



# Troubleshooting



Mine Çetinkaya-Rundel

@minebocek   
mine-cetinkaya-rundel   
mine@rstudio.com 

**Writing robust code**

# Writing robust code

- ▶ Complexity is the problem; abstraction is the solution
  - ▶ Software programs are far too large to reason about in their entirety
  - ▶ Good programs are broken into fragments that you can reason about locally, and compose reliably
  - ▶ In other words, we break the program into simple fragments, and if we verify that each fragment is correct, then the whole program is correct
- ▶ **Are our fragments simple enough to understand?**
- ▶ **Do they compose reliably?**

# Understandable fragments

- ▶ Indent your code! (Ctrl+I/Cmd+I)
- ▶ Extract out complicated processing logic (as opposed to UI logic) into top-level functions so you can test them separately
- ▶ Each function, reactive, observer, or module should be small, and do one thing
  - ▶ Function/reactive/observer bodies that don't fit on a single screen is a bad code smell
  - ▶ If you're having trouble giving something a meaningful name, maybe it's doing too much
- ▶ When you encounter unavoidable complexity, at least try to firewall the complexity behind as simple/straightforward an API as possible
  - ▶ Even if it's hard to verify if the scary piece itself is correct, it's still easy to verify that its callers are correct

# Reliable composition

- ▶ Prefer "pure functions" — functions without side effects. Much less likely to surprise you.
- ▶ When you do need side effects, don't put them in surprising places. Consider following command-query separation — "asking a question should not change the answer"
- ▶ Reactive expressions must not have side effects
- ▶ Avoid observers and reactive values, where possible; use reactive expressions if you can help it
- ▶ Don't pass around environments and reactive values objects; this is similar to sharing global variables, which is always good to avoid
- ▶ For ease of reasoning, prefer: pure functional > reactive > imperative (observers)

# Debugging tools

# Standard R debugging tools

- ▶ Tracing
  - ▶ `print()/cat()/str()`
  - ▶ `renderPrint` eats messages, must use `cat(file = stderr(), ...)`
  - ▶ Also consider `shinyjs` package's `logjs`, which puts messages in the browser's JavaScript console
- ▶ Debugger
  - ▶ Set breakpoints in RStudio
  - ▶ `browser()`
  - ▶ Conditionals: `if (!is.null(input$x)) browser()`
- ▶ Learn more: Jenny Bryan | Object of type 'closure' is not subsettable - [youtu.be/vgYS-F8opgE](https://youtu.be/vgYS-F8opgE)

# Shiny debugging tools

**Symptom:** Outputs or observers don't execute when expected, or execute too often

- Reactlog
  - Restart R process
  - Set `options(shiny.reactlog = TRUE)`
  - In the browser, Ctrl+F3 (or Cmd+F3)
  - Learn more: [rstudio.github.io/reactlog](https://rstudio.github.io/reactlog)
- Showcase mode:
  - DESCRIPTION file
  - `runApp(display.mode = "showcase")`
  - Learn more: [shiny.rstudio.com/articles/display-modes.html](https://shiny.rstudio.com/articles/display-modes.html)



# Shiny debugging tools

**Symptom:** Red error messages in the UI or session abruptly terminates

- ▶ This means an R error has occurred
- ▶ Look in R console for stack traces
  - ▶ By default, Shiny hides "internal" stack traces. Use `options(shiny.fullstacktrace = TRUE)` if necessary to show.
- ▶ Shiny/Shiny Server "sanitize" errors, for security reasons (every error message is displayed as "An error has occurred")
- ▶ Learn more: [shiny.rstudio.com/articles/sanitize-errors.html](https://shiny.rstudio.com/articles/sanitize-errors.html)

# Shiny debugging tools

**Symptom:** Server logic seems OK, but unexpected/broken/missing results in browser

- Check browser's JavaScript console for errors
- Listen in on conversation between client and server
  - `options(shiny.trace = TRUE)` logs messages in the R console
  - Use Chrome's Network tab to show individual websocket messages

# Your turn

- ▶ Open **05-troubleshooting/01-troubleshoot/app.R**. It is broken in a not-very-subtle way. See if you can find and fix the bug.
- ▶ Continue on for **05-troubleshooting/02-troubleshoot.R** and **05-troubleshooting/03-troubleshoot/app.R**.

10<sub>m</sub> 00<sub>s</sub>



# Common errors

# Common errors

"Object of type 'closure' is not subsettable"

- You forgot to use () when retrieving a value from a reactive expression  
`plot(userData)` should be `plot(userData())`

# Common errors

"Unexpected symbol"

"Argument xxx is missing, with no default"

- ▶ Missing or extra comma in UI.
- ▶ Sometimes Shiny will realise this and give you a hint, or use RStudio editor margin diagnostics.

# Common errors

"Operation not allowed without an active reactive context.  
(You tried to do something that can only be done from  
inside a reactive expression or observer.)"

- ▶ Tried to access an input or reactive expression from directly inside the server function. You must use a reactive expression or observer instead.
- ▶ Or if you really only care about the value of that input at the time that the session starts, then use `isolate()`.

# Testing



# Why automated testing with Shiny?

- ▶ There are many possible reasons for an application to stop working. These reasons include:
  - ▶ An upgraded R package has different behavior. (This could include Shiny itself!)
  - ▶ You make modifications to your application.
  - ▶ An external data source stops working, or returns data in a changed format.
- ▶ Automated tests can alert you to these kinds of problems quickly and with almost zero effort, after the tests have been created.


# shinytest

- ▶ Shinytest uses snapshot-based testing strategy.
- ▶ The first time it runs a set of tests for an application, it performs some scripted interactions with the app and takes one or more snapshots of the application's state.
- ▶ These snapshots are saved to disk so that future runs of the tests can compare their results to them.
- ▶ Learn more: Mastering Shiny, Chapter 21. [mastering-shiny.org/scaling-testing.html](https://mastering-shiny.org/scaling-testing.html)

# Troubleshooting



Mine Çetinkaya-Rundel

@minebocek   
mine-cetinkaya-rundel   
mine@rstudio.com 