Isomorphic web apps with Rum
by Nikita Prokopov
1985
not existing

2001
not programming

2005
studying programming

not programming Clojure :(

2012
programming Clojure

2015
programming ClojureScript

2016
speaking at ClojuTRE (we’re here)
HISTORY
Om

Centralized state atom
Asynchronous rendering
Leveraged immutability
React

May 2013

Om

Reagent

Jan 2014
Reagent

Reactive atoms

Hiccup syntax
Quiescent

Minimal

No state model

Pure functional, no OOP
IDEA
Rum motivation

Borrow all the good stuff
Mix Om, Reagent, Quiescent models
Be compatible with DataScript
Rum motivation

Borrow all the good stuff
Mix Om, Reagent, Quiescent and other models
Be compatible with DataScript any storage
Be future-proof
Solution

Not opinionated
Don’t commit to anything
Build for extension
Library, not a framework
No mandatory state model
Open, decomplexed architecture
Well-defined internals are treated as API
Clear Rum/React mapping, escape hatches
Minimal code base (~900 LOC)
IMPLEMENTATION
(rum/defc label [text class]
   [:div.lbl {:class class} text])

(rum/mount (label "Hello" "header")
   js/document.body)
Simpler than React

functions, not classes
arguments, not props
no this (duh)
DOM as data
(def mixin
  { :will-mount
    (fn [state]
      (assoc state :key (atom nil))) })

(rum/defc label < mixin [text class]
  [:div.label {:class class} text])
State

is immutable
lifecycle methods are pure
can’t update component directly
can’t close over a thing that will later change
Mixins

Pure, composable, no methods

Powerful & decomplexed:
- Update logic defined per-component
- Emulate Reagent, Quiescent/PureComponent, Om
- Local state is a mixin
SERVER-SIDE
(rum/defc label [text]
    [:.label text])

React.createClass({
    render: function() {
        return React.createElement("div",
            { class: "label" },
            this.props.text);
    }
});
(rum/defc label [text]
    [::{label text}])

(defn label [text]
    (str "<div class='label'>"
        text
        "</div>"))
Server-side rendering

Renders to a string
Use same components via CLJC
3× faster than Hiccup in interpreting mode
No state, no lifecycle
Compatible with React.js server-side renderer
CONCLUSION
What’s Rum good for?

Complex single-page apps with fine control
Custom/mixed state models
Server-side rendering and templating
What’s Rum bad for?

Doesn’t teach you how to write apps
Contacts

github.com/tonsksy/rum
#tonsksy/rum at gitter.im
#rum at clojurians.slack.com
@nikitonsksy