

DDD with CQRS and Redux

A wireframe illustration of a pear on the left and an apple on the right, both rendered in a low-poly, geometric style. The background is light gray with scattered white dots and faint circular patterns.

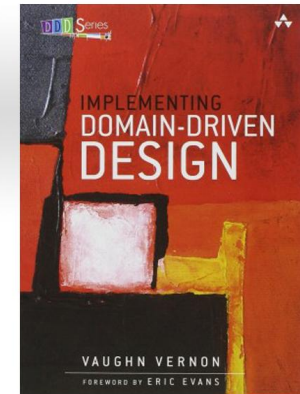
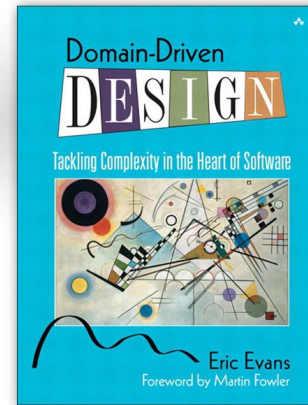
Domain Driven Design Cologne/Bonn Meetup / 17.07.17

Christoph Baudson / @sustainablepace

REWE digital

Christoph Baudson

- **Software developer** at REWE Digital since 08/2015
- [@sustainablepace](https://twitter.com/sustainablepace)
- sustainablepace.net



Agenda

- 1) **Redux**
- 2) **Redux live coding**
- 3) **Redux and Flux**
- 4) **React-Redux live coding**
- 5) **React-Redux and CQRS**
- 6) **Domain driven design: Combining reducers**

A wireframe sphere composed of many small triangles, rendered in a light gray color. It is positioned on the left side of the image, surrounded by a faint, larger wireframe sphere and scattered small gray dots.

Redux

A wireframe banana composed of many small triangles, rendered in a light gray color. It is positioned on the right side of the image, surrounded by a faint, larger wireframe banana and scattered small gray dots.

Redux

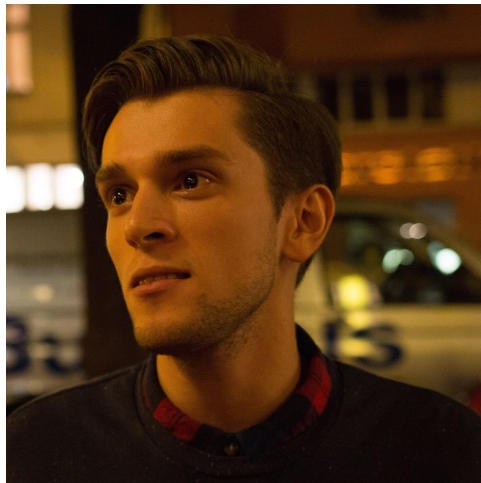
“My goal was to create a **state management library** with **minimal API** but completely **predictable behavior**”

Dan Abramov, creator of Redux

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

[The Changelog #187](#)

[Getting Started With Redux](#)



Redux - Three Principles

- The **state** of your whole application is stored in an object tree within **a single store**.
- The **only way to change the state is to emit an action**, an object describing what happened.
- To specify how the **state tree is transformed by actions**, you write pure **reducers**.

<http://redux.js.org/docs/introduction/ThreePrinciples.html>

Reducer

- “the **most important concept** in Redux”
- Inspired by Elm updaters
- type Reducer<**S**, **A**> = (state: **S**, action: **A**) => **S**
- must be **pure functions**
 - exact same output for given inputs
 - free of side-effects
 - do not put API calls into reducers!
- make state mutations **predictable!**

<http://redux.js.org/docs/Glossary.html#reducer>

A wireframe sphere is positioned on the left side of the image, and a wireframe banana is on the right. Both objects are rendered in a low-poly, geometric style with a light gray wireframe and a semi-transparent gray fill. The background is a light gray gradient with scattered small white dots.

Redux live coding

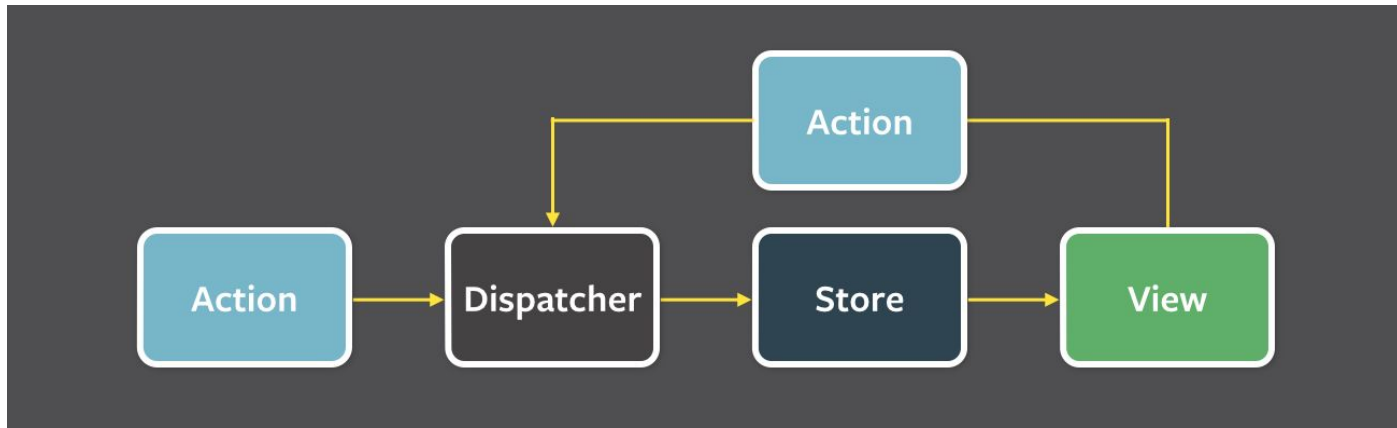
Components and concepts

- [Forsyth-Edwards-Notation \(FEN\)](#)
- [Chess.js](#)
- Redux

Redux and Flux

Flux

- a **pattern** for **managing data flow** in your application
- most important concept is that **data flows in one direction**



<https://github.com/facebook/flux/tree/master/examples/flux-concepts>

Flux and Redux

Flux	Redux
Action	Action
A single Dispatcher	No Dispatcher (but a (Redux) Store has a dispatch method)
Many (Flux) Stores	A single (Redux) Store, many Reducers
State is mutated	State is immutable
View	Listener

<http://redux.js.org/docs/introduction/PriorArt.html#flux>

React-Redux

Presentational and Container components

	Presentational	Container
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

CQRS and React-Redux

CQRS	React-Redux
Command	MapDispatchToProps
Query	MapStateToProps

```
connect (mapStateToProps, mapDispatchToProps) (Chessdiagram)
```

A wireframe sphere with a semi-transparent grey interior, surrounded by a cloud of small grey dots.A wireframe banana with a semi-transparent grey interior, surrounded by a cloud of small grey dots.

React-Redux live coding

Components and concepts

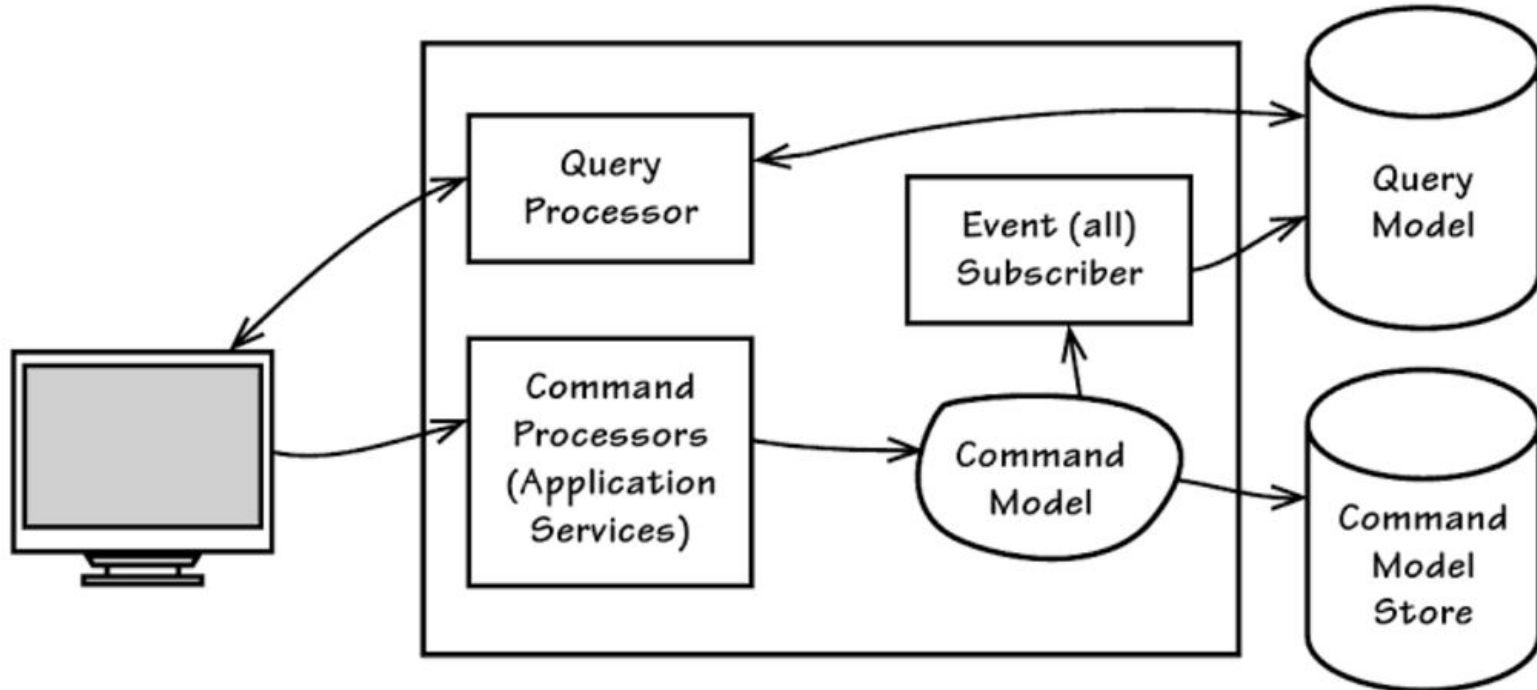
- [React](#)
- [React-Redux](#)
- [React-chessdiagram](#)
- Forsyth-Edwards-Notation (FEN)
- Chess.js
- Redux

A wireframe sphere with a semi-transparent grey interior, surrounded by a cloud of small grey dots.A wireframe banana with a semi-transparent grey interior, surrounded by a cloud of small grey dots.

React-Redux and CQRS

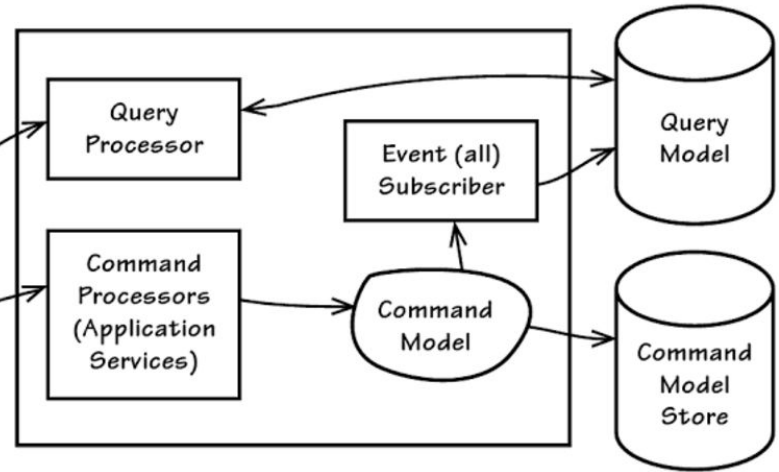
CQRS

“Implementing DDD”, Vaughn Vernon

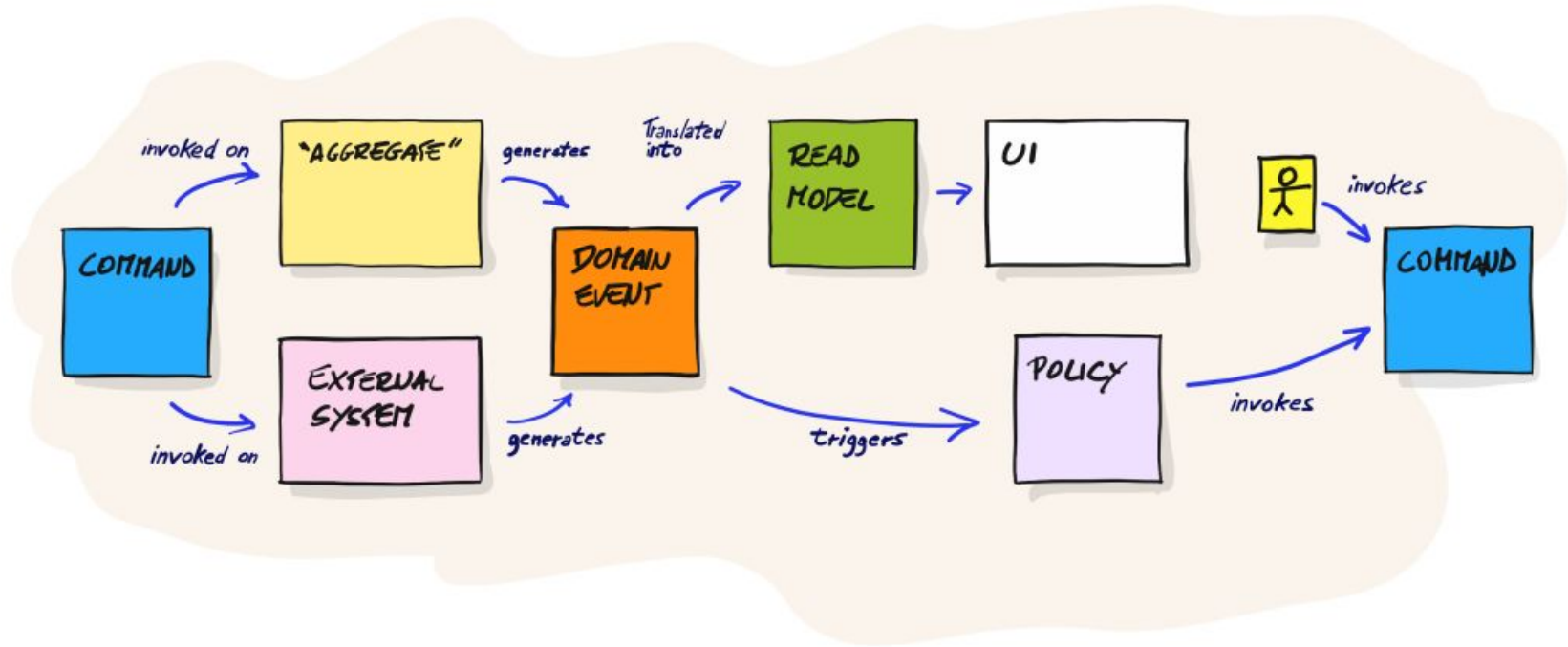


CQRS and React-Redux

CQRS	React-Redux
Command Processor	MapDispatchToProps
Command Model	Action Creator, Action
Command Model Store	Middleware
Event Subscriber	Reducer
Query Model	State
Query Processor	MapStateToProps



Event Storming



Event Storming vs. React-Redux

Event Storming	(React) Redux
Command	MapDispatchToProps Methods
Aggregate	Action Creator
Domain Event	Action
Read Model	Reducer → State
UI	MapStateToProps: State → Props

DDD: React + Redux =



Components and concepts

- [Redux combineReducers](#)
- React
- React-Redux
- React-chessdiagram
- Forsyth-Edwards-Notation (FEN)
- Chess.js
- Redux

React and Redux hand in hand

- Files grouped by bounded context in separate folders
- Domain slicing via
 - Sub-components (React)
 - combineReducers (Redux)
- Tree hierarchy with root reducer and component



Elegant and simple

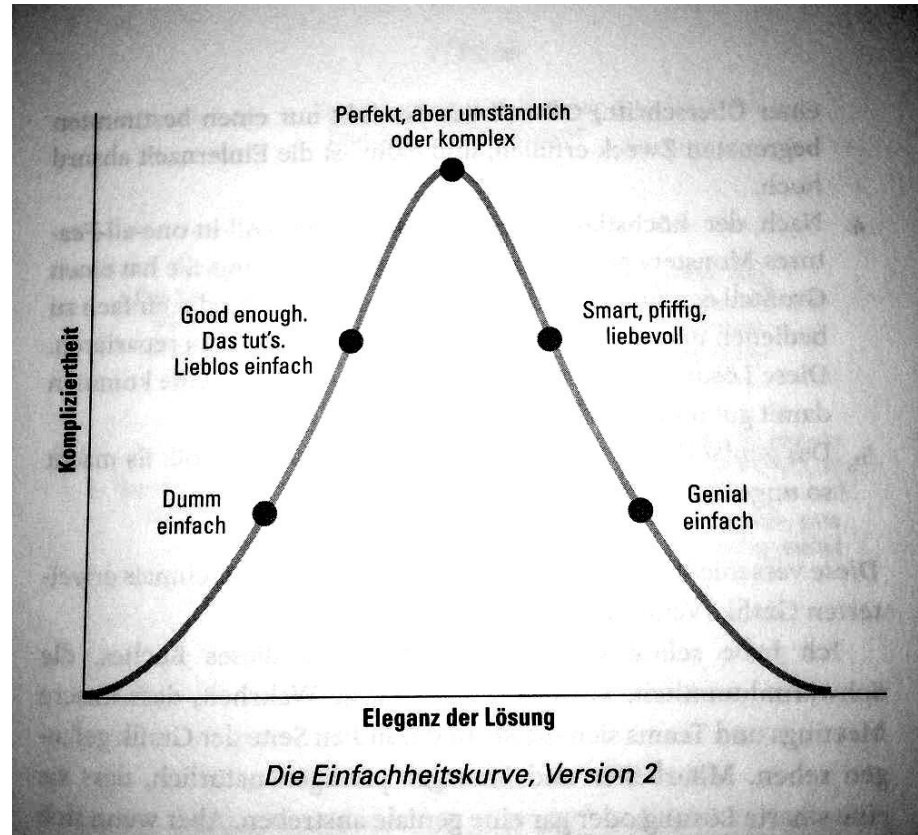
Redux is

“genial einfach”

not

“dumm einfach”

Gunter Dueck “Schwarmdumm”





Vielen DDDank :)

Christoph Baudson / @sustainablepace

Demo, slides, source code at <http://chess.baudson.de>