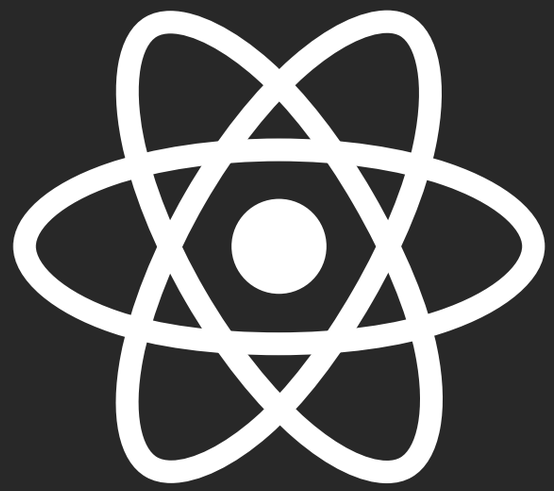
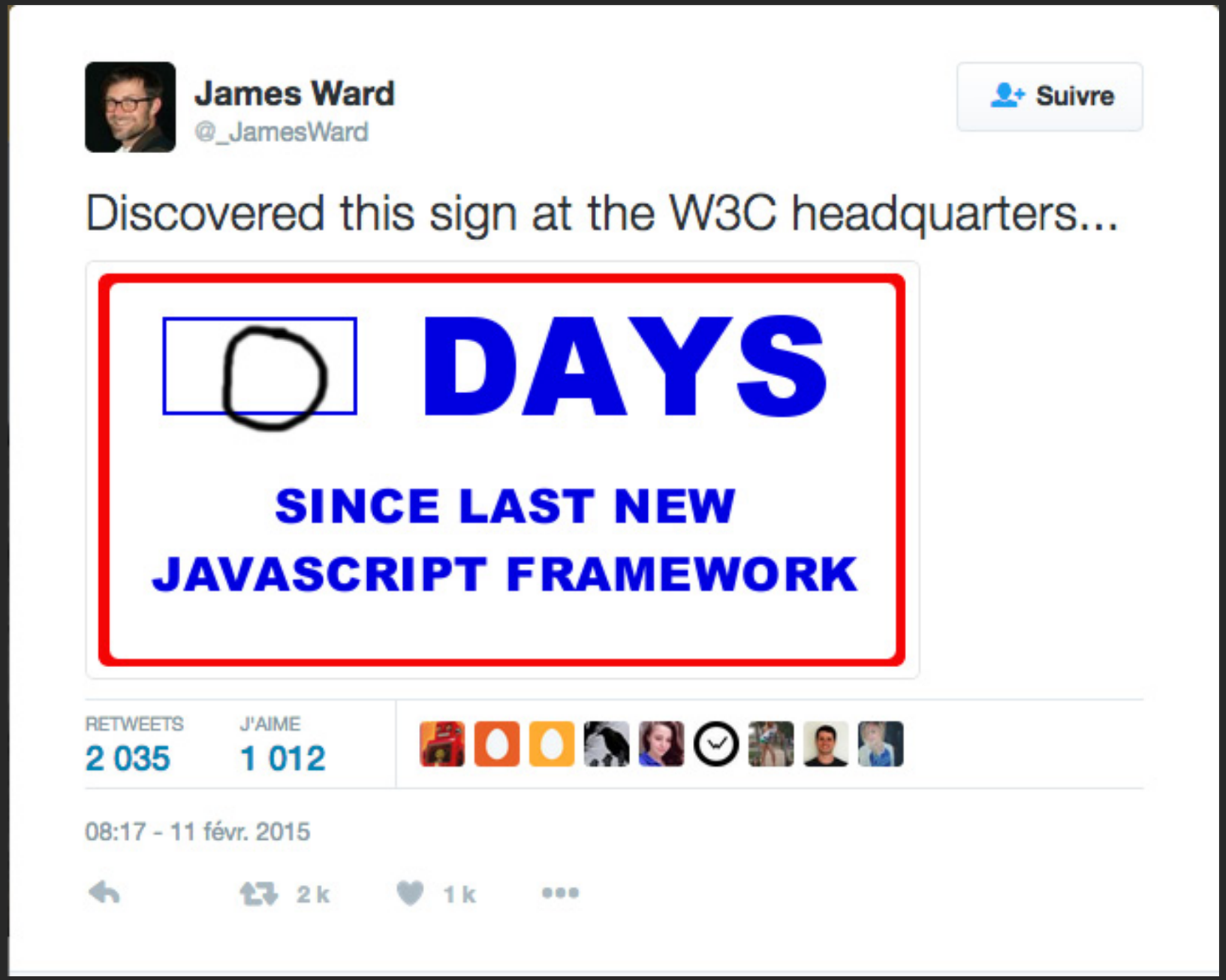


ReactJS



Sources
https://twitter.com/_jamesward/status/565545160332046336

An (another) javascript framework

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

Sources
http://stackoverflow.com/questions/148747/what-is-the-difference-between-a-framework-and-a-library
https://web.archive.org/web/20070504053354/http://www.ddj.com/blog/architectblog/archives/2006/07/frameworks_vs_l.html

What's React ?

A javascript library developped, open sourced and maintained by facebook
React is designed for (large) applications **with data that changes over time**
Released in 2013
facebook *dogfooding* it with facebook and Instagram

What's React ?

A javascript library developped, open sourced and maintained by facebook
React is designed for large applications **with data that changes over time**
Released in 2013
facebook *dogfooding* it with facebook and Instagram

No MVC
No MVVM
No DM-VM-CM-VC-V*
…

Sources
https://github.com/xlasne/MVVM-C

ReactJS is only **views**

ReactJS is only **views**
ReactJS is **components**

React components are

Reutilisable
Testable
Maintanable
Use on shadow DOM

React components are

Reutilisable
Testable
Maintanable
Use on shadow DOM ⟶ Extremely fast

React components are

Reutilisable
Testable
Maintanable
Use shadow DOM ⟶ Extremely fast
Managed only by their parent

React components are not Web Components

Web components are **for strong encapsulation**
React components are **made to be sync with datas**

Sources
https://facebook.github.io/react/docs/webcomponents.html

Let’s write a 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 a 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

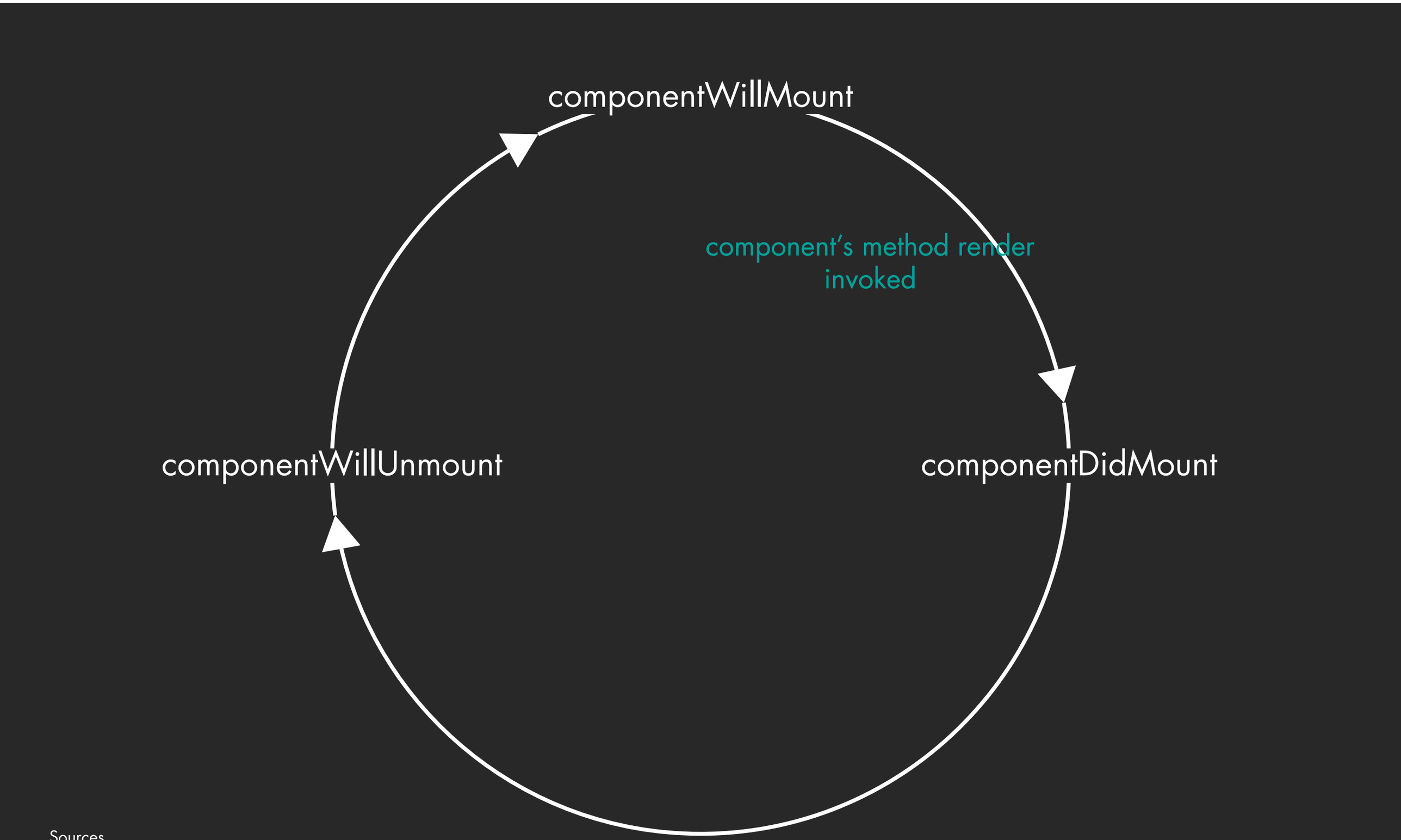
Let's write a component

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


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

We start the React app

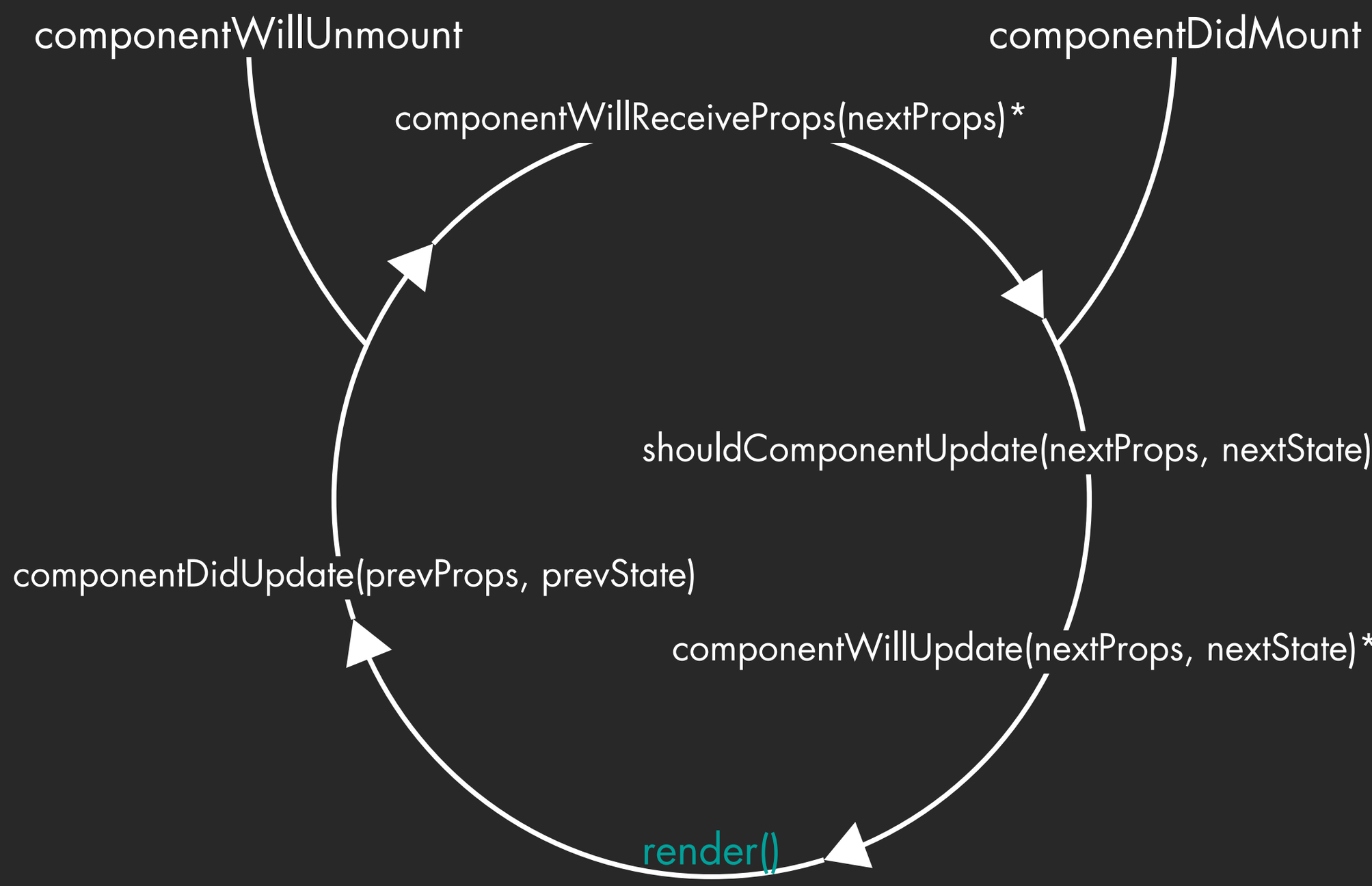
Note : There can be only one render() per page

Component life cycle



Sources
https://facebook.github.io/react/docs/component-specs.html#lifecycle-methods

Component life cycle
(between componentWillUnmount and componentDidMount)



Sources
https://facebook.github.io/react/docs/component-specs.html#lifecycle-methods

* : These methods are not called in the initial render

Props and state

Props and state allow to change component's content/datas

Props are **immutable**

Props are **read-only**

Props are **transmitted by their parent**

Props and state

Props and state allow to change component's content/datas

Props are **immutable**

Props are **read-only**

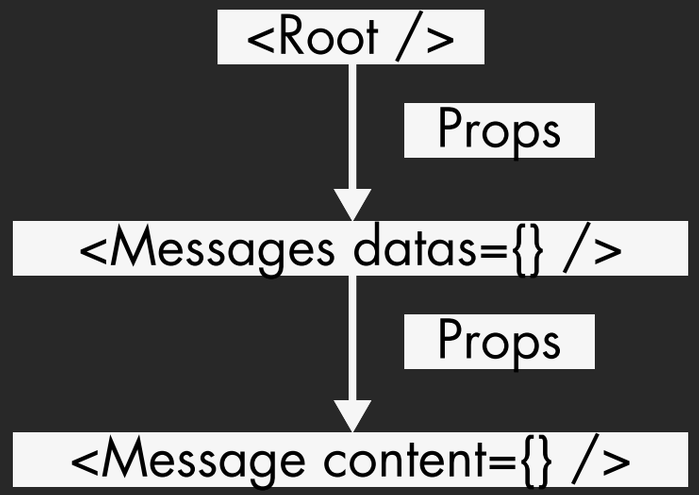
Props are **transmitted by their parent**

State is **mutable**

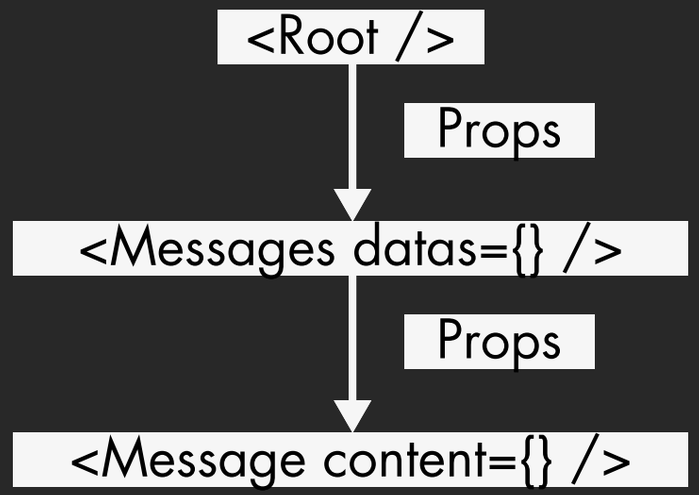
State **mutates in component**

ReactJS's data flow is **unidirectional**

Data flow (down)

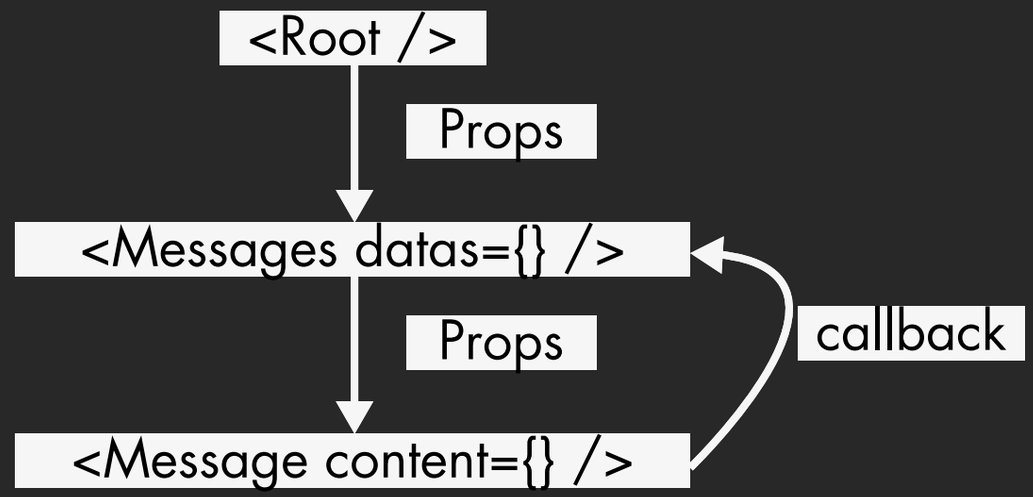


Data flow (down)

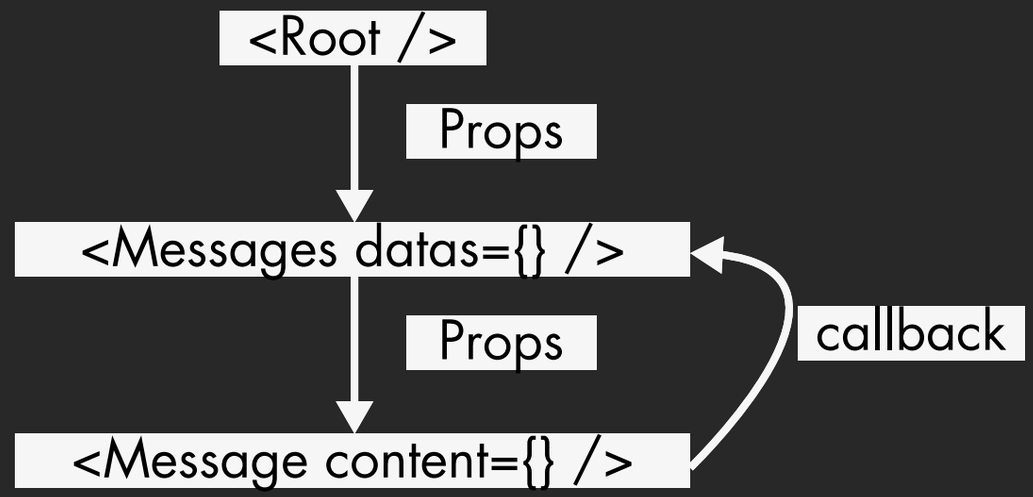



```
import React from 'react'

export default class Messages extends React.Component {
  constructor (props) {
    super(props);
  }

  render() {
    return (
      <Message content={this.props.datas[0]} />
    );
  }
}
```

Data flow (up)



Data flow (up)




```
export default class Messages extends React.Component {
  myCallback () {
    console.log(‘‘my child talks to me !’’);
  }

  render() {
    return (
      <Message content={this.props.datas[0]} onClickCB={this.myCallback} />
    );
  }
}
```

Sources
https://facebook.github.io/react/tips/communicate-between-components.html

Data flow (up)

```
export default class Message extends React.Component {
  messageClick () {
    this.props.onClickCB()
  }

  render() {
    return (
      <p onClick={this.messageClick}> {this.props.content} </p>
    );
  }
}
```

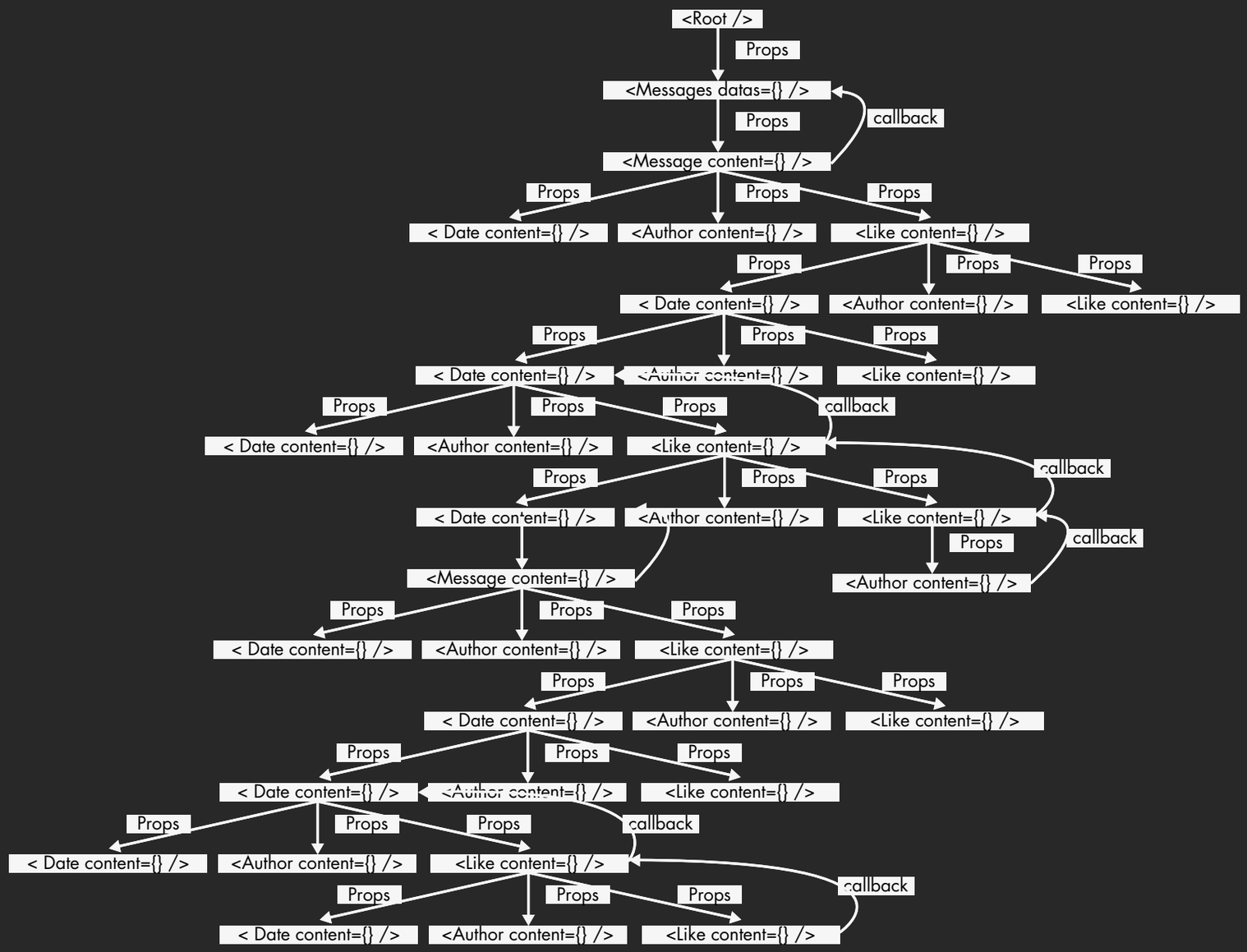
Sources
https://facebook.github.io/react/tips/communicate-between-components.html

Demo

Sources
https://github.com/DanYellow/react-presentation-project/tree/noredux

This data flow is (totally) a mess

If we scaled up, this kind of thing can happen




9
9
NEWS
ABSOLUTELY DISGUSTING

Demo

Sources
https://github.com/danyellow/google-bouffe/tree/css-modules

Inline style

To be a real component, it needs to overcome any extra files like stylesheets
React brings **inline style** with JS to solve this issue.

```
export default class Messages extends React.Component {
  [...]
  render() {
    return (
      <Message style={Styles.message} content={this.props.datas[0]} />
    );
  }
}

const Styles = {
  message: {
    fontSize: 25,
    textAlign: ‘center’
  }
}
```

Sources
https://facebook.github.io/react/tips/inline-styles.html

Inline style

Inline style in ReactJS is a nice feature, but there are some cons :

- No pseudo-format/class support (:hover, :last-child...)
- No media-queries support
- No autoprefixer for exotic css attribute
- No relative units

Inline style

Inline style in ReactJS is a nice feature, but there are some cons :
- No pseudo-format/class support (:hover, :last-child...)
- No media-queries support
- No autoprefixer for exotic css attributes
- No relative units

Inline style

Libs for uncomplete style management in React

- classnames
- radium
- CSSModules

Sources
https://github.com/JedWatson/classnames
https://github.com/FormidableLabs/radium
https://github.com/css-modules/css-modules

First part in a nutshell

React bring real component (DOM + CSS + JS in same file)
JSX is strongly encourage for templating
React allows a full control of the component with its lifecycle
Props allow parent to set datas to its children
Props **are immutables**

First part in a nutshell

React bring real component (DOM + CSS + JS in same file)
JSX is strongly encourage for templating
React allows a full control of the component with its lifecycle
Props allow parent to set datas to its children
Props **are immutables**
State is **updated by component himself**
State is **mutable**

Inline styles missing features are solved by Radium & CSS Modules

Data flow in ReactJS is a mess… except if we use…

Sources
https://github.com/FormidableLabs/radium
https://github.com/css-modules/css-modules

Redux

Redux ?

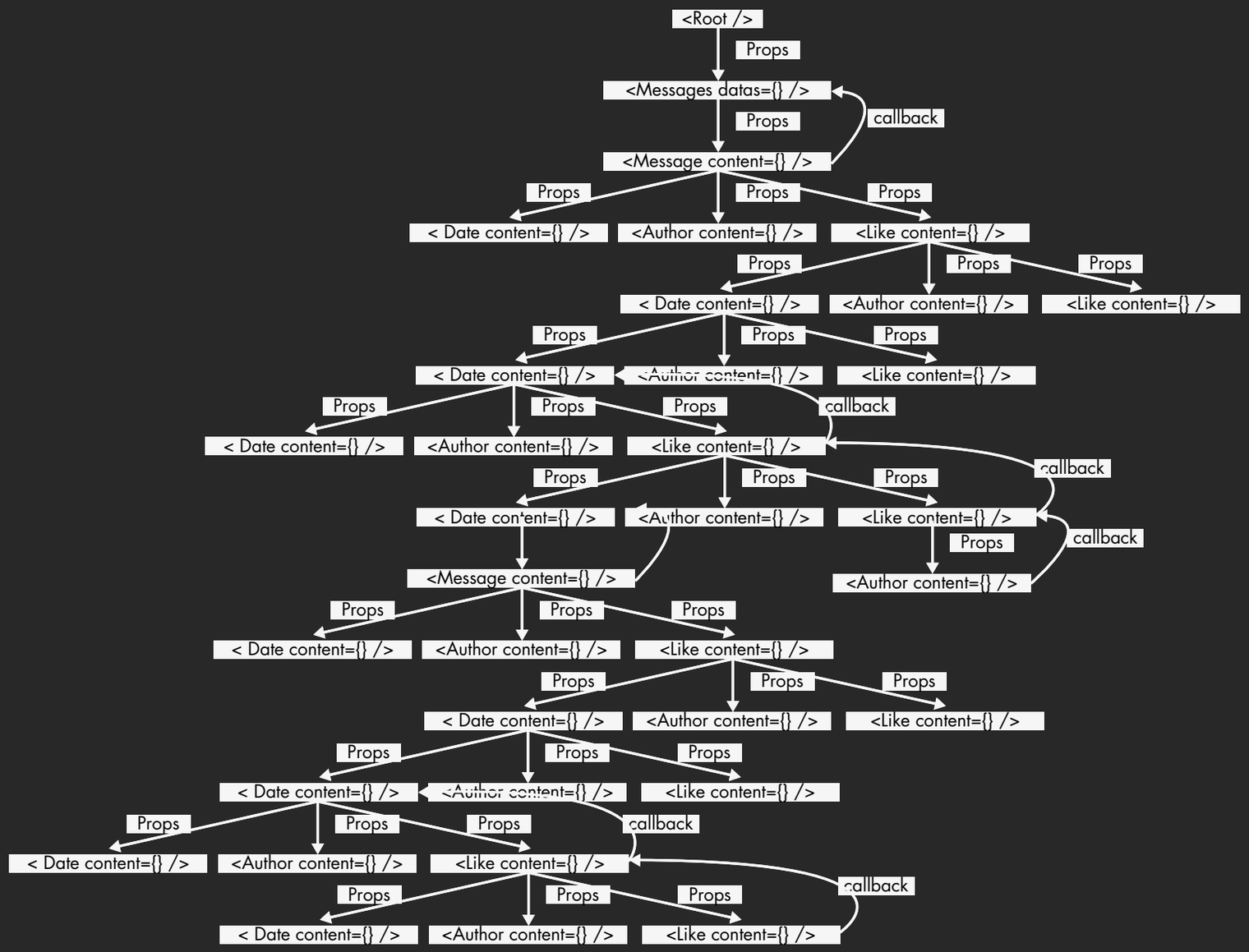
Redux is predicable state container for **javascript application**

Redux ?

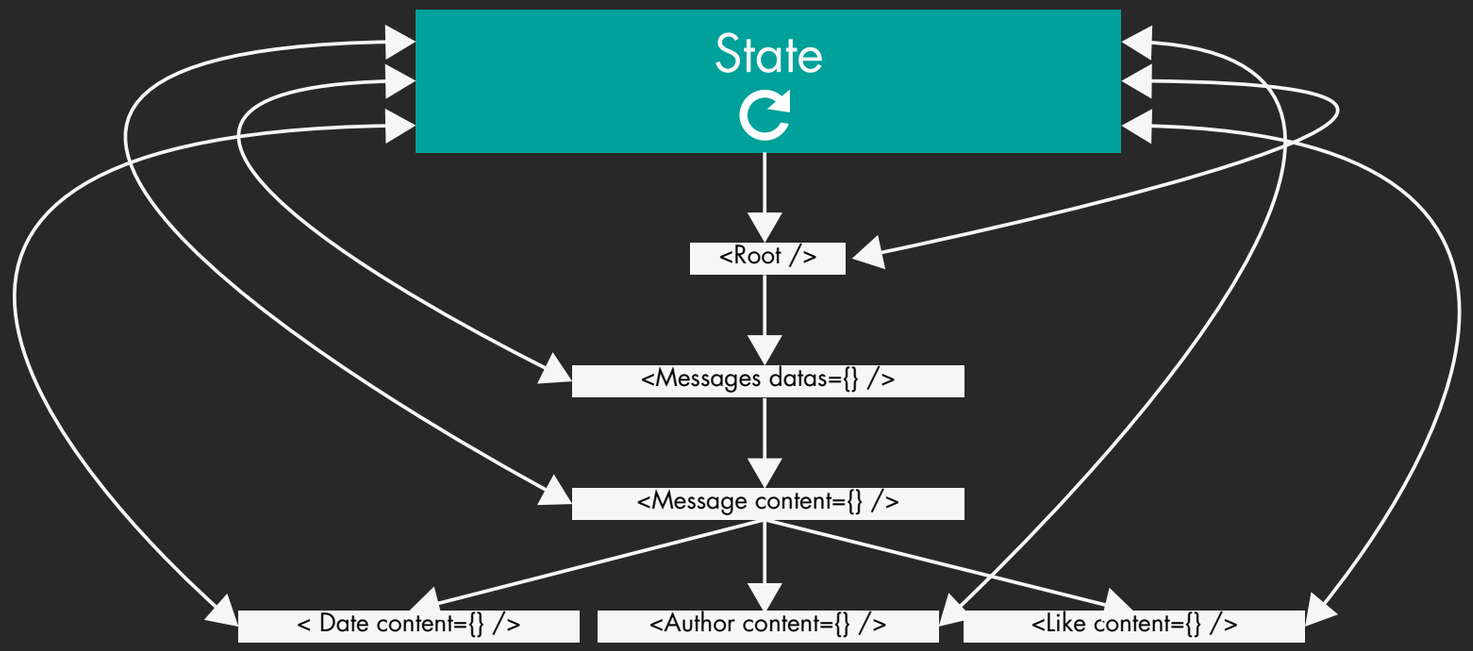
Redux is predicable state container for **javascript application**

With Redux, ReactJS app **has only one state for the whole app**

Dataflow (up and down) without Redux



Dataflow (up and down) with Redux



Principles

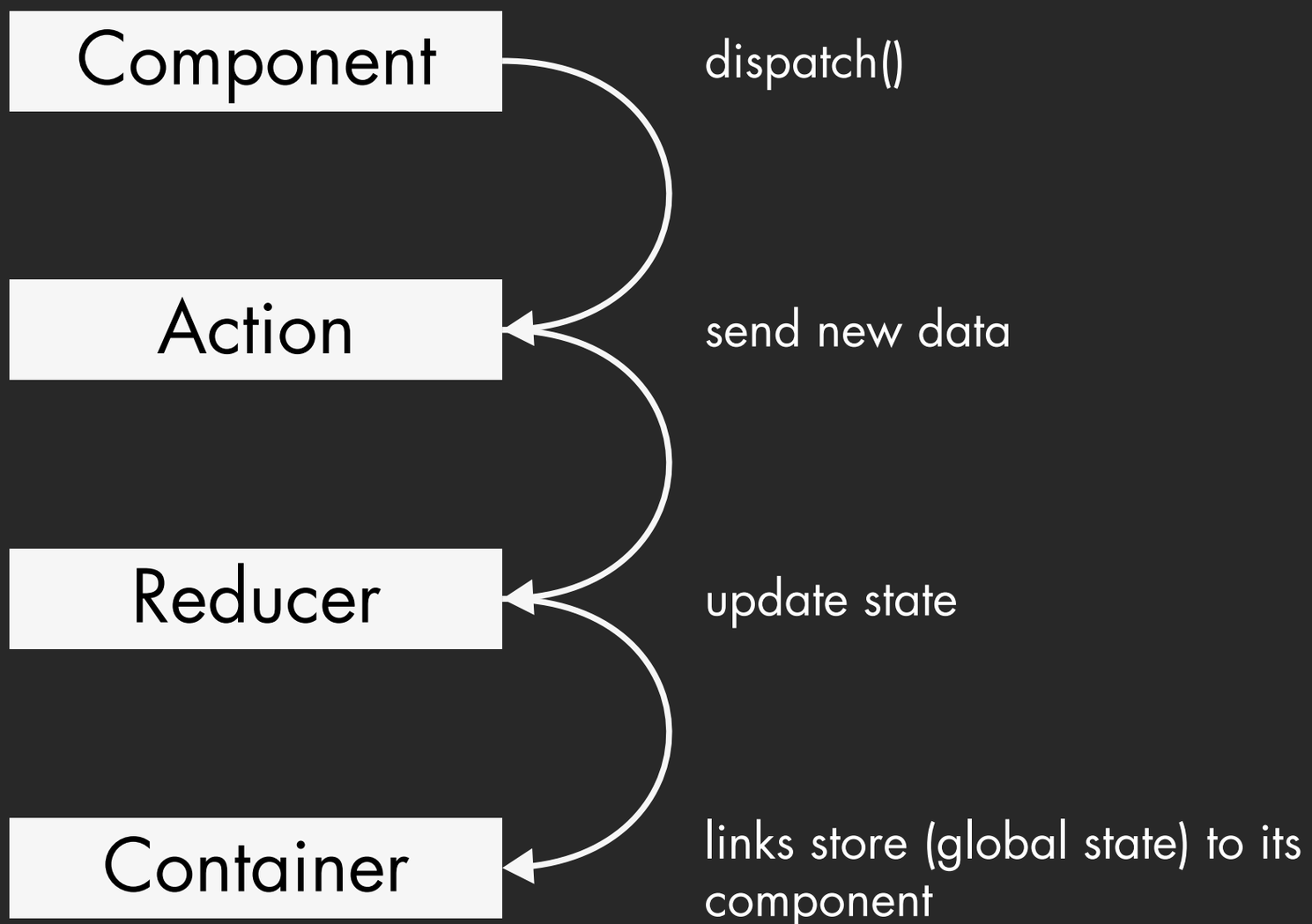
- Single source of truth
- State **is immutable**
- Changes are made with pure functions

Sources
https://github.com/reactjs/redux/blob/master/docs/introduction/ThreePrinciples.md
https://en.wikipedia.org/wiki/Pure_function

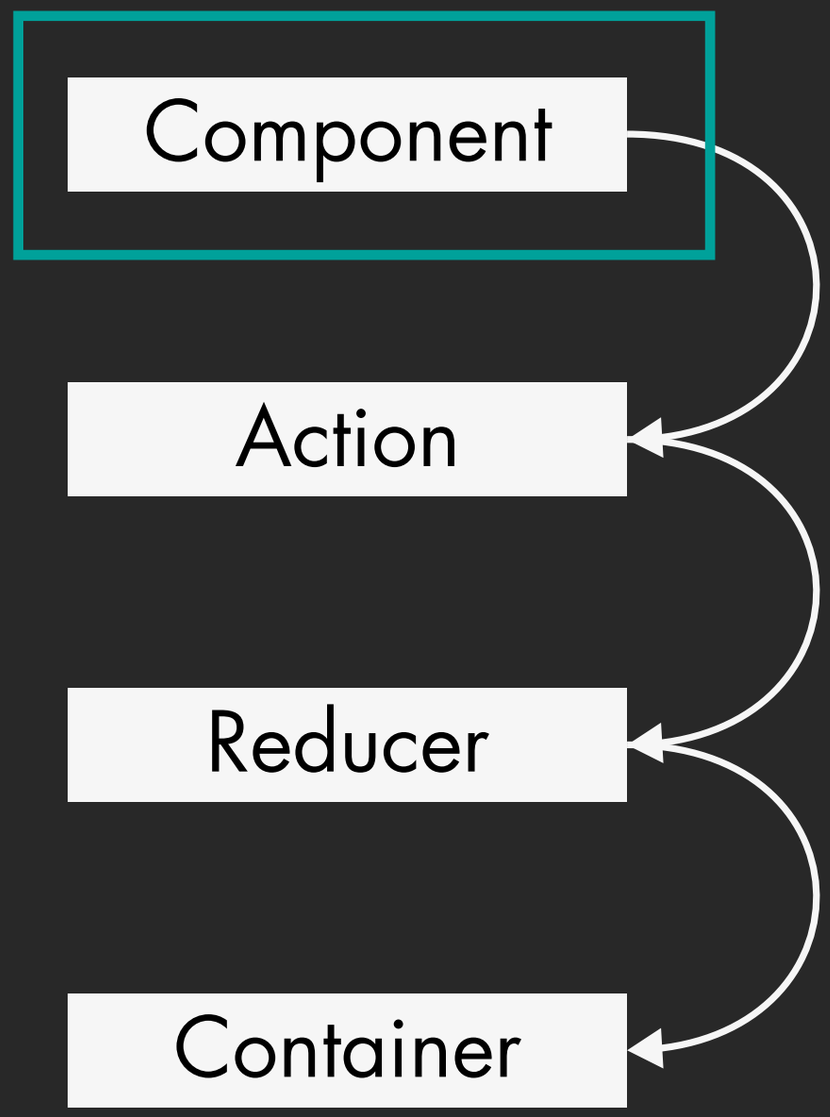
Advantages

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

Redux dataflow in details



Redux dataflow in details



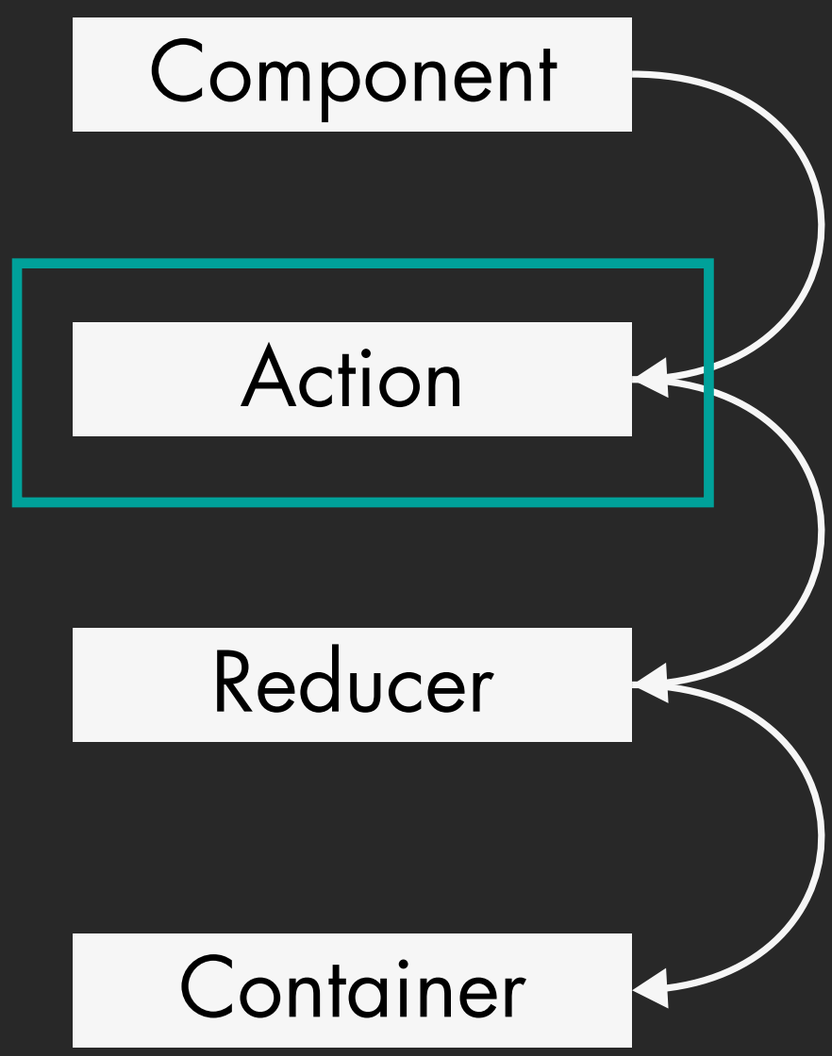
Component
It's the View in MVC pattern
Calls an action creator

Redux dataflow in details - Component

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

```
export default class Message 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 name=’’message’’ value={this.props.content} />
         <button>Add message</button>
      </form>

    );
  }
}
```

Redux dataflow in details



Action

Notifies the app that something happened

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

Redux dataflow in details - Action

Notifies the app that something happened

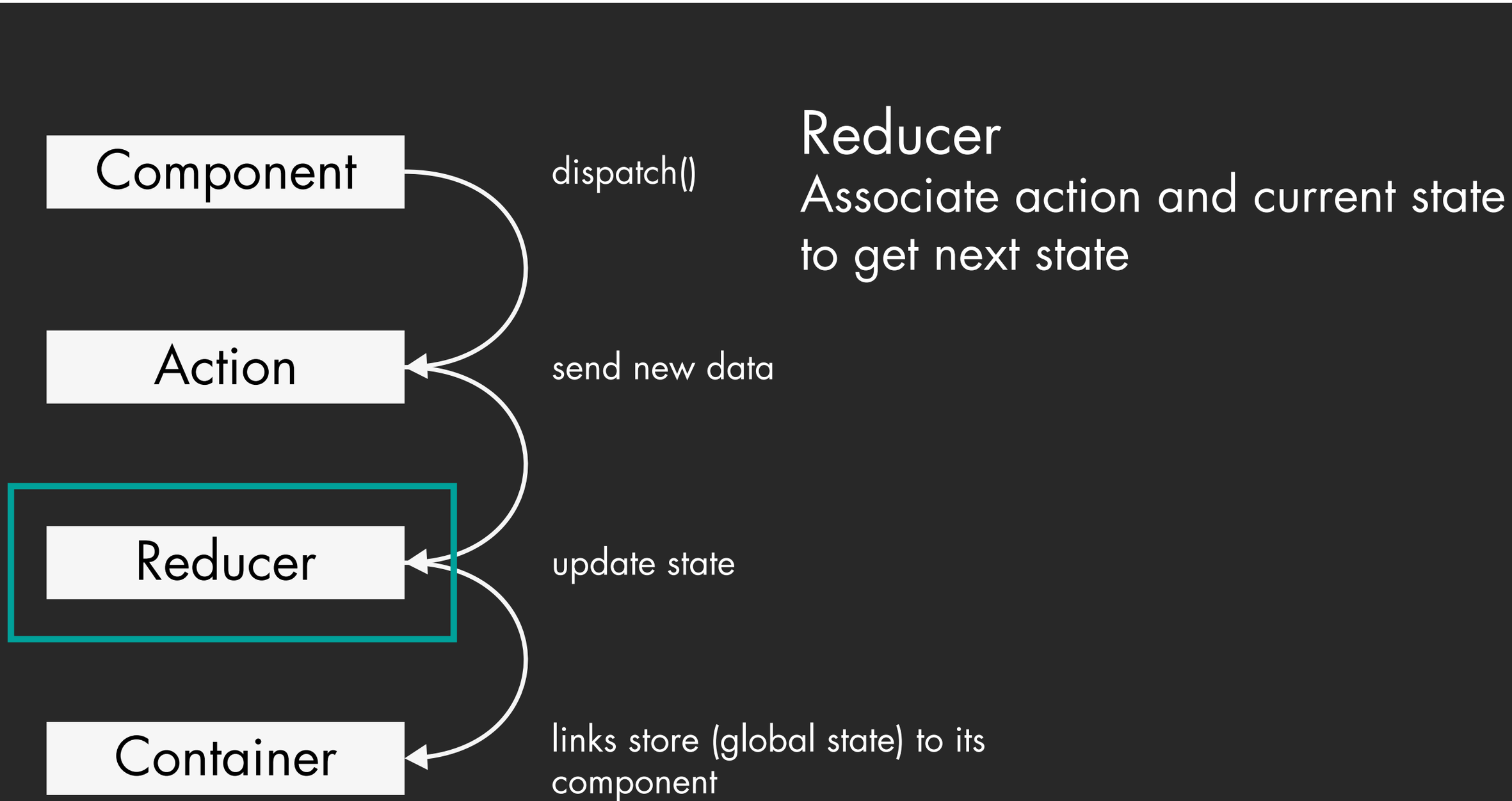
```
export const addMessage = function (id) {
  return {
    type: ‘ADD_MESSAGE’,
    text: text
  }
}
```

Action

Action creator

Sources
http://redux.js.org/docs/basics/Actions.html

Redux dataflow in details



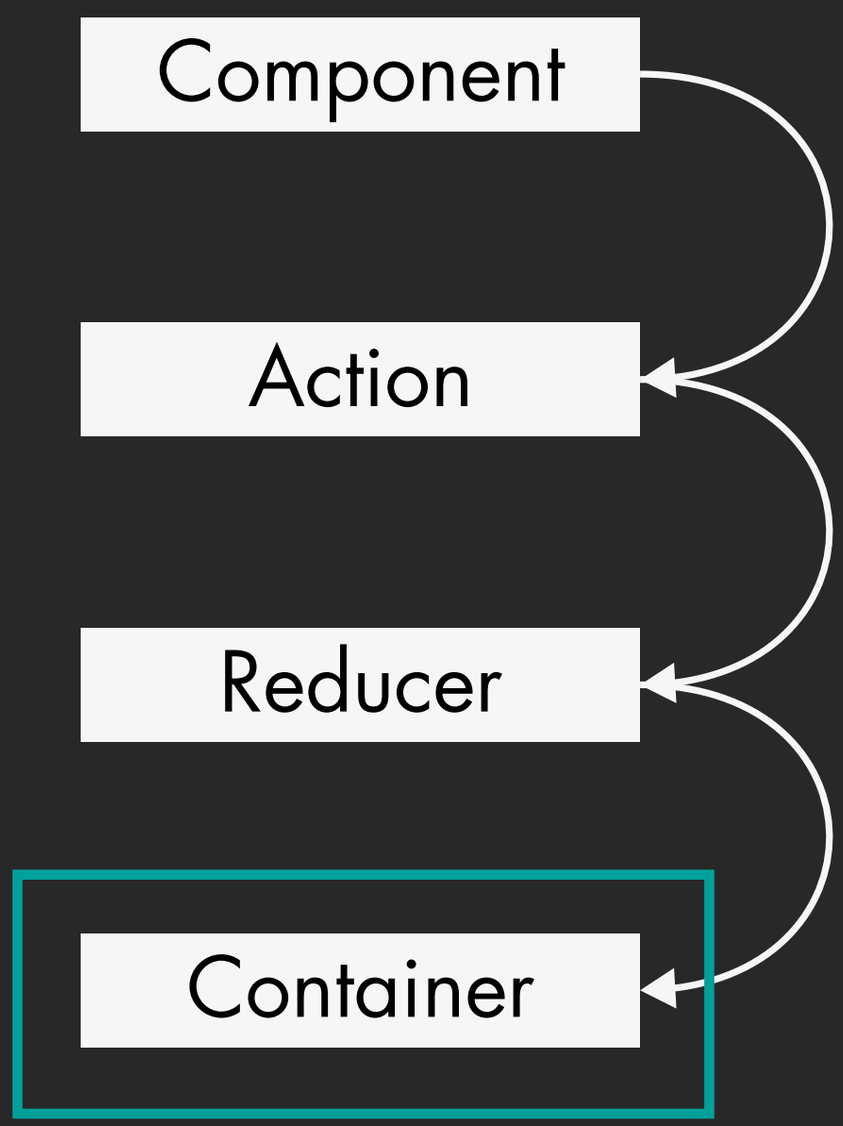
Redux dataflow in details - Reducer

Associate action and current state to get next state.
(previousState, action) => newState

```
const message = function (state = ‘’, action) {
  switch (action.type) {
    case ‘ADD_MESSAGE’:
      return action.text;
    default:
      return state;
  }
}
```

Sources
http://redux.js.org/docs/basics/Reducers.html

Redux dataflow in details



dispatch()

send new data

update state

links store (global state) to its component

Container

Links store (global state) to its component

It's the Controller in MVC pattern

Note : By convention Container and its component share same name

Redux dataflow in details - Container

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

Note : By convention Container and its component share same name

```
import Messages from '../components/Messages'

function mapStateToProps(state) {
  return {
    messages: state.messages
  }
}

export default connect(mapStateToProps)(Messages)
```

Sources
https://github.com/reactjs/react-redux/blob/master/docs/api.md#connectmapstatetoprops-mapdispatchtoprops-mergeprops-options

« 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 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

Demo

Sources
https://github.com/DanYellow/react-presentation-project/tree/redux
https://github.com/DanYellow/React-Pokemon/tree/redux

Redux's competitors

- Flux (facebook)
- DIY Facebook Flux
- Fluxxor
- McFly
- NuclearJS
- Flummox
- Fluxette
- Lux
- Marty
- Reflux
- Alt (number two on the list)
- Yahoo-Fluxible
- Delorean
- Barracks
- Fluxy
- MobX

Sources
http://devnacho.com/2016/02/24/what-is-the-best-way-to-style-react-components/
http://stackoverflow.com/questions/32461229/why-use-redux-over-facebook-flux#answer-32920459

Last part in a nutshell

Redux is not related to ReactJS
Redux is really simple don’t be afraid by over-engineered tutorials
Redux is not mandatory but highly recommanded to **avoid spaghetti code**
Redux brings missing Controller to React

My opinion about ReactJS

- Powerful javascript library
- Easy to learn
- Useful for many cases. E.g. : Dealer locator
- Maybe the best library to manage view and data currently

Sources
http://stateofjs.com/2016/api/

For further

React router

Sources
https://github.com/ReactTraining/react-router

React storyboard

Sources
https://github.com/kadirahq/react-storybook

Thank you

Questions ?

For further
https://js.coach/react
https://github.com/facebookincubator/create-react-app
https://packagecontrol.io/packages/Babel%20Snippets