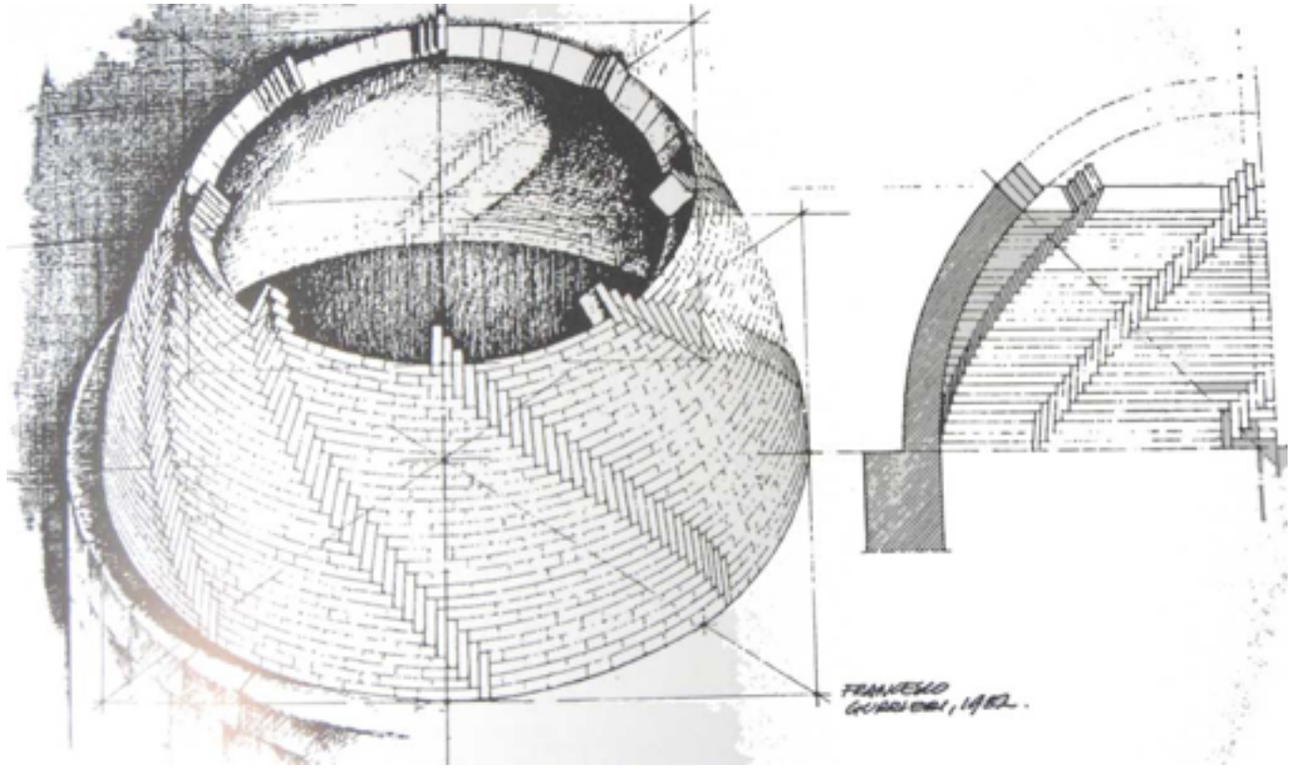


The Front End Architecture Revolution

GOTO Chicago 2015


cognitect

Live Coverage of Election Day

Americans went to the polls on Tuesday and Times reporters around the country will be providing live updates, analysis and results throughout the day.

HIGHLIGHTS

11:43 pm	**The Scene at Romney Headquarters**
10:18 pm	**Warren Wins in Massachusetts**
9:55 pm	**Mood Swings in Chicago**
9:26 pm	**Obama Wins Pennsylvania, Networks Project**
8:45 pm	**Exit Polls: Blaming Bush**

2:30 am That's a Wrap

The New York Times

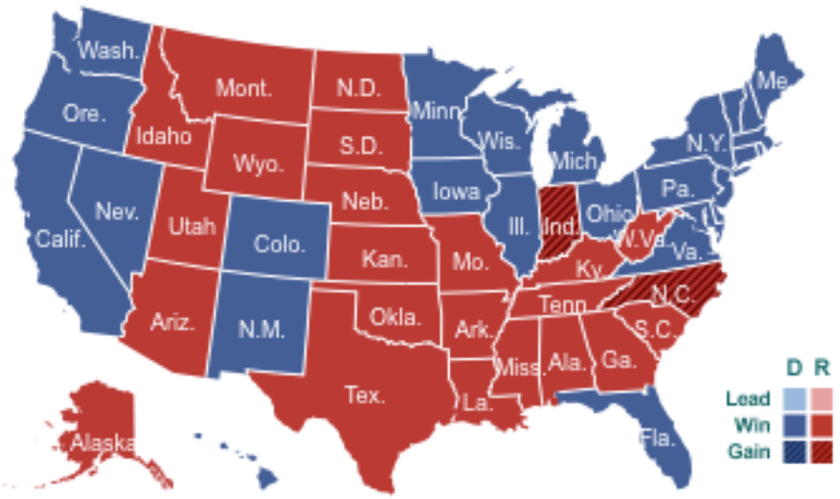
Damon Winter/The New York Times

President » Updated Nov. 29

332 Obama ✔ — **Romney 206**

270 to win

	Fla.	Ohio	N.C.	Va.	Wis.
Obama	✔ 50%	✔ 50%	48%	✔ 51%	✔ 53%
Romney	49%	48%	✔ 51%	48%	46%
Reporting	*100%*	*100%*	*100%*	*99%*	*99%*



Senate »

54 ✔ — 1 IND. — **45**

DEM. 50 REP.

Democrats gain 1 seat

Republicans need +4 for control

House »

201 — **233** ✔

DEM. 218 REP.

Democrats gain 8 seats

Democrats need +25 for control

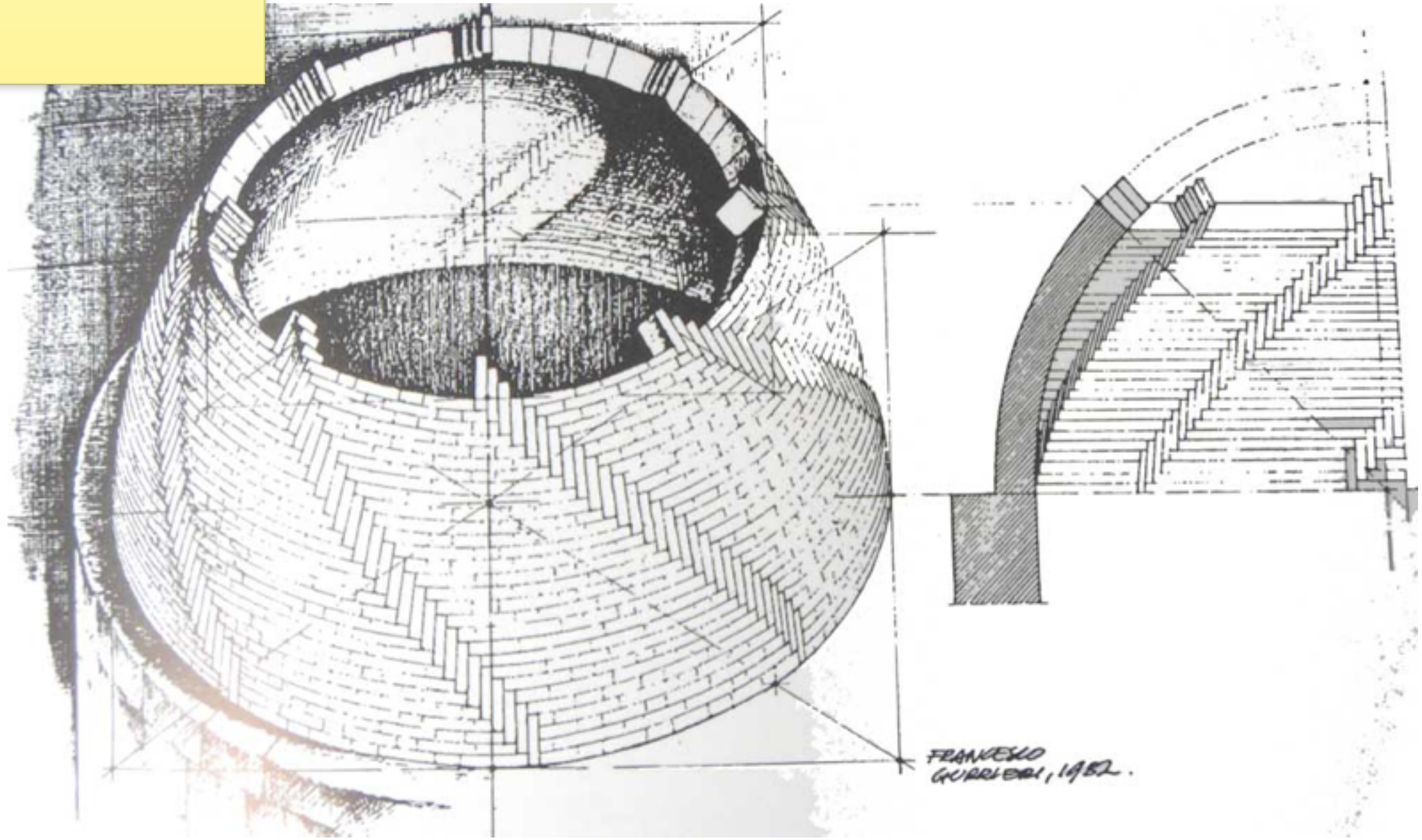

Haldane's
CONDENSED
PRIMORDIAL
SOUP

Simplicity

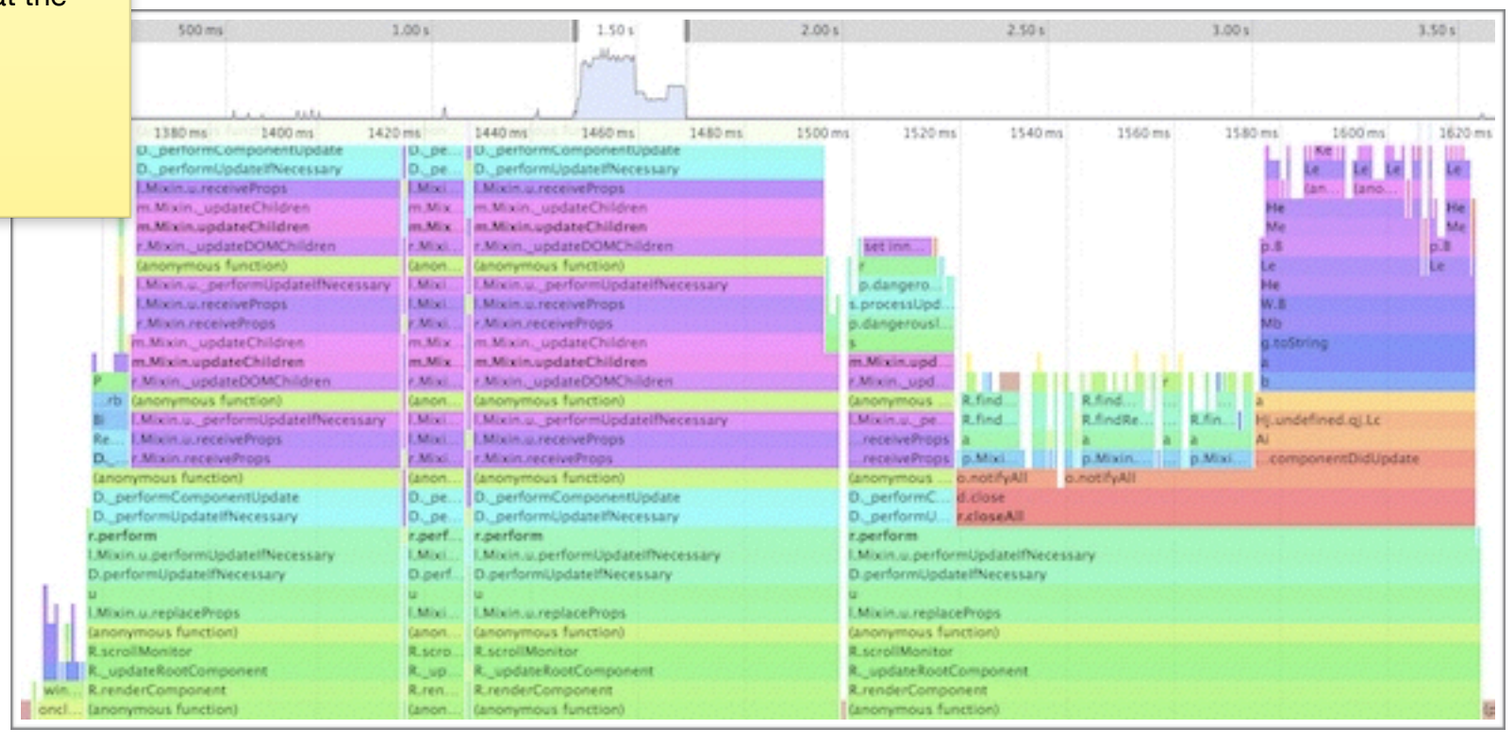
spina de pesce

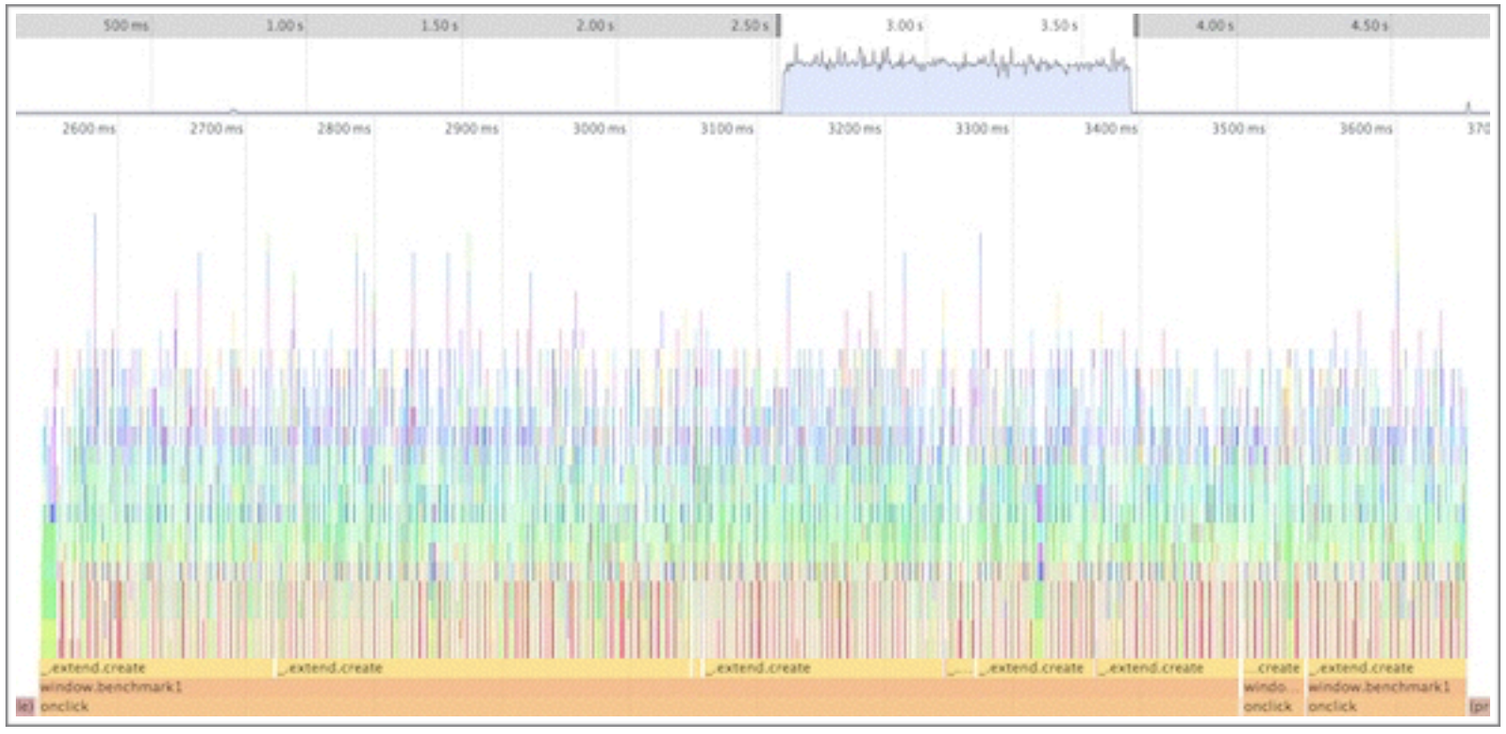
herring bone

brick!



many mainstream practice emphasize only the separation of concerns at the detriment of global coordination opportunities





- Simplicity scales
- Pervasive simplicity permits more opportunity for global optimization
- Question designs, tools, processes that don't lead to global optimization
- Global optimization is not at odds with modularity (Garbage Collection)

Possible Stack

- React / Relay
- ClojureScript (Google Closure)
- Transit
- Relay / Datomic

Support immutability at every layer

Benefits

- Enable simpler reasoning (which permits wider / deeper reasoning)
- Agility (remove needless coordination)
- Performance

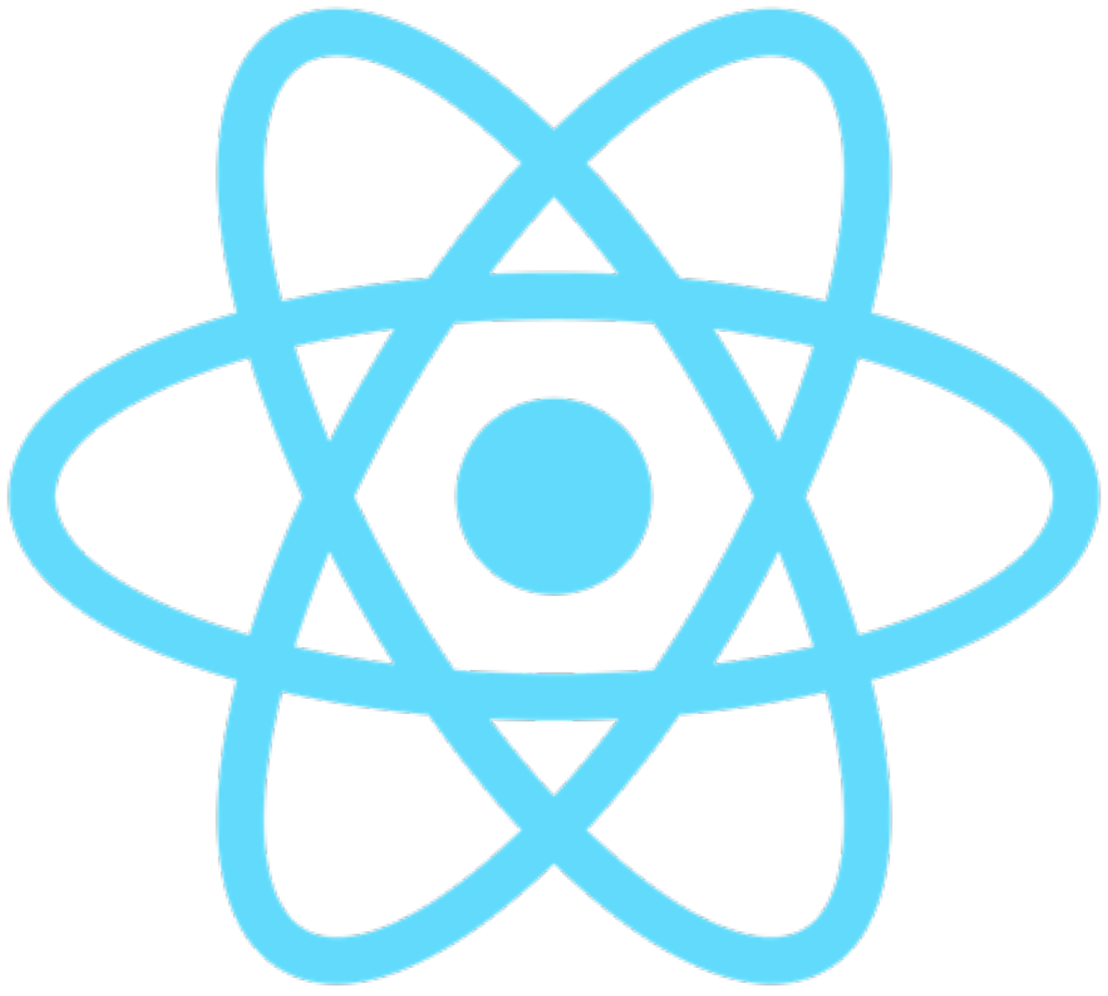
Client Layer


BACKBONE.JS


ember

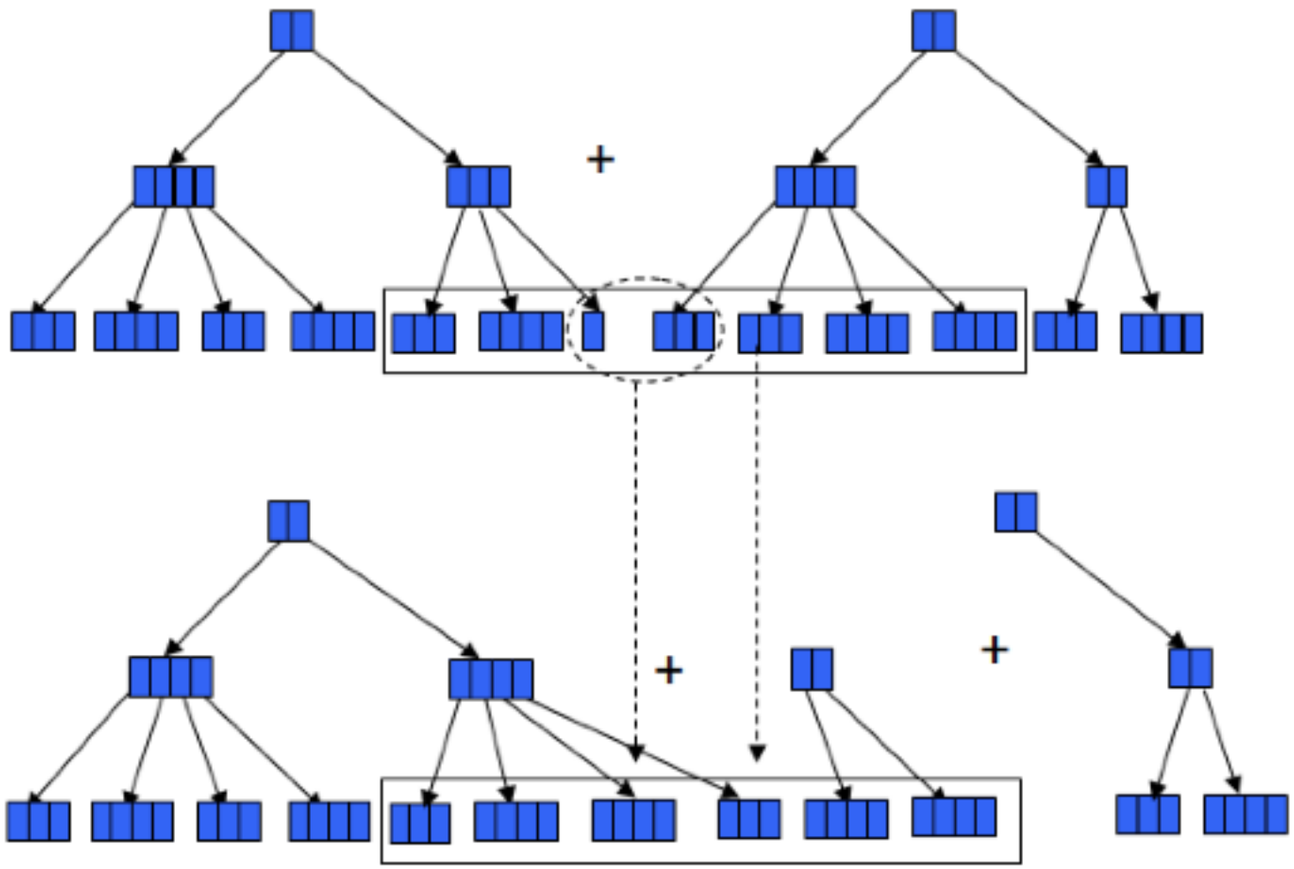
Forcing mutability is like forcing someone pick a database, this is just bad design

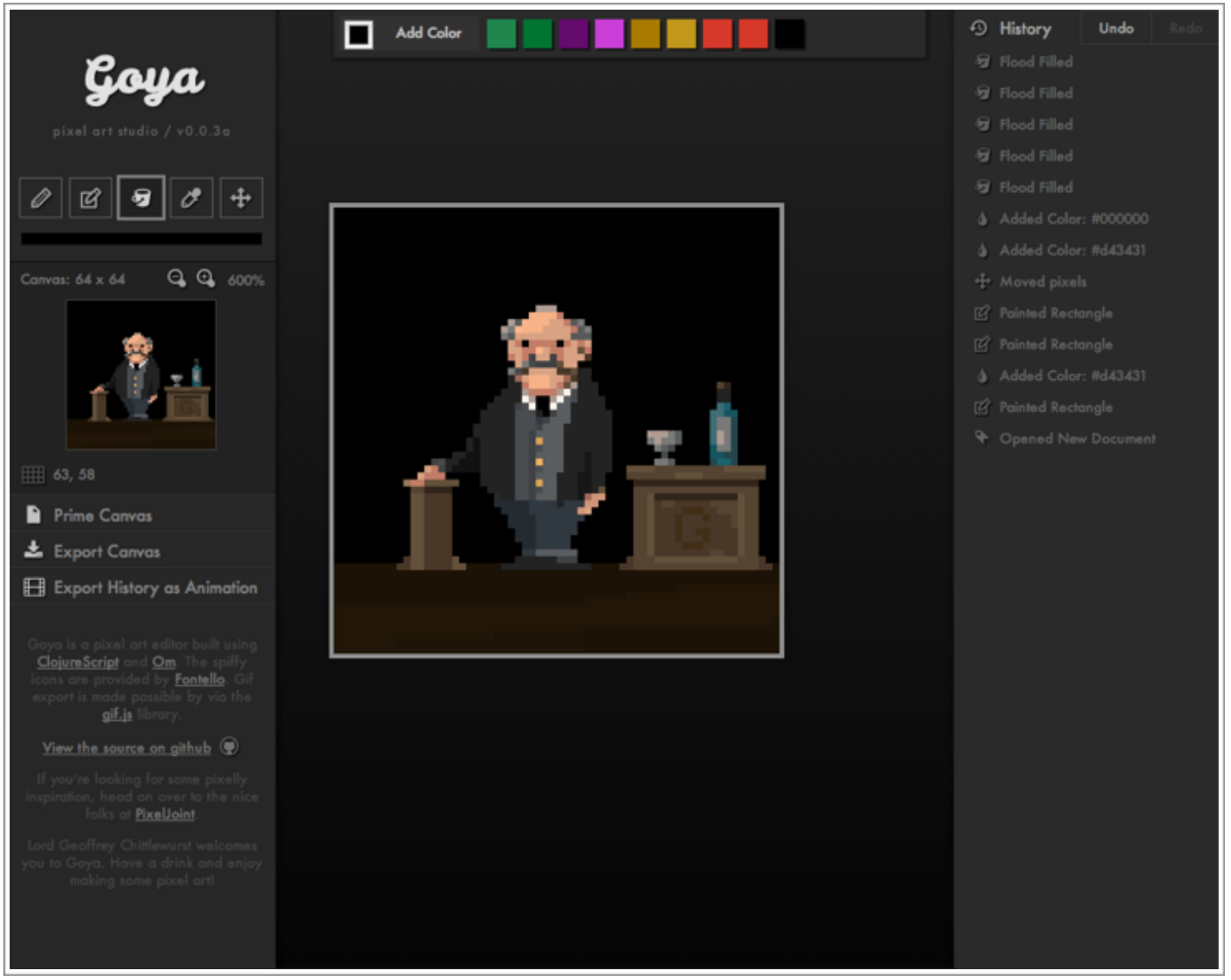
Mutability should be an implementation detail



Rule 5. Data dominates. If you've chosen the right data structures and organized things well, the algorithms will almost always be self evident. Data structures, not algorithms, are central to programming.

- Rob Pike




Goya
pixel art studio / v0.0.3a
Add Color
History
Undo
Redo
Flood Filled
Flood Filled
Flood Filled
Flood Filled
Flood Filled
Added Color: #000000
Added Color: #d43431
Moved pixels
Painted Rectangle
Painted Rectangle
Added Color: #d43431
Painted Rectangle
Opened New Document
Canvas: 64 x 64
600%
63, 58
Prime Canvas
Export Canvas
Export History as Animation
Goya is a pixel art editor built using ClojureScript and Om. The spiffy icons are provided by Fontello. Gif export is made possible by via the gif.js library.
View the source on github
If you're looking for some pixelly inspiration, head on over to the nice folks at PixelJoint.
Lord Geoffrey Chittlewurst welcomes you to Goya. Have a drink and enjoy making some pixel art!

branch: master · goya / src / cljs / goya / timemachine.cljs

swannodette 13 days ago Project layout refactor, better production settings

1 contributor

file 62 lines (41 sloc) 1.85 kb — Open | Edit | Raw | Blame | History | Delete

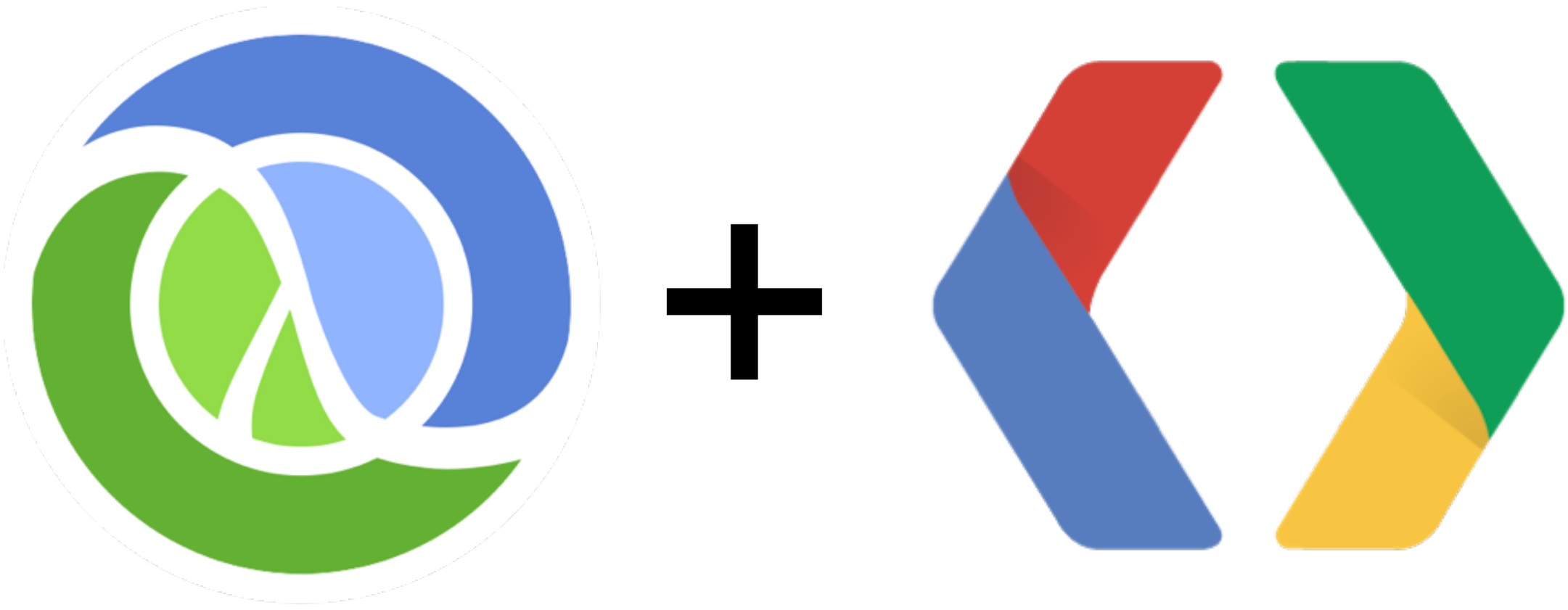
```
(ns goya.timemachine
        (:require [goya.appstate :as app]
            [goya.previewstate :as previewstate]))


;; ================================================================================
;; Credits to David Nolen's Time Travel blog post.

(def app-history (atom [(get-in @app/app-state [:main-app])]))
(def app-future (atom []))



;; ================================================================================

(defn update-preview []
  (reset! previewstate/preview-state
          (assoc-in @previewstate/preview-state [:main-app :image-data]
                    (get-in @app/app-state [:main-app :image-data]))))

(defn show-history-preview [idx]
  (reset! previewstate/preview-state
          (assoc-in @previewstate/preview-state [:main-app :image-data]
                    (get-in (nth @app-history idx) [:image-data]))))

(add-watch app/app-state :preview-watcher
  (fn [_ _ _ _] (update-preview)))


(defn undo-is-possible []
  (> (count @app-history) 1))

(defn redo-is-possible []
  (> (count @app-future) 0))


(defn push-onto-undo-stack [new-state]
  (let [old-watchable-app-state (last @app-history)]
    (when-not (= old-watchable-app-state new-state)
      (swap! app-history conj new-state))))


(defn do-undo []
  (when (undo-is-possible)
    (swap! app-future conj (last @app-history))
    (swap! app-history pop)
    (reset! app/app-state (assoc-in @app/app-state [:main-app] (last @app-history)))))

(defn do-redo []
  (when (redo-is-possible)
    (reset! app/app-state (assoc-in @app/app-state [:main-app] (last @app-future)))
    (push-onto-undo-stack (last @app-future))
    (swap! app-future pop)))


(defn handle-transaction [tx-data root-cursor]
  (when (= (:tag tx-data) :add-to-undo)
    (reset! app-future [])
    (let [new-state (get-in (:new-state tx-data) [:main-app])]
      (push-onto-undo-stack new-state))))
```

Which Language?

TypeScript

Scala

BABEL



ClojureScript

- Now industry leading experts on effective UI/UX over immutable data
  - React Native permits targeting all major platforms with one language
    - Without throwing out multi-threaded server side programs

Closure Compiler

- Whole program optimization
  - Dead Code Elimination
  - Optimal code splitting
    - Cross module code motion
- ES2015, CommonJS, AMD consumption

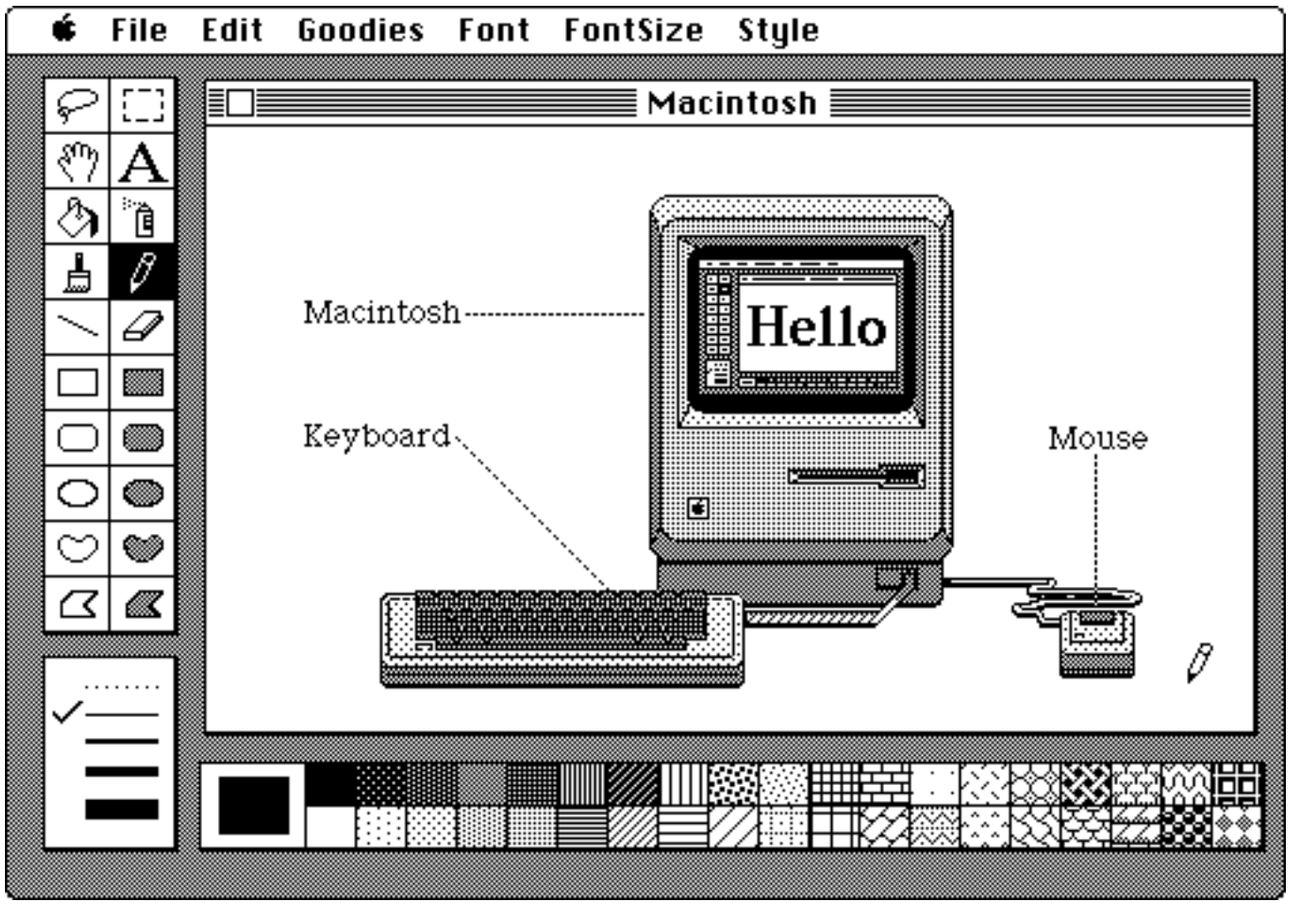
Moving Data

Transit

- Leverage JSON (pervasive)
- Provide richer types out of the box
  - Extensible
- In some cases can decode Transit payload into immutable data structures as fast as equivalent JSON

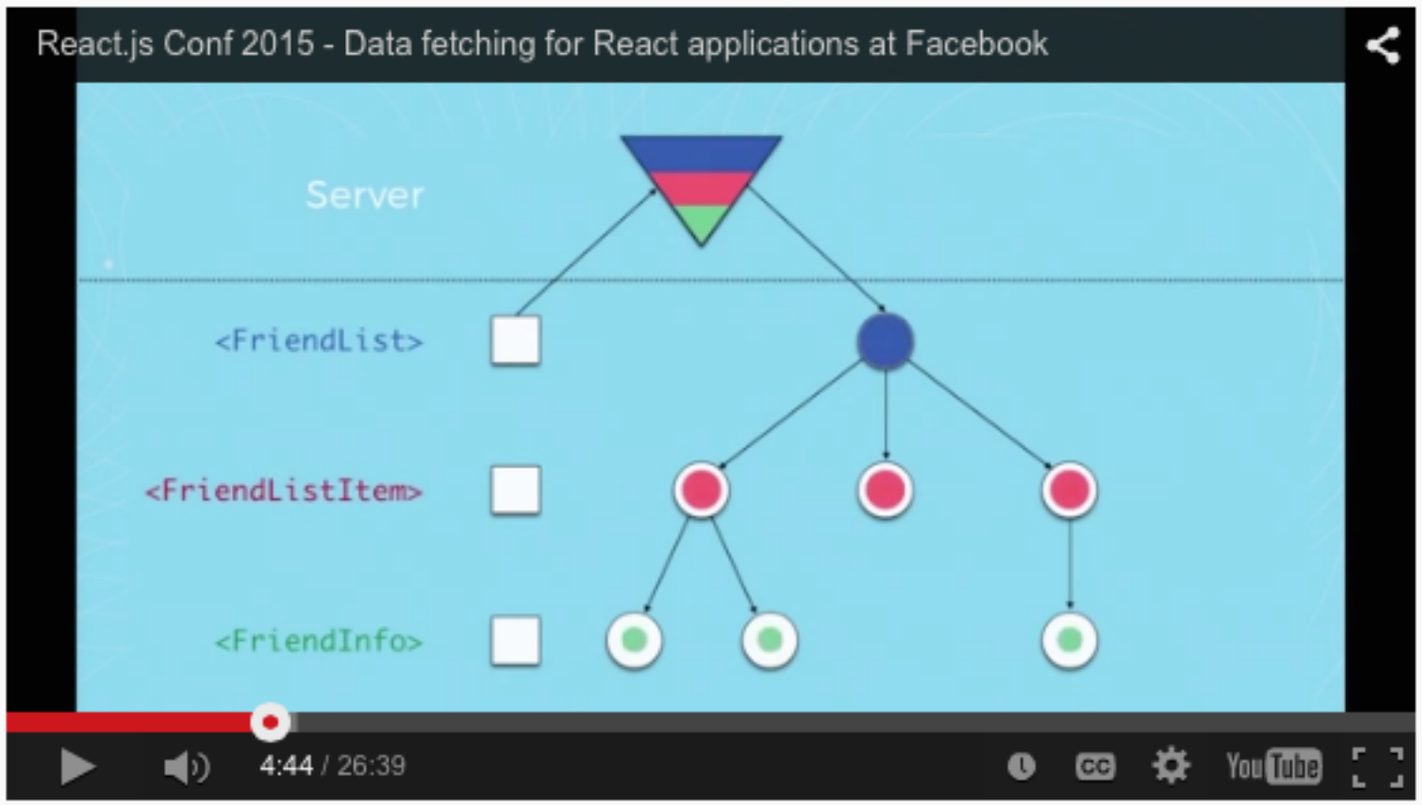
[["~#point",[1.5,2.5]],["~#cache",1],["^1",1]]

Server Side


File Edit Goodies Font FontSize Style
Macintosh
Hello
Macintosh
Keyboard
Mouse

Data fetching for React applications

There's more to building an application than creating a user interface. Data fetching is still a tricky problem, especially as applications become more complicated. At React.js Conf we announced two projects we've created at Facebook to make data fetching simple for developers, even as a product grows to include dozens of contributors and the application becomes as complex as Facebook itself.



The two projects — Relay and GraphQL — have been in use in production at Facebook for some time, and we're excited to be bringing them to the world as open source in the future. In the meantime, we wanted to share some additional information about the projects here.


Netflix had a RESTful API.
NETFLIX
2:09 / 54:57
Keynote - JSON Graph: Reactive REST at Netflix
TheOfficialACM
Subscribe
2,410
1,185
Add to
Share
More
11
0

This is the story of how

NETFLIX

eliminated 90% of the

networking code in our app.

Big Ideas

- UI components define what they need
  - Use a recursive description (JSON, EDN, etc)
- Batching

Datomic

- Immutable relational database
  - Powerful auditing capabilities
    - Efficient queries of arbitrary points in the past
- Datalog-style queries are themselves data (easy to compose)

```
[:artist/name :artist/startYear]
```

```
[{:release/media
  [{:medium/tracks
    [:track/name {:track/artists [:artist/name]}]}]}]}]
```

Radical Simplicity

- Revisit your assumptions / biases about any element of your stack that *creates* complexity
- Examine unfamiliar but time-tested ideas for complexity reduction

Questions?