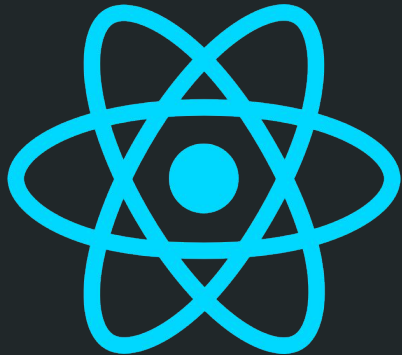


“What the Flux?! Let's Redux.”





Júnior Oliveira

IT Manager at PHIPASA

CTO at NomenaLista.net

PHP Old School

React and Redux beginner, but already in love with

<https://github.com/arojunior>

*Before start we need to know
three things about Redux...*

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.

*And then we can start to talk
about these libs...*

React-redux

Official React bindings for Redux. Connects a React component to a Redux store

Redux-actions

Flux Standard Action utilities for Redux. Wraps an action creator so that its return value is the payload of a Flux Standard Action

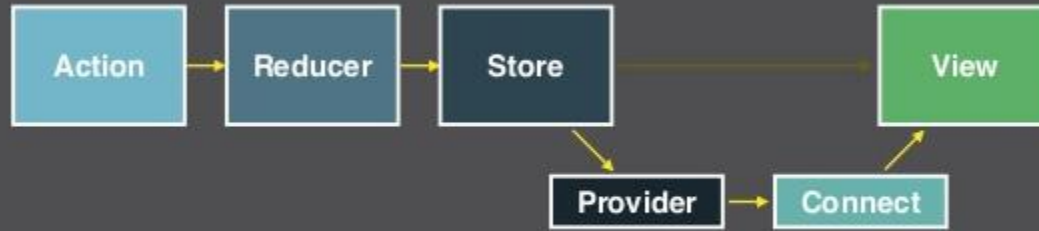
Redux-promise

FSA-compliant promise middleware for Redux (Async Actions)

Redux-form

Manage form state in Redux

Data flow using react-redux



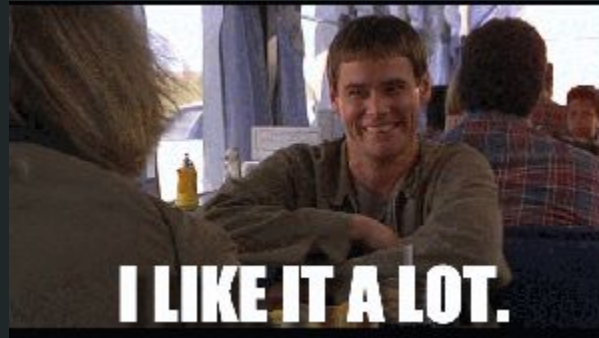
Presentational and Container Components

	Presentational Components	Container Components
Purpose	How things look (markup, styles)	How things work (data fetching, state updates)
Aware of Redux	No	Yes
To read data	Read data from props	Subscribe to Redux state
To change data	Invoke callbacks from props	Dispatch Redux actions
Are written	By hand	Usually generated by React Redux

Presentational and Container Components

“Most of the components we'll write will be presentational, but we'll need to generate a few container components to connect them to the Redux store. This and the design brief below do not imply container components must to be near the top of the component tree.”

*No, we are not going to talk about
“TODO list”... Relax and enjoy it.*



React-redux

Provider

“Makes the Redux store available to the connect() calls in the component hierarchy below. Normally, you can’t use connect() without wrapping the root component in <Provider>.”

React-redux

Provider

```
import {Provider} from 'react-redux'
import {store} from './modules'

render(
  <Provider store={store}>
    <Router history={browserHistory}>
      <Route path="/" component={App}/>
      <Route path="/netflix" component={Netflix}/>
      <Route path="/spotify" component={Spotify}/>
    </Router>
  </Provider>,
  document.getElementById('root')
)
```

React-redux

Connect

Inject just dispatch and don't listen to store

```
export default connect()(Spotify)
```


Inject all action creators without subscribing to the store

```
import * as actions from '../modules/Spotify/actions'
```

```
export default connect(null, actions)(Spotify)
```

Inject dispatch and every field in the global state

```
export default connect(state => state)(Spotify)
```



Don't do this! It kills any performance optimizations because App will rerender after every action. It's better to have more granular connect() on several components in your view hierarchy that each only listen to a relevant slice of the state.

React-redux

Connect

“[`mapStateToProps`(state, [ownProps]): stateProps] (Function): If this argument is specified, the new component will subscribe to Redux store updates. This means that any time the store is updated, `mapStateToProps` will be called. The results of `mapStateToProps` must be a plain object, which will be merged into the component's props.”*

React-redux

Connect

```
const mapStateToProps = state => {  
  return {  
    netflix : state.netflix  
  }  
}
```

Now store is
available in
this.props.netflix

 
export default connect(mapStateToProps)(Netflix)

React-redux

Connect

*“[`mapDispatchToProps(dispatch, [ownProps]): dispatchProps`]
(Object or Function): If an object is passed, each function inside it is assumed to be a Redux action creator. An object with the same function names, but with every action creator wrapped into a dispatch call so they may be invoked directly, will be merged into the component’s props. If a function is passed, it will be given dispatch. If you don’t want to subscribe to store updates, pass null or undefined in place of `mapStateToProps`.”*

React-redux

Connect

```
import {connect} from 'react-redux'
import {bindActionCreators} from 'redux'

const mapStateToProps = state => state.spotify
      ↓
const mapDispatchToProps = dispatch =>
  bindActionCreators(spotifyActions, dispatch)
      ↙
export default connect(
  mapStateToProps,
  mapDispatchToProps
)(Spotify)
```

Redux-actions

Flux Standard Actions (FSA)

An action MUST

- be a plain JavaScript object.
- have a **type** property.

An action MAY

- have an **error** property.
- have a **payload** property.
- have a **meta** property.

An action MUST NOT include properties other than type, payload, error, and meta.

Redux-actions

createAction(type, payloadCreator = Identity, ?metaCreator)

```
let netflixSearch = createAction('NETFLIX_SEARCH', value => value);  
// same as  
netflixSearch = createAction('NETFLIX_SEARCH');  
  
expect(netflixSearch('breaking bad')).to.deep.equal({  
  type: 'NETFLIX_SEARCH',  
  payload: 'breaking bad'  
});
```

Redux-actions

```
const netflixSearch = () => {  
  return {  
    type: 'NETFLIX_SEARCH'  
  }  
}
```



```
const netflixSearch = createAction('NETFLIX_SEARCH')
```

Redux-actions

handleAction(type, reducer | reducerMap = Identity, defaultState)

```
export default (state = initialState, action) => {  
  switch (action.type) {  
    case 'NETFLIX_SEARCH':  
      return {  
        ...state,  
        result : action.payload  
      }  
    default:  
      return state  
  }  
}
```



```
const reducer = (state = initialState, action) => ({  
  ...state,  
  result : action.payload  
})  
  
export default handleAction('NETFLIX_SEARCH', reducer, initialState)
```

Redux-actions

handleActions(reducerMap, defaultState)

```
const initialState = {
  fetching : false,
  user     : null
}

const reducer = handleActions({
  [GITHUB_FETCHING]: (state, action) => ({
    ...state,
    sending : true
  }),

  [GITHUB_SUCCESS]: (state, action) => ({
    sending : false,
    user    : action.payload.data
  }),

  [GITHUB_ERROR]: (state, action) => ({
    ...state,
    sending : false,
  })
}, initialState);

export default reducer
```

References

<http://redux.js.org/>

<https://github.com/reactjs/react-redux>

<https://github.com/acdlite/redux-actions>

<http://redux-form.com/>

<https://blog.andyet.com/2015/08/06/what-the-flux-lets-redux/>

Thank you folks

<https://github.com/arojunior/what-the-flux-lets-redux>