

React and Redux 101

July 2018



Danielo **JEAN-LOUIS**
Front-end developer



James Ward

@_JamesWard

Follow



Discovered this sign at the W3C headquarters...



8:17 AM - 11 Feb 2015

1,992 Retweets 1,037 Likes



40



2.0K



1.0K

An (another) javascript framework

An (another) javascript framework **library!**

Sources

<http://www.nicoespeon.com/en/2015/01/pure-functions-javascript/>

<https://web.archive.org/web/20070504053354/>

http://www.ddj.com/blog/architectblog/archives/2006/07/frameworks_vs_l.html

What's React ?

- Javascript library developed, open sourced and maintained by facebook
- Designed for (large) applications **with data that change over the time**
- Released in 2013
- Dogfed by facebook with facebook and instagram

Sources

https://en.wikipedia.org/wiki/Eating_your_own_dog_food

<https://reactjs.org/docs/getting-started.html>

What's React ?

- Javascript library developed, open sourced and maintained by facebook
- Designed for (large) applications **with data that change over the time**
- Released in 2013
- Dogfed by facebook with facebook and instagram
- Relies on Virtual DOM

Sources

https://en.wikipedia.org/wiki/Eating_your_own_dog_food
<https://reactjs.org/docs/getting-started.html>

What's React ?

- Javascript library developed, open sourced and maintained by facebook
- Designed for (large) applications **with data that change over the time**
- Released in 2013
- Dogfed by facebook with facebook and instagram
- Relies on Virtual DOM —————> Extremely fast

Sources

https://en.wikipedia.org/wiki/Eating_your_own_dog_food
<https://reactjs.org/docs/getting-started.html>

What's React ?

- No MVC
- No MVVM
- No MVW
- No DM-VM-CM-VC-V*
- ...

*<https://github.com/xlasne/MVVM-C>

React is only **view**

React is only **view**

React is **components**

React components are

Reusable

Testable

Maintainable

React components are not Web components

Web components are **for strong encapsulation**

React components are **made to be sync with data**

Sources

<https://reactjs.org/docs/web-components.html>

Let's write our first component

Let's write our first component

```
import React from 'react';

export default class MyFirstComponent extends React.Component {
  render() {
    return (
      <p>It's my first component with React !</p>
    );
  }
}
```


Let's write our first component

```
import React from 'react';
```

```
export default class MyFirstComponent extends React.Component {
```

```
  render() {  
    return (  
      <p>It's my first component with React !</p>  
    );  
  }  
}
```

render() method displays
the template of the
component




Let's write our first component

```
import { render } from 'react-dom'  
import MyFirstComponent from './MyFirstComponent'
```

```
render(  
  <MyFirstComponent />,  
  document.getElementById('content')  
);
```

We start the React app
(A div can contain only one
react app)



Component's life cycle

Mount

Update

Unmount

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Component's life cycle

Mount

Update

Unmount

Error handling

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Props and state

Props and state allow to change/set component's content/data

Props are **immutable**

Props are **read-only**

Props are **passed only by component's closest parent**

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Props and state

Props and state allow to change/set component's content/data

Props are **immutable**

Props are **read-only**

Props are **passed only by the closest parent**

State is **mutable**

State is **handled by the component itself**

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Props and state

Props and state allow to change/set component's content/data

Props are **immutable**

Props are **read-only**

Props are **passed only by the closest parent**

State is **mutable**

State is **handled by the component itself**

Update component's state **will call** component's render method

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Props and state

Props and state allow to change/set component's content/data

Props are **immutable**

Props are **read-only**

Props are **passed only by the closest parent**

State is **mutable**

State is **handled by the component itself**

Update component's state **will call** component's render method

Sources

<https://reactjs.org/docs/react-component.html#the-component-lifecycle>

Do not update the state inside render!

Props and state

```
import React from 'react';
import MyChildComponent from './my-child-component';

export default class MyFirstComponent extends React.Component {
  constructor(props) {
    // [...]
    this.state = {
      hello: "el mundo"
    }

    setTimeout(this.myMethod, 5000);
  }
  myMethod() {
    this.setState({
      hello: "world"
    })
  }
  render() {
    return (
      <MyChildComponent text={this.state.hello} />
    );
  }
}
```

Example – props & state

Sources

<https://github.com/DanYellow/presentations/tree/master/react-redux-101/examples/props-and-state>

Data flow (up)

Callback function to communicate **with the closest parent**

Data flow (up) - Parent

Callback function to communicate **with the closest parent**

```
export default class MyFirstComponent extends React.Component {  
  // [...]  
  myCallback () {  
    console.log("my child talks to me!");  
  }  
  render() {  
    return (  
      <MyChildComponent  
        text={this.state.hello}  
        onClickCB={this.myCallback}  
      />  
    );  
  }  
}
```

Data flow (up) - Child

Callback function to communicate **with the closest parent**

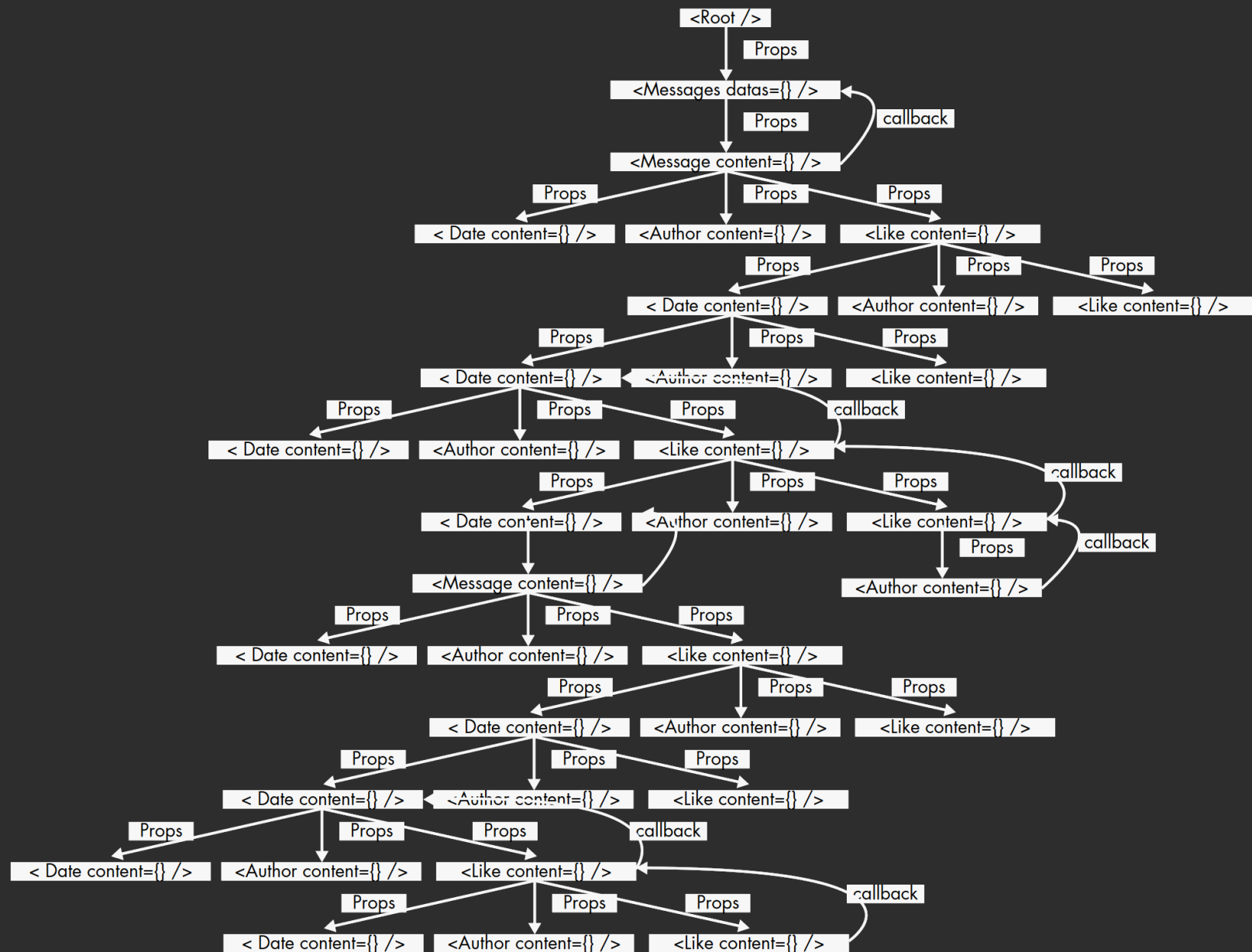
```
export default class MyChildComponent extends Component {  
  constructor(props) {  
    super(props);  
  
    this.handleClick = this.handleClick.bind(this);  
  }  
  
  handleClick() {  
    this.props.onClickCB()  
  }  
  
  render() {  
    return (  
      <p onClick={this.handleClick}>{ this.props.text }</p>  
    );  
  }  
}
```

Example – callback

Sources

<https://github.com/DanYellow/presentations/tree/master/react-redux-101/examples/callback>

Data flow (up)





ABSOLUTELY DISGUSTING

REPORT BY NEWS 9 CORRESPONDENT

Advanced topics

Stateless component

When a component doesn't need to update its state or rely on its lifecycle, it should be written as a function also called **stateless component**

Stateless component

When a component doesn't need to update its state, it should be written as a function also called **stateless component**

Import React from 'react';

```
const MyStatelessComponent = (props) => {  
  return (  
    <p>  
      I turn myself into a stateless component. I'm a stateless component!  
    </p>  
  );  
}
```

Stateless component - Advantages

- Easier to test
- Better performance (since React ≥ 16)
- No need to bind this inside methods
- Decreases bundle size within the app

Debugging

Browser extension for React (Firefox and Chrome)

Create React App

- Wonderful tool to set up a React project in less than 1 min
- Create dev and prod env
- Provide linters and unit tests setup

Sources

<https://github.com/facebook/create-react-app>

First part summary

React is **only view in the MVC pattern**

React **relies on VDOM, it's fast thank to it**

JSX is strongly encouraged for templating

React allows a full control of the component with its lifecycle

Props **allows parent to set/update children's data**

Props are immutables

First part summary

React is **only view in the MVC pattern**

React **relies on VDOM, it's fast thank to it**

JSX is strongly encouraged for templating

React allows a full control of the component with its lifecycle

Props **allows parent to set/update children's data**

Props are immutables

State is updated by the component itself

State is mutable

First part summary

React is **only view in the MVC pattern**

React **relies on VDOM, it's fast thank to it**

JSX is strongly encouraged for templating

React allows a full control of the component with its lifecycle

Props **allows parent to set/update children's data**

Props are immutables

State is updated by the component itself

State is mutable

The data flow (up) in React is a mess... except if we use...



Redux

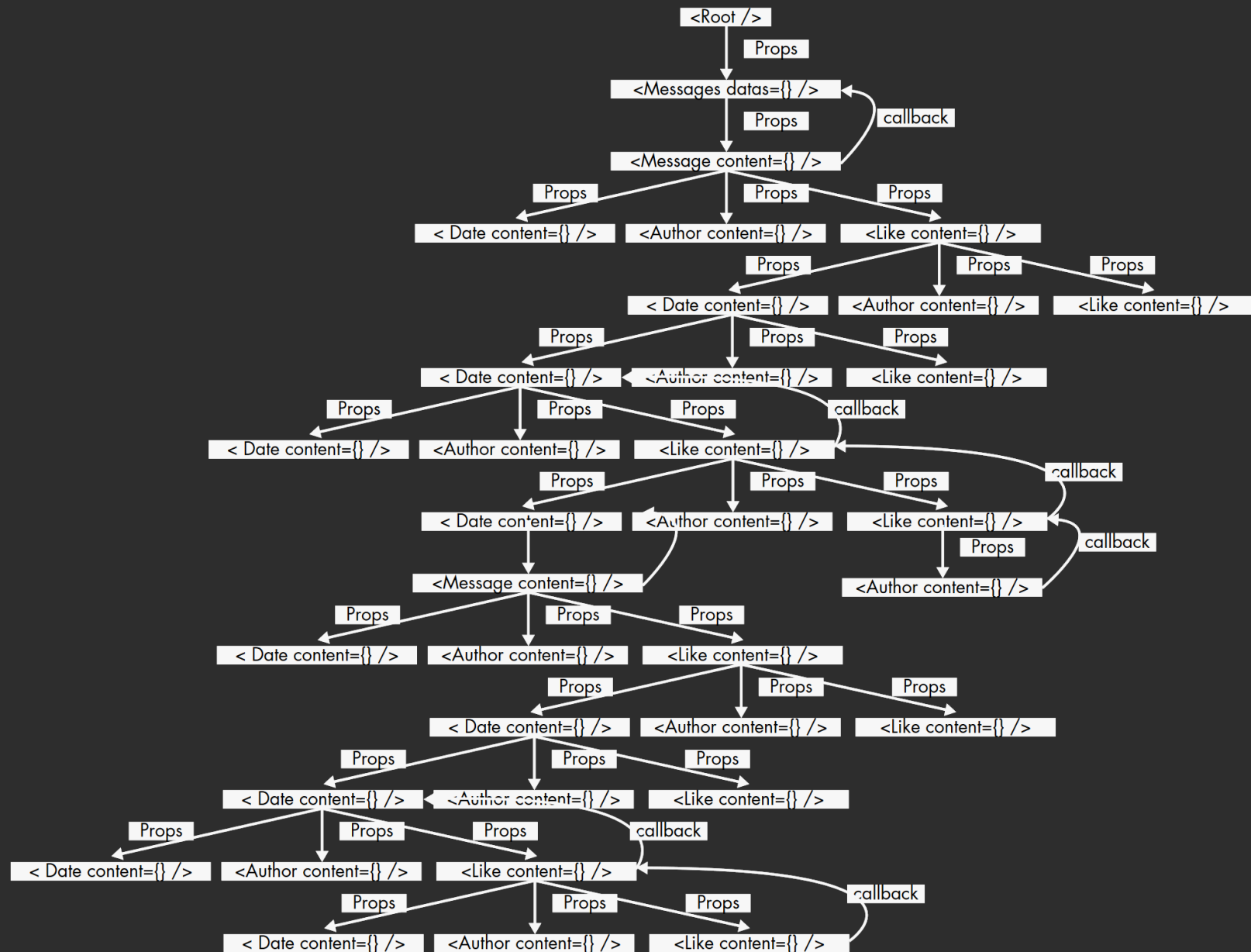
What's Redux ?

- Predicable state container for **any javascript application**
- With Redux, a React app **has only one state for the whole app**
- Redux **is agnostic**, it works with React, VueJS, Angular and of course vanilla js

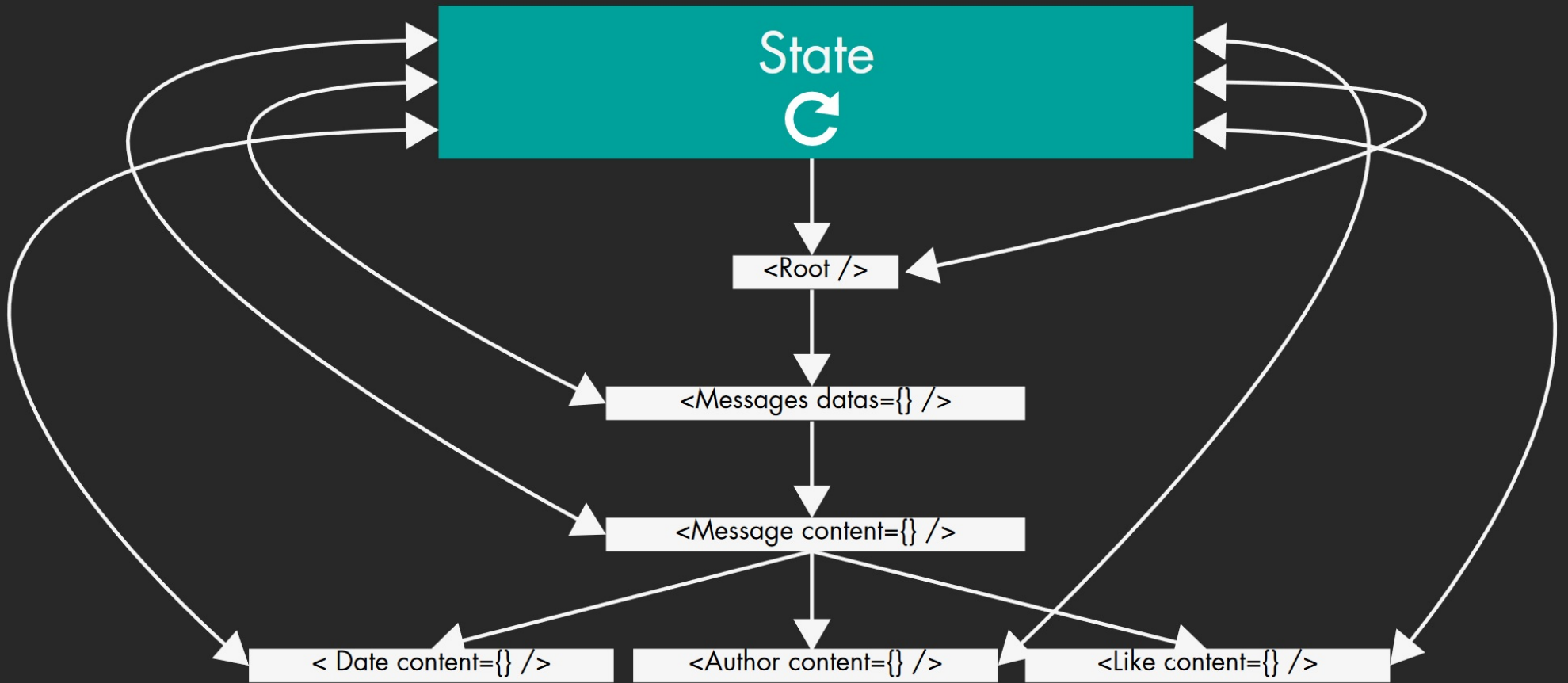
Sources

<https://github.com/angular-redux/ng-redux>
<https://github.com/reduxjs/react-redux>
<https://github.com/titouancreach/vuejs-redux>

Dataflow (up and down) without Redux



Dataflow (up and down) with Redux



Redux's principles

- Single source of truth for **the entire app**
- State **is immutable**
- Changes are made with pure and synchronous functions

Sources

<https://github.com/reactjs/redux/blob/master/docs/introduction/ThreePrinciples.md>

https://en.wikipedia.org/wiki/Pure_function

Redux's advantages

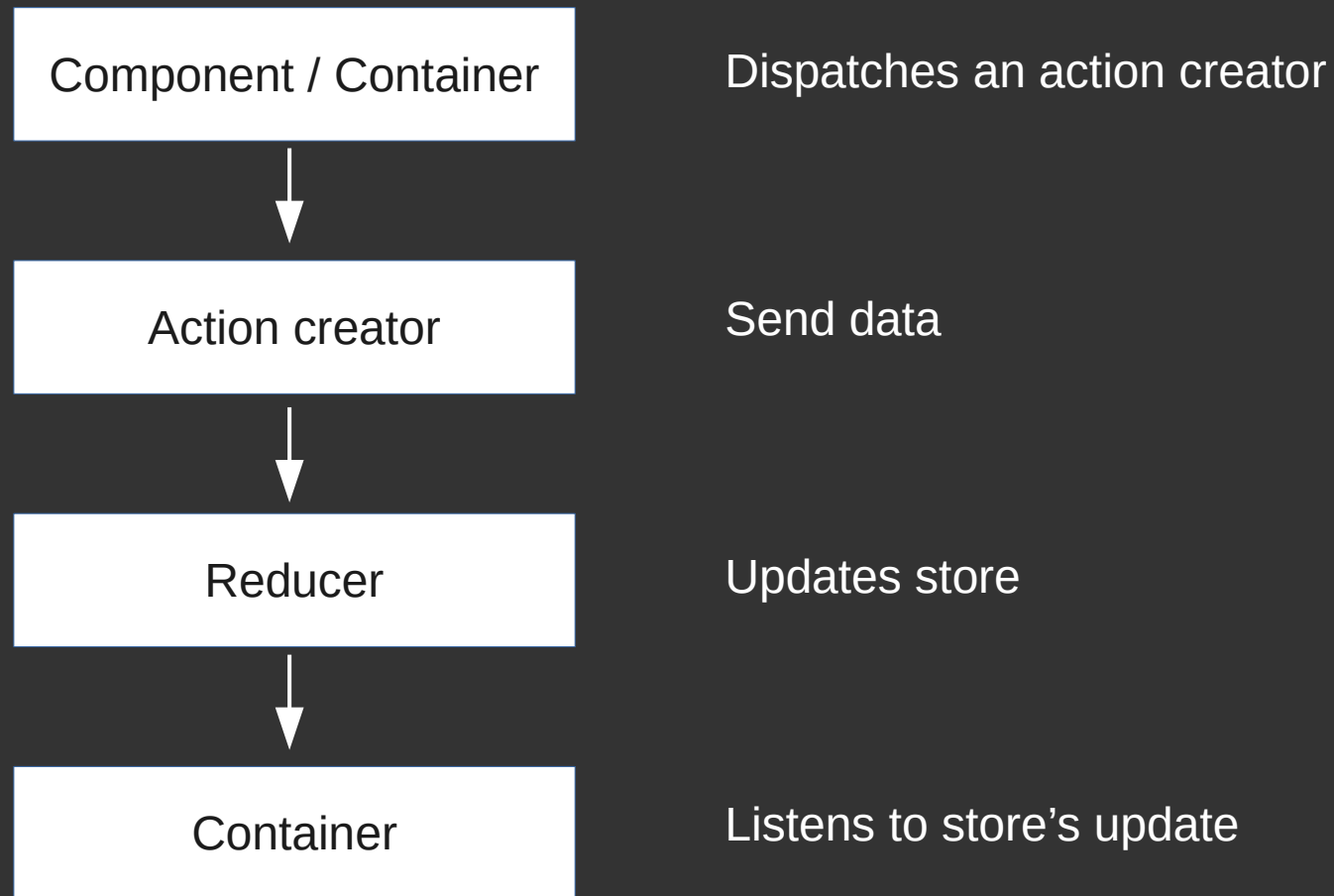
- Easier to debug ReactJS apps ; All data transit in the same place
- Brings (M)VC pattern to React (React is only View)
- Limits corrupted datas ; Redux rewrites state at each change

Sources

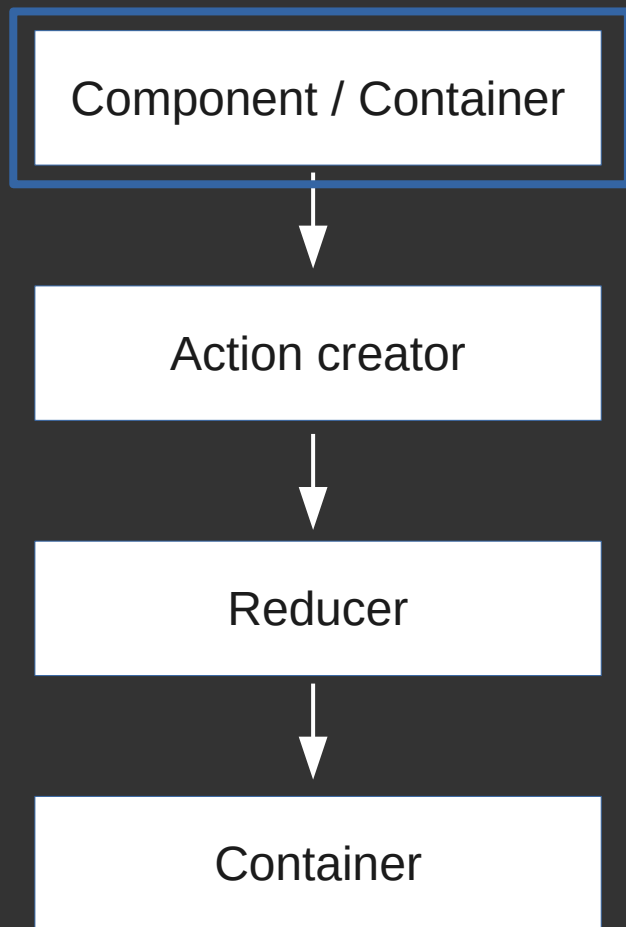
<https://github.com/reactjs/redux/blob/master/docs/introduction/ThreePrinciples.md>

https://en.wikipedia.org/wiki/Pure_function

Redux's flow



Redux's flow



Dispatches
an action creator

Component / Container
It's the View in MVC pattern
Calls an action creator

Send data (an action)

Updates store

Listens to store's update

Redux's flow – Component / container

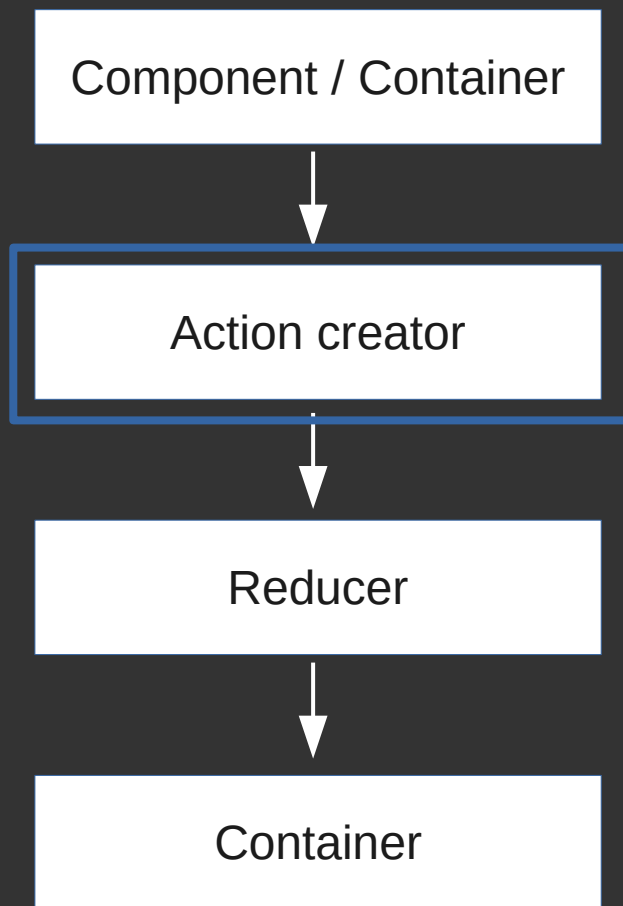
It's the View in MVC pattern. Calls an action creator

```
import React from 'react';

export default class MessageForm extends React.Component {
  // [...]
  submitMessage (e) {
    e.preventDefault();
    this.props.dispatch(addMessage("Hello world"));
  }

  render() {
    const { messages } = this.props;
    return (
      <h1>{ messages[messages.length - 1] }</h1>
      <form>
        <textarea value={this.props.content} />
        <button>Add message</button>
      </form>
    );
  }
}
```

Redux's flow



Dispatches
an action creator

Send data (an action)

Updates store

Listens to store's update

Action creator

Notifies the app that something happened

Note : An action must return an object, to use a function use redux-thunk

Redux's flow – Action creator

Notifies the app that something happened

```
export const addMessage = function (text) {  
  return {  
    type: 'ADD_MESSAGE',  
    payload: { text: text }  
  }  
}
```

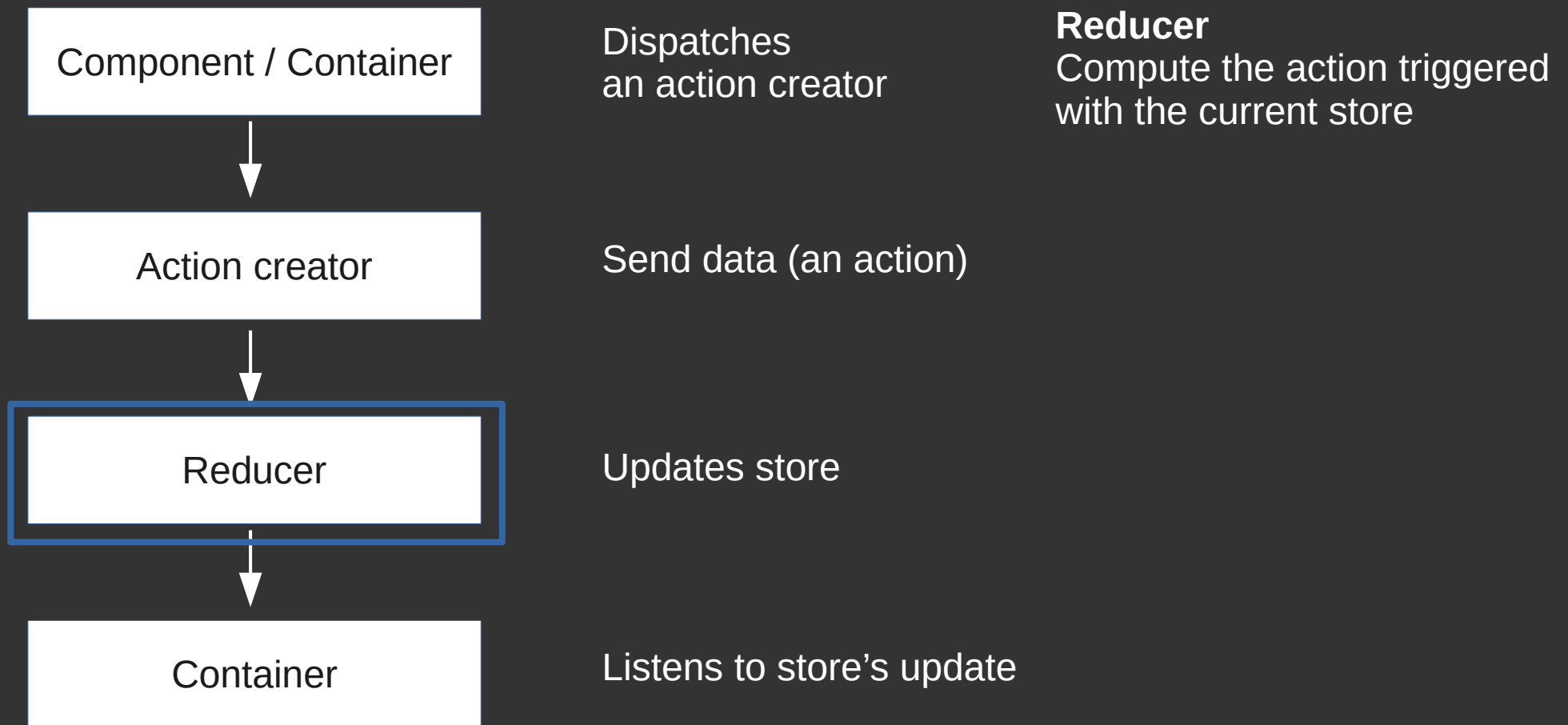
Action

Action creator

Sources

<http://redux.js.org/docs/basics/Actions.html>

Redux's flow



Redux's flow – Reducer

Associates action and current state to get next state.
Indicates how the store should respond to any action
(previousState, action) => newState

```
const DEFAULT_STATE = { text: "" }
```

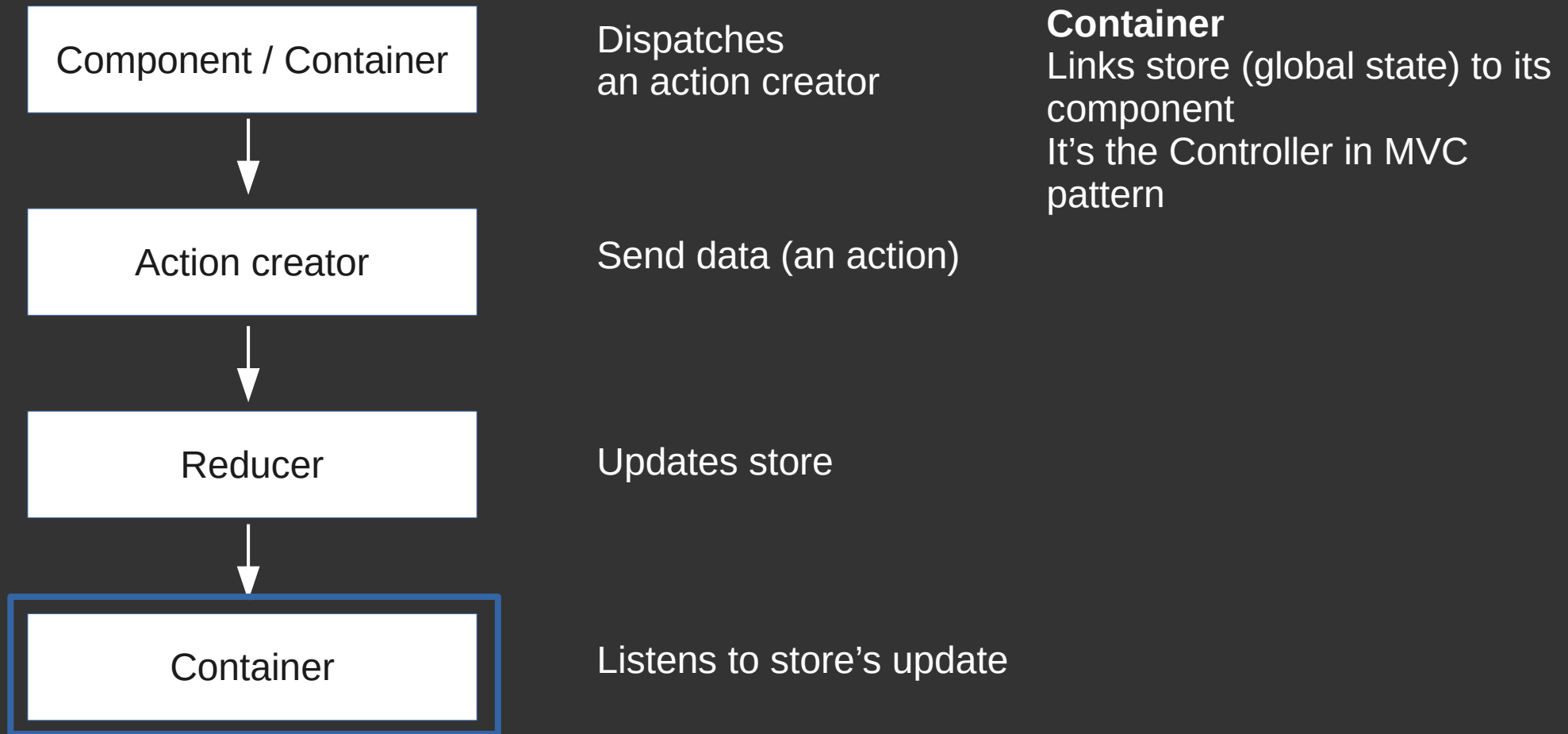
```
const messageReducer = function (state = DEFAULT_STATE, action) {  
  switch (action.type) {  
    case 'ADD_MESSAGE':  
      return action.payload.text;  
    default:  
      return state;  
  }  
}
```

```
export default messageReducer
```

Sources

<http://redux.js.org/docs/basics/Reducers.html>

Redux's flow - Container



Redux's flow – Container

Add some logic to component. It's the Controller in MVC pattern

```
// [...]  
import { connect } from 'react-redux';  
import Messages from '../components/Messages'  
  
// Mainly Message Container definition  
  
function mapStateToProps(state) {  
  return {  
    messages: state.messages  
  }  
}  
  
export default connect(mapStateToProps)(Messages)
```

Sources

<http://redux.js.org/docs/basics/Reducers.html>

Redux's vocabulary

The best way to communicate with a developer, it's to have the same language

Container

React component aware of redux. It calls the `connect()` method.
A container is also called a “smart component”

Component

“Dumb component” (also called “presentational component”)
it just consumes props from its parents

Action creator

Function which **triggers** a store update. It doesn't update the store just indicates what's happened.

Reducer

Indicates how the store should respond to any action

Sources

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

Example – messages

Sources

<https://github.com/DanYellow/presentations/tree/master/react-redux-101/examples/messages>

« You might get the wrong impression from over-engineered tutorials and all the stuff that community has built around it. But Redux itself is very simple. »

Dan Abramov, Redux's co-creator

Sources

https://www.reddit.com/r/reactjs/comments/4npzq5/confused_redux_or_mobx/d46k2bl

<http://www.slideshare.net/tedpennings/how-to-redux>

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

The background is a dark gray color. In the top-left corner, there is a yellow triangle pointing towards the center. In the bottom-right corner, there is a light pink triangle pointing towards the center.

Good practices

Good practices

- Name your action type like... an action
- redux-ducks architecture
- **Never ever** use a function like push or reassign the state inside a reducer. Use Object.assign or the spread operator

Sources

<https://github.com/erikras/ducks-modular-redux>

Advanced topics

Context API – React \geq 16.3

- Built-in “equivalent” of redux inside react
- Allows to bypass container-component hierarchy
- Relies between two members:
 - Provider: Redux’s smart component
 - Consumer: Redux’s dumb component

Sources

<https://reactjs.org/docs/context.html>

Context API – React \geq 16.3

- Built-in “equivalent” of redux inside react
- Allows to bypass container-component hierarchy
- Relies on two parts:
 - Provider: Redux’s smart component
 - Consumer: Redux’s dumb component

Note: React has a deprecated context API. You should never use it

Sources

<https://reactjs.org/docs/context.html>

MapDispatchToProps

Second parameter of redux high-order function (HOF) connect
Allows to bind actions creator to any container

3 (main) ways to bind actions as props to a container

- connect(..., object): short-hand syntax
- connect(..., function): useful to split the logic between store's and actions creator
- connect(..., undefined): adds a props function called dispatch inside the container to call directly inside the **component** an action creator.

E.g.: `this.props.dispatch(actionCreator(param))`

MapDispatchToProps

Second parameter of redux high-order function (HOF) connect
Allows to bind actions creator to any container

Debugging

Browser extension for redux (redux dev-tools)

Use it only on dev!

Sources

<https://github.com/zalmoxisus/redux-devtools-extension>

Questions ?

More resources

- Presentation + examples :

<https://github.com/DanYellow/presentations/tree/master/react-redux-101>

Stateless component

When a component doesn't need to update its state, it should be written as a function also called **stateless component**