





# {shinytest2}

Unit testing for Shiny applications

Barret Schloerke

RStudio / Shiny Team

  @schloerke

# Raise a hand🙋 if...

You have written a  
Shiny App





**Raise a hand 🙋 if...**

You remember all of  
your App's features

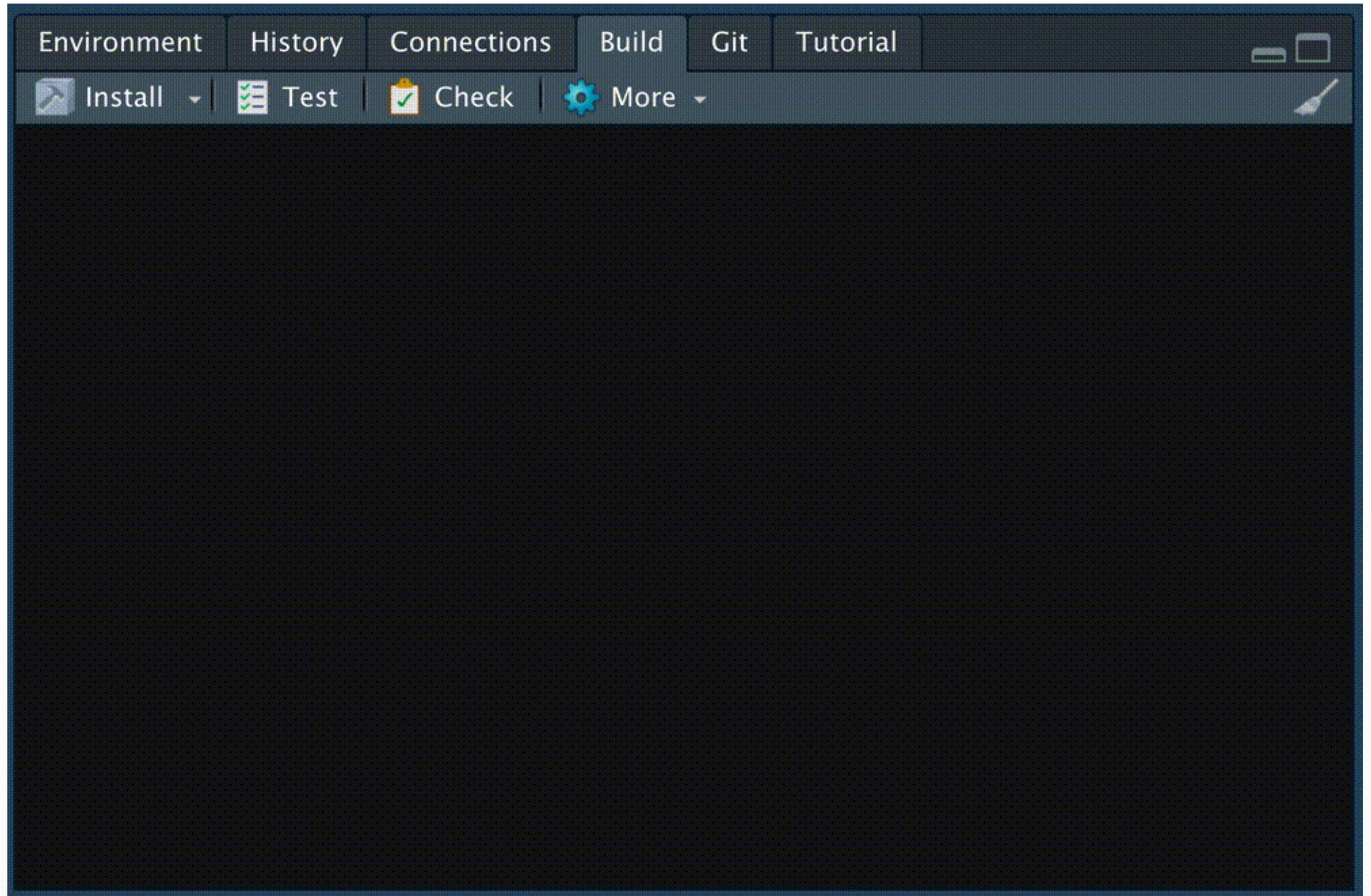


# Raise a hand 🙋 if...

You would feel  
comfortable having a  
coworker rewrite  
part of your App







5x playback speed







- Quickly build full-stack interactive web apps
- Executes reactive expressions
  - When a reactive value is changed...  
All downstream reactive values are invalidated and re-calculated



# *All too familiar* workflow

1. Add / adjust some reactivity
2. Click "Run App"
3. Manually experiment with new feature to see if it works
4. Rinse and repeat

# *All too familiar* workflow

1. Add / adjust some reactivity
2. Click "Run App"
3. **Manually experiment** with new feature to see if it works
4. Rinse and repeat









- Unit Testing for R
- Executes **formal** unit tests
- Goals
  - Fewer bugs
  - Robust code
  - Better code structure
  - Call to action

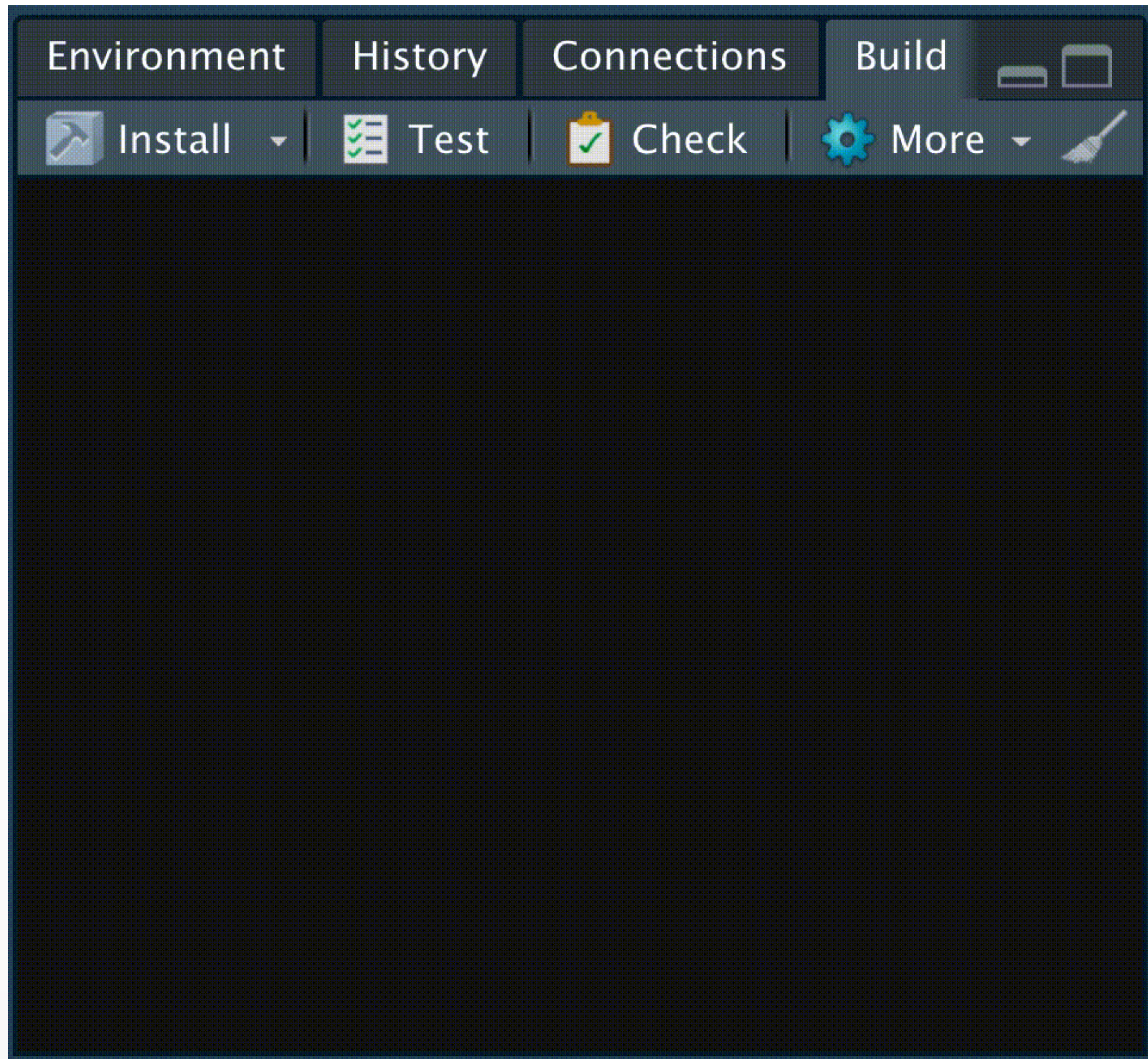


```
# File: tests/testthat/test-example.R

test_that("examples work", {
  expect_equal(2 * 2, 4)
  expect_equal(2 * NA, NA_integer_)

  ex_file <- make_file()
  expect_snapshot_file(ex_file)
})
```









- **Unit testing** for Shiny applications






- **Regression testing** for Shiny applications
  - Testing that **existing App behavior** is consistent over time
- Executes within `{testthat}` tests
  - `{shinytest2}` expectation methods leverage `testthat::expect_snapshot_file()`
- Built on `{chromote}`
  - View live test App in Chrome browser
  - Familiar debugging tools in Chrome browser





- What about `{shinytest}`?
  - Entering **maintenance** mode
  - Headless browser reached **End-Of-Life** 👻
  - Not compatible with Bootstrap v5+
- `{shinytest2}`
  - Different headless browser 
  - Different file API
  - Different R API





127.0.0.1:5608

127.0.0.1:5608

What is your name?

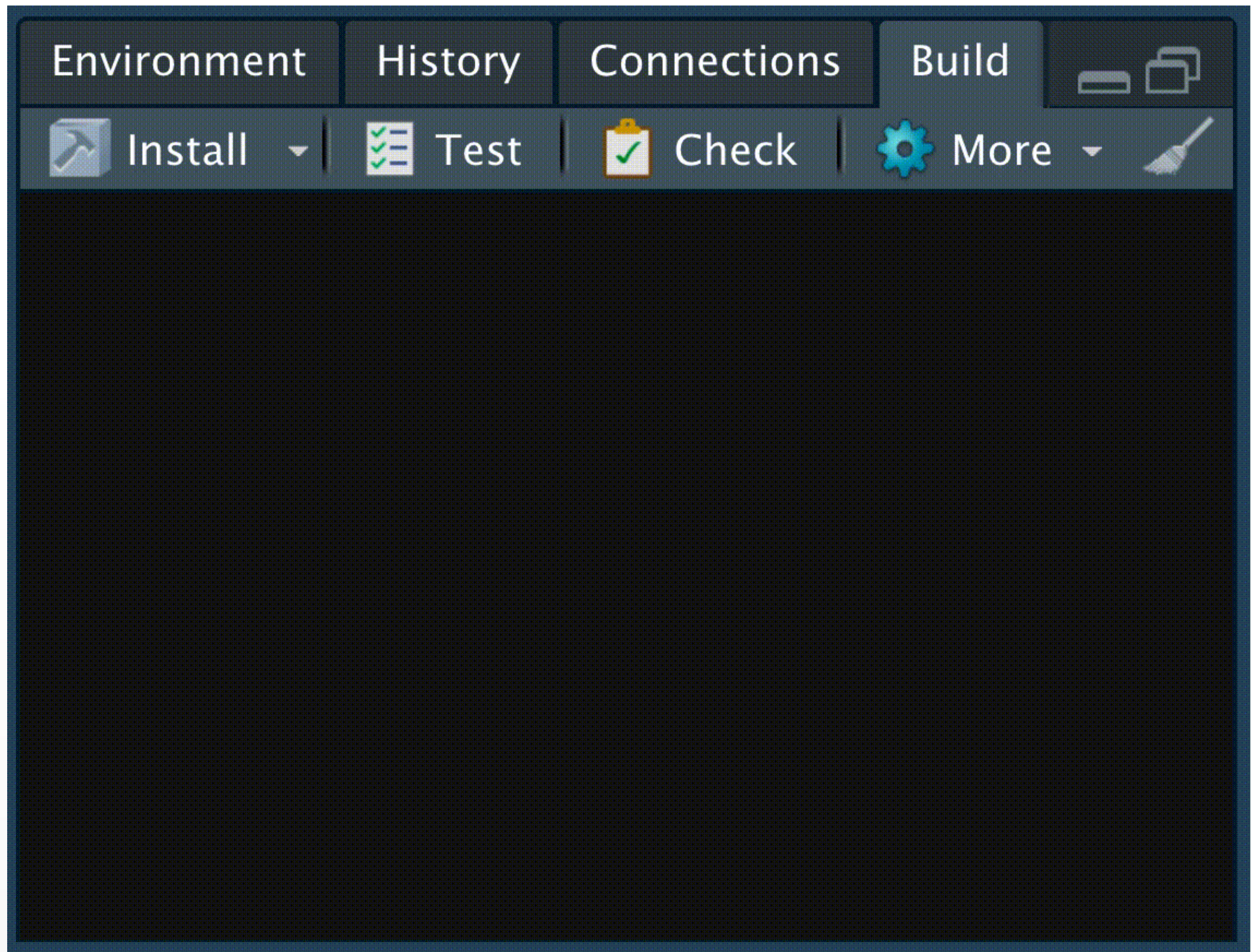
Greet



```
# File: tests/testthat/test-hello.R
library(shinytest2)

test_that("App says hello Barret", {
  app <- AppDriver$new(name = "say-hello")
  app$set_inputs(name = "Barret")
  app$click("greet")
  app$expect_values()
})
```









“But writing tests is hard!”  
- Everyone except Hadley





“But writing tests is hard!”

- Everyone except Hadley

- `record_test()`
- Captures all **Shiny interactions** as a `{testthat} test`
- “If you have time to ~~test~~,  
you have time to ~~record~~ a test”






127.0.0.1:5608


127.0.0.1:5608

What is your name?

Greet

{shinytest2} expectations

 Expect Shiny values ?

 Expect screenshot ?

At least one expectation must be made


Code

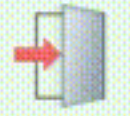
1 | app\$set\_window\_size(width = 341, height = 403)

Save

Test name:  ?

Random seed:  ?

 Exit

 Save test and exit





```
# File: tests/testthat/test-shinytest2.R
library(shinytest2)

test_that("{shinytest2} recording: hello", {
  app <- AppDriver$new(name = "hello")
  app$set_inputs(name = "Barret")
  app$click("greet")
  app$expect_values()
})
```





```
# File  
library  
  
test_th  
  app  
  app$  
  app$  
  app$  
})
```

Environment | History | Connections | Build | Tutorial

Install | Test | Check | More

R

o", {





```
# File: tests/testthat/test-shinytest2.R
library(shinytest2)

test_that("{shinytest2} recording: hello", {
  app <- AppDriver$new(name = "hello")

  app$view()

  app$set_inputs(name = "Barret")
  app$click("greet")
  app$expect_values()
})
```





New Tab

Search Google or type a URL

Search Google or type a URL

127.0.0.1

presentation-2022-07-28-rstudioconf22-shinytest2 - RStudio

test-shinytest2.R

Run Tests

```
1 library(shinytest2)
2 test_that("{shinytest2} recording: hello", {
3   app <- AppDriver$new(name = "hello", height =
4   app$view()
5   app$set_inputs(name = "Barret")
6   app$click("greet")
7   app$expect_values()
8 })
```

1:1 (Top Level) R Script

Console Terminal Render Background Jobs

R 4.1.3 · ~/Documents/git/presentations/presentation-2022-07-28-  
RESTARTING R SESSION...

>



# Suggestions...



- Use `shiny::exportTestValues()`
  - Limit testing to objects under your control
- Initial snapshot file contents **must be verified** by hand
  - Use explicit expectations when possible (`expect_equal()`)
- Minimize number of screenshot expectations
  - **Brittle tests** and **cannot compare screenshots** across Operating Systems or R versions



RECAP

- Regression testing for Shiny Apps
- *“If you have time to rest,  
you have time to record a test”*
- Interactively `$view()` your live App





# `{shinytest2}`:

## Regression testing for Shiny applications

- Website: <https://rstudio.github.io/shinytest2/>
  - *Getting Started* and many more articles!
- Slides: TODO

Barret Schloerke  
RStudio / Shiny Team

  @schloerke



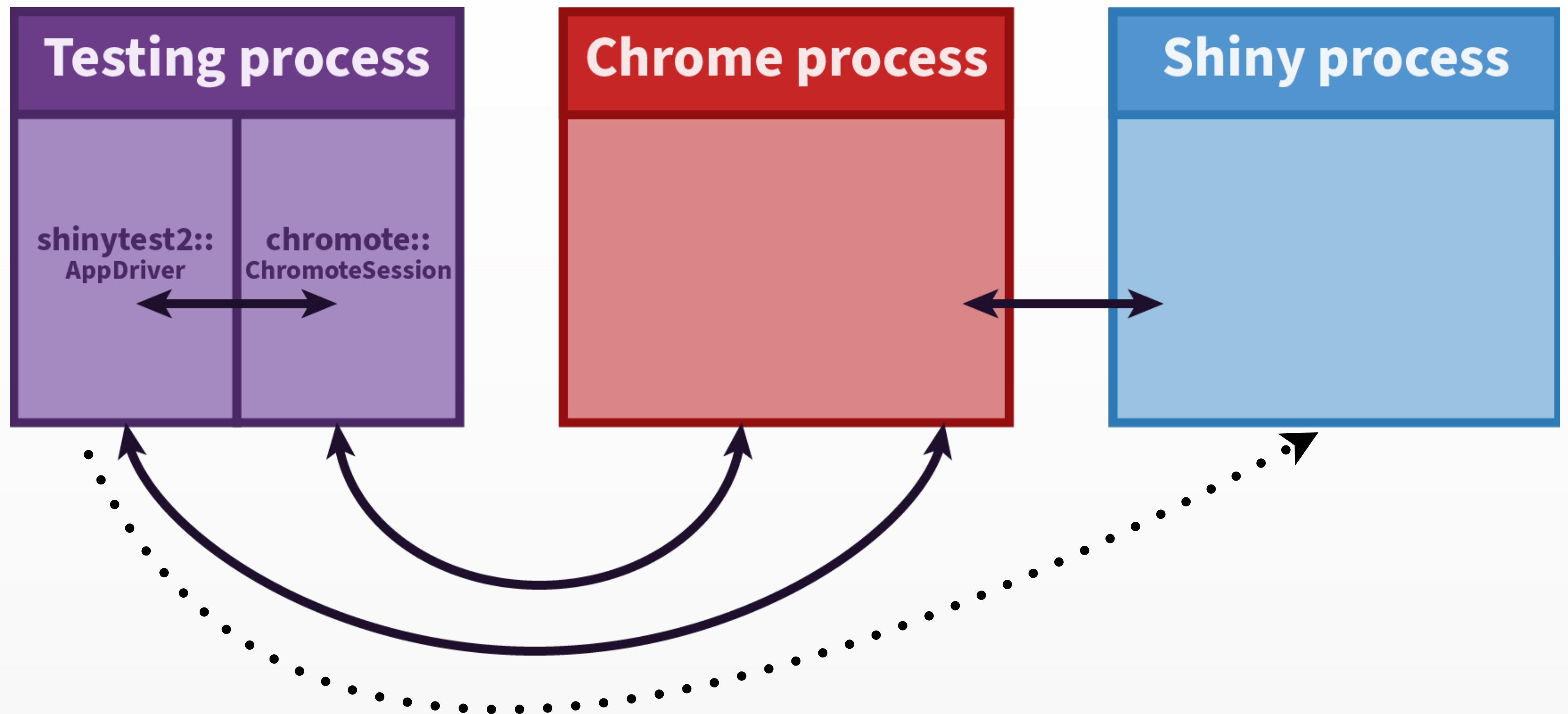
# Questions?

```
# File: ./tests/testthat/test-questions.R
library(shinytest2)

test_that("Questions are answered", {
  app <- AppDriver$new(name = "questions")
  app$set_inputs(ready = TRUE)
  app$click("ask")
  if (interactive()) app$view()
  expect_true(
    app$get_value(export = "has_answer")
  )
})
```



How does {shinytest2}  
orchestrate everything?



# Shiny App development

- Core
  - `{shinyuieditor}`: Visual tool for organizing Shiny UI
  - `{bslib}`: Custom Bootstrap Sass themes for {shiny} and {rmarkdown}
  - `{shinytest2}`: Unit testing for R Shiny Apps
- Debug
  - `{profvis}`: R Shiny App profiler
  - `{reactlog}`: R Shiny reactivity visualizer
- Performance
  - `shiny::bindCache()` & `{cachem}`: Cache and store any reactive output
  - `{promises}` & `{future}`: Async code execution
- Admin
  - `{shinyloadtest}`: Load testing for R Shiny Apps
- Extra credit
  - `{plumber}`: R web API
  - `{shinymeta}`: Expose Shiny app logic using meta-programming



# Shiny performance workflow

From Joe Cheng's Shiny in Production keynote...

1. Use `{shinyloadtest}` to see if it's fast enough
2. If not, use `{profvis}` to see what's making it slow
3. Optimize
  - a. Move work outside of `{shiny}` (very often)
  - b. Make code faster (very often)
  - c. Use caching (sometimes)
  - d. Use `async` (occasionally)
4. Repeat!