“What the Flux?! Let's Redux.”

Júnior Oliveira

IT Manager at PHIPASA
CTO at NomenaLista.net

PHP Old School
Javascript lover

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-form

Manage form state in Redux

Redux-promise

FSA-compliant promise middleware for Redux (Async Actions)

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

Presentational and Container Components

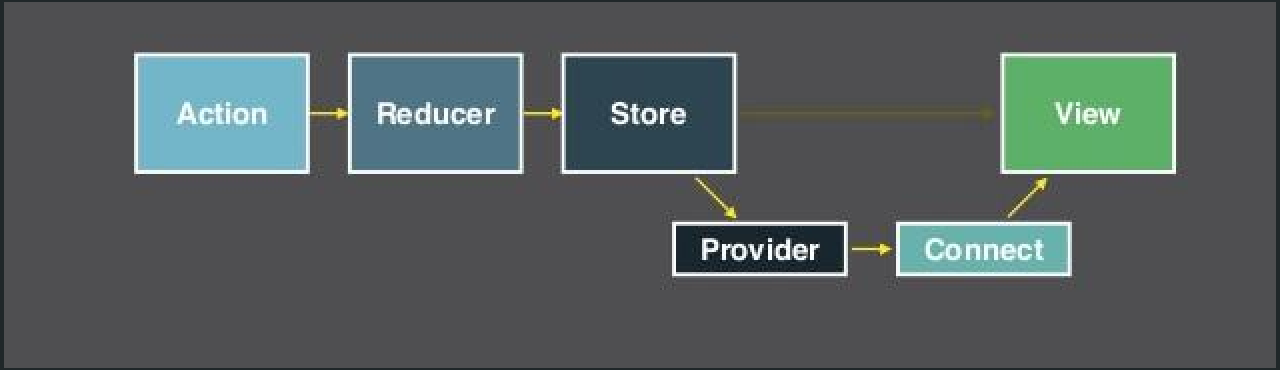
	Presentational Components	Container Components
Purpose	How things look (markup, styles)	How things work (data fetching, state updates)
Aware of Redux	No	Yes
To read data	Read data from props	Subscribe to Redux state
To change data	Invoke callbacks from props	Dispatch Redux actions
Are written	By hand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

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>.”

Data flow using react-redux



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')
)
```

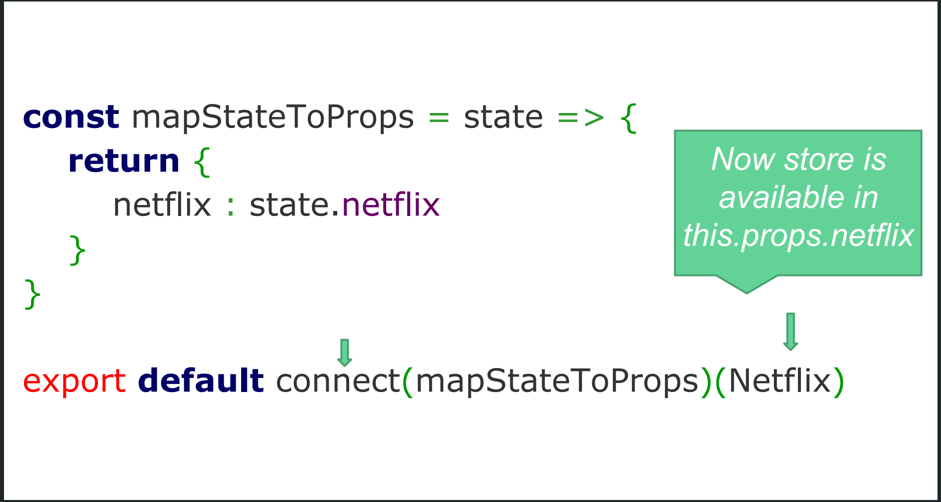
React-redux

Connect

“[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
    }
}

export default connect(mapStateToProps)(Netflix)
```

Now store is available in this.props.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

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-form

redux-form primarily consists of four things:

1. A Redux reducer that listens to dispatched redux-form actions to maintain your form state in Redux.

2. A React component decorator that wraps your entire form in a Higher Order Component (HOC) and provides functionality via props.

3. A Field component to connect your individual field inputs to the Redux store.

4. Various Action Creators for interacting with your forms throughout the application.

Redux-form

```
import { Field, reduxForm } from "redux-form"

const Form = props => {
  const { handleSubmit, pristine, submitting } = props
  return (
    <form onSubmit={handleSubmit}>
      <div className="form-group">
        <Field
          name="searchString"
          component="input"
          type="text"
          placeholder="Search by Movie Title"
          className="form-control"
        />
      </div>
      <button
        type="submit"
        className="btn btn-primary"
        disabled={pristine || submitting}> Send
      </button>
    </form>
  )
}

export default reduxForm({
  form: "netflixSearch"
})(Form)
```

Redux-form

```
class Netflix extends Component {

  handleSubmit(values, dispatch) {
    dispatch(netflixSearch(values.searchString))
  }

  render() {
    return (
      <div className="container">
        <div className="jumbotron">
          <Form onSubmit={this.handleSubmit.bind(this)} />
        </div>
      </div>
    )
  }
}
```

Redux-promise

If it receives a promise, it will dispatch the resolved value of the promise. It will not dispatch anything if the promise rejects.

If it receives an **Flux Standard Action** whose payload is a promise, it will either

- Dispatch a copy of the action with the resolved value of the promise, and set status to success.

- Dispatch a copy of the action with the rejected value of the promise, and set status to error.

The middleware returns a promise to the caller so that it can wait for the operation to finish before continuing.

Redux-promise

```
const searchAction = value => {
  return axios
    .get(apiServer + value.replace(/\s/ig, "%20"))
    .then(res => res.data)
    .catch(err => err)
}


export const NETFLIX_SEARCH = "modules/Netflix/SEARCH"
export const netflixSearch = createAction(NETFLIX_SEARCH, searchAction)
```

Redux-promise

```
export default function promiseMiddleware({ dispatch }) {
  return next => action => {
    if (!isFSA(action)) {
      return isPromise(action)
        ? action.then(dispatch)
        : next(action);
    }

    return isPromise(action.payload)
      ? action.payload.then(
          result => dispatch({ ...action, payload: result }),
          error => {
            dispatch({ ...action, payload: error, error: true });
            return Promise.reject(error);
          }
        )
      : next(action);
  };
}
```

React-redux

Dan Abramov: Redux creator, create-react-app and React developer

Redux-actions

Andrew Clark: Redux co-creator, redux-promise creator, FSA and React/Facebook developer

Redux-form

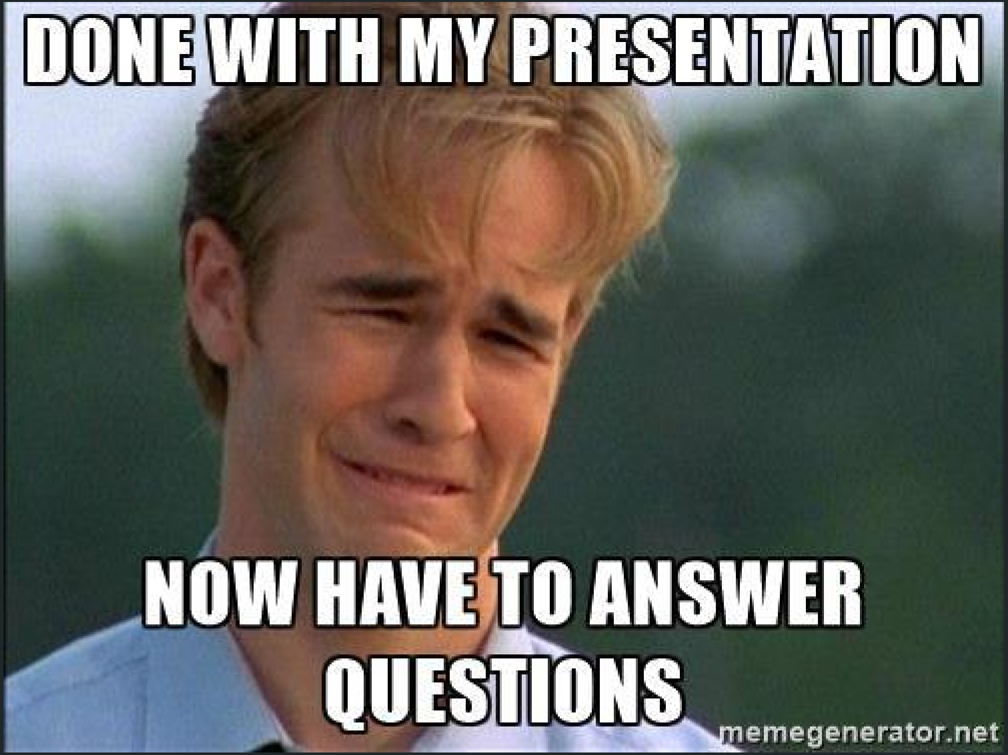
Erik Rasmussen: react-redux-universal-hot-example creator

Redux-promise

Same of redux-actions

Thank you folks

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


DONE WITH MY PRESENTATION
NOW HAVE TO ANSWER QUESTIONS
memegenerator.net

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/