State of the State

WTF IS REDUX?

FRONTEND

connect
mapStateToProps
dispatch

BACKEND

Reducer
Thunk
Saga
Observable
Immer

WTF WITH REDUX?

FRONTEND

mmm nothing wrong? Just use reselect.

BACKEND

ok-ish?

Immutable immutability

WHY ONE NEED REDUX?

THE CHANGE

To change the state in "predicable" way

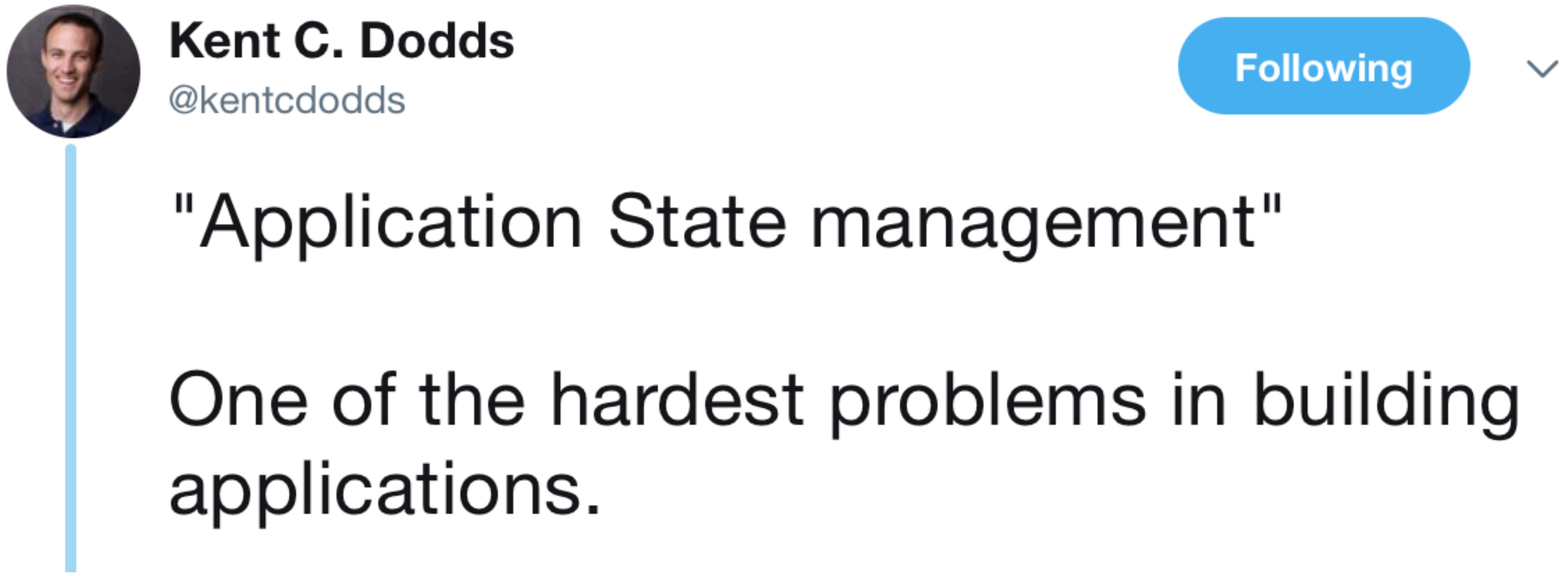
(as seen on TV)

TO FETCH

A global state is at your beck and call.

(single point of truth)

WTF THE STATE?

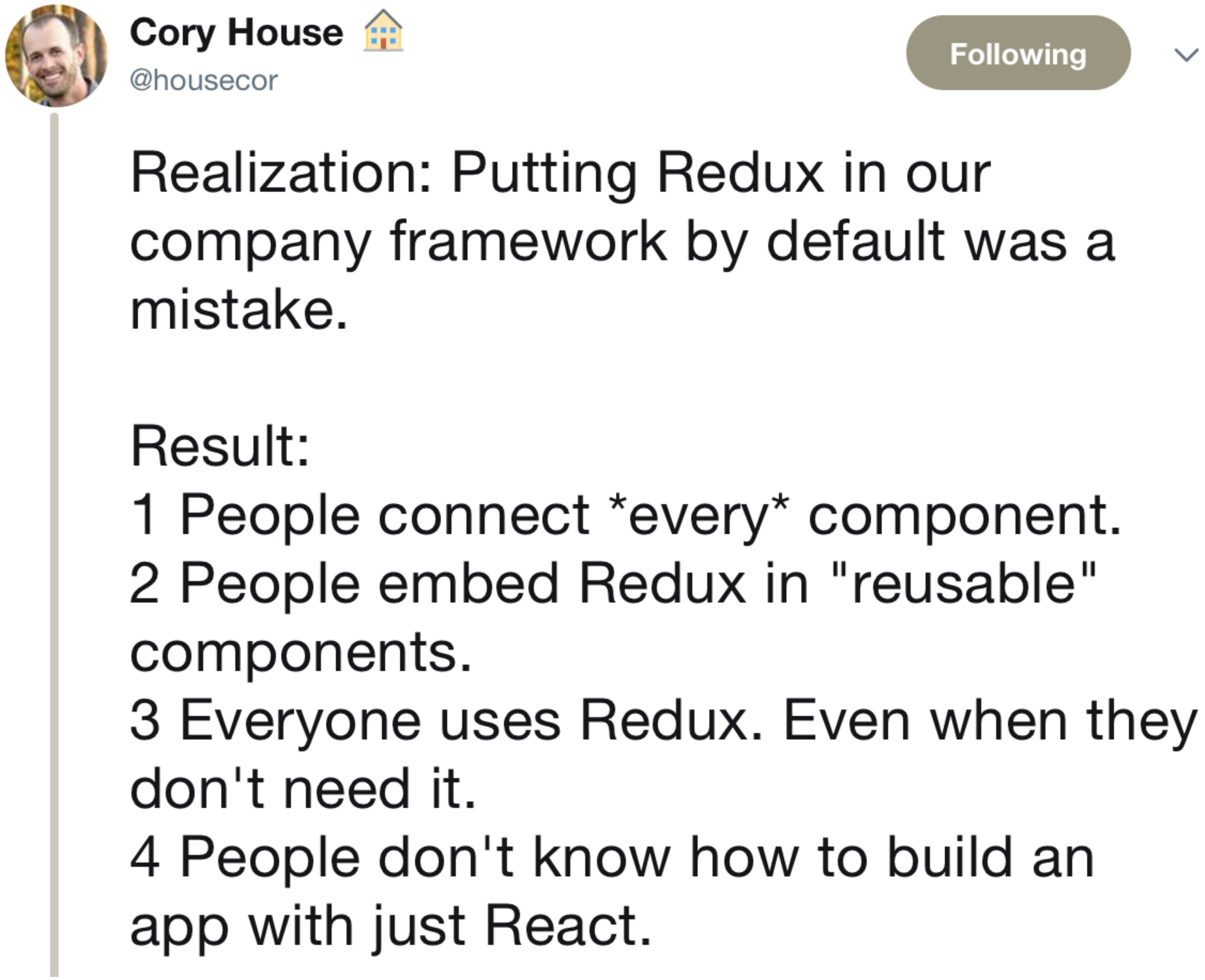


Kent C. Dodds
@kentcdodds

"Application State management"

One of the hardest problems in building applications.

OK! REDUX WILL SOLVE IT!



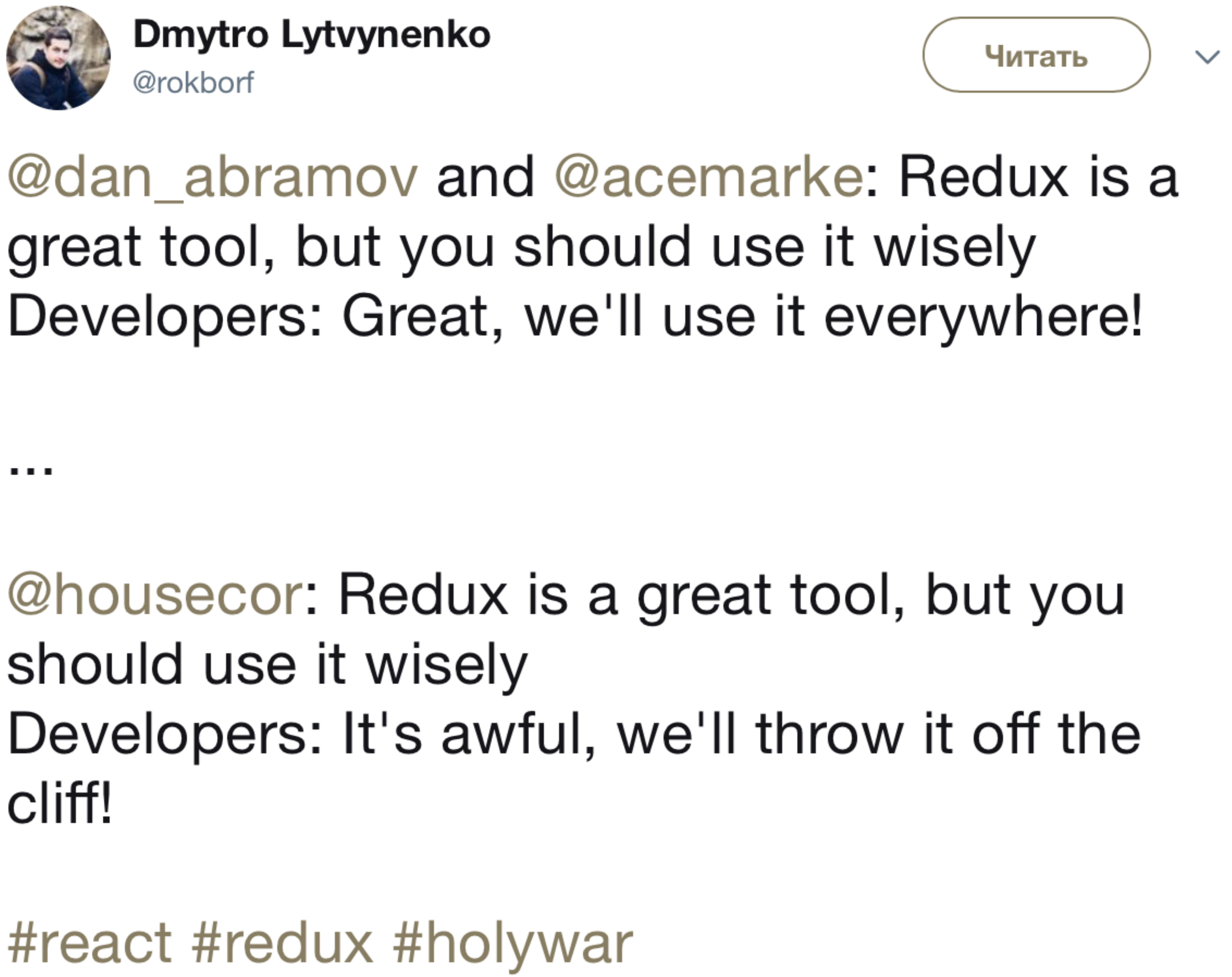
Cory House 🏠
@housecor

Realization: Putting Redux in our company framework by default was a mistake.

Result:
1 People connect *every* component.
2 People embed Redux in "reusable" components.
3 Everyone uses Redux. Even when they don't need it.
4 People don't know how to build an app with just React.

YOU DONT NEED REDUX

STATE != STATE MANAGEMENT



Dmytro Lytvynenko
@rokborf

Читать

@dan_abramov and @acemarke: Redux is a great tool, but you should use it wisely
Developers: Great, we'll use it everywhere!

...

@housecor: Redux is a great tool, but you should use it wisely
Developers: It's awful, we'll throw it off the cliff!

#react #redux #holywar

YOU SHALL NOT PUT EVERYTHING IN A SINGLE REDUX STORE

3 PILARS….

- ***Single source of truth:*** *The state of your whole application is stored in an object tree within a single store.*
- ***State is read-only:*** *The only way to change the state is to emit an action, an object describing what happened.*
- ***Changes are made with pure functions:*** *To specify how the state tree is transformed by actions, you write pure reducers.*

"Single source of truth" is wrong, because, you don't have to put everything into Redux, the store state doesn't have to be an object, and you don't even have to have a single store.

HTTP://BLOG.ISQUAREDSOFTWARE.COM/2017/05/IDIOMATIC-REDUX-TAO-OF-REDUX-PART-1/

YOU SHALL NOT PUT EVERYTHING IN A SINGLE REDUX STORE

Some valid reasons for using multiple stores in Redux:

- Solving a performance issue caused by too frequent updates of some part of the state, when confirmed by profiling the app.
- Isolating a Redux app as a component in a bigger application, in which case you might want to create a store per root component instance.

OFFICIAL DOCUMENTATION

JFYI: THERE IS NO RING

Three Stores for the Elven-kings under the sky,
Seven for the Dwarf-lords in their halls of stone,
Nine for Mortal Men doomed to die,
One for the Dark Lord on his dark throne
In the Land of Mordor where the Shadows lie.
One Store to rule them all, One Store to find them,
One Store to bring them all and in the darkness bind them, In the Land of Mordor where the Shadows lie.

Spoiler alert: It got melted. Doesn’t dispatch events properly.

YOU SHALL NOT PUT EVERYTHING IN A SINGLE REDUX STORE

YOU HAVE NOT PUT EVERYTHING IN A SINGLE REDUX STORE

YOU COULD NOT PUT EVERYTHING IN A SINGLE REDUX STORE

“

Any component wrapped with connect() call will receive a dispatch function as a prop, and any state it needs from the global state.

ADAM RACKIS

“

Connect - Connects a React component to a Redux store.
Any component. Anywhere.

README.MD

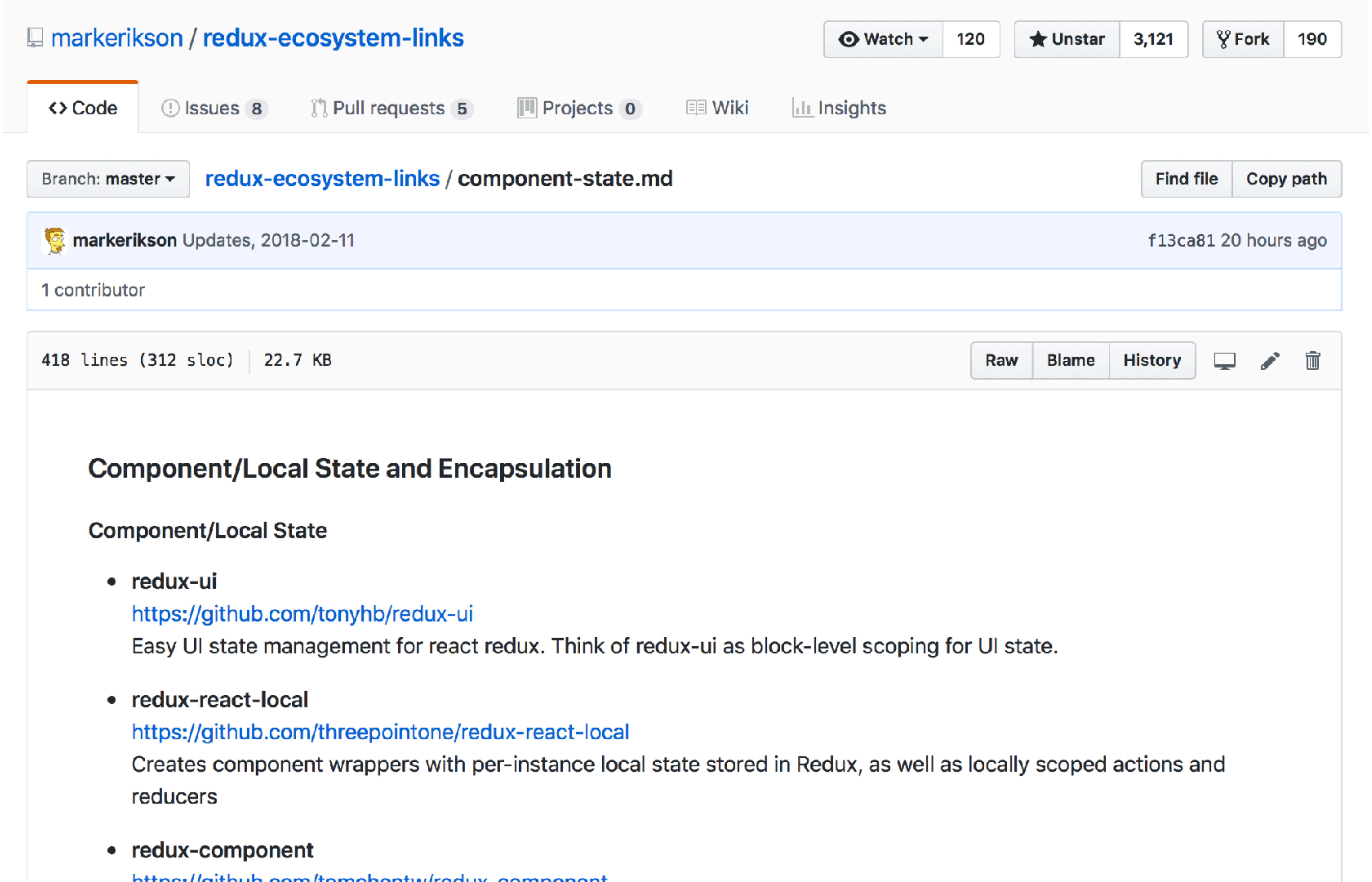
HOW TO COULD?

LOCAL STORE

A “my precious” for each one.

You can store anything in “your own store”.
Your local store might be a part of a global store.

A Store inside a Store.
A State of a State



redux-ecosystem-links / component-state.md

markerikson Updates, 2018-02-11 f13ca81 20 hours ago

1 contributor

418 lines (312 sloc) | 22.7 KB

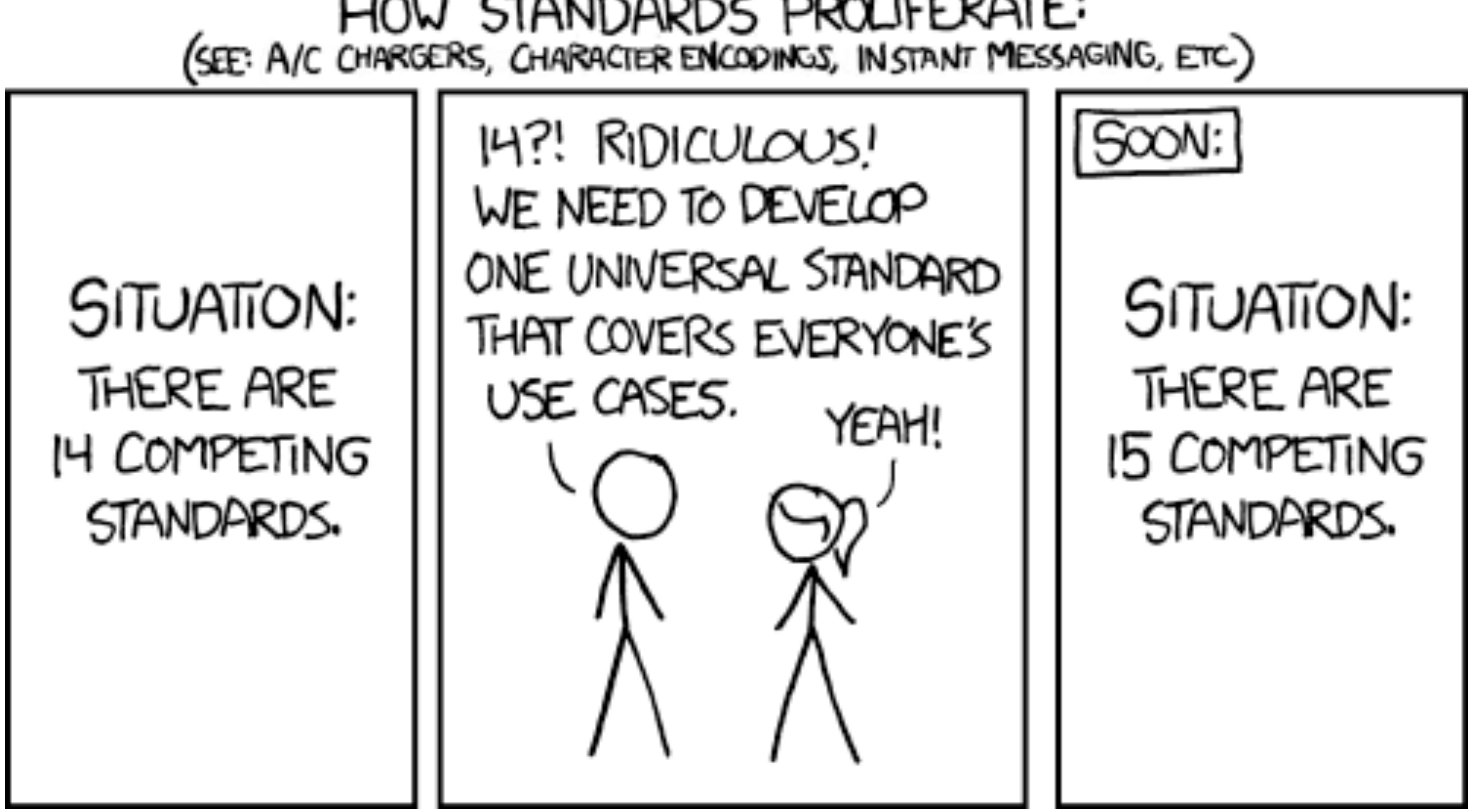
Component/Local State and Encapsulation

Component/Local State

- **redux-ui**
  https://github.com/tonyhb/redux-ui
  Easy UI state management for react redux. Think of redux-ui as block-level scoping for UI state.

- **redux-react-local**
  https://github.com/threepointone/redux-react-local
  Creates component wrappers with per-instance local state stored in Redux, as well as locally scoped actions and reducers

- **redux-component**


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

78

awesome things

7-8

super awesome things

Just to name a few

Redux-react-local

Redux state

Redux-subspace

```
// connect your components
@local({
  ident: 'app',
  initial: { count: 0 },
  // optionally -
  reducer(state, action) {
    if(action.me) { // happened 'locally'
      switch(action.meta.type) {
        // case: increment decrement etc
      }
    }
    // reduce on other global dispatches here
    return state
  }
})
class App extends React.Component {
  render() {
    let { state, dispatch, $ } = this.props
    return (<div onClick={() => dispatch($({ type: 'increment' }))}>
      clicked {state.count} times
    </div>)
  }
}
```

Just to name a few

Redux-react-local

Redux state

Redux-subspace

```
const mapStateToProps = (localState, props, state) => ({
    ...
});
const mapDispatchToProps = (localDispatch, props, dispatch) => ({
    localAction: bindActionCreators(actions.localAction,
localDispatch),
    globalAction: bindActionCreators(actions.globalAction,
dispatch)
    ...
});
const mergeProps = (stateProps, dispatchProps, props) => ({
    ...
});

const Component = (props)=>(
...
);

export default connectState(mapStateToProps, mapDispatchToProps,
mergeProps, localReducer)(Component);
```

Just to name a few

Redux-react-local

Redux state

Redux-subspace

```
const rootReducer = combineReducers({
  todo: todoReducer
  counter1: namespaced('counter1')(counterReducer),
  counter2: namespaced('counter2')(counterReducer)
})

const store = createStore(rootReducer)

const App = () => (
  <Provider store={store}>
    <SubspaceProvider mapState={(state) => state.todo}>
      <TodoApp />
    </SubspaceProvider>
    <SubspaceProvider mapState={(state) => state.counter1}
namespace="counter1">
      <CounterApp />
    </SubspaceProvider>
    <SubspaceProvider mapState={(state) => state.counter2}
namespace="counter2">
      <CounterApp />
    </SubspaceProvider>
  </Provider>
)
```

QUESTION?

Do they solve the problem?

ANSWER!

They solves, their problems.

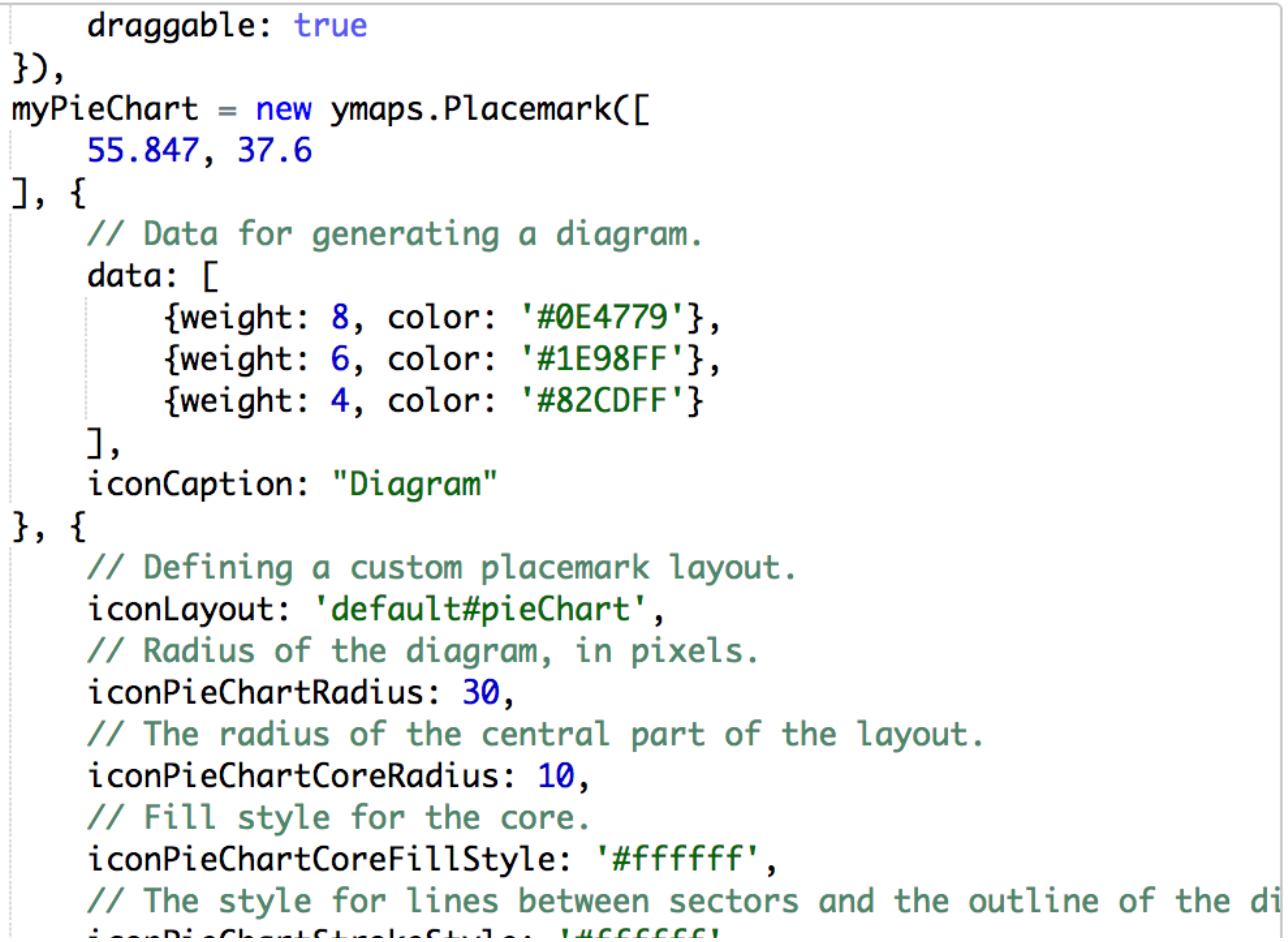
ANSWER!

They solves, their problems, in their own ways. They all have strong theory behind.

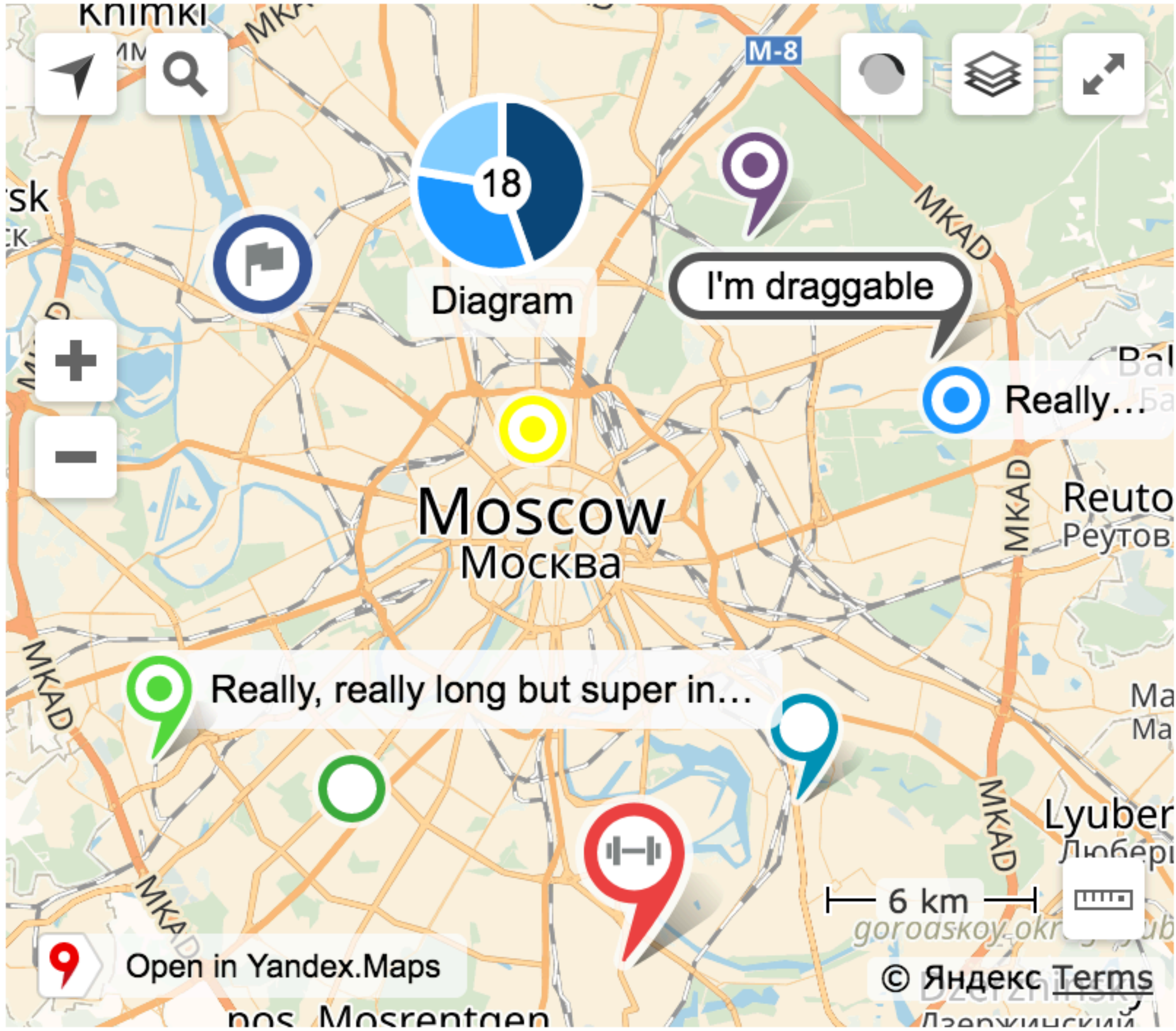
Meanwhile in Russia

Someone also solved their problems.

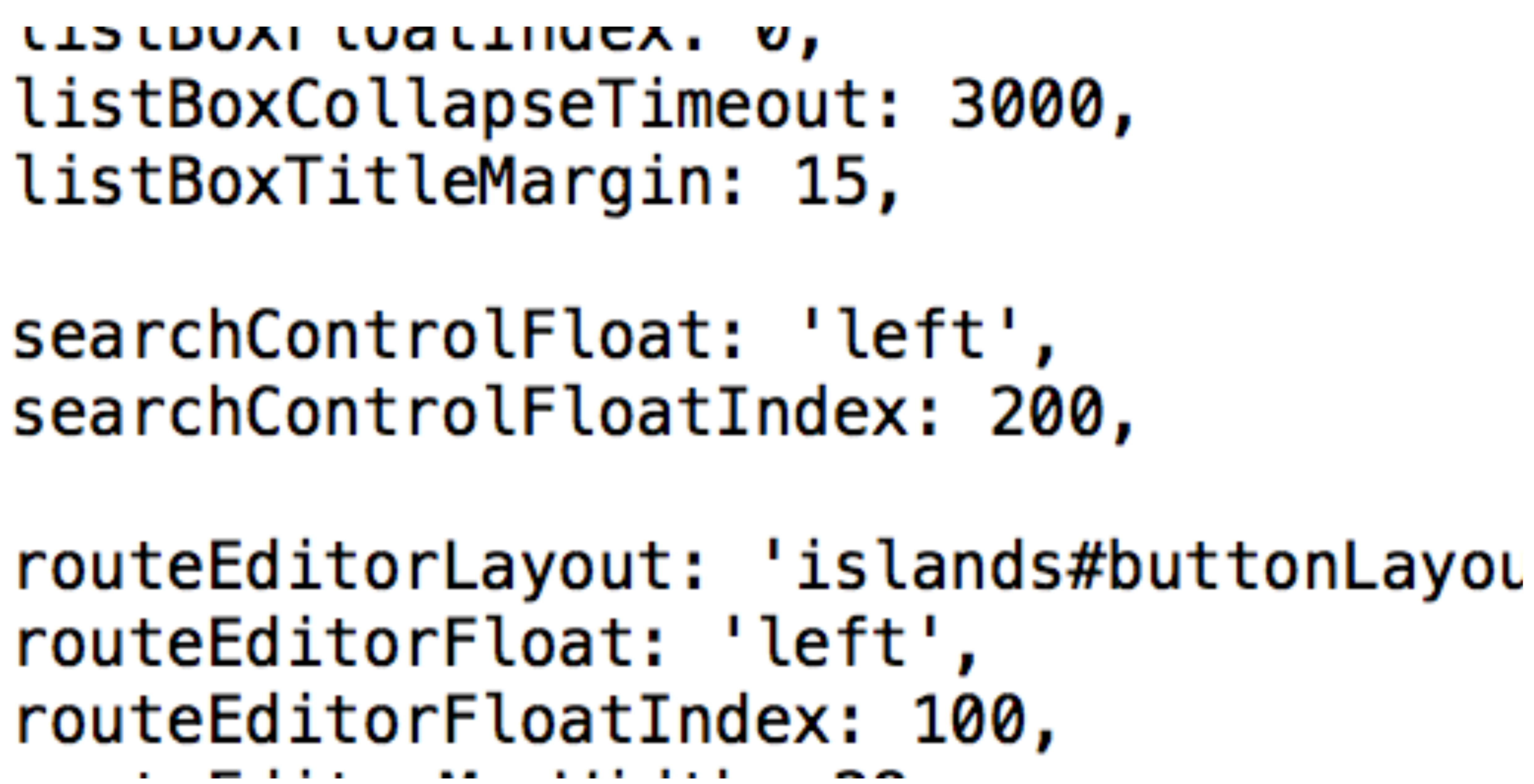
YANDEX MAP API



```
    draggable: true
}),
myPieChart = new ymaps.Placemark([
    55.847, 37.6
], {
    // Data for generating a diagram.
    data: [
        {weight: 8, color: '#0E4779'},
        {weight: 6, color: '#1E98FF'},
        {weight: 4, color: '#82CDFF'}
    ],
    iconCaption: "Diagram"
}, {
    // Defining a custom placemark layout.
    iconLayout: 'default#pieChart',
    // Radius of the diagram, in pixels.
    iconPieChartRadius: 30,
    // The radius of the central part of the layout.
    iconPieChartCoreRadius: 10,
    // Fill style for the core.
    iconPieChartCoreFillStyle: '#ffffff',
    // The style for lines between sectors and the outline of the di
    iconPieChartStrokeStyle: '#ffffff'
```



ON THE LEFT



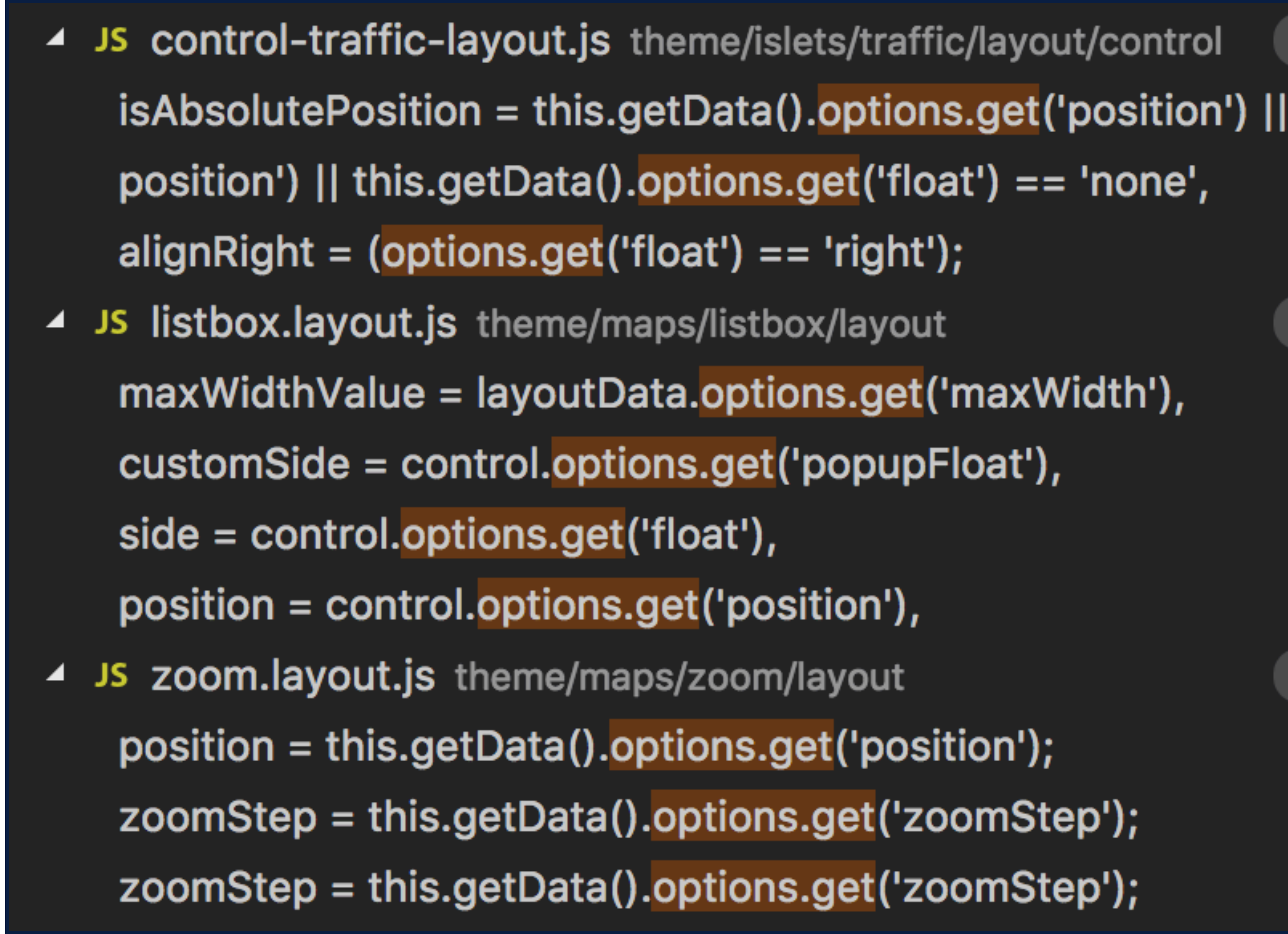
```
listBoxFloatIndex: 0,
listBoxCollapseTimeout: 3000,
listBoxTitleMargin: 15,

searchControlFloat: 'left',
searchControlFloatIndex: 200,

routeEditorLayout: 'islands#buttonLayou
routeEditorFloat: 'left',
routeEditorFloatIndex: 100,
```

Names are “long” to set

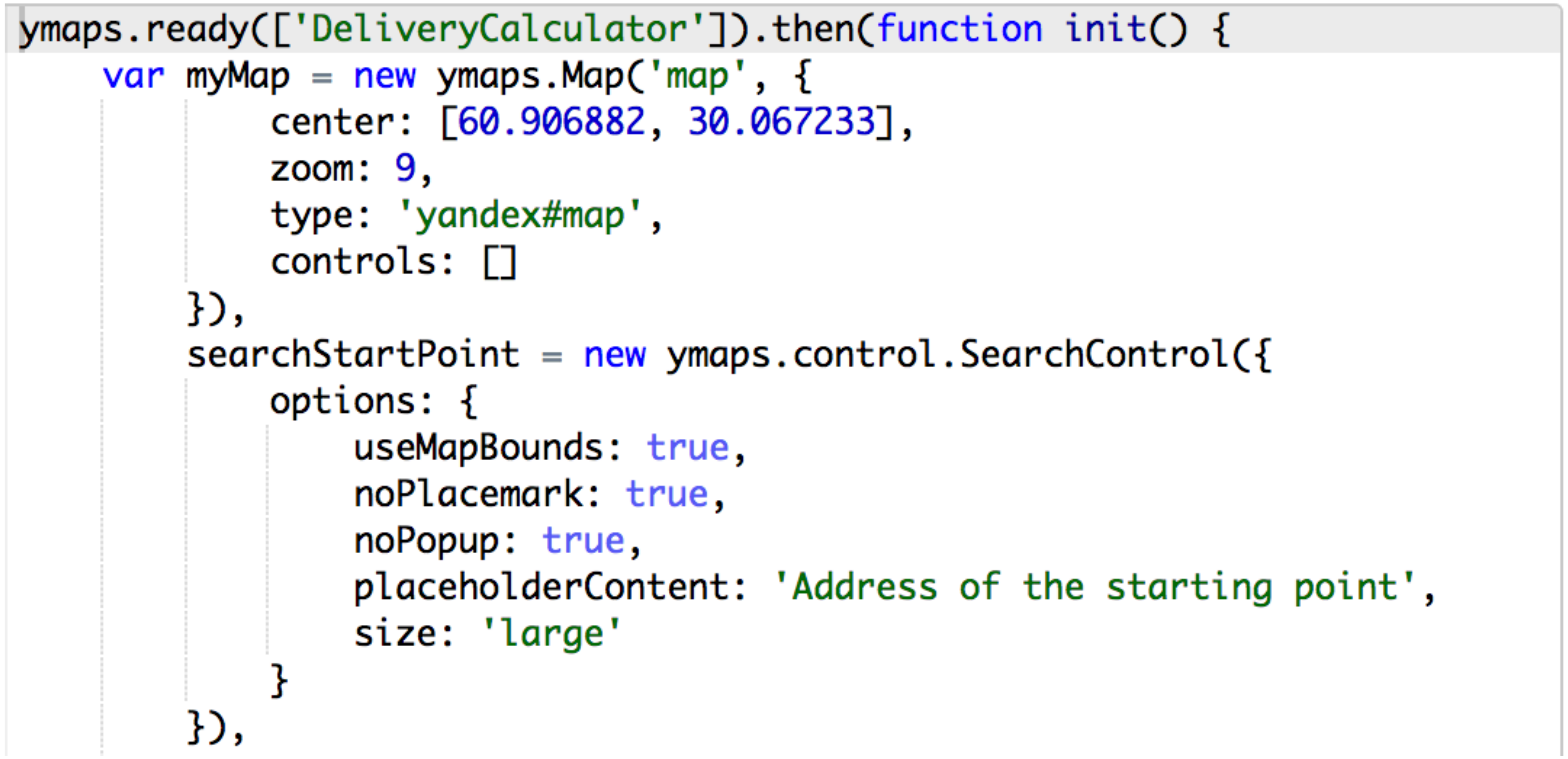
ON THE RIGHT SIDE



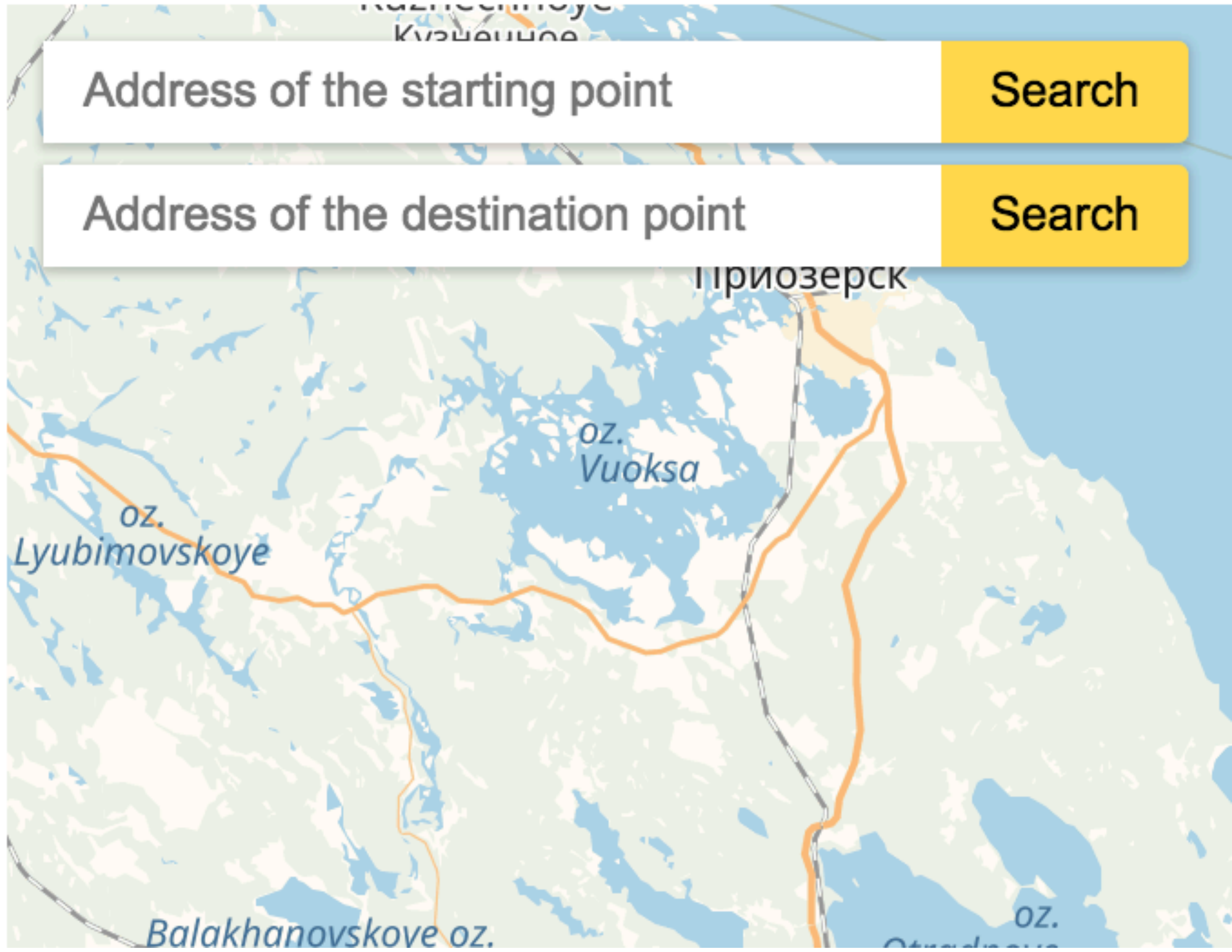
```
JS control-traffic-layout.js theme/islets/traffic/layout/control
    isAbsolutePosition = this.getData().options.get('position') ||
    position') || this.getData().options.get('float') == 'none',
    alignRight = (options.get('float') == 'right');
JS listbox.layout.js theme/maps/listbox/layout
    maxWidthValue = layoutData.options.get('maxWidth'),
    customSide = control.options.get('popupFloat'),
    side = control.options.get('float'),
    position = control.options.get('position'),
JS zoom.layout.js theme/maps/zoom/layout
    position = this.getData().options.get('position');
    zoomStep = this.getData().options.get('zoomStep');
    zoomStep = this.getData().options.get('zoomStep');
```

Names are “short” to get

“LOCAL” STATE



```
ymaps.ready(['DeliveryCalculator']).then(function init() {
    var myMap = new ymaps.Map('map', {
            center: [60.906882, 30.067233],
            zoom: 9,
            type: 'yandex#map',
            controls: []
        }),
        searchStartPoint = new ymaps.control.SearchControl({
            options: {
                useMapBounds: true,
                noPlacemark: true,
                noPopup: true,
                placeholderContent: 'Address of the starting point',
                size: 'large'
            }
        }),
```



OPTION MAPPER

DONT HAVE VALUE IN A STATE?
PREFIX KEY AND ASK THE PARENT! (for both keys 🙀)

OPTION MAPPER

It “**bubbles**” from components, looking for any value matching the requested one.

6Y

in production

that was not a good idea

Changing a value “reflows” everything. Not “predictable”.

Redux-restate

redemption

DIVE IN

mapStateToState

```
newState = parentState => ({
  …parentState.subBranch,
  …parentState.somethingElse
})
```

Form a State from a State

BUBBLE OUT

routeDispatch

```
(dispatch, event) =>
   dispatch({…mod(event), secret:42})
```

“Restore” the information

AND THEN.....

A New State will become a True State for all the nested components.
Scoping updates, btw

AND THEN.....

You can use tree-shaped, non-normalized states. As mobx-state-tree could.

AND THEN.....

You can decoupe your application into the “micro-frontends”.
(introduced in redux-subspace)

DIVE IN

mapStatesToState

```
newState = parentStates => ({
  …firstState.subBranch,
  …secondState.somethingElse
  …componentProps.state
})
```

Form a State from a States

BUBBLE OUT

routeDispatch

```
(dispatch, event) =>
   dispatch.store1(event)
```

“Restore” and “Route”

one
STORE
to
RULE THEM ALL!

and to darkness bind them, of course

AND THEN.....

You can split and rejoin "Stores". And it still could be the single Store (at the bubble end)

DIVE IN

mapStatesToState

```
newState = (states, props) => ({
  …props
})
```

Form a State

BUBBLE OUT

routeDispatch

```
(dispatch, event, props) =>
   props.dispatch(event)
```

“Restore” and “Route”

DIVE IN

mapStatesToState

```
newState = (states, props) => ({
  …props
})
```

Form a State

BUBBLE OUT

routeDispatch

```
(dispatch, event, props) =>
   setState(state => event(state))
```

“Restore” and “Route”

“

It's important that actions being objects you have to dispatch is not boilerplate, but one of the fundamental design choices of Redux.
If there are no serializable plain object actions.... you don't need Redux.

REDUX DOCUMENTATION

AND THEN.....

You can put React state inside “Redux”, mapStateToProps, and form a new React State.

(dont ask, you just can do it)

(No Time-Travel, sorry mate)

Subpackages

Unbranch

Scope updates

Semaphore

Freeze the time, and freeze the state

Restate

Rule multiple stores at once

Focus

Dive into a new state

+ react-redux-delay, beautiful-react-redux. More to come each week.

100

lines of code

0

magical hacks inside

0

magical hacks inside

except unbranch, it has lots of magics underneath

“

If one said that redux is so bad…
Why not to fix it?

🤷‍♂️

FRACTAL STATE