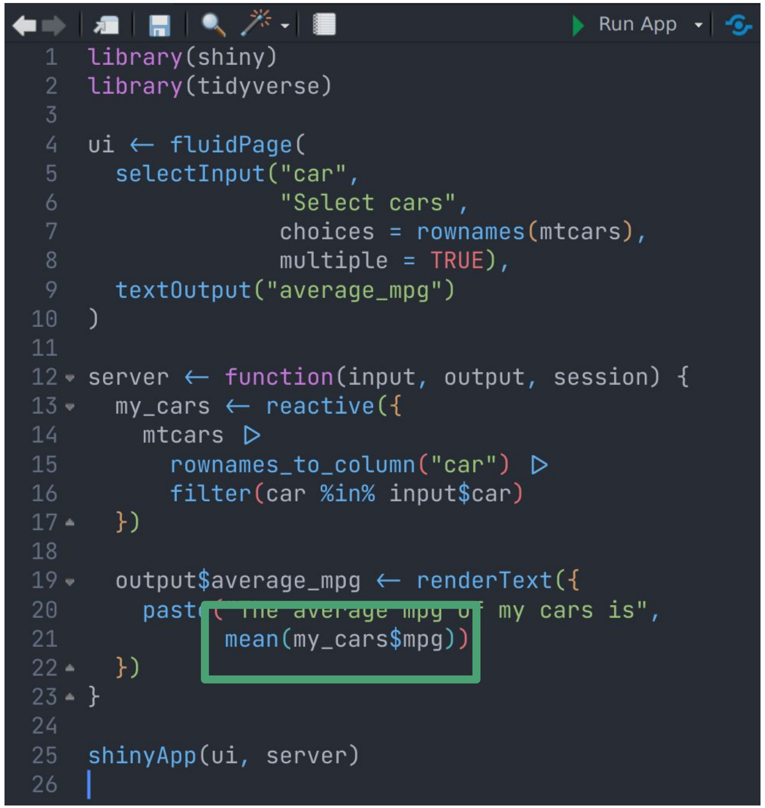


```
library(shiny)
library(DT)
library(tidyverse)

ui ← fluidPage(
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
              ),
  DTOutput("my_car")
)

server ← function(input, output, session) {
  my_car ← reactive({
    mtcars ▷
      rownames_to_column("car") ▷
      filter(car %in% input$car)
  })

  output$my_car ← renderDT({
    my_car ▷
      select(car, cyl, hp, mpg)
  })
}

shinyApp(ui, server)
```

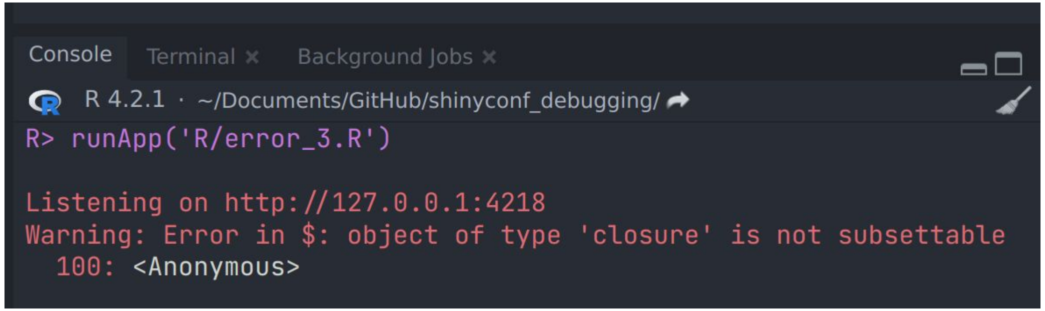


```
R 4.2.1 · ~/Documents/GitHub/shinyconf_debugging/
✖ dplyr::lag()    masks stats::lag()

Listening on http://127.0.0.1:7619
Warning: Error in UseMethod: no applicable method for 'select' applied
to an object of class "c('reactiveExpr', 'reactive', 'function')"
  105: select
  104: exprFunc [/home/tan/Documents/GitHub/shinyconf_debugging/R/error
_2.R#21]
  103: widgetFunc
  102: ::
htmlwidgets
shinyRenderWidget
  101: func
   88: renderFunc
   87: renderFunc
   83: renderFunc
   82: output$my_car
    1: runApp
```

What does this error mean, part 3



```
library(shiny)
library(tidyverse)

ui ← fluidPage(
  selectInput("car",
              "Select cars",
              choices = rownames(mtcars),
              multiple = TRUE),
  textOutput("average_mpg")
)

server ← function(input, output, session) {
  my_cars ← reactive({
    mtcars ▷
      rownames_to_column("car") ▷
      filter(car %in% input$car)
  })

  output$average_mpg ← renderText({
    paste("The average mpg of my cars is",
          mean(my_cars$mpg))
  })
}

shinyApp(ui, server)
```

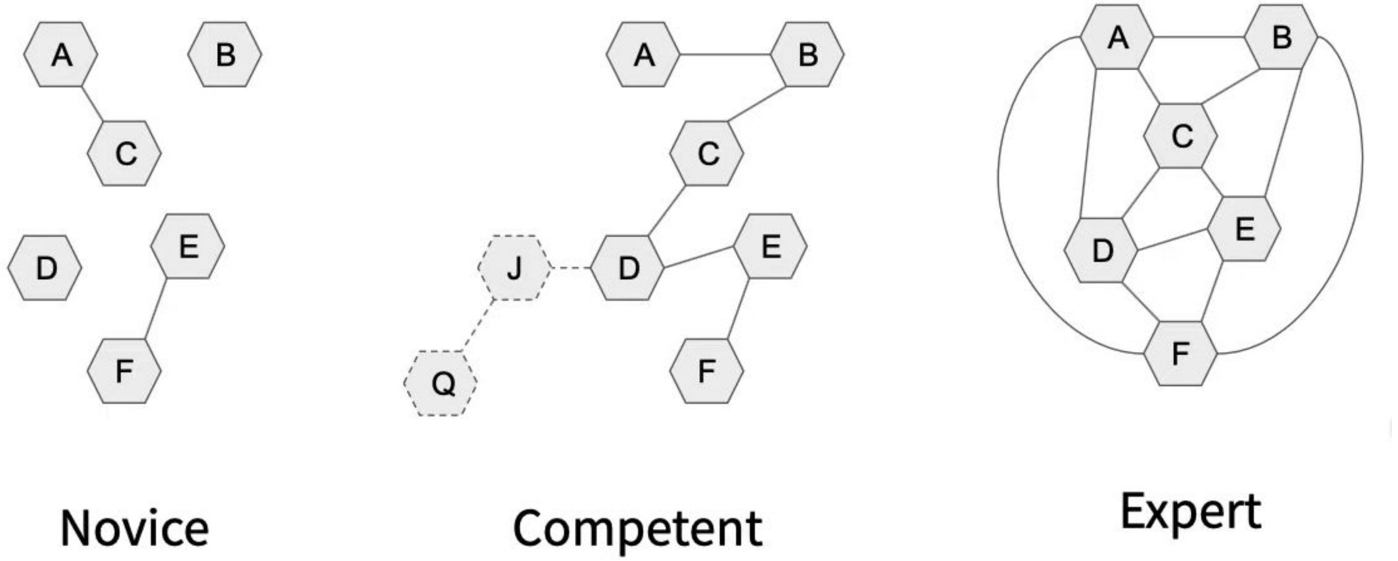


```
R> runApp('R/error_3.R')

Listening on http://127.0.0.1:4218
Warning: Error in $: object of type 'closure' is not subsettable
  100: <Anonymous>
```

A closure is the C type name for an R function
- Advanced R, chapter 12

How can we learn to recognize these on sight?

We build mental models



What Everyone In Tech Should Know About Learning and Teaching - Greg Wilson

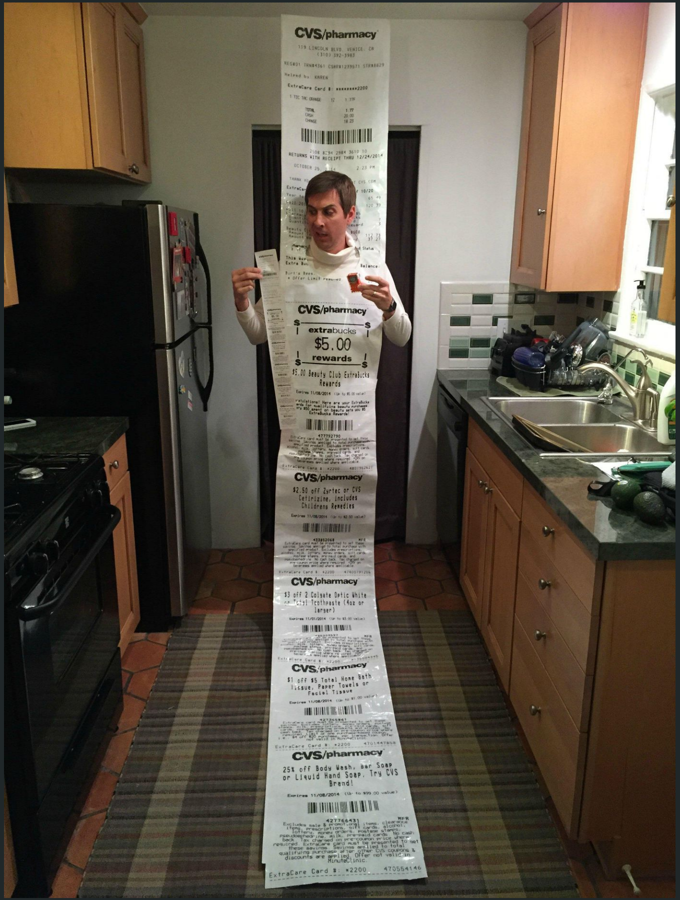
How do we build better mental models for Shiny?

Debugging!

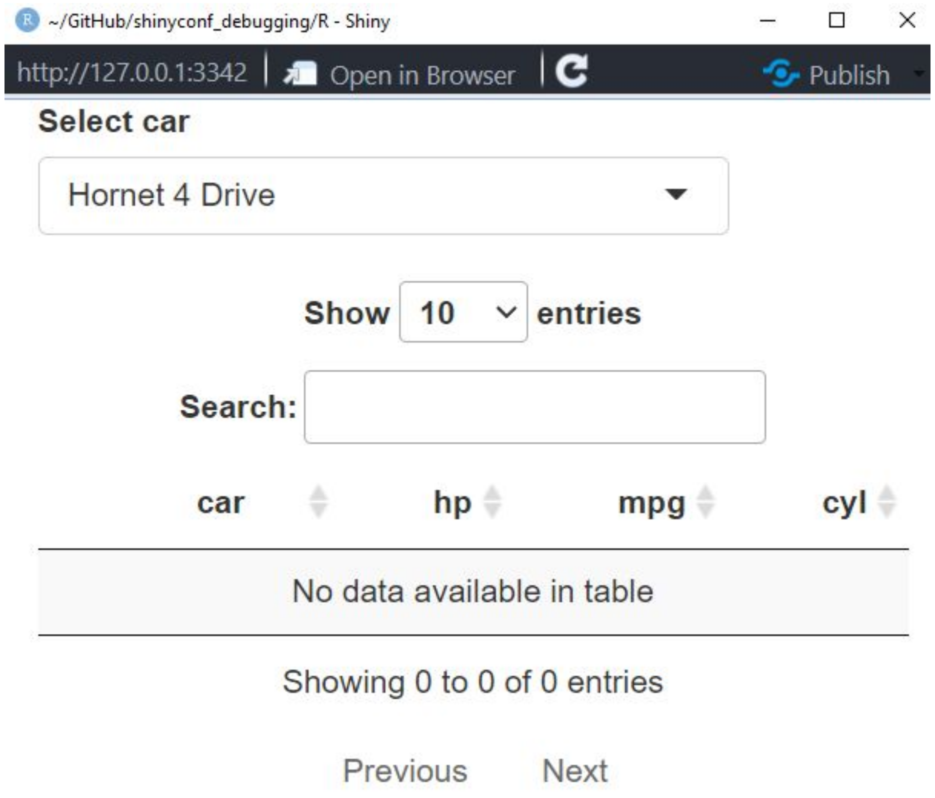
Observe - it’s your code!

We’ve all made this app before…

browser() is your best friend!



browser() is your best friend



Select car

Hornet 4 Drive

Show 10 entries

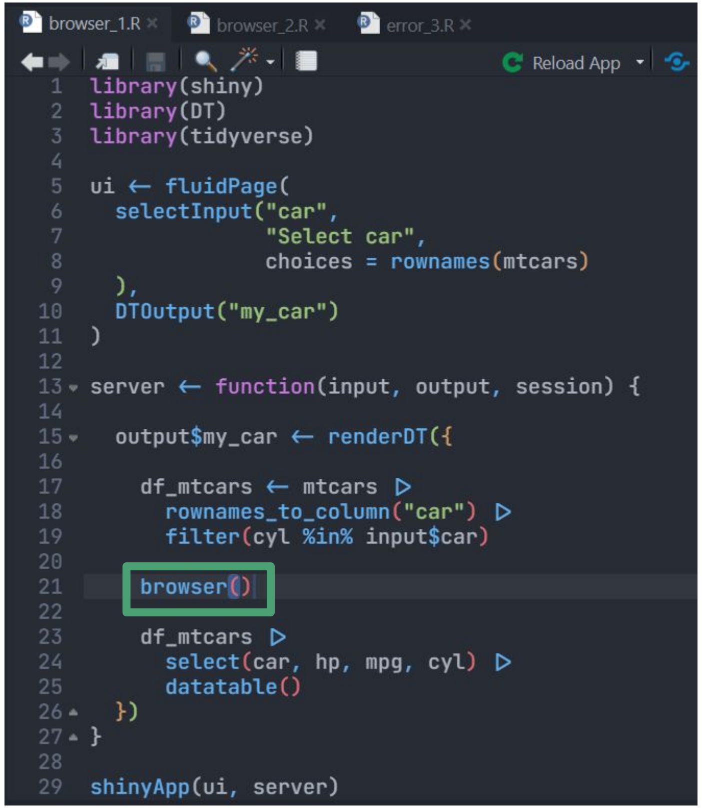
Search:

car	hp	mpg	cyl
No data available in table			

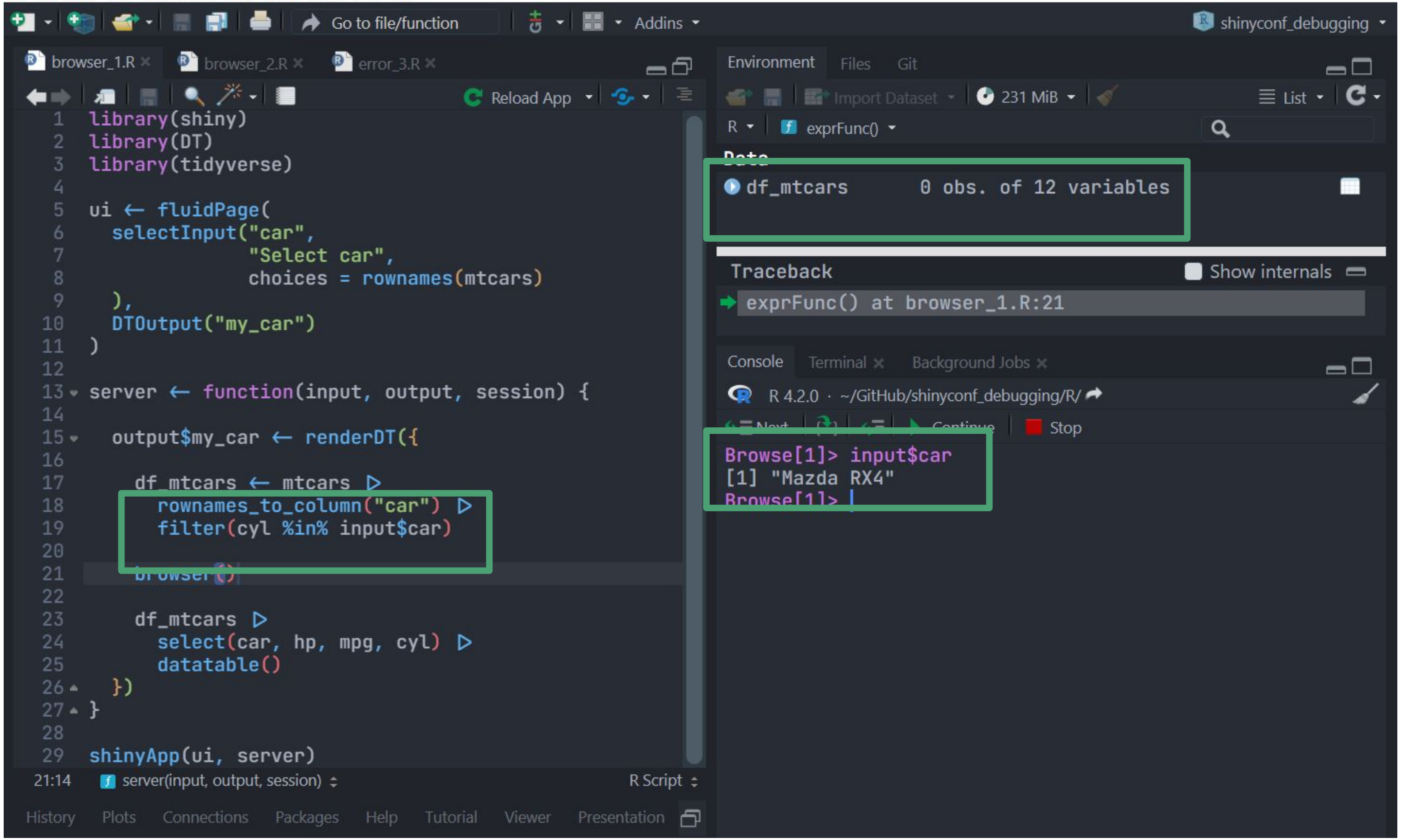
Showing 0 to 0 of 0 entries

Previous Next

browser() is your best friend



```r
library(shiny)
library(DT)
library(tidyverse)

ui <- fluidPage(
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
  ),
  DTOutput("my_car")
)

server <- function(input, output, session) {

  output$my_car <- renderDT({

    df_mtcars <- mtcars |>
      rownames_to_column("car") |>
      filter(cyl %in% input$car)

    browser()

    df_mtcars |>
      select(car, hp, mpg, cyl) |>
      datatable()
  })
}

shinyApp(ui, server)
```

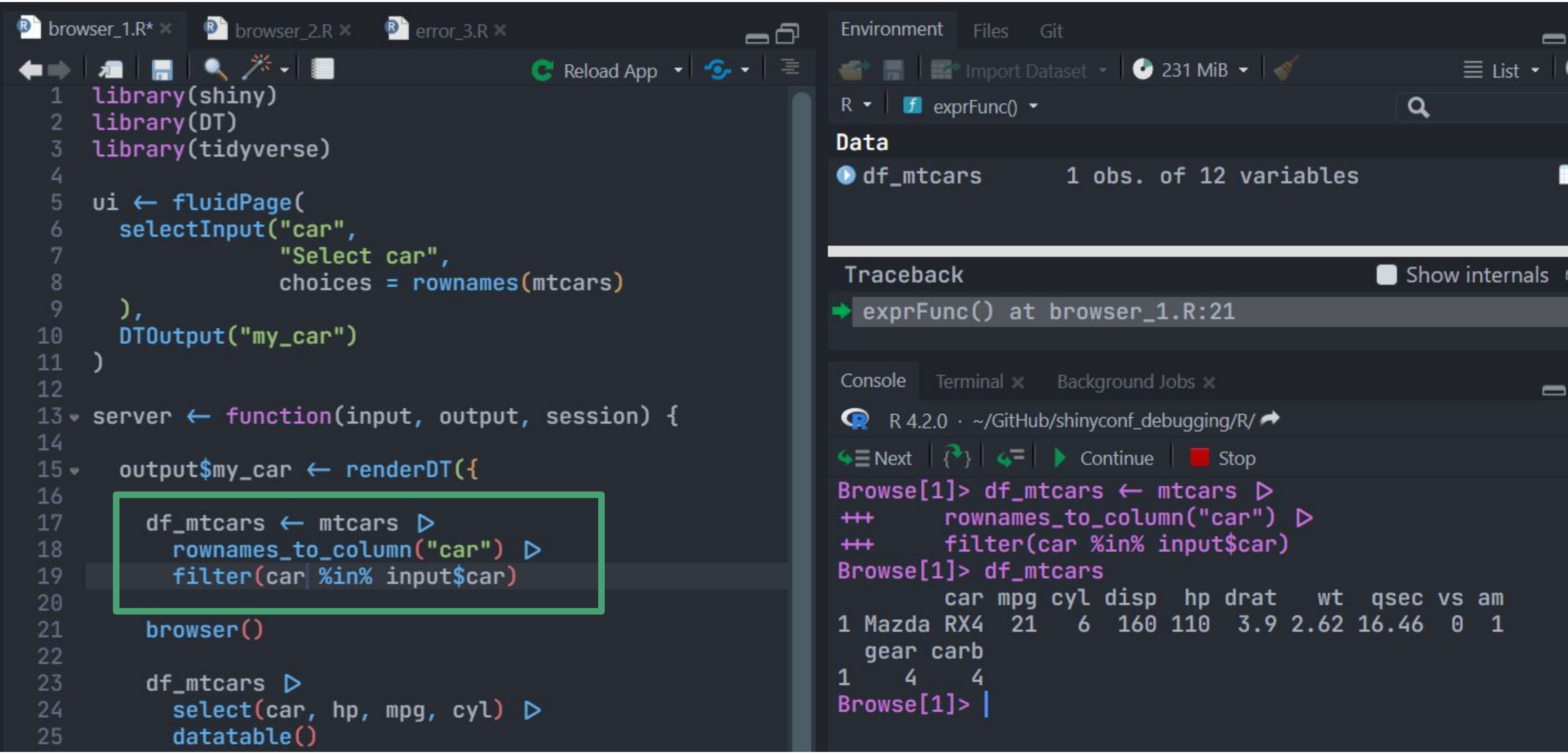


```r
library(shiny)
library(DT)
library(tidyverse)

ui <- fluidPage(
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
  ),
  DTOutput("my_car")
)

server <- function(input, output, session) {

  output$my_car <- renderDT({

    df_mtcars <- mtcars |>
      rownames_to_column("car") |>
      filter(cyl %in% input$car)

    browser()

    df_mtcars |>
      select(car, hp, mpg, cyl) |>
      datatable()
  })
}

shinyApp(ui, server)
```

```
df_mtcars       0 obs. of 12 variables
```

```
exprFunc() at browser_1.R:21
```

```
Browse[1]> input$car
[1] "Mazda RX4"
Browse[1]>
```



```r
library(shiny)
library(DT)
library(tidyverse)

ui <- fluidPage(
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
  ),
  DTOutput("my_car")
)

server <- function(input, output, session) {

  output$my_car <- renderDT({

    df_mtcars <- mtcars |>
      rownames_to_column("car") |>
      filter(car %in% input$car)

    browser()

    df_mtcars |>
      select(car, hp, mpg, cyl) |>
      datatable()
```

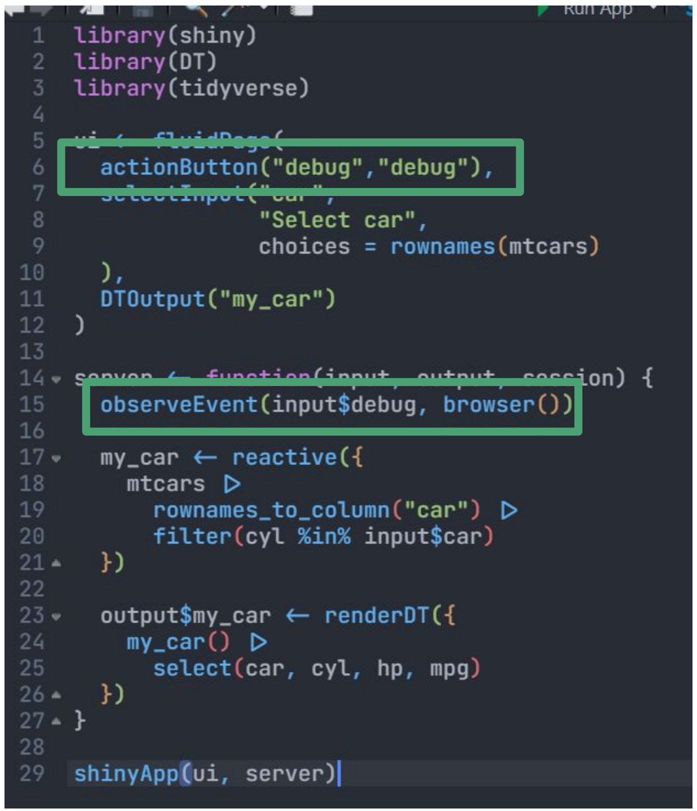
Data
df_mtcars   1 obs. of 12 variables

Traceback
exprFunc() at browser_1.R:21

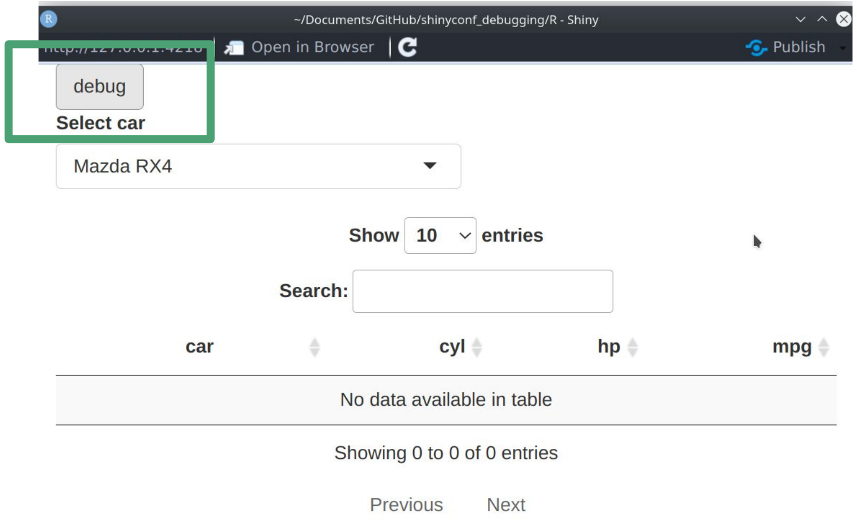
R 4.2.0 · ~/GitHub/shinyconf_debugging/R/

```
Browse[1]> df_mtcars <- mtcars |>
+     rownames_to_column("car") |>
+     filter(car %in% input$car)
Browse[1]> df_mtcars
        car mpg cyl disp  hp drat   wt  qsec vs am
1 Mazda RX4  21   6  160 110  3.9 2.62 16.46  0  1
  gear carb
1    4    4
Browse[1]> 
```

browser() ALL THE THINGS!



```r
library(shiny)
library(DT)
library(tidyverse)

ui <- fluidPage(
  actionButton("debug","debug"),
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
  ),
  DTOutput("my_car")
)

server <- function(input, output, session) {
  observeEvent(input$debug, browser())

  my_car <- reactive({
    mtcars |>
      rownames_to_column("car") |>
      filter(cyl %in% input$car)
  })

  output$my_car <- renderDT({
    my_car() |>
      select(car, cyl, hp, mpg)
  })
}

shinyApp(ui, server)
```



Select car

Mazda RX4

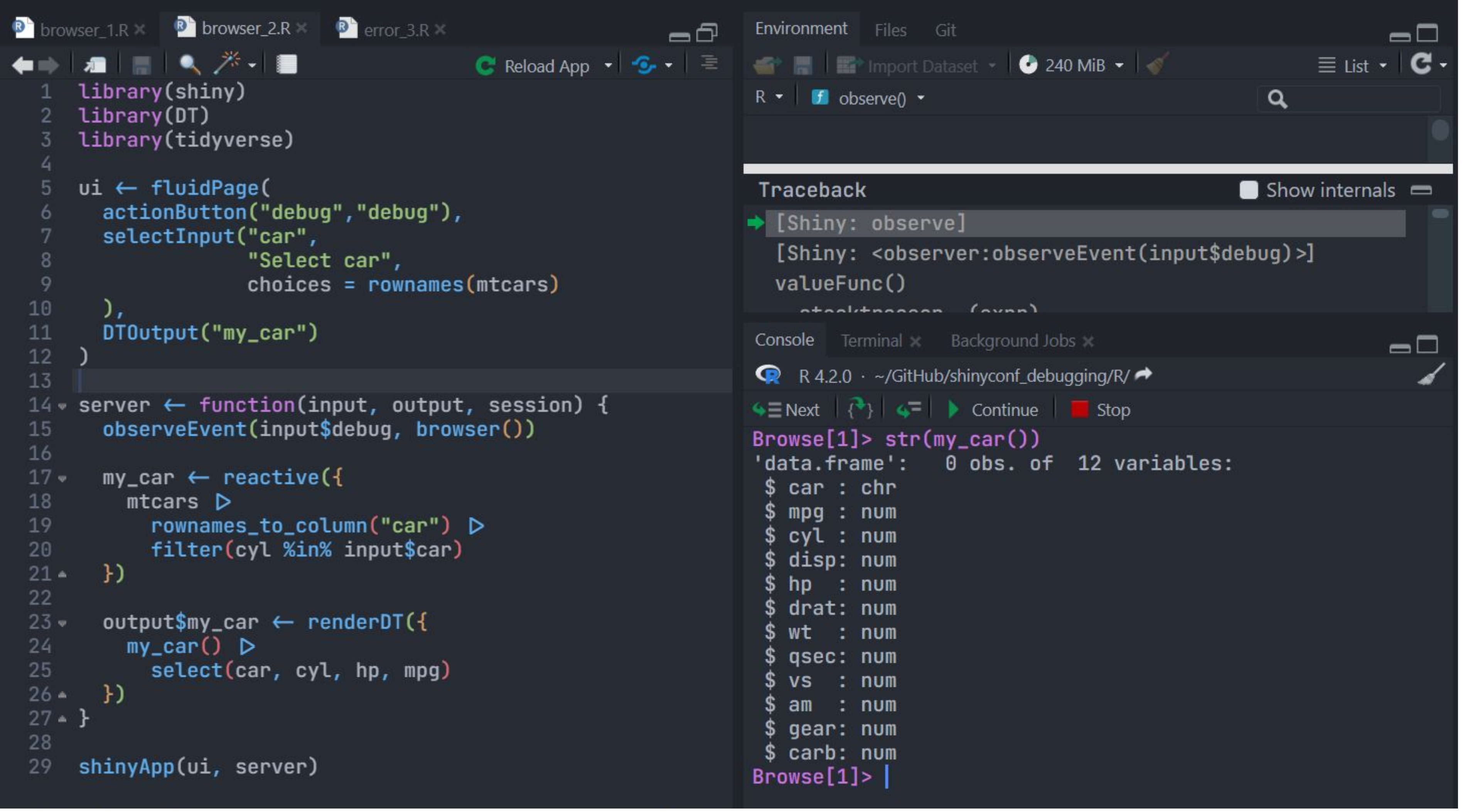
Show 10 **entries**

Search:

car	cyl	hp	mpg
No data available in table			

Showing 0 to 0 of 0 entries

Previous Next



```r
library(shiny)
library(DT)
library(tidyverse)

ui ← fluidPage(
  actionButton("debug","debug"),
  selectInput("car",
              "Select car",
              choices = rownames(mtcars)
  ),
  DTOutput("my_car")
)

server ← function(input, output, session) {
  observeEvent(input$debug, browser())

  my_car ← reactive({
    mtcars ▷
      rownames_to_column("car") ▷
      filter(cyl %in% input$car)
  })

  output$my_car ← renderDT({
    my_car() ▷
      select(car, cyl, hp, mpg)
  })
}

shinyApp(ui, server)
```

Traceback

```
[Shiny: observe]
[Shiny: <observer:observeEvent(input$debug)>]
valueFunc()
```

R 4.2.0 · ~/GitHub/shinyconf_debugging/R/

```
Browse[1]> str(my_car())
'data.frame':   0 obs. of  12 variables:
 $ car : chr
 $ mpg : num
 $ cyl : num
 $ disp: num
 $ hp  : num
 $ drat: num
 $ wt  : num
 $ qsec: num
 $ vs  : num
 $ am  : num
 $ gear: num
 $ carb: num
Browse[1]>
```

You’ve tried solving the problem in context…

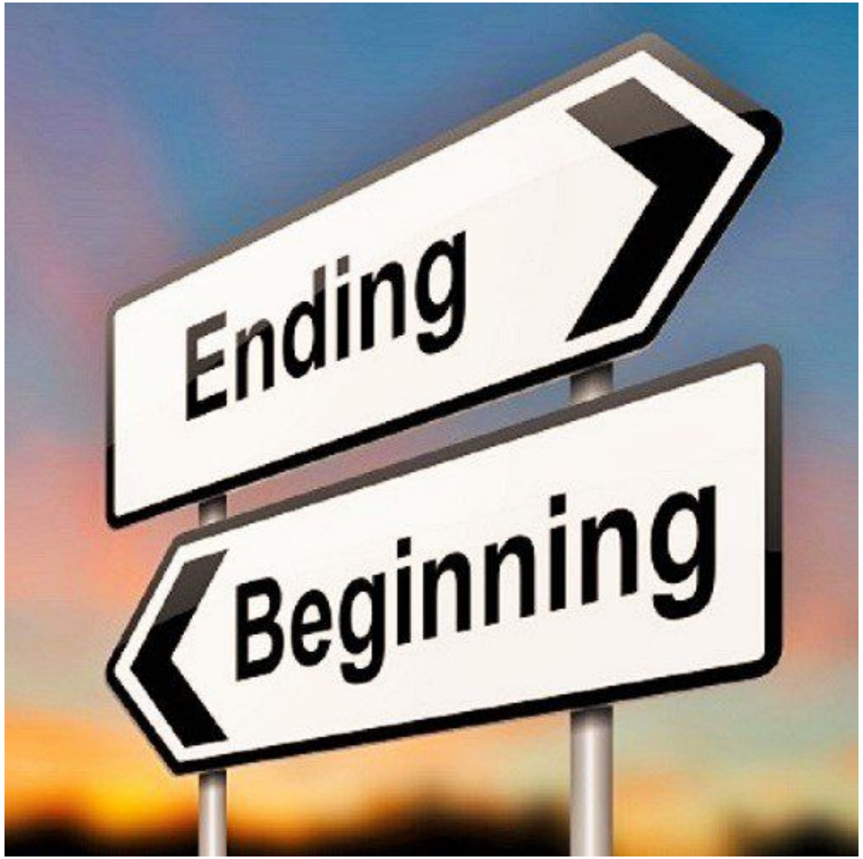
- and you’re stuck.
- and you need to hit a million buttons to reproduce the problem.
- and you’re frustrated.

What next?

*Repr*oducible *Ex*amples (reprexes)

Two approaches to reprexes in Shiny

- Peel away complexity from the current app
    - Video example by Hadley Wickham
    - A similar Twitch stream I recorded
- Start from scratch and progressively add code



Overall Goal

Reproduce the bug with the absolute minimum...

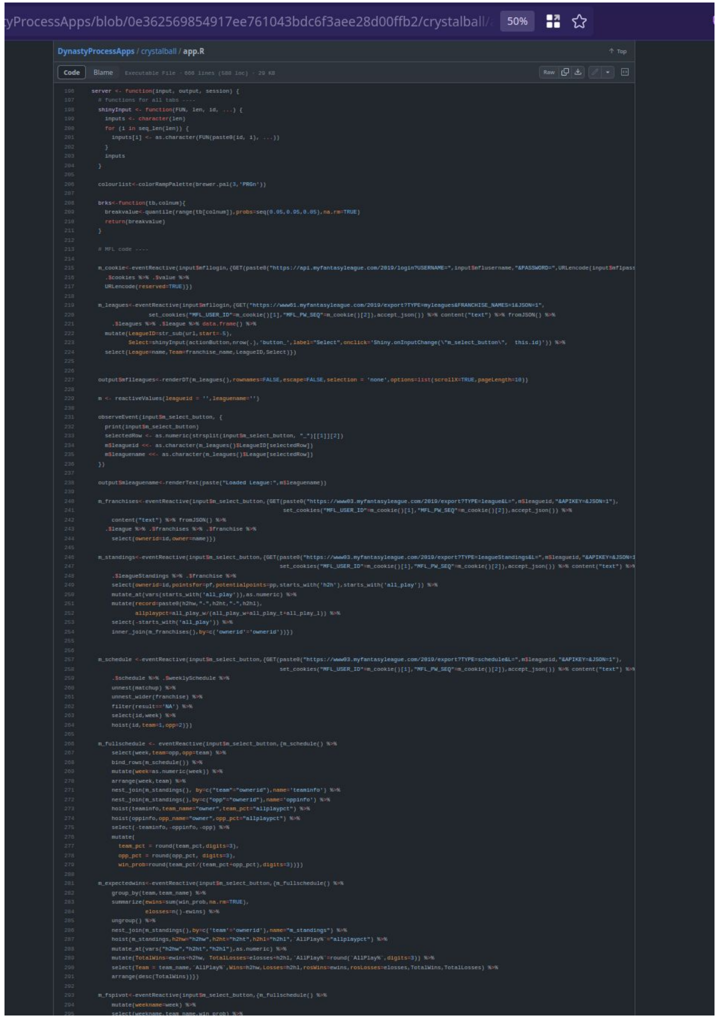
- dependency packages
- lines of code
- context/domain knowledge involved

So that...

- a solution (maybe) becomes evident
- you can isolate the cause of the problem
- you can start asking for help!

Shiny is NOT where the magic happens

FUNCTIONS are where the magic happens



tyProcessApps/blob/0e362569854917ee761043bdc6f3aee28d00ffb2/crystalball/

50%

DynastyProcessApps / crystalball / app.R

Top

Code Blame Executable File · 666 lines (588 loc) · 29 KB

Raw

```r
server <- function(input, output, session) {
  # functions for all tabs ----
  shinyInput <- function(FUN, len, id, ...) {
    inputs <- character(len)
    for (i in seq_len(len)) {
      inputs[i] <- as.character(FUN(paste0(id, i), ...))
    }
    inputs
  }

  colourlist<-colorRampPalette(brewer.pal(3,'PRGn'))

  brks<-function(tb,colnum){
    breakvalue<-quantile(range(tb[colnum]),probs=seq(0.05,0.95,0.05),na.rm=TRUE)
    return(breakvalue)
  }

  # MFL code ----

  m_cookie<-eventReactive(input$mfllogin,{GET(paste0("https://api.myfantasyleague.com/2019/login?USERNAME=",input$mflusername,"&PASSWORD=",URLencode(input$mflpass
    .$cookies %>% .$value %>%
    URLencode(reserved=TRUE)})

  m_leagues<-eventReactive(input$mfllogin,{GET("https://www61.myfantasyleague.com/2019/export?TYPE=myleagues&FRANCHISE_NAMES=1&JSON=1",
                set_cookies("MFL_USER_ID"=m_cookie()[1],"MFL_PW_SEQ"=m_cookie()[2]),accept_json()) %>% content("text") %>% fromJSON() %>%
      .$leagues %>% .$league %>% data.frame() %>%
    mutate(LeagueID=str_sub(url,start=-5),
           Select=shinyInput(actionButton,nrow(.),'button_',label="Select",onclick='Shiny.onInputChange(\"m_select_button\",  this.id)')) %>%
    select(League=name,Team=franchise_name,LeagueID,Select)})


  output$mflleagues<-renderDT(m_leagues(),rownames=FALSE,escape=FALSE,selection = 'none',options=list(scrollX=TRUE,pageLength=10))

  m <- reactiveValues(leagueid = '',leaguename='')

  observeEvent(input$m_select_button, {
    print(input$m_select_button)
    selectedRow <- as.numeric(strsplit(input$m_select_button, "_")[[1]][2])
    m$leagueid <<- as.character(m_leagues()$LeagueID[selectedRow])
    m$leaguename <<- as.character(m_leagues()$League[selectedRow])
  })

  output$mleaguename<-renderText(paste("Loaded League:",m$leaguename))

  m_franchises<-eventReactive(input$m_select_button,{GET(paste0("https://www03.myfantasyleague.com/2019/export?TYPE=league&L=",m$leagueid,"&APIKEY=&JSON=1"),
                                                        set_cookies("MFL_USER_ID"=m_cookie()[1],"MFL_PW_SEQ"=m_cookie()[2]),accept_json()) %>%
      content("text") %>% fromJSON() %>%
    .$league %>% .$franchises %>% .$franchise %>%
      select(ownerid=id,owner=name)})

  m_standings<-eventReactive(input$m_select_button,{GET(paste0("https://www03.myfantasyleague.com/2019/export?TYPE=leagueStandings&L=",m$leagueid,"&APIKEY=&JSON=1
                                                       set_cookies("MFL_USER_ID"=m_cookie()[1],"MFL_PW_SEQ"=m_cookie()[2]),accept_json()) %>% content("text") %>
      .$leagueStandings %>% .$franchise %>%
      select(ownerid=id,pointsfor=pf,potentialpoints=pp,starts_with('h2h'),starts_with('all_play')) %>%
      mutate_at(vars(starts_with('all_play')),as.numeric) %>%
      mutate(record=paste0(h2hw,"-",h2ht,"-",h2hl),
             allplaypct=all_play_w/(all_play_w+all_play_t+all_play_l)) %>%
      select(-starts_with('all_play')) %>%
      inner_join(m_franchises(),by=c('ownerid'='ownerid'))})


  m_schedule <-eventReactive(input$m_select_button,{GET(paste0("https://www03.myfantasyleague.com/2019/export?TYPE=schedule&L=",m$leagueid,"&APIKEY=&JSON=1"),
                                                       set_cookies("MFL_USER_ID"=m_cookie()[1],"MFL_PW_SEQ"=m_cookie()[2]),accept_json()) %>% content("text") %>
      .$schedule %>% .$weeklySchedule %>%
      unnest(matchup) %>%
      unnest_wider(franchise) %>%
      filter(result=='NA') %>%
      select(id,week) %>%
      hoist(id,team=1,opp=2)})

  m_fullschedule <- eventReactive(input$m_select_button,{m_schedule() %>%
      select(week,team=opp,opp=team) %>%
      bind_rows(m_schedule()) %>%
      mutate(week=as.numeric(week)) %>%
      arrange(week,team) %>%
      nest_join(m_standings(), by=c("team"="ownerid"),name='teaminfo') %>%
      nest_join(m_standings(),by=c("opp"="ownerid"),name='oppinfo') %>%
      hoist(teaminfo,team_name="owner",team_pct="allplaypct") %>%
      hoist(oppinfo,opp_name="owner",opp_pct="allplaypct") %>%
      select(-teaminfo,-oppinfo,-opp) %>%
      mutate(
        team_pct = round(team_pct,digits=3),
        opp_pct = round(opp_pct, digits=3),
        win_prob=round(team_pct/(team_pct+opp_pct),digits=3))})

  m_expectedwins<-eventReactive(input$m_select_button,{m_fullschedule() %>%
      group_by(team,team_name) %>%
      summarize(ewins=sum(win_prob,na.rm=TRUE),
                elosses=n()-ewins) %>%
      ungroup() %>%
      nest_join(m_standings(),by=c('team'='ownerid'),name="m_standings") %>%
      hoist(m_standings,h2hw="h2hw",h2ht="h2ht",h2hl="h2hl",`AllPlay%`="allplaypct") %>%
      mutate_at(vars("h2hw","h2ht","h2hl"),as.numeric) %>%
      mutate(TotalWins=ewins+h2hw, TotalLosses=elosses+h2hl,`AllPlay%`=round(`AllPlay%`,digits=3)) %>%
      select(Team = team_name,`AllPlay%`,Wins=h2hw,Losses=h2hl,rosWins=ewins,rosLosses=elosses,TotalWins,TotalLosses) %>%
      arrange(desc(TotalWins))})

  m_fspivot<-eventReactive(input$m_select_button,{m_fullschedule() %>%
      mutate(weekname=week) %>%
      select(weekname,team_name,win_prob) %>%
```

A flashback to my first-ever Shiny app: DynastyProcess Crystal Ball

(and this is just the first 200ish lines of the server function!)

DP Crystal Ball, as explained to a human

A fantasy football app that:

- Logs on to the user’s league via API
- Downloads the current standings and the remaining schedule
- Determines relative strengths of each team based on standings
- Creates probability of winning the remaining games on the schedule
- Returns to the user a table with the expected wins for the rest of the season and where the model thinks they’ll finish at the end of the year

How much of this logic NEEDS Shiny?

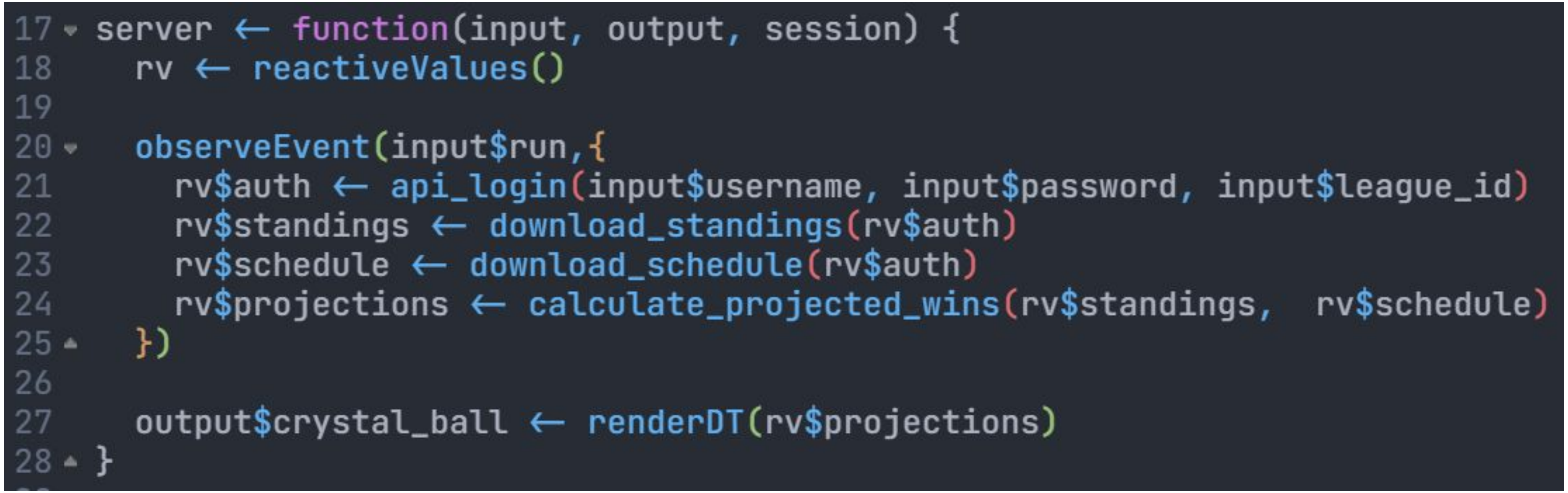
Shiny

- The user supplies their username, password, and league ID
- The user receives the output projections

Not Shiny

- Login to API
- Download standings
- Download schedule
- Do Fancy Maths

How I’d write this app today



```r
server ← function(input, output, session) {
  rv ← reactiveValues()

  observeEvent(input$run,{
    rv$auth ← api_login(input$username, input$password, input$league_id)
    rv$standings ← download_standings(rv$auth)
    rv$schedule ← download_schedule(rv$auth)
    rv$projections ← calculate_projected_wins(rv$standings,  rv$schedule)
  })

  output$crystal_ball ← renderDT(rv$projections)
}
```

Benefits of moving logic to functions

- Work iteratively in your normal code workflow
- Abstract into well-named, self-contained parts
- Add argument checks and helpful errors
- Write unit tests to ensure that logic works consistently

Keep the Shiny Simple - pass the inputs to your functions, and present the outputs!

Takeaways

Three Debugging Tools

- Investigate problem in context with **browser()**
- Create a **reprex** to drill down to relevant parts
- Split out the business logic into **functions**

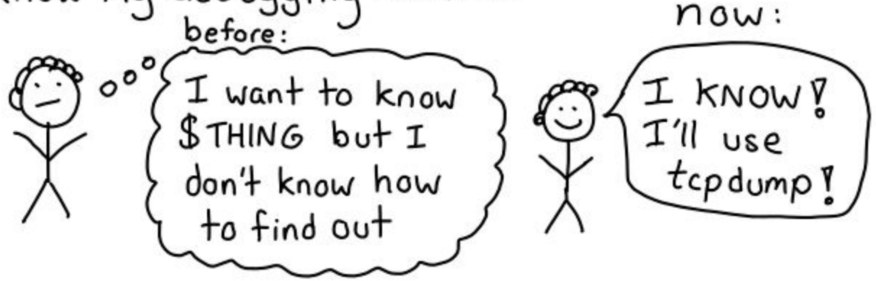
how I got better at debugging

Remember: the bug is happening for a logical reason.

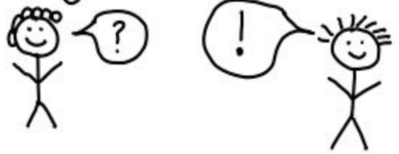
It's never magic. Really. Even when it makes no sense.

Be confident I can fix it

know my debugging toolkit


before:
I want to know $THING but I don't know how to find out
now:
I KNOW! I'll use tcpdump!

Talk to my coworkers



most importantly: I learned to like it

before:
oh no! a bug!
I think I'm about to learn something
facial expression: determination

JULIA EVANS
@b0rk

https://wizardzines.com/comics/better-at-debugging/

Thank you!

@_TanHo
fosstodon.org/@TanHo

Resources

- A debugging manifesto - Julia Evans
- Making a minimal reprex for a Shiny app - Hadley Wickham
- Advanced R: Interactive Debugging
- Mastering Shiny: Debugging
- What Everyone in Tech Should Know About Teaching and Learning - Greg Wilson
- R4DS Slack Community