# Reagent

Minimalistic React for ClojureScript

# About Us

Dmitri Sotnikov

- Lead Developer @ UHN
- https://yogthos.net/
- https://github.com/yogthos

Scot Brown

- Senior Software Developer @ UHN
- GitHub: https://github.com/SVMBrown

# Why ClojureScript?

- Simple Syntax
- Fast
- Immutable by default
- Hotloading
- REPL driven workflow
- Auto code pruning
- Auto minification
- Seamless JS interop

# Syntax

JSX

- DSL with  its own syntax
- Requires preprocessing
- Verbose

S-Expressions

- Plain data
- Concise
- Structural editing

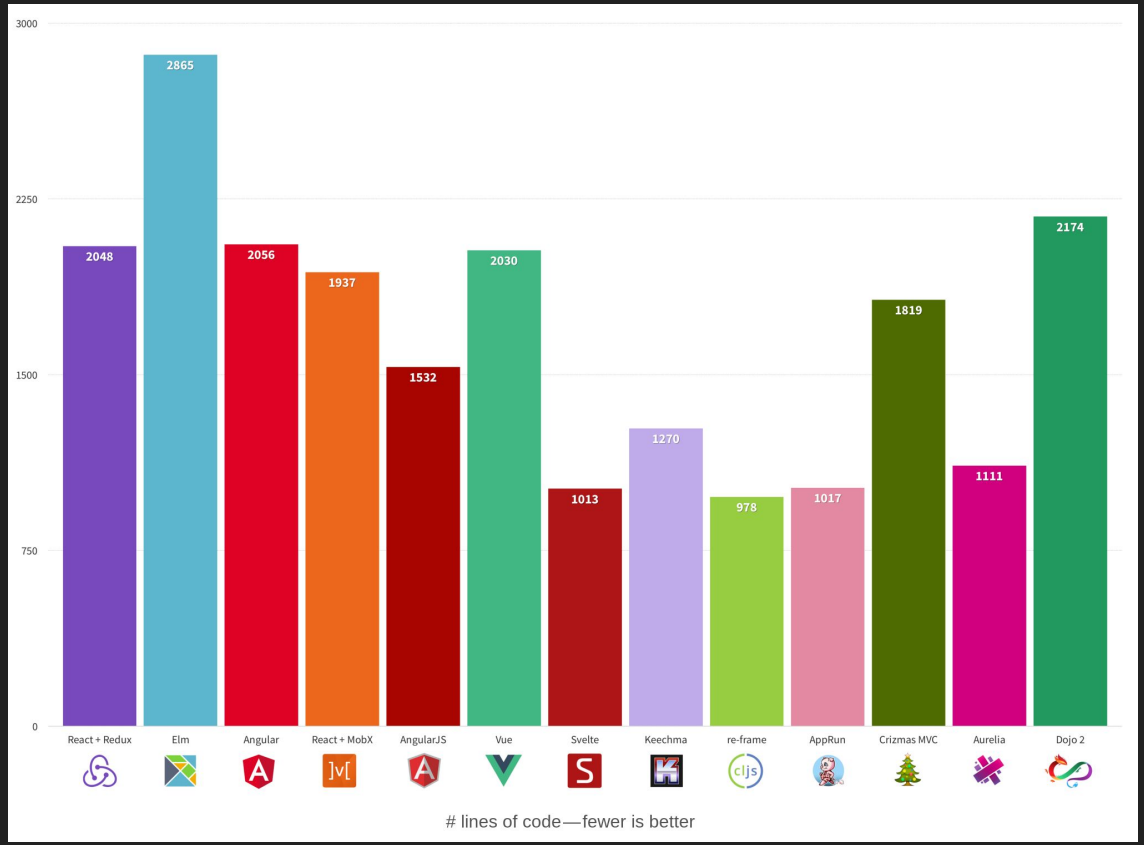
# Comparison of Modern Web Frameworks

- Comparison of minimal real-world applications
- Holistic comparison with practical metrics
  - Lines of Code
  - First meaningful render (ms)
  - Transfer size (KB)

# Lines of Code

Re-Frame: 978
React+Redux: 2048

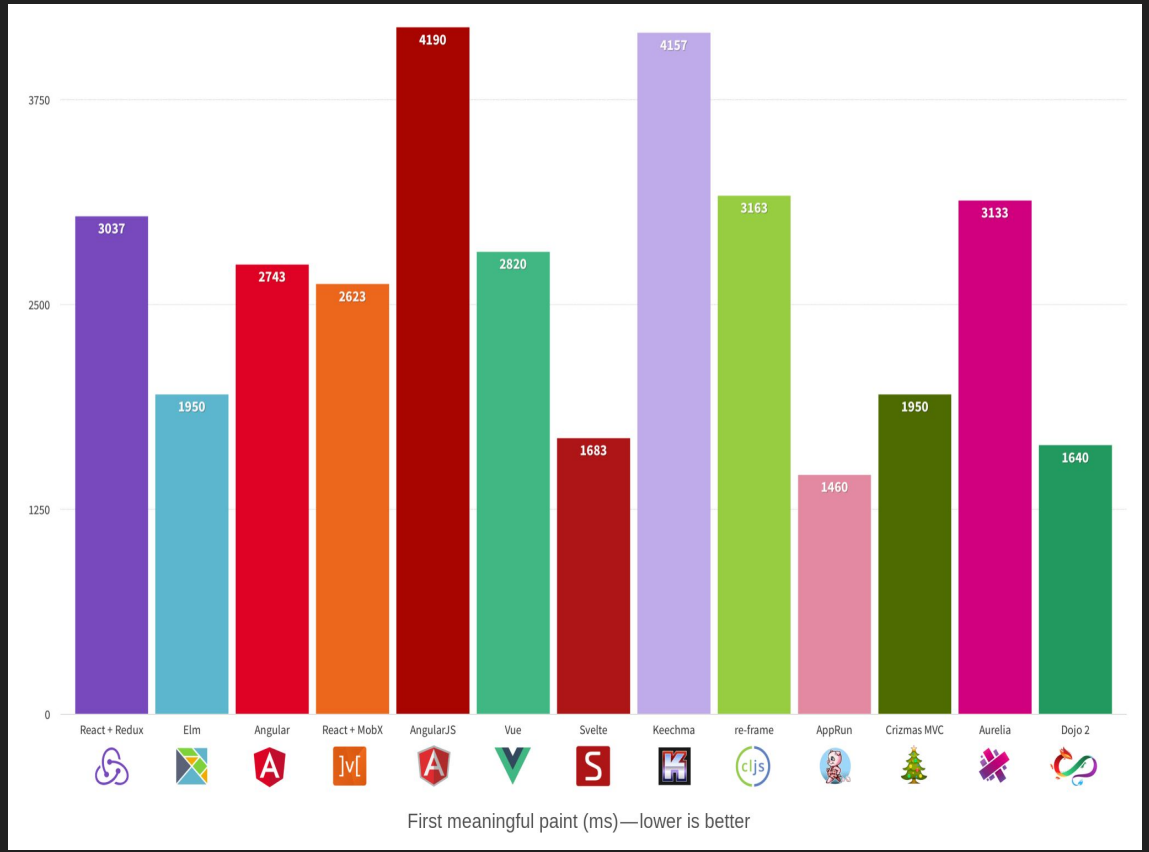
- S-expressions are concise
- Most of lifecycle is automatic



# Performance

Re-Frame: 3163
React+Redux: 3037

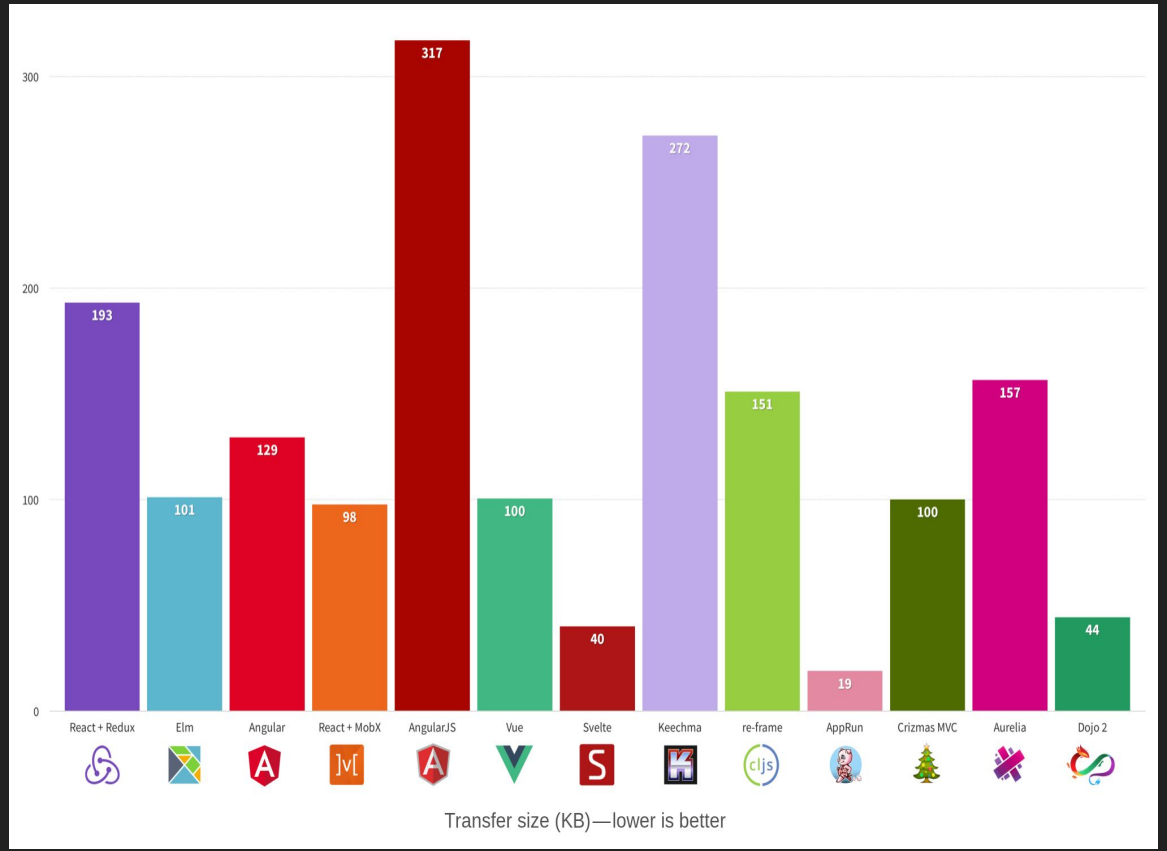
- Both frameworks can render server-side to mitigate this
- Reagent can beat React on repaints due to immutability



# Transfer Size

Re-Frame: 151
React+Redux: 193

- Google Closure compiler
- Dead code pruning
- Modules



# Live Demo

- https://github.com/yogthos/reagent-talk

# Links

Reagent

https://reagent-project.github.io/

Re-Frame

https://github.com/Day8/re-frame

ClojureScript FAQ

https://github.com/clojure/clojurescript/wiki/FAQ-(for-JavaScript-developers)#how-does-clojurescript-compare-to-the-newer-ecmascript-versions

Benchmarks

https://medium.freecodecamp.org/a-real-world-comparison-of-front-end-frameworks-with-benchmarks-2018-update-e5760fb4a962

https://github.com/gothinkster/realworld