Outline

Intro

I've got the keys! Using lenses and transducers simplify your life.

- Brooke Mitchell
- `https://github.com/brookemitchell`
- `https://twitter.com/brocmit`

Welcome

- These are my experiences using lenses and transducers to simplify data manipulation in a redux app I built.
- Spoiler... there are tradeoffs.

Slides and source code

Slides

`https://github.com/brookemitchell/talk-redux-lenses-transducers`

Example Project with refactored branches

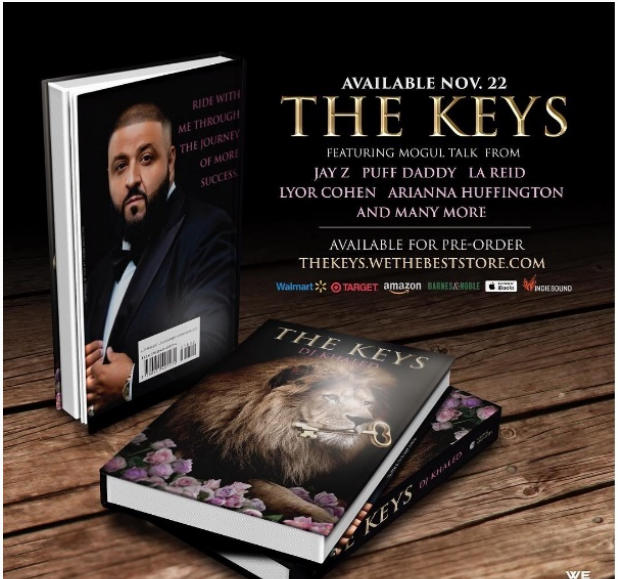
`https://github.com/brookemitchell/redux-lenses-example`

Me and Redux

- This isn't a talk about react.
- This talk is a little about my experiences using redux.
- This is really about {} . Plain old javascript objects, aka pojos.
- This is also a talk about DJ Khaled.

DJ Khaled

DJ Khaled



djkhaled S'abonner

15,9k J'aime 12 h

djkhaled New book alert ! #THEKEYS NOV.22 pre order now http://bit.ly/thekeysbn

afficher les 581 commentaires

taste_the_sauce @dvovictor

mitchrey @wesleymurillo @anthokneee

elsheunextdoor @daniellaaaordaz

minimouze4 Ur a positive man so I might have to scoop this!

andrewv262 @nickyhong found your bday gift

eduardocruz @jo5on mogul talk

eduardocruz @mitchhhhhhhhh

criminalct @kazman1987 @benjaman12

cathalkearney6 @peteroregan99

raielmusic @ttammyc

Connectez-vous pour aimer ou

Ramda

- Ramda Repl (`http://ramdajs.com/repl/`)
- Auto currying

```
const R = require('ramda')
const addOne = R.map(x => x + 1)
addone([1, 2, 3]) // => [2, 3, 4]
```

Ramda

- Composition

```
const fn1 = a => a + 3
const fn2 = b => b * 2
const twoFns = c => fn2(fn1(c))
const twoFnsCompose = c => R.compose(fn2, fn1)(c)
// or pointfree
const twoFnsComposePf = R.compose(fn2, fn1)
twoFnsComposePf(0) // => 6
```

Ramda

- R.ap, related to applicatives. Don't worry about that.

```
const fn1 = a => a + 3
const fn2 = b => b * 2
const fn3 = c => c / 2
const crossProduct = R.compose(R.ap[fn1, fn2, fn3], R.of)
crossProduct(10) // => [13, 20, 5]
```

DJ Khaled

Crash course in Redux

- There is a 'store' that keeps all of your state in one object
- You can subscribe to store updates to map them to your view.

Redux steps

1. You call store.dispatch(action)
2. The Redux store calls any reducer functions (called 'reducers') that will make changes to the state.
3. The Redux store replaces the old state tree with a new state tree.

DJ Khaled

Redux Reducer

Imaginary redux reducer - nb: List of actions is really provided over time

```
const initialState = {loading: true, filter: all}
['ACTION 1', 'STOP_LOADING', 'ACTION 3']
  .reduce((state, action) => {
    if (action.type === 'STOP_LOADING') {
      return Object.assign({}, state, {loading: false})
    }
    else return state
    }
  },  initialState)
```

- Explaining Immutability [Link]

DJ Khaled

Redux Mapper

- Using nanocomponent...

```
const component = require('nanocomponent');
const html = require('bel');
const mapStateToProps = state => ({widgets: state.widgets});
const props = mapStateToProps(store.getState());
var WidgetList = component({
  render: function (props) {
    return html`
      <ul>${props.widgets.map(
        e => html`<li>${e}</li>`)}</ul>`
  }
});
```

DJ Khaled

Refactoring a real app

Starting point - Troubled Mapper

```
const mapStateToProps = state => {
  return {
    user: state.users[state.routeParams.uid]
    userDetails: state.usersDetails[user.uid] userDetails,
    noUser: typeof user === 'undefined',
    name: noUser ? '' : user.info.name,
    lastUpdatedUser: user ? user.lastUpdated : 0,
    isFetching: user.isFetching || usersDetails.isFetching,
    error: users.error || usersDetails.error,
    ...
  };
};
```

DJ Khaled

Bad map fix, step 1: create selectors

```
// reducers.js
const editing = state => state.works.editing
const user$ = state => state.users[state.route.uid]
const editing$ = R.compose(
                    R.propOr([], 0),
                    R.toPairs,
                    editing)
//container.js
export const mapStateTo = (state) => {
  return {
    user: user$(state),
    editing: editing$(state)
    ...
  };
```

DJ Khaled

Bad map fix, step 2: Composing with ramda

- Major key: compose selectors.

```
// selectors.js
export const stateToProps$ = R.compose(
  R.zipObj(['user', 'error', 'editing']),
  R.ap([
    user$,
    error$,
    editing$
    ]),
  R.of)
//container.js
const mapStateToProps = stateToProps$(store.getState())
// => {'user': ..., 'error': ..., 'editing': ...}
```

DJ Khaled

Alternative Step 2: Reselect

- https://github.com/react/reselect

```
import { createSelector } from 'reselect'
const isFetching = createSelector(
  [ user, userDetails ],
  (user, userDetails) => user.isFetching
    || usersDetails.isFetching,
)

export const stateToProps$ = createSelector(
  [name$, userDetails$, error$, editing$],
  (name, userDetails, error, editing) =>
    ({name, userDetails, error, editing})
)
```

DJ Khaled

Alternative Step 2.5: Ramda Reselect

- written by me! [http://npmjs.com/ernusame/ramda-reselect]
- write one yourself!

```
const createSelector = (...fns) =>
  R.compose(
    R.apply(R.memoize(R.last(fns)))),
    R.ap(R.slice(0, -1, fns))
    R.of
  )
```

Alternative Step 2.5: Ramda Reselect

- Mapper looks better now.

```
export const stateToProps$ = createSelector(
  name$, userDetails$, error$, editing$,
  (name, userDetails, error, editing) =>
    ({name, userDetails, error, editing})
)
```

DJ Khaled

Awkward reducer

- Reducer for the auckland roadworks app,
  - This is the function for setting the new shape of the state called every time an 'action' is dispatched.

```
export default function works(state = initialState, action) {
  switch (action.type) {
    case WORKS_FETCH_FAILED: {
      return {
        ...state,
        appState: "error",
        error: action.message
      };
    }
```

Awkward reducer cont...

```
case SET_TEXT: {
  const oldItem = state.works[
    action.changedEntry.id];
  const newItem = action.changedEntry[
    action.changedEntry.id];

  const mergedEntry = {
    works: {
      ...state.works,
      [action.changedEntry.id]: {
        ...oldItem,
        ...newItem
      }
    }
```

Awkward reducer cont...

```
      return {
        ...state,
        ...mergedEntry
      };
    }
    default:
      return state;
  }
}
```

DJ Khaled

Reducer refactor pt1.

- break out sub-reducers.

```
function setText (state, action) {...}
export default function works(state = initialState, action) {
    ...
    case SET_TEXT: {
      setText(state, action)
    }
    ...
}
```

- I don't really like this. – Separation of concerns RANT - MPJ's Musings - FunFunFunction #47 https://www.youtube.com/watch?v=0ZNIQOO2sfA

DJ Khaled

Reducer refactor pt2.

- What if we could use selectors in reducers. Interchangably.
- This won't work. But suggests something cool.

```
const editTextReducer = createSelector(
  state, editing$,
  (state, editing) => Object.assign({}, state, {editing})
)
// warning:  this doesn't work
function works(state = initialState, action) {
    case SET_TEXT:
      return editTextReducer(state)
      };
    }
}
```

DJ Khaled


Ride wit me though the journey of more success
The key to more success🙏 trust me

Reducer refactor pt3. - Lenses!

```
// selector
export const worksLens = R.lensProp("works");
const getWorks = R.view(worksLens, store.getState());
const worksItemLens =  (id, key) => R.compose(
        worksLens,
        R.lensPath([id, key]));
// reducer
function works(state = initialState, action) {
    case SET_TEXT: {
      const { id, key, value } = action
      return R.set(worksItemLens(id, key), value, state);
    }
}
```

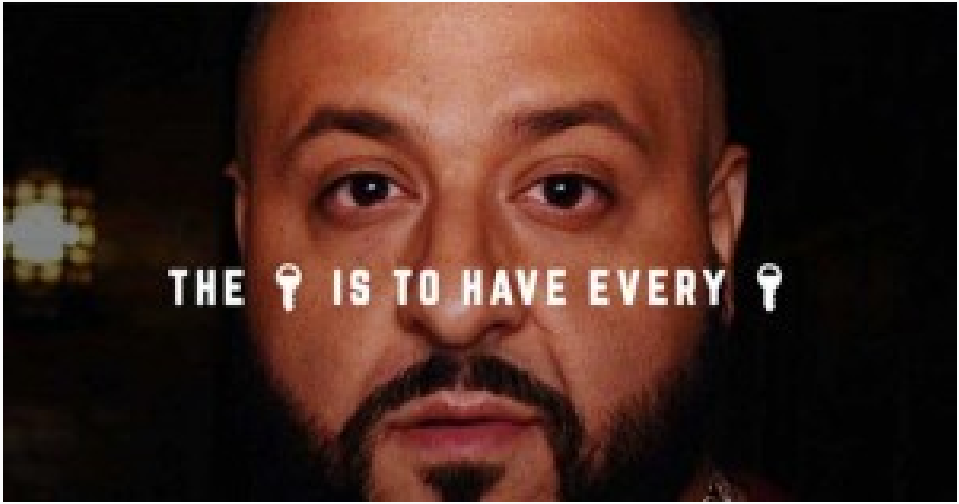
DJ Khaled

Reducer refactor pt4. - Transducers!

- Didn't end up being useful.
- Check out transducers.js or ramdas transducer function.
- Definite future use for complex text filters and form validation.

```
const t = require("transducers.js")
const xform =  t.compose(
        t.map(function(kv){return [kv[0], kv[1] + 10]}),
        t.map(function(kv){return [kv[0], kv[1] * 9]}),
        t.filter(function(kv){return kv[1] % 2 !== 0}),
       )
t.seq({ one: 1, two: 2, three: 3 }, xform);
// => {one: 99, three: 117}
```

DJ Khaled



Major Keys

- The key is to have every key. And compose them.
- Don't play yourself. Refactor honestly.
- Stay away from they. Don't get discouraged if things don't work out.

Some related talks.

- Think about refactoring as a spectrum of abstraction. Cheng Lou - On the Spectrum of Abstraction
- "Build things with knowledge and technique." Alan Kay - Is it really "Complex"? Or did we just make it "Complicated"?
- Learning to make things with basic tools. MPJ - Coding and Cooking

Outro

- Let's keep winning.