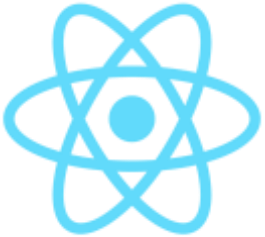
React + Redux + TypeScript === ♥

An introduction to the development of universal JavaScript applications with React, Redux & TypeScript

By Remo H. Jansen

Live demo & source code

https://github.com/remojansen/typescript-redux-react-demo



React

A JavaScript library for building user interfaces.

Lots of people use React as **the V in MVC**.

Makes no assumptions about the rest of your technology stack.

Abstracts away the DOM from you.

Can also render on the **server** using Node.

Can power **native apps** using React Native.

One-way reactive data flow.

React.Component

Components are the main building block of a React application.

A component represents a self-contained piece of UI.

Display some data and be able handle some kind of user interaction.

A component can contain child components.

```
<Cell color={color} figure={figure} />
```

React.Component: Properties

Properties are a Component's configuration. Received from above and immutable..

```
class Cell extends React.Component<ICellProps, ICellState> {

    constructor(props) {
        super(props);
    }

    public render() {
        return (
            <div className={`cell ${this.props.color} ${this.props.figure.type} figure`}
                 data-guid={this.props.figure.id} />
        );
    }
}
```

```
interface IFigure {
  id: string;
  color: string;
  type: string;
}

interface ICell {
  figure: IFigure,
  color: string,
  available: boolean
}
```

```
<div class="cell b pawn_w figure" data-guid="9ef3231e-bf6c-5e16-22e6-68bbf4c110f5"</div>
```

React.Component: Life cycle

Mounting: componentWillMount

Invoked once, both on the client and server, immediately before the initial rendering occurs.

Mounting: componentDidMount

Invoked once, only on the client (not on the server), immediately after the initial rendering occurs.

Unmounting: componentWillUnmount

Invoked immediately before a component is unmounted from the DOM.

```
constructor(props) {
  super(props)
}

public componentWillMount() {
  this.actions.fetchAppetiteFilterConfig();
  this.actions.fetchAppetiteBegin();
}

render() {
  const children = this.props.children;
  return (
    <div>
```

Updating: componentWillReceiveProps

Invoked when a component is receiving new props. This method is not called for the initial render.

Updating: shouldComponentUpdate

Invoked before rendering when new props or state are being received. `forceUpdate` is used.

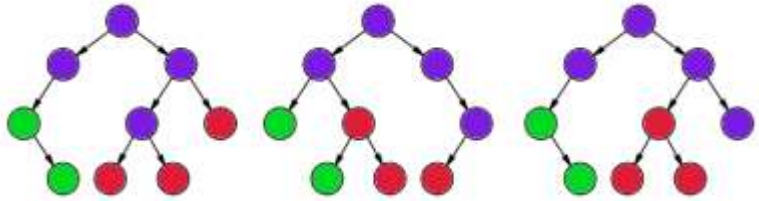
Updating: componentWillUpdate

Invoked immediately before rendering when new props or state are being received. This method is not called for the initial render.

Updating: componentDidUpdate

Invoked immediately after the component's updates are flushed to the DOM. This method is not called for the initial render.

Everything is a tree!



Properties (JSON):

```
{ color : "b",  available : false, figure : { id : "", color: "w", type: "pawn" } }
```

Components (JSX/TSX)

```
<Cell background={background} x={x} y={y} figure={Figure} />
```

Output (HTML)

```
<div class="cell b pawn_w figure" data-guid="9ef3231e-bf6c-5e16-22e6-68bbf4c110f5"</div>
```

React.Component: State, JSX/TSX and Events

```
class TodoApp extends React.Component<IAppProps, IAppState> {

    public state : IAppState;

    constructor(props : IAppProps) {
        super(props);
        this.state = {
            nowShowing: ALL_TODOS,
            editing: null
        };
    }

    public save(todoToSave : ITodo, text : String) {
        this.props.model.save(todoToSave, text);
        this.setState({editing: null});
    }
```

```
    var todoItems = shownTodos.map((todo) => {
        return (
            <TodoItem
                key={todo.id}
                todo={todo}
                onToggle={this.toggle.bind(this, todo)}
                onDestroy={this.destroy.bind(this, todo)}
                onEdit={this.edit.bind(this, todo)}
                editing={this.state.editing === todo.id}
                onSave={this.save.bind(this, todo)}
                onCancel={ e => this.cancel() }
            />
        );
    });

});
```

The state starts with a default value when a Component mounts.

The state suffers from mutations in time (mostly generated from user events).

A Component manages its own state internally (you could say the state is private).

React.Component: The state problem

The Problem:

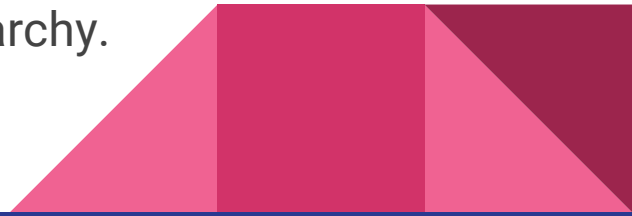
Managing the internal state of the components makes them hard to maintain.

Components stop being a “pure function” when we use state: (porps) => HTML

The solution:

Move the state to the parent component and inject it as properties.

Inject properties from the top of the component’s hierarchy.



Redux

Predictable **state container** for JavaScript apps.

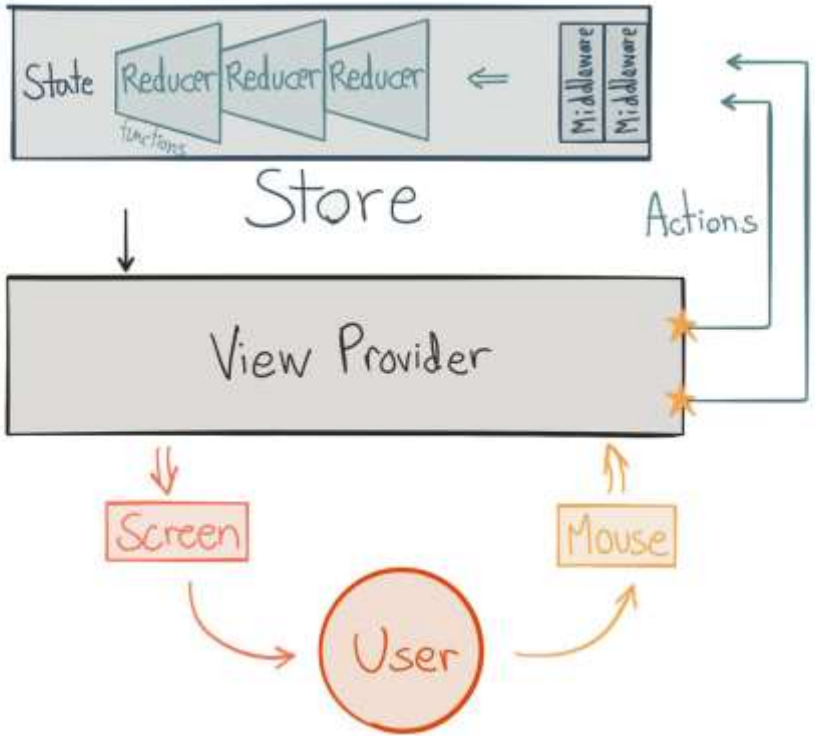
Applications that behave consistently.

Run in **client, server, and native**

easy to test.

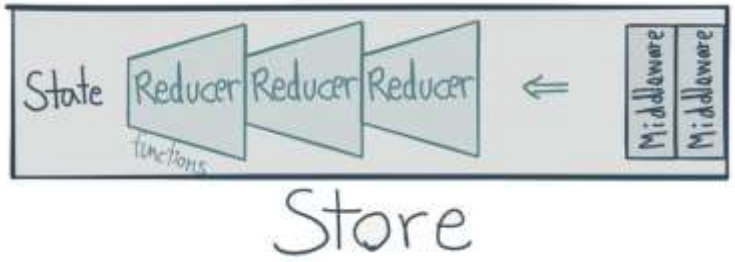
Great developer experience.

Unidirectional data flow.



Making the state “global”: The Store

```
import * as thunk from "redux-thunk";
import * as createHistory from "history/lib/createBrowserHistory";
import { createStore, applyMiddleware, compose } from "redux";
import { reduxReactRouter } from "redux-router";
import routes from "../config/routes.tsx";
import rootReducer from "../reducers/root_reducer";

let win: any = window;
let router = reduxReactRouter({ routes, createHistory });
let middleware = applyMiddleware(thunk);
let devToolsExtension = win.devToolsExtension ? win.devToolsExtension() : f => f;

const finalCreateStore = compose(
    middleware,
    router,
    devToolsExtension
)(createStore);

function configureStore(initialState: any = {}) {
    return finalCreateStore(rootReducer, initialState);
}

export default configureStore;
```



Making the state “global”: The provider

```
import * as React from "react";
import { Provider } from "react-redux";
import { ReduxRouter } from "redux-router";
import configureStore from "../../config/store.ts";

const store = configureStore();

class Root extends React.Component<any, any> {

    constructor(props) {
        super(props);
    }

    render() {
        return (
            <Provider store={store}>
                <ReduxRouter />
            </Provider>
        );
    }
}
```

The provider owns the store and makes the it available to other components (in the component hierarchy below) using the @connect() calls.

Redux: @Connect

```
import * as React from "react";
import { connect } from "react-redux";
import { Link } from "react-router";
import { bindActionCreators } from "redux";
import * as lobbyActions from "../../actions/lobby_actions";
import Loading from "../../components/loading.tsx";
import Error from "../../components/error.tsx";

function mapStateToProps(state: ApplicationProps) {
  return state.lobby;
}

function mapDispatchToProps(dispatch) {
  return { actions : bindActionCreators(lobbyActions, dispatch) };
}

@connect(mapStateToProps, mapDispatchToProps)
class LobbyPage extends React.Component<ILobbyProps, void> {

    public constructor(props) {
        super(props);
    }
```

Components decorated with @connect are aware of the Store and because of that we call them **smart components.**

Dumb components receive their properties from its parent component and are not aware of the Store.

Smart components are less reusable.

Redux: Actions and action creators

An action creator is just a function that returns an action (a JSON object). Actions must have an action type:

```
funcion selectFigure(guid) {
    return {
        type: ACTION_TYPES.SELECT_FIGURE,
        guid : guid
    };
}
```

```
export const INIT_LOBBY_START   = "INIT_LOBBY_START";
export const REMOVED_GAME       = "REMOVE_GAME";
export const ADDED_GAME         = "ADD_GAME";
export const LOGGED_IN_SUCCESS  = "LOGGED_IN_SUCCESS";
export const LOGGED_IN_ERROR    = "LOGGED_IN_ERROR";
export const SELECT_FIGURE      = "SELECT_FIGURE";
export const MOVE_FIGURE        = "MOVE_FIGURE";
export const INIT_GAME          = "INIT_GAME";
```

Using the makeActionCreator helper:

```
const selectFigure = makeActionCreator(ACTION_TYPES.SELECT_FIGURE, "guid");
```

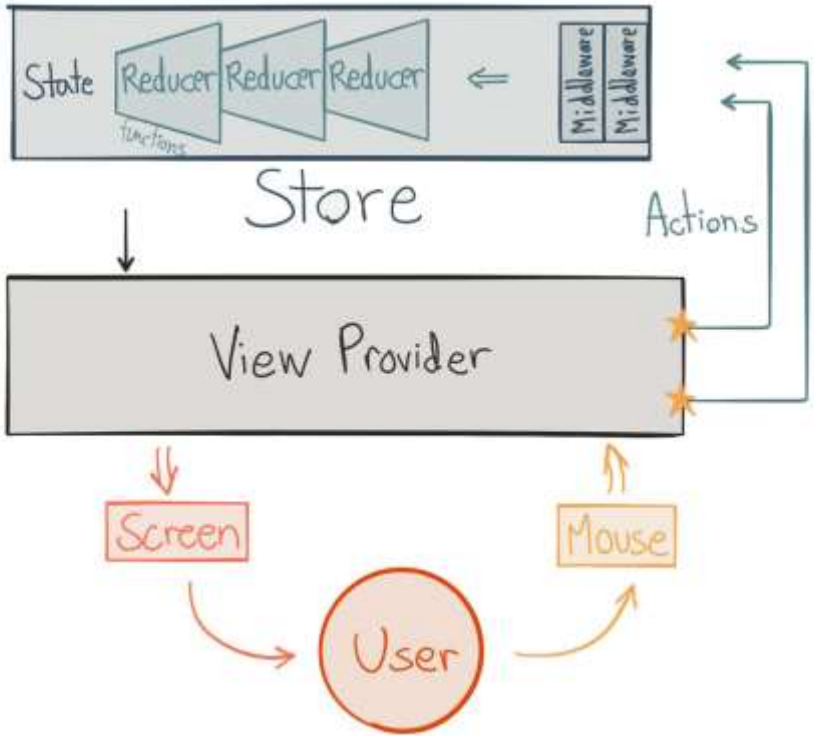
Redux: Reducers

Actions describe the fact that something happened, but don’t specify how the application’s state changes in response. This is the job of a reducer.

```
function gameReducer(state = defaultState, action) {

    switch (action.type) {

        case ACTION_TYPES.INIT_GAME:
            return action.game;

        case ACTION_TYPES.SELECT_FIGURE:
            return chessUtils.selectFigure(state, action);

        case ACTION_TYPES.MOVE_FIGURE:
            return chessUtils.moveFigure(state, action);

        default:
            return state;
    }
}
```

(Previous State, Action) => new

Redux: Summary



The Redux ecosystem: Asynchronous actions

Thunk middleware for Redux https://github.com/gaearon/redux-thunk

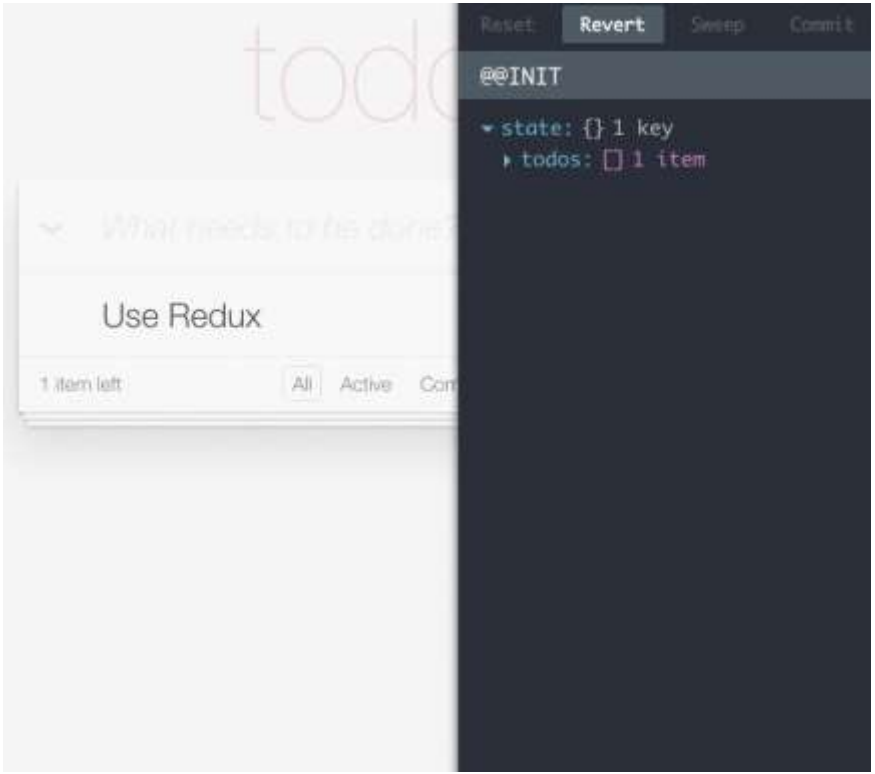
```
export const fetchAppetiteBegin = actionCreatorFactory(ACTION_TYPES.FETCH_FILTERS_BEGIN);
export const fetchAppetiteSuccess = actionCreatorFactory(ACTION_TYPES.FETCH_FILTERS_SUCCESS, "appetite");
export const fetchAppetiteError = actionCreatorFactory(ACTION_TYPES.FETCH_FILTERS_ERROR, "error");

export const fetchAppetiteAsync =  function() {
  return function (dispatch) {

    dispatch(fetchAppetiteBegin());

    webServices.appetite.get()
        .then((data) => {
            var result= data;
            dispatch(fetchAppetiteSuccess(result));
            }).catch((error) => {
                dispatch(fetchAppetiteError(error));
     });
  }
}
```

The Redux ecosystem: Router

React router for Redux https://github.com/rackt/react-router-redux

```
import * as React from "react";
import { Route, IndexRoute } from "react-router";

// LAYOUT
import AppLayout from "../containers/layout/app_container.tsx";

// PAGES
import GamePage from "../containers/pages/game_page.tsx";
import LobbyPage from "../containers/pages/lobby_page.tsx";

let routes = (
  <Route component={AppLayout}>
    <Route path="/" component={LobbyPage} />
    <Route path="/game/:id" component={GamePage} />
  </Route>
);

export default routes;
```

The Redux ecosystem: Development tools



Redux Devtools
https://github.com/zalmoxisus/redux-devtools-extension

React Hot Loader
http://gaearon.github.io/react-hot-loader/

Questions?

Thanks!

Contact me at www.remojansen.com or @OweR_ReLoaDeD