

Things to Remember

- Shiny outputs that are not visible, are not calculated. . . This is on purpose!

(Effective Reactive Programming - Joe Cheng @ Shiny DevCon 2016 “Part 1” @46:34)

- If new reactive objects are being created for **every** user interaction, you may have coded an “anti-solution”
(Shiny DevCon 2016 “Part 1”@17:00)
 - Will cause `reactlog` graph to become **very large**
 - Typically caused by calling `render*()` within `observe*()`
 - Tease out values using `reactive()` and render the reactive value

- `reactLog` is **not** a *performance* debugger. `reactLog` is a *reactivity* debugger

- Use `profvis` for *performance analysis*

- **Do not keep when deploying to production**

```
options(show.reactlog == TRUE)
```

Things to Remember

- Shiny outputs that are not visible, are not calculated. ...This is on purpose!
([Effective Reactive Programming - Joe Cheng @ Shiny DevCon 2016 “Part 1”@46:34](#))
- If new reactive objects are being created for **every** user interaction, you may have coded an “anti-solution”
([Shiny DevCon 2016 “Part 1”@17:00](#))
 - Will cause `reactlog` graph to become **very large**
 - Typically caused by calling `render*()` within `observe*()`
 - Tease out values using `reactive()` and render the reactive value
- `reactlog` is **not** a *performance* debugger. `reactlog` is a *reactivity* debugger
 - Use [profvis](#) for *performance analysis*
- Do **not** keep `options(show.reactlog = TRUE)` when deploying to production

Future Ideas