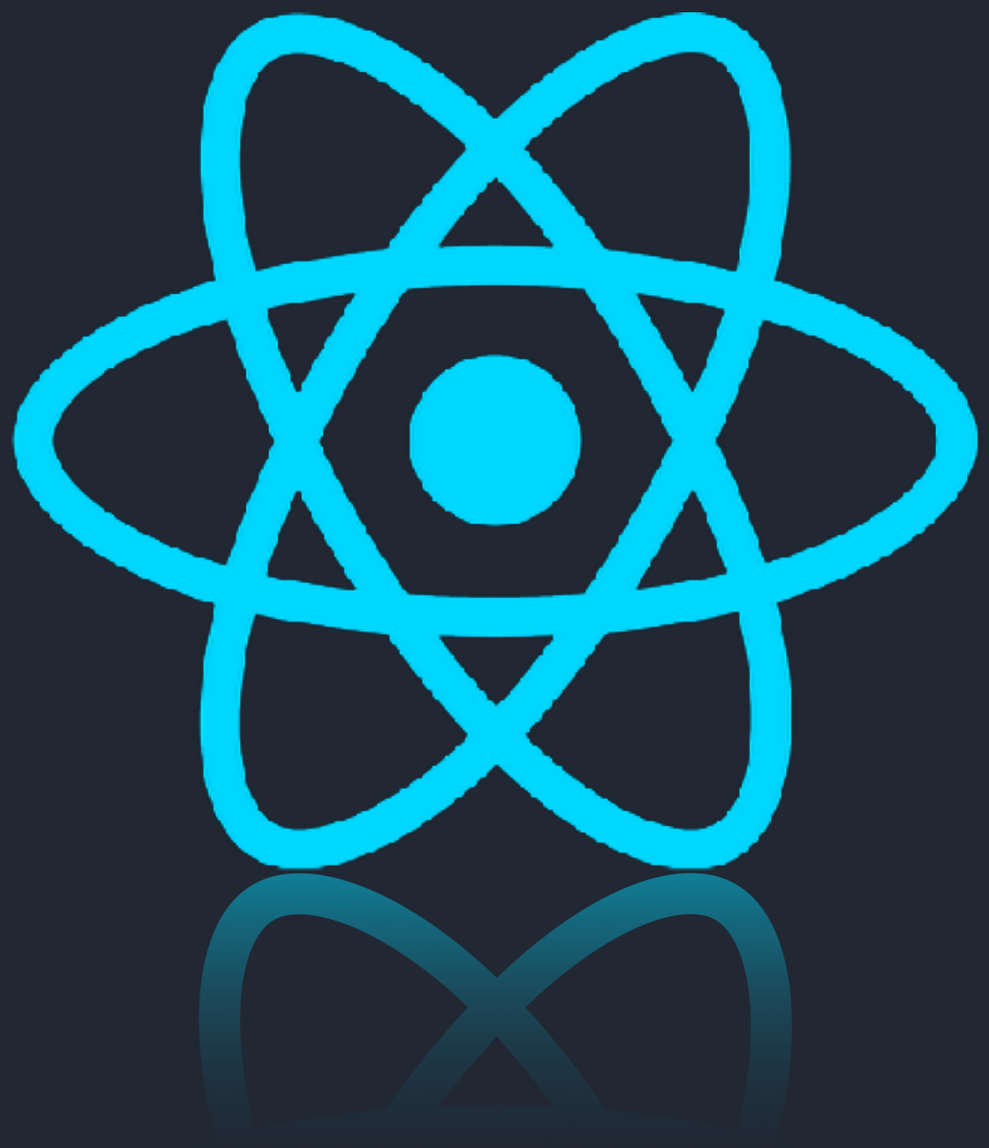


Front-End Entwicklung mit React

Mail: jonas.bandi@ivorycode.com

Twitter: @jbandi

Client Side Routing

Client Side Routing in a SPA

The traditional web is built on the concept of linked documents (URL, bookmarking, back-button …)

In a SPA the document is just the shell for an application. When you navigate away from the document, the application is "stopped".

As a consequence a SPA should "emulate" the traditional user-experience on the web:

- navigate via urls, links & back-button
- bookmarks and deep links

Client-Side Routing in a SPA

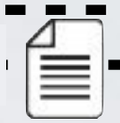
http://thedomain:80/app#/first

Browser

Server

initial request
http://thedomain:80/app

html-shell



html-shell is loaded

assets

"navigate" to "first" screen

optional ajax request
http://thedomain:80/api/data/first

data

load data &
render screen with JS

http://thedomain:80/app#/**first**

navigation

"navigate" to "second" screen

optional ajax request
http://thedomain:80/api/data/second

data

load data &
render screen with JS

http://thedomain:80/app#/**second**

app bootstrapping

http://localhost:8080/app

<head>
</head>
<body>
... **App Component**

```
<Route path="path1"
    component={Comp1}>
<Route path="path2"
    component={Comp2}>
```

...
</body>

SPA Shell

/app/path1

compontent 1

/app/path2

component 2

React Router

React Router is a collection of navigational components.

With React Router routing is dynamic, it takes place as the app is rendering.
Routing is not configured statically during initialization.
This allows a very neat integration with the React component model.

```
npm install react-router-dom
```

https://reacttraining.com/react-router/

React Router

Routes are represented as components.

```
import {BrowserRouter as Router,
      Route, Link, Switch} from 'react-router-dom';

<Router>
  <div>
    <ul>
       <li><Link to="/">Home</Link></li>
       <li><Link to="/about">About</Link></li>
    </ul>
    <Switch>
       <Route exact path="/" component={Home}/>
       <Route exact path="/about" component={About}/>
    </Switch>
  </div>
</Router>
```

A `Router` is a container component:

A `route` renders a component if the route matches the path:

The `Router` provides some properties to all contained components:
`match, location, history`

https://reacttraining.com/react-router/

Backend Access

axios

axios is a promise based HTTP client library for the browser and node.js

```
axios.get(API_URL)
   .then((response) => console.log(response))
   .catch((error) => console.log(error));
```

```
axios.post(API_URL, payload)
   .then((response) => console.log(response))
   .catch((error) => console.log(error));
```

```
axios.delete(`${API_URL}/${id}`)
   .then((response) => console.log(response))
   .catch((error) => console.log(error));
```

https://github.com/axios/axios

Advanced State Management

Redux

MobX

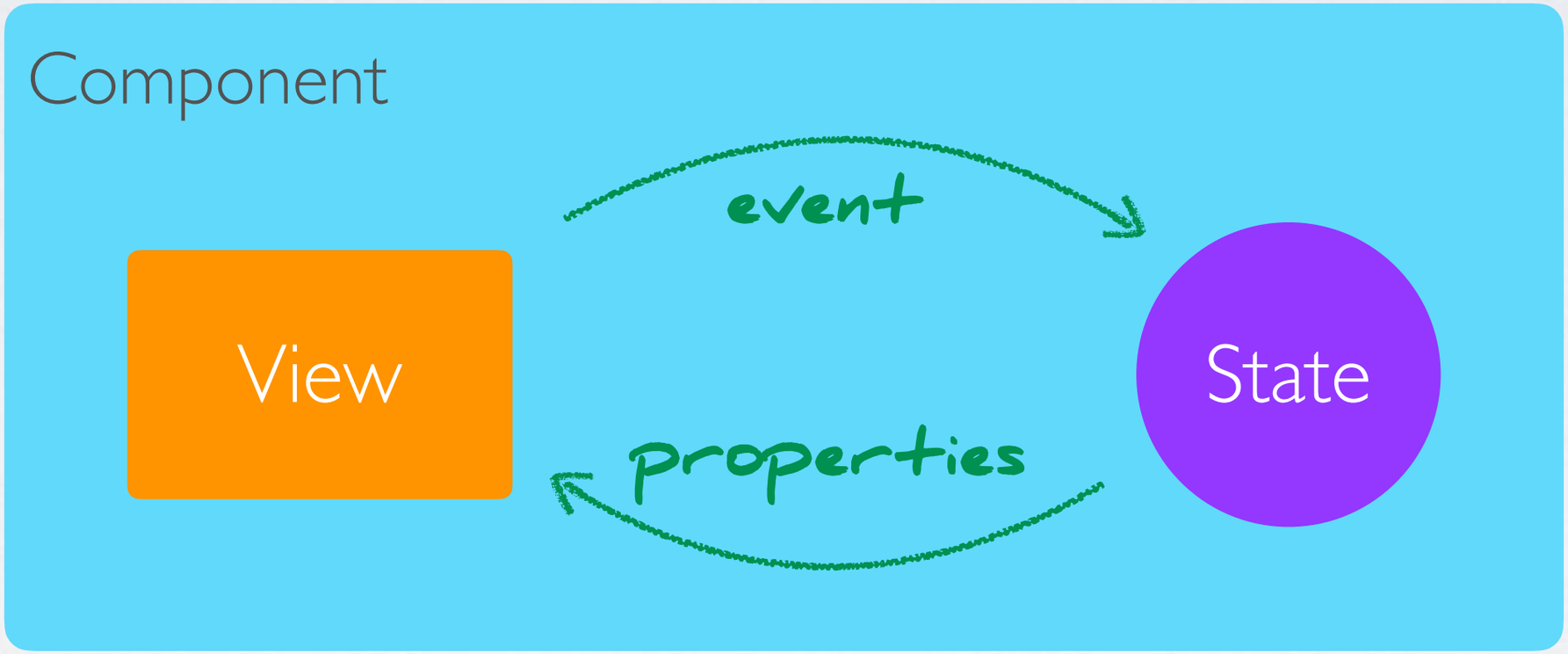
Managing state in a complex web application is not a solved problem! New ideas and implementations are popping up all the time ...

https://stateofjs.com/2017/state-management/results

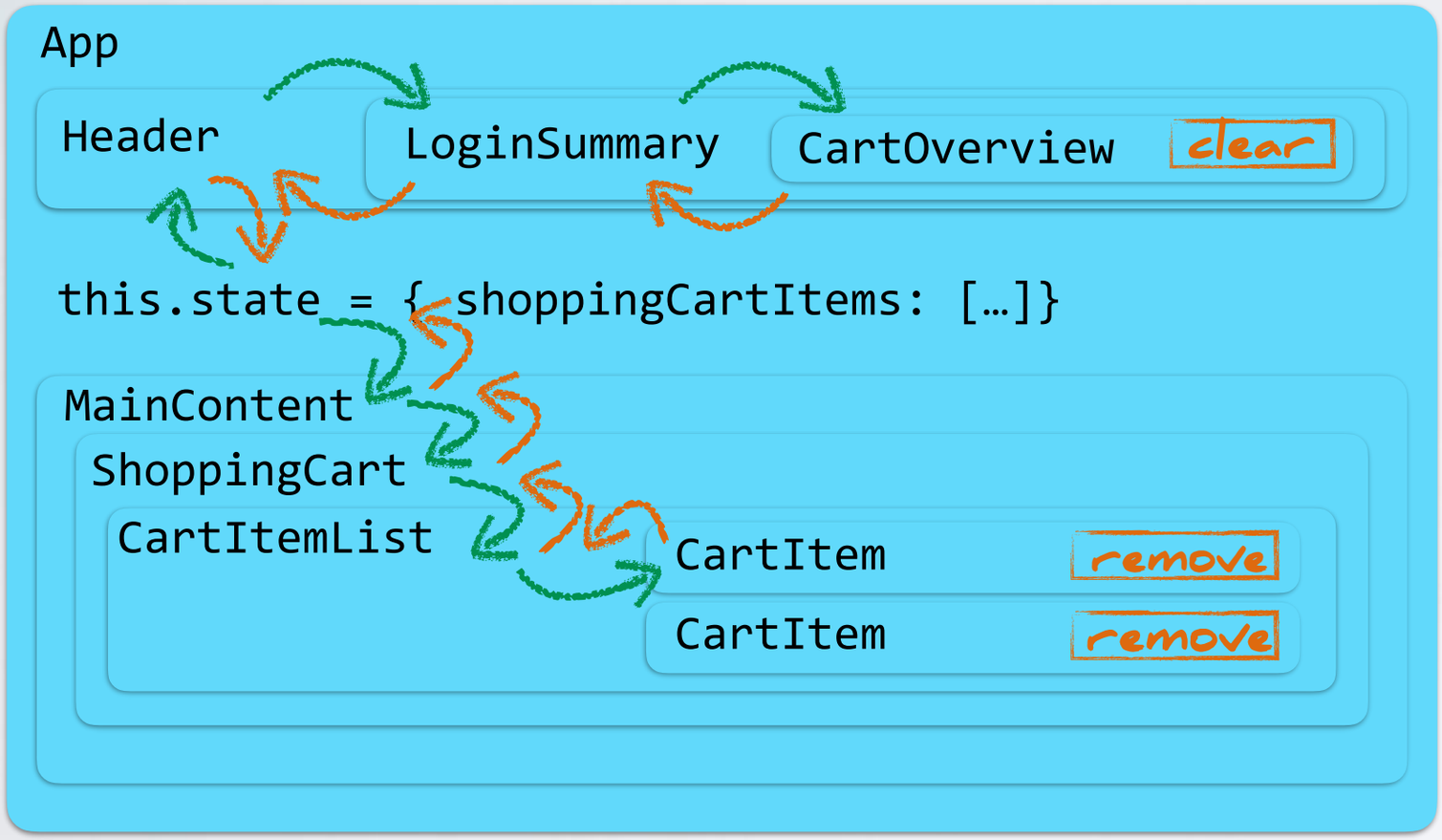
MobX State Tree: https://github.com/mobxjs/mobx-state-tree
Immer: https://github.com/mweststrate/immer
Satchel: https://github.com/Microsoft/satcheljs
Rx & Recompose: https://github.com/acdlite/recompose
...

A single component

Managing state in a component is simple:



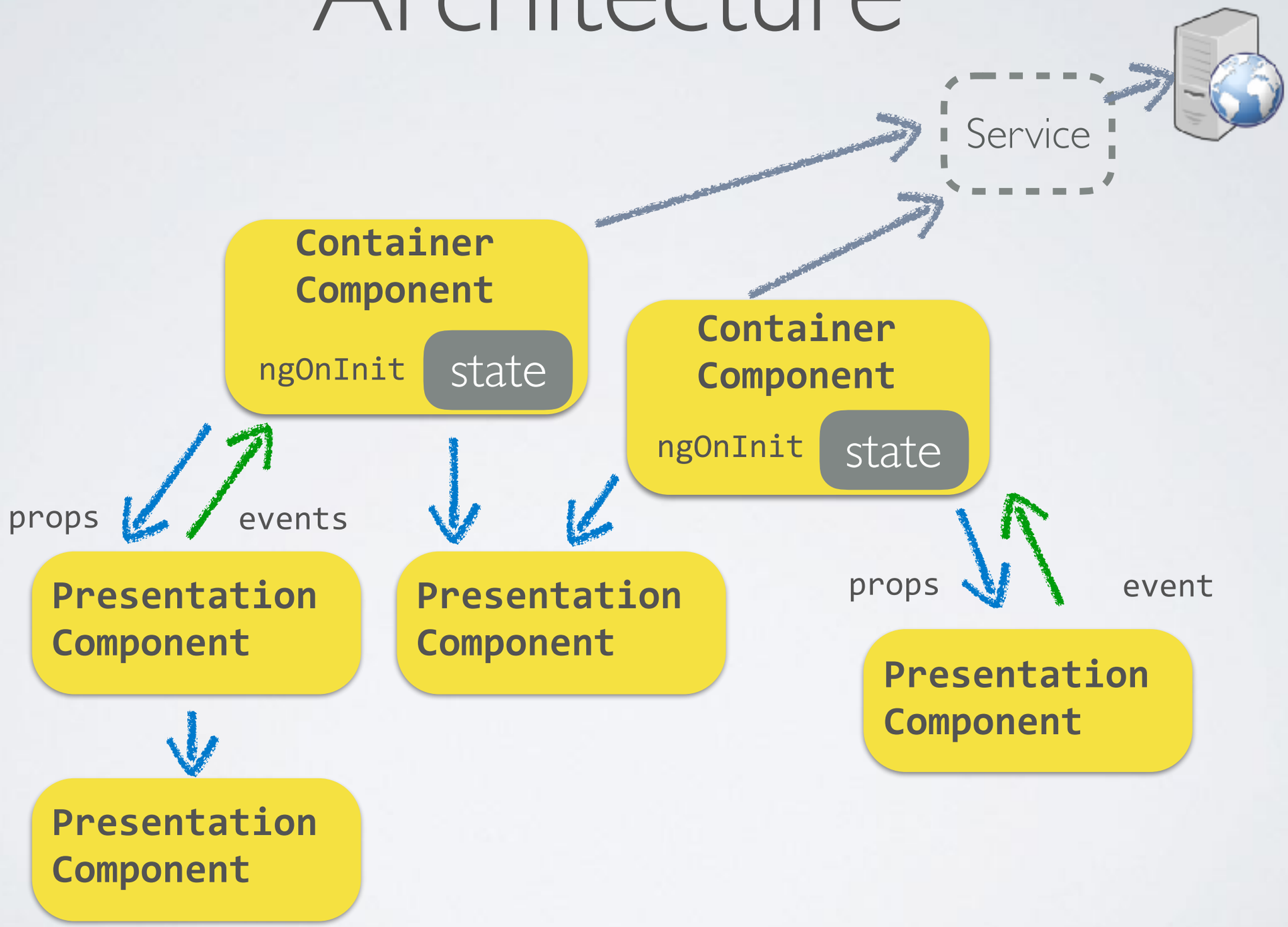
State Management



Challenges when managing state in a component tree:

- Multiple components may depend on the same piece of state.
- Different components may need to mutate the same piece of state.

State Management: Component Architecture



https://www.robinwieruch.de/learn-react-before-using-redux/

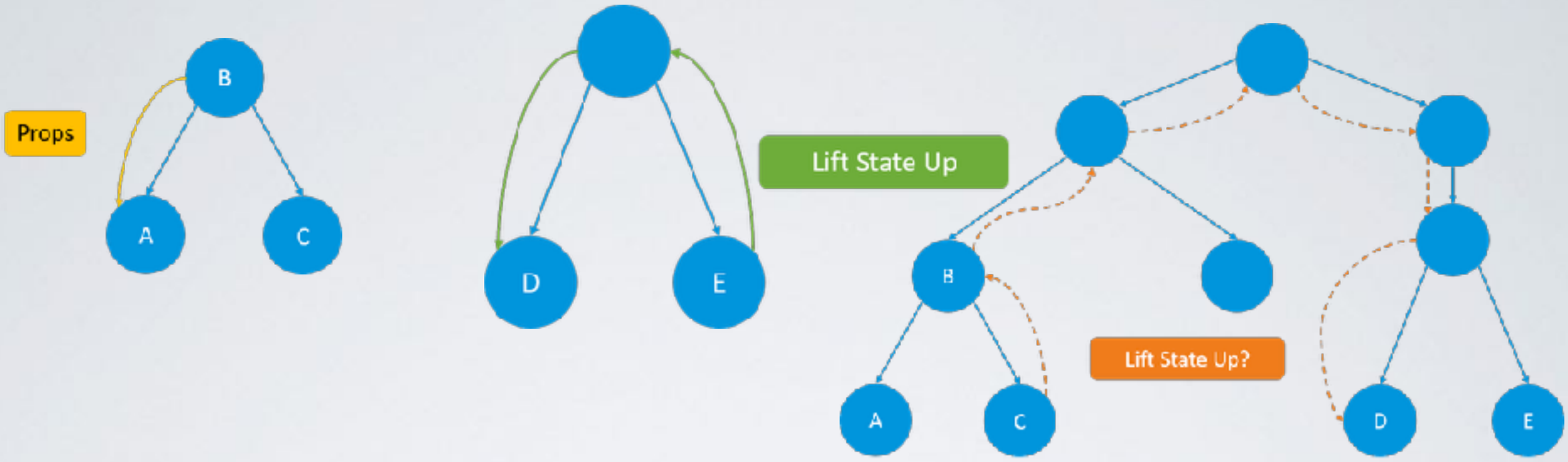
Container vs. Presentation Components

Application should be decomposed in container- and presentation components:

Container	**Presentation**
Little to no markup	Mostly markup
Pass data and actions down	Receive data & actions via props
typically stateful / manage state	mostly stateless
	better reusability

aka: Smart- vs. Dumb Components

https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0



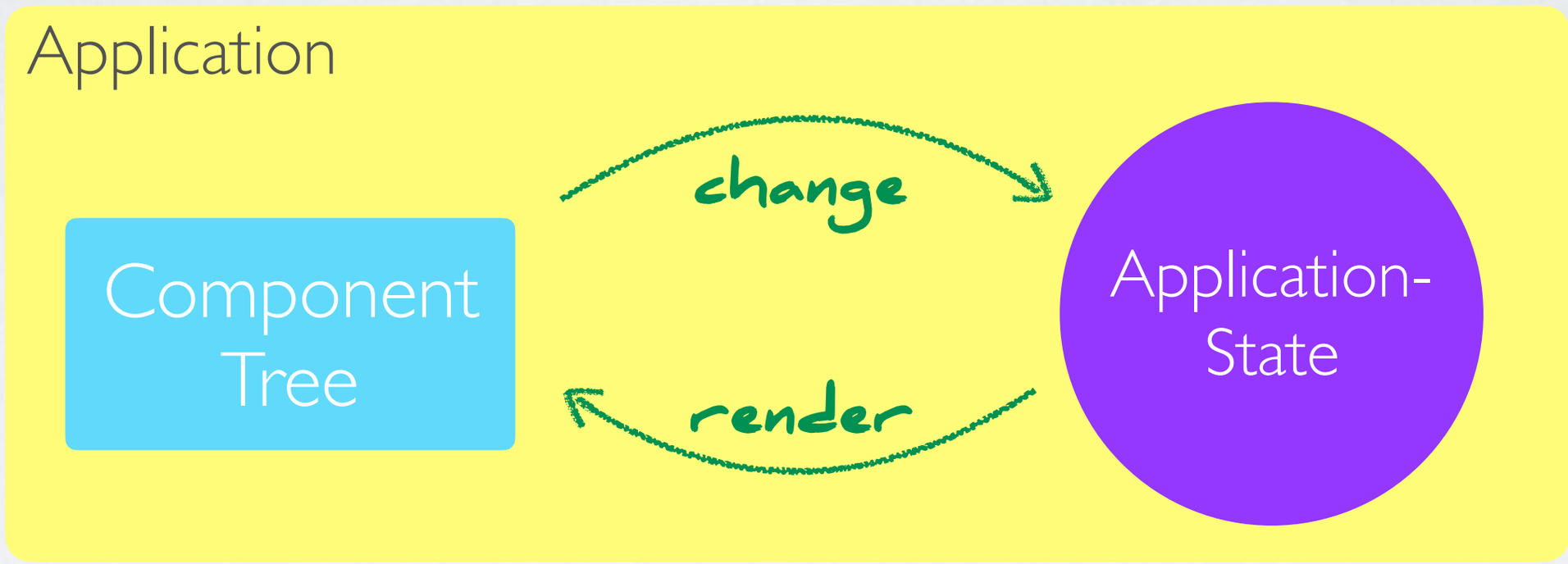
"Thinking in React" approaches data flow by using
props to pass data around and lifting state up.
At a certain point though, doing this will hurt the
"Composability and Reusablity" principles

https://reactjs.org/docs/thinking-in-react.html
https://reactjs.org/docs/lifting-state-up.html

https://medium.com/@nimelrian/thinking-in-react-a-paradox-statement-33c19e2eb9e2

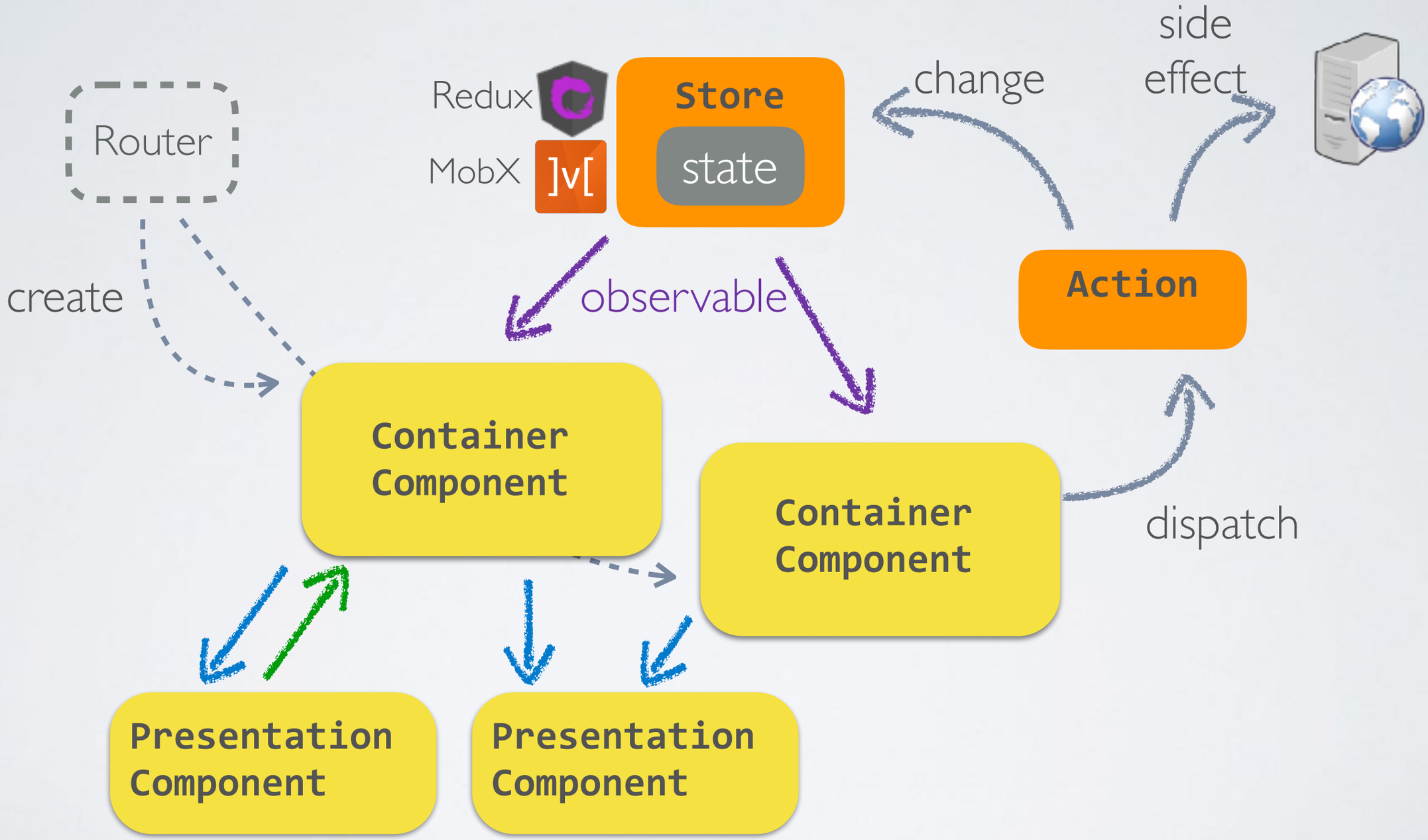
Application with State Container

A state container extracts the shared state out of the components, and manages it in a global singleton.



The component tree becomes a big "view", and any component can access the state or trigger actions, no matter where they are in the tree!

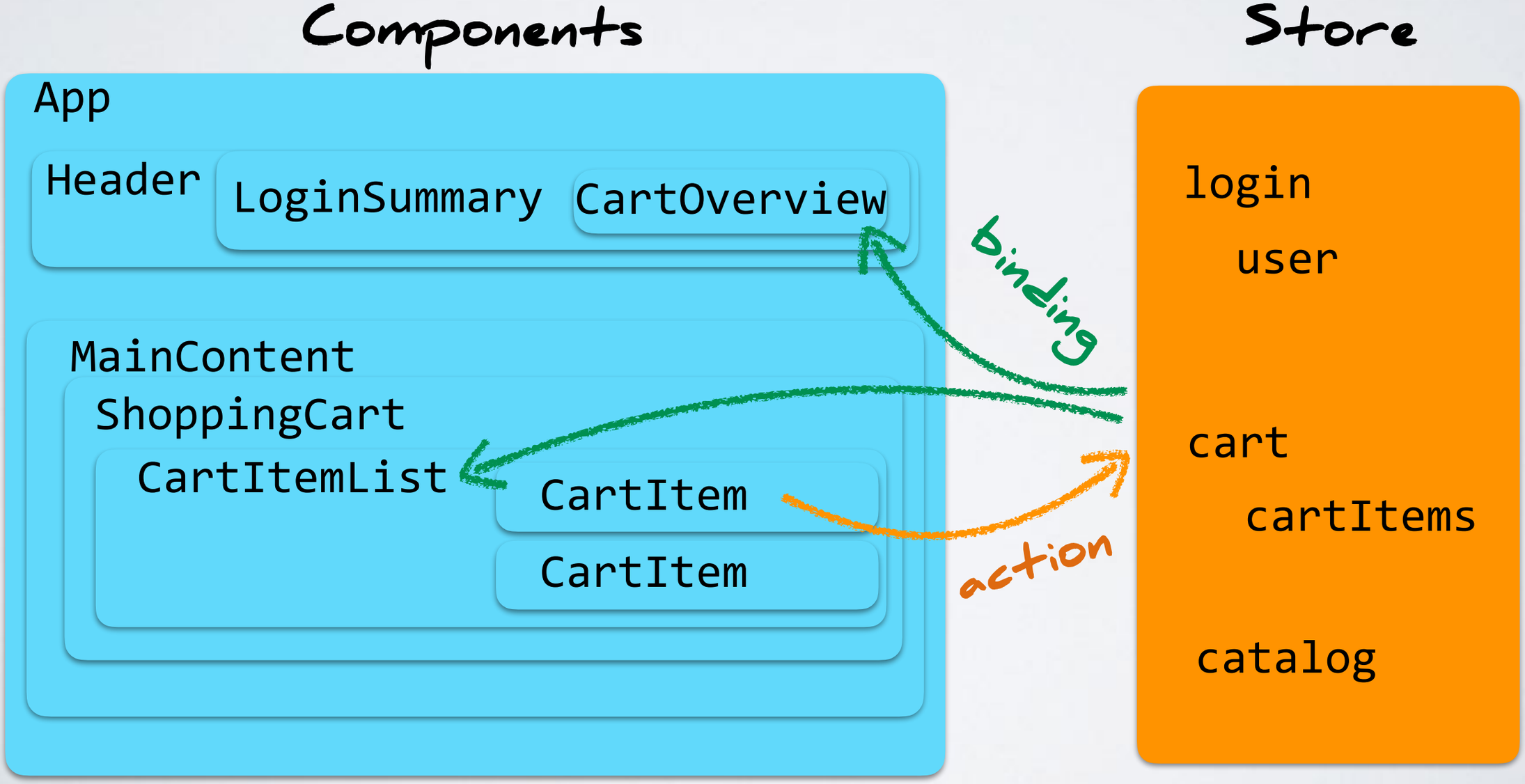
State Management: State Container



ngrx: https://github.com/ngrx/
MobX: https://mobx.js.org/

Managing State with a State Container

State can be managed outside the components.
Components can be bound to state.



Do I need a state container?

Holding state in React components works well when the state is hierarchical and more or less matches the component structure.

If distant distant parts of the app want to have access to the same state, you'll end up with a bunch of very large components at the top of the component tree that pass a myriad of props down through some intermediate components that don't use them, just to reach a few leaf components that actually care about that data.
In this scenario you can extract state & state management into an external container and connect leaf components directly.

http://stackoverflow.com/questions/36631761/when-should-i-add-redux-to-a-react-app
https://medium.com/@dan_abramov/you-might-not-need-redux-be46360cf367

Redux is often (usually?) not needed

https://medium.com/@daveford/redux-flux-is-often-usually-not-needed-13c734b416cd

https://twitter.com/ryanflorence/status/861603318824550400

Dan Abramov
@dan_abramov

Following

I think Redux is popular partly because of deficiencies in React state model. I also think local component state is ultimately better.

10:05 PM - 26 Aug 2016

19 Retweets **43** Likes

2 19 43

Tweet your reply

Dan Abramov @dan_abramov · 26 Aug 2016

Replying to @dan_abramov

I also hoped that Redux would encourage innovation around React state model, not create a wall of “best practices” and “right ways”.

5 13 31

Dan Abramov @dan_abramov · 26 Aug 2016

Think: what does Redux give you? What properties are at play? What properties does React state model lack? Can we fix this?

19 2 16

https://twitter.com/dan_abramov/status/769325010691428353

https://twitter.com/housecor/status/962754389533429760

TJ Holowaychuk 🐥
@tjholowaychuk

Follow

my js state management library: {}

2:51 AM - 29 Jan 2018

328 Retweets **1,575** Likes

49 328 1.6K

Redux Basics: Array Reduce

```
var a = [1,2,3,4,5];

var result = a.reduce(
  // reducer function
  (acc, val) => {
    const sum = acc.sum + val;
    const count = acc.count + 1;
    const avg = sum/count;
    return {sum, count, avg};
  },
  // state object
  {sum:0, count:0, avg:0}
);

console.log('Statistics:', result);
```

The reducer function is a pure function.

What is Redux?

Redux is a predictable state container for JavaScript apps.

Created in 2015 by Dan Abramov.

Redux itself is the implementation of a very simple concept.
There is a big ecosystem around Redux (middleware).

Redux is very small (~2KB).

Redux is widely used in the React community. Technically it is not tied to React.

Redux can be used with Angular. For Angular there is also ngrx, which is an alterative implementation of the same concept.
(https://github.com/angular-redux/ng-redux and https://github.com/ngrx)

http://redux.js.org/

State Reduce

Redux implements store mutations in a "functional way".

(old state , action) => new state

aka: "reducer function"

Using Redux to manage the Global State

Demo: `32-redux-basics/12-state-redux`

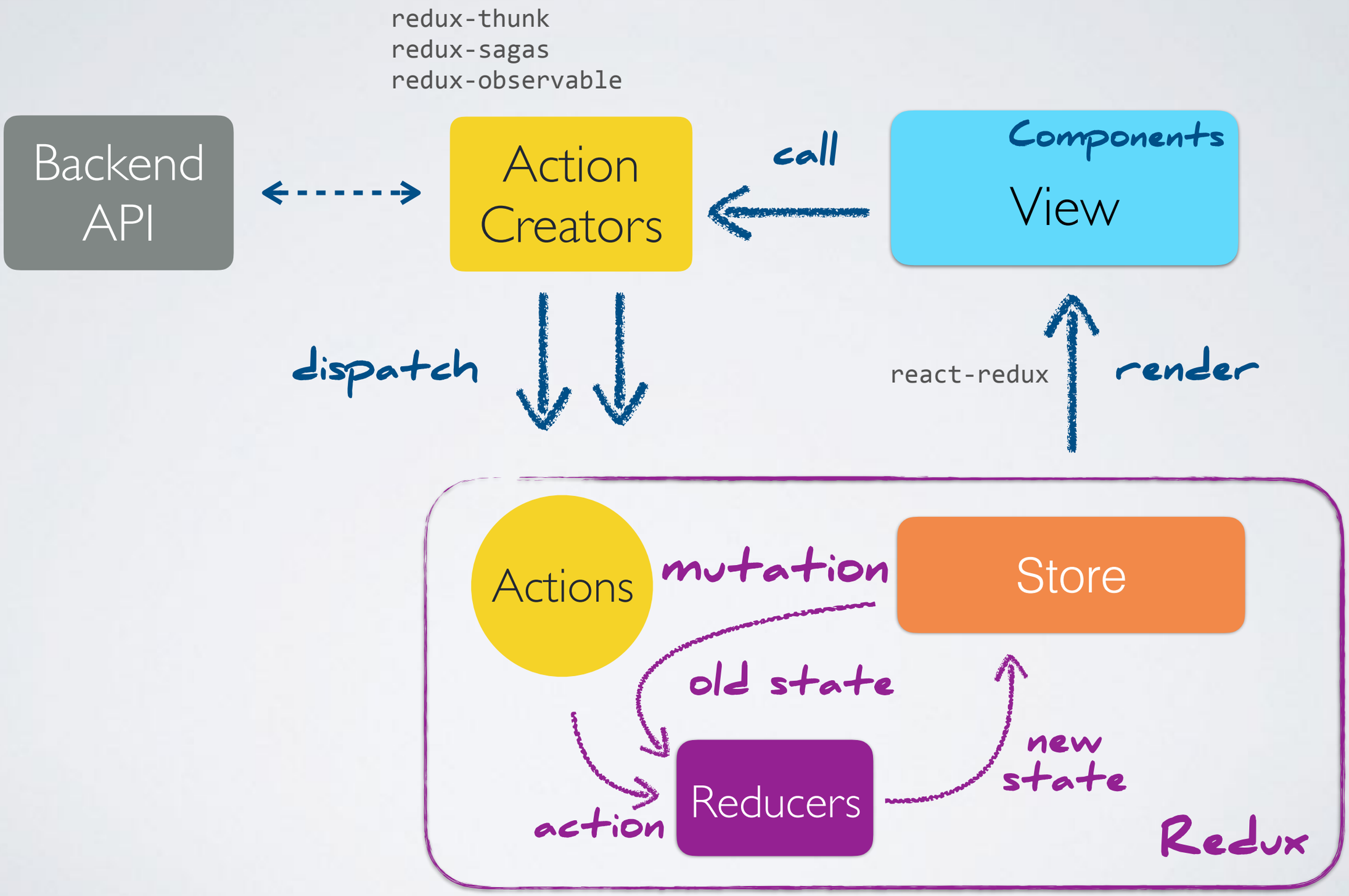
Inspect the example.

Run the example:

```
npm install
npm run:watch
```

Implement the functionality to mark a todo as completed.

Redux Architecture



Redux Thunk

https://github.com/gaearon/redux-thunk

Redux Thunk is a middleware for Redux that enables action creators can return a function.

With thunks you can:

- dispatch multiple actions
- dispatch actions asynchronously at a later point in time
- dispatch actions based on certain conditions

Alternatives:

- Redux Saga: https://redux-saga.js.org/
- Redux Observables: https://redux-observable.js.org/

MobX

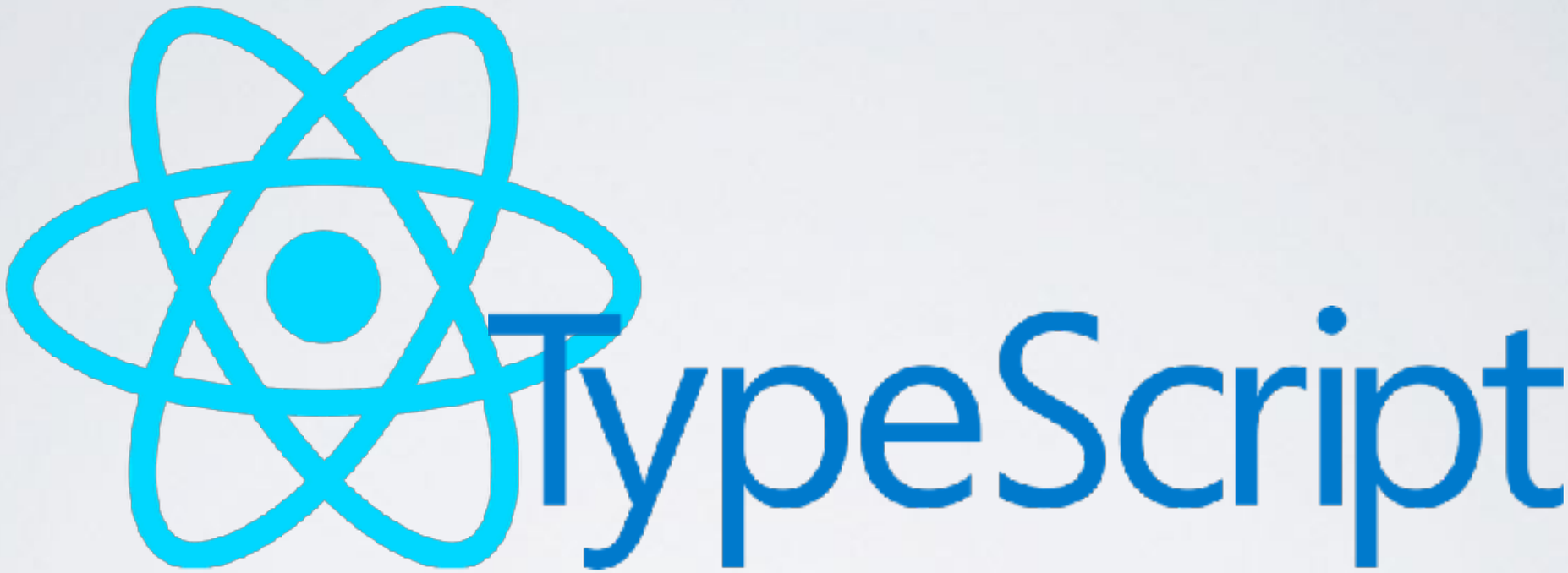
MobX is a state management library.

https://mobx.js.org/

Interesting Library: MobX State Tree https://github.com/mobxjs/mobx-state-tree

https://www.robinwieruch.de/redux-mobx-confusion/

React works great with TypeScript!




```
npm i -g create-react-app
create-react-app awesome-app --scripts-version=react-scripts-ts
cd awesome-app
npm start
```

https://github.com/wmonk/create-react-app-typescript

Alternative: Flow
https://flow.org/

Have Fun with React!

JavaScript / Angular / React
Schulungen & Coachings,
Project-Setup & Proof-of-Concept:

http://ivorycode.com/#schulung
jonas.bandi@ivorycode.com