<beamer>metropolis

Contents

1 Welcome

- I'm Brooke Mitchell.
- These are my experiences using lenses and transducers to simplify data manipulation in a redux app I built.
- Spoiler... there are tradeoffs when trying to simplify.

2 Slides and source code

2.1 Slides

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

2.2 Example Project with refactored branches

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

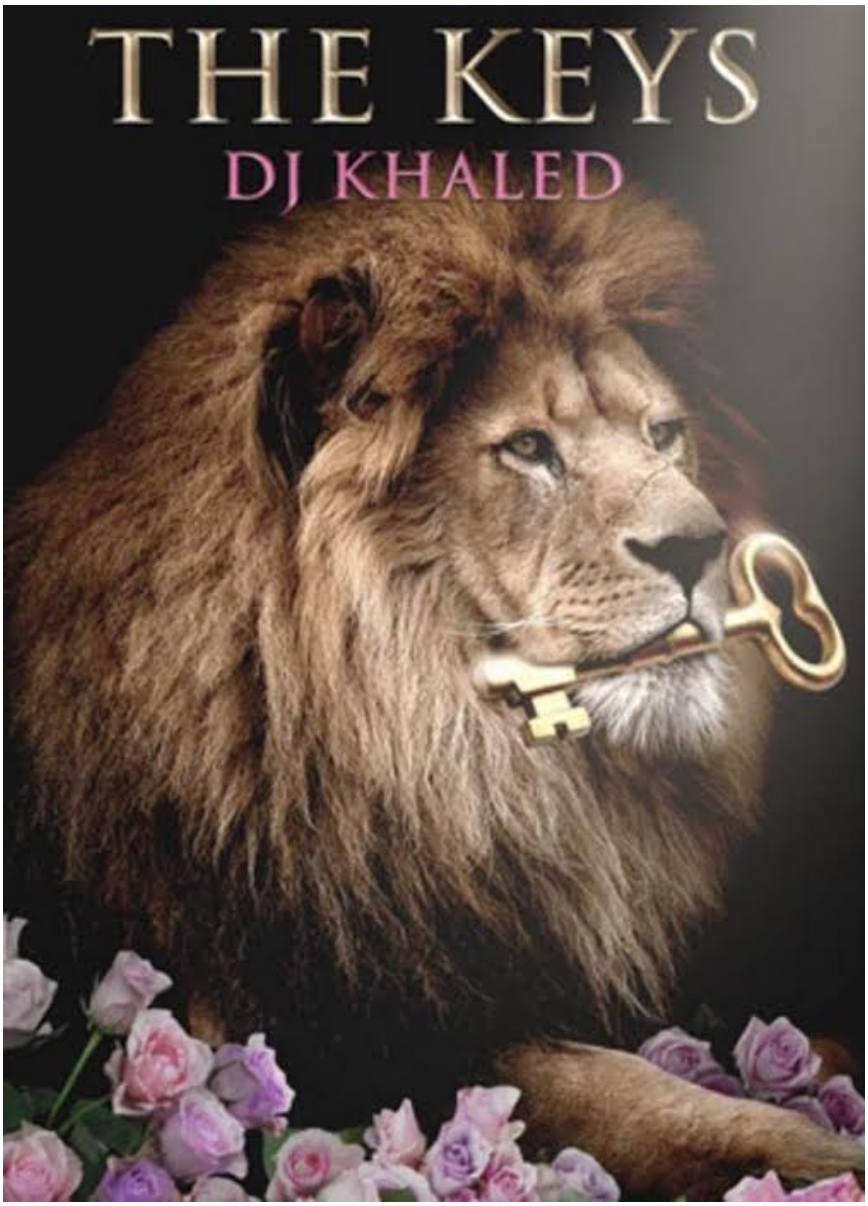
3 Me and Redux

3.1 {}

- 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.

4 DJ Khaled

5 DJ Khaled



6 DJ Khaled

7 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]
```

8 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
```

9 Ramda

- Applicatives

```
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]
```

10 Redux

10.0.1 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.

10.0.2 Redux steps

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

11 DJ Khaled

12 Redux Reducer

12.0.1 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)
```

- Immutability links here... (todo)

13 DJ Khaled

14 Redux Mapper

- People typically use libs like react-redux, but lets try...

```
import h from 'hyperscript'
const widgetsList = widgets =>
  h('div',
    h('ul',
      widgets.map(w => h('li', w))));
let divWithState;
store.subscribe(() => {
  const currentState = store.getState(); // => {widgets: [...]}
  divWithState = widgetsList(currentState.widgets);
})
```

15 DJ Khaled

16 Redux Mapper 2

- Same again 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>`
  }
});
```

17 DJ Khaled

18 Refactoring a real app

18.1 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,
    ...
  };
};
```

19 DJ Khaled

20 Bad map fix, step 1: create selectors

```
// selectors.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)
    ...
```

```
  };
};
```

21 DJ Khaled

22 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())
```

23 DJ Khaled

24 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})
)
```

25 DJ Khaled

26 Alternative Step 2.5: Ramda Reselect

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

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

27 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})
)
```

28 DJ Khaled

29 Awkward reducer

- Reducer for the roadworks editing 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
      };
    }
```

30 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
      }
    }
  };
```

31 Awkward reducer cont...

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

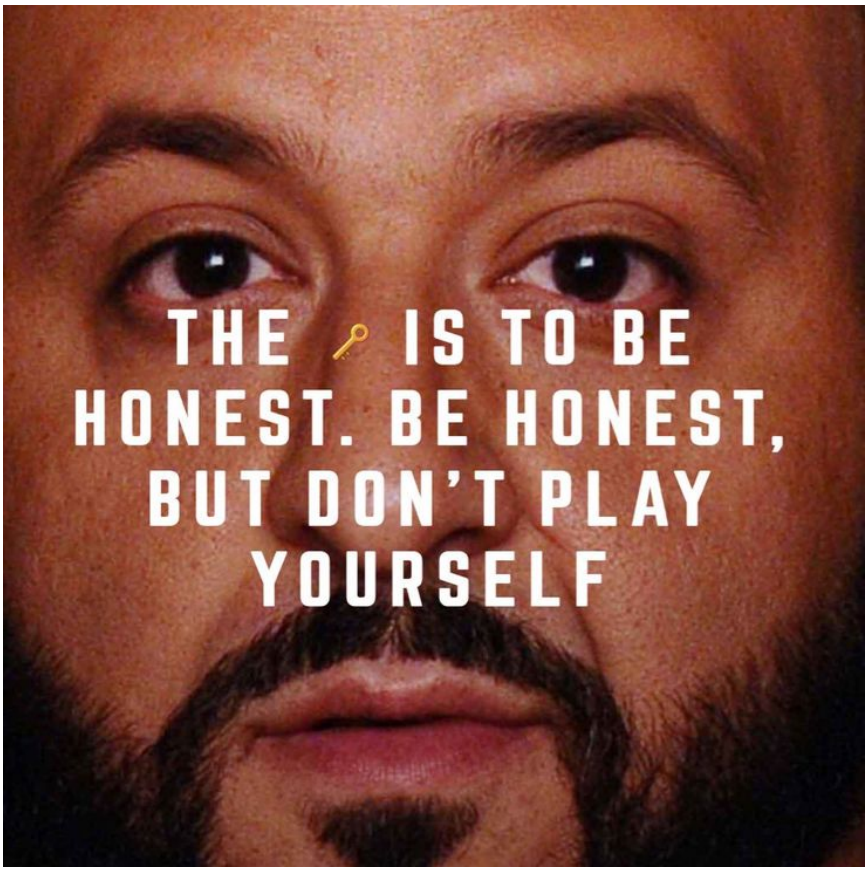
32 DJ Khaled

33 Reducer refactor pt1.

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

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

34 DJ Khaled



35 Reducer refactor pt2.

- What if we could use selectors in reducers. Interchangable.
- x this won't work. But suggests something bigger.

```
const editTextReducer = createSelector(
  state, editing$,
  (state, editing) => Object.assign({}, state, {editing})
)

function works(state = initialState, action) {
    case SET_TEXT:
```

```
      return editTextReducer(state)
      // x this doesn't work
      };
    }
}
```

36 DJ Khaled

37 Reducer refactor pt3. - Lenses!

- Now we can use the lens in both places!

```
// selector
const worksItemLens = R.lensPath(["works", id, key]);
// reducer
function works(state = initialState, action) {
    case SET_TEXT: {
      const { id, key, value } = action
      const worksItemLens = R.lensPath(["works", id, key]);
      return R.set(worksItemLens, value, state);
    }
}
```

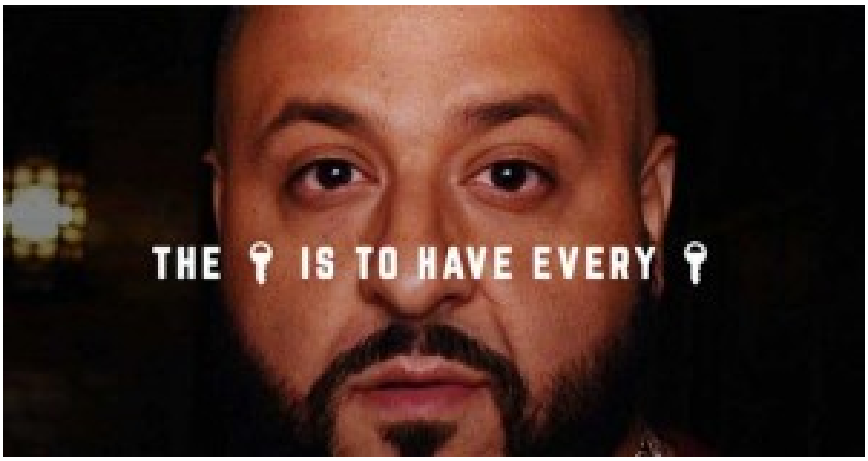
38 DJ Khaled

39 Reducer refactor pt4. - Transducers!

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

```
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}
```

40 DJ Khaled



41 Major Keys

- The key is to have every key.
- Don't play yourself.
- Stay away from they.
- 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"?
- Compose and use basic tools. MPJ - Coding and Cooking

41.1 Outro

- Let's keep winning.