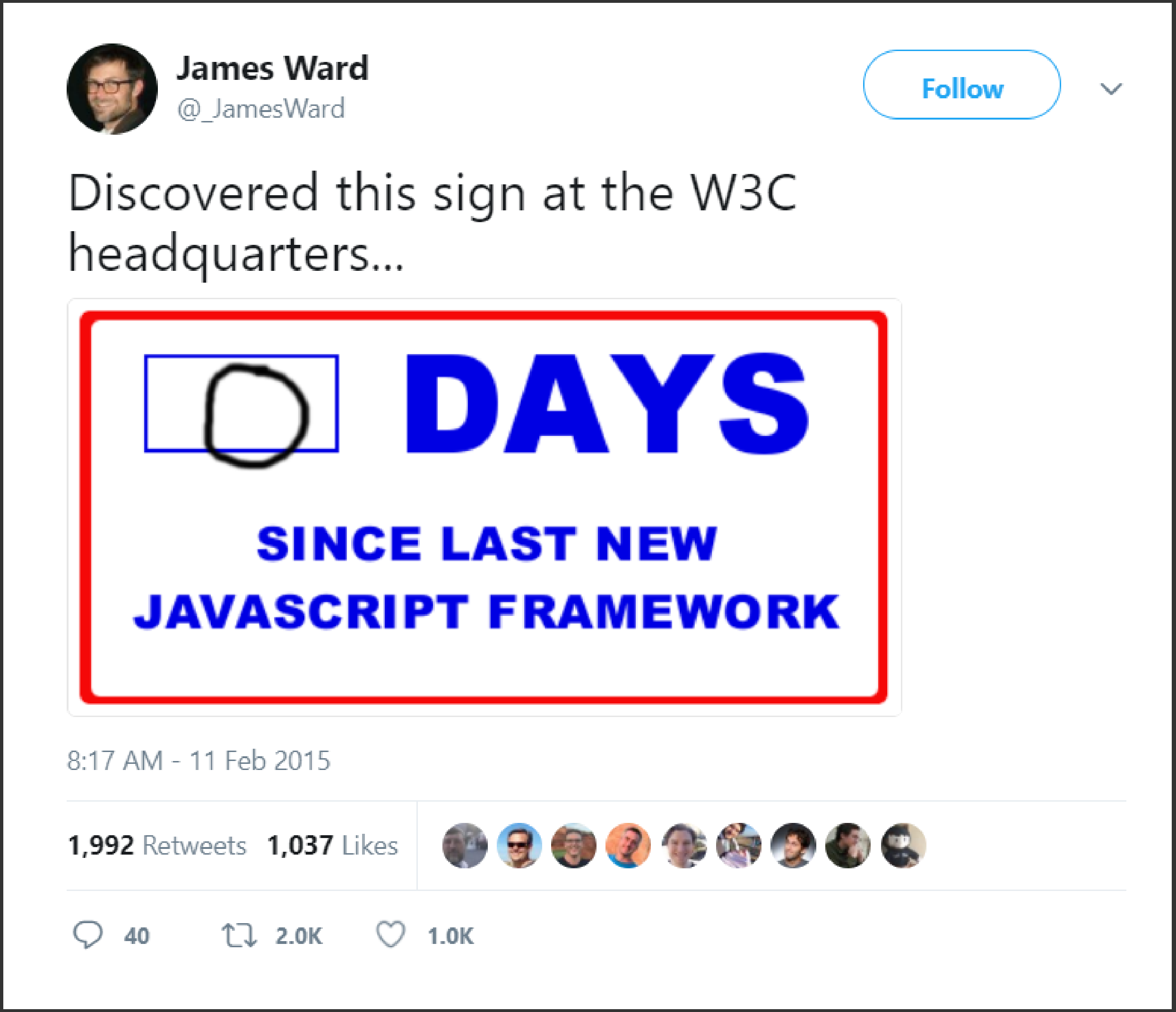
React and Redux 101

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 developped, open sourced and maintened by facebook
- Designed for (large) applications **with data that change over the time**
- Released in 2013
- Dogfeed by facebook with facebook and instagram

What’s React ?

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

- Relies on Virtual DOM

What’s React ?

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

- Relies on Virtual DOM $\longrightarrow$ Extremely fast

What’s React ?

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

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

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>
    );
}
```

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

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

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

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 the 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 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} />
    );
}
```

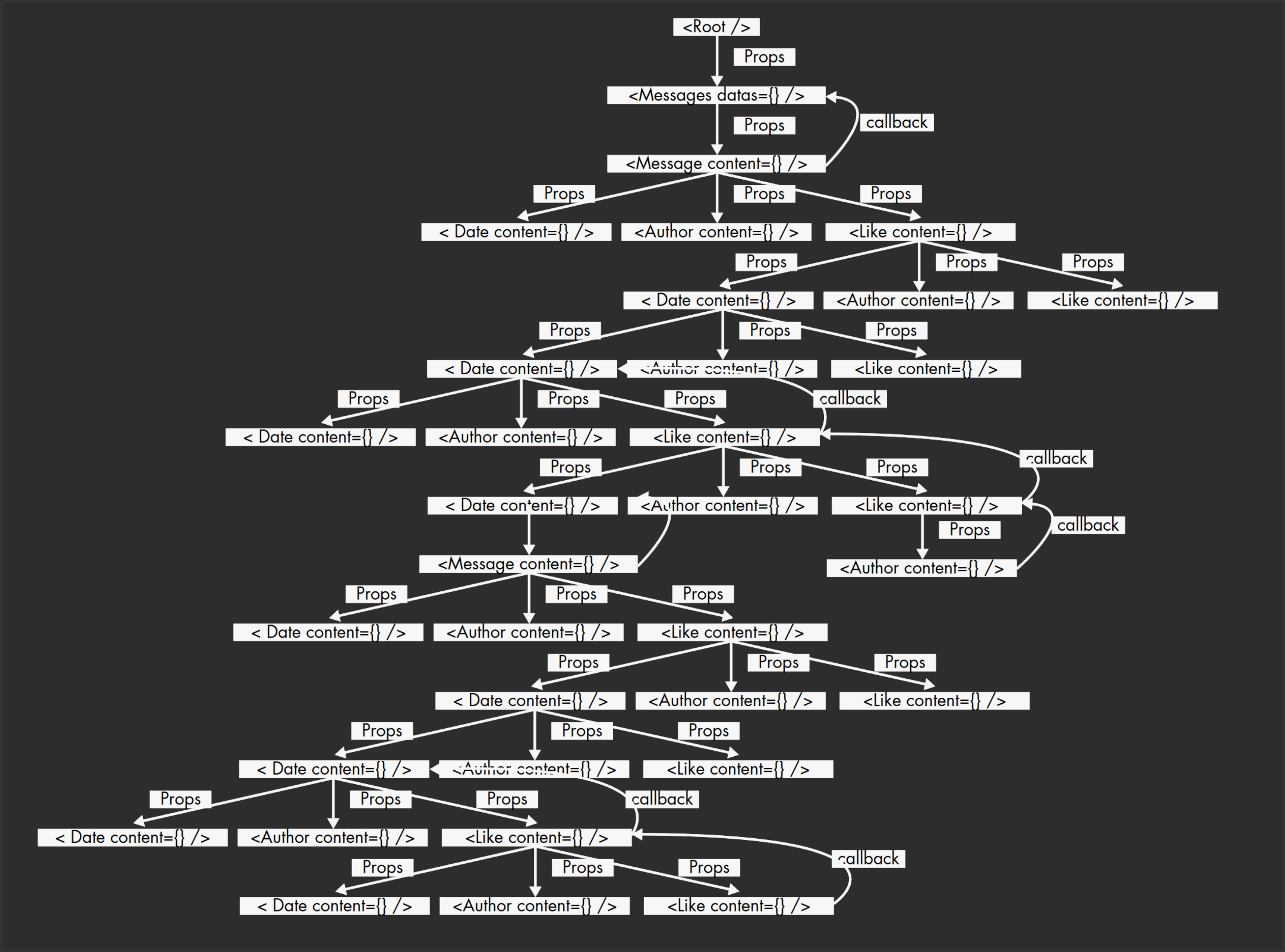
Example – state & props

Data flow (up)

Use callback function to communicate with the closest parent

Example – callback

Data flow (up)



9
NEWS
ABSOLUTELY DISGUSTING

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

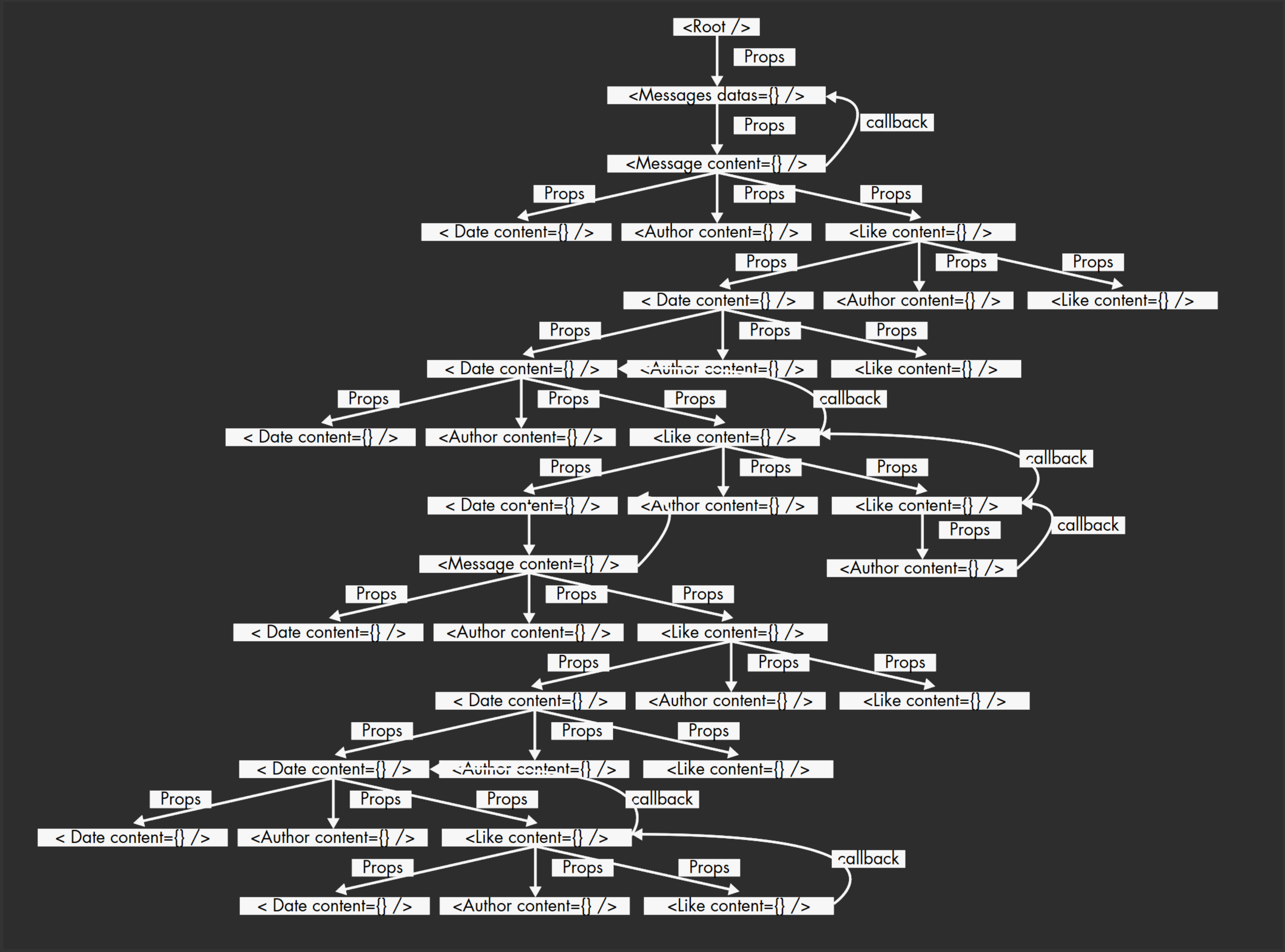
The data flow (up) in React is a mess... except if we use...

Redux

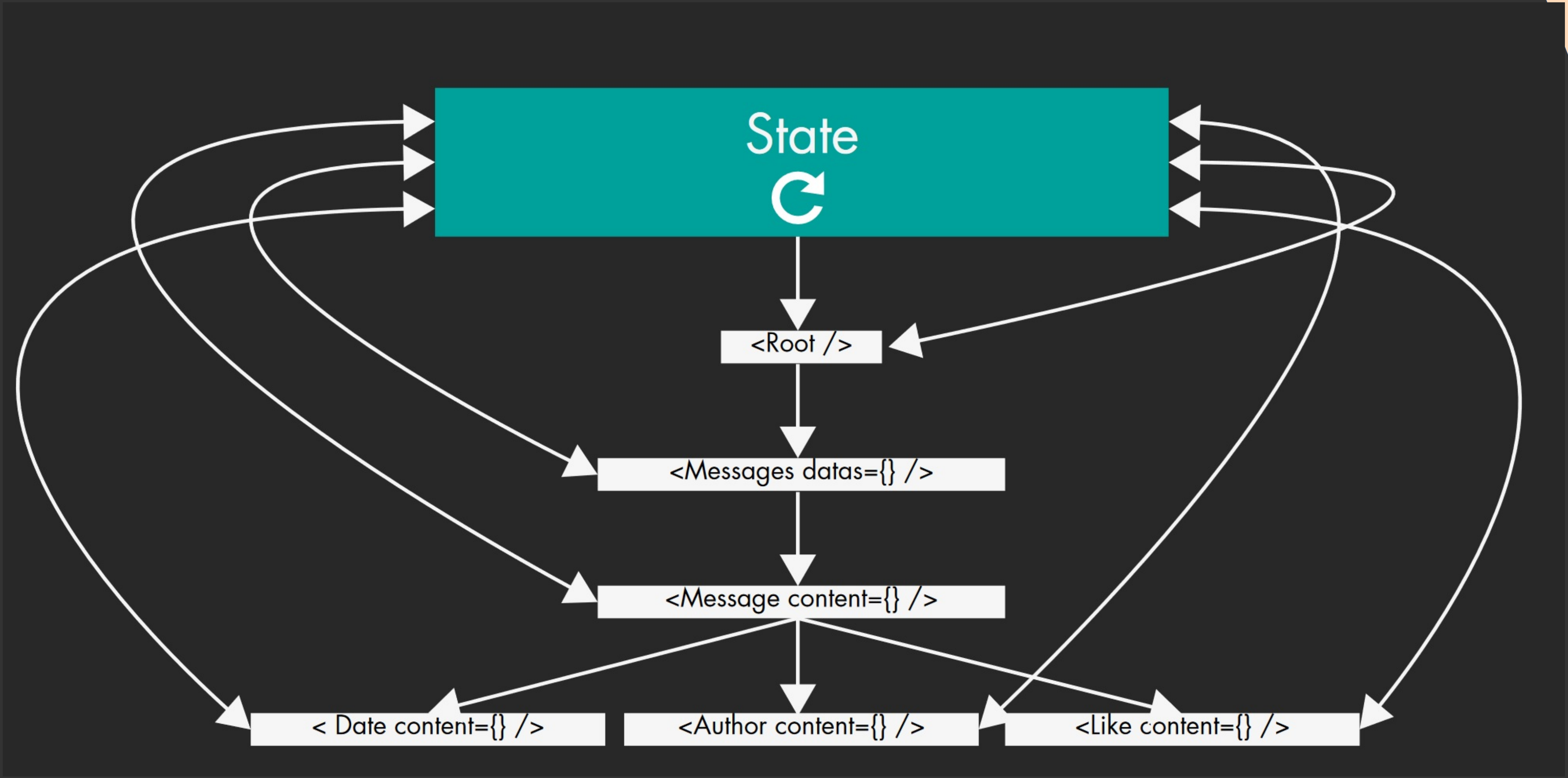
What’s Redux ?

Predicable state container for **any javascript application**
With Redux, a React app **has only one state for the whole app**

Dataflow (up and down) without Redux



Dataflow (up and down) with Redux



Redux’s principles

- Single source of truth
- 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

Example – todo-list

« 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

For further

Context API – React ≥ 16.3

```
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} />
    );
}
```

Web extensions (Chrome and FF)

- React
- redux dev-tools

Architecture

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

Questions ?

But your code has to be ready for it

Jest

- Unit test framework for javascript
- Created by facebook
- Agnostic but made to work well with React
- Inspired by Jasmine
- Allows snapshot
- Jest = Jasmine + sinon + istanbul

Sources
- https://facebook.github.io/jest/

Pure function is the first step

Pure function

- No side effects
    - Doesn’t update variables outside its own scope
- Always return the same results
    - E.g.: Math.random() (js) is impure

Sources
- http://www.nicoespeon.com/en/2015/01/pure-functions-javascript/
- https://medium.com/javascript-scene/master-the-javascript-interview-what-is-a-pure-function-d1c076bec976

Example

Impure function

```
let a = 2;
export const sumImpure = (b) => {
    a = a + b; // Access var outsite func
    return a
}
```

Let’s test it...

with Jest

Jest

- Unit test framework for javascript
- Created by facebook
- Agnostic but made to work well with React
- Inspired by Jasmine
- Allows snapshot
- Jest = Jasmine + sinon + istanbul

Sources

- https://facebook.github.io/jest/

Now test our function

Let’s analyze the command

```
jest*  impure.test.js
```

* Assuming jest is installed in global or run from our npm scripts.
If not you’ll need to prefix it with ‘node_modules/.bin/’

Let’s analyze the command

Jest command

```
jest* impure.test.js
```

File(s) to test

* Assuming jest is installed in global or run from our npm scripts.
If not you’ll need to prefix it with ‘node_modules/.bin/’

Example

Pure function

```
export const sumPure = (a=0, b=0) => {
    return a + b;
}
```

It works well but...

Did we manage every case?

Code coverage

- Works hand in hand with unit tests
- Detects useless code
- Forces to manage every case / scenario
- Returns the percentage of code executed

- “--coverage” param with Jest
- Jest relies on istanbul for it

Let’s check the coverage

Code coverage details

Statements (Stmts):

Proportion of statements / instructions executed

Examples:

```
    const a = 42; // It’s a statement
    const b = 42; console.log(b) // It’s two statements
```

Branches (Branch):

Conditional statements executed (if / else, switch...)

Sources

- https://www.w3schools.com/js/js_statements.asp
- http://2ality.com/2012/09/expressions-vs-statements.html
- https://github.com/dwyl/learn-istanbul
- https://github.com/gotwarlost/istanbul/issues/639

Code coverage details

Functions (Funcs):

Proportion of functions / methods called

Lines (lines):

Proportion of lines executed

Sources
- http://2ality.com/2012/09/expressions-vs-statements.html
- https://github.com/dwyl/learn-istanbul

Let’s improve the coverage

Code coverage details

- /coverage/ to get more details about the latest code coverage

100 % code coverage,
it’s not a dream.
It can be a reality.

Best practices

- Be explicit (You know what you’re testing)
- Don’t try to replace QA / functional tests
- Add them to your precommit
- TDD (Write your test then code)
- Define a naming convention
- Create a mocks folder for your mocked data
- Considerate **unit tests** as a task

Best practices - Mocking

```
export const listItemsTpl = (apiRes) => (
  apiRes
    .filter(entry => entry.name)
    .map(entry => (
      `<li>${entry.name}</li>`
    ))
)
```

Best practices - Mocking

- Avoid (real) API calls
- Improves unit tests’ speed
- Enforce the concept:
  One function, one functionality /
  Single Responsibility Principle

Sources
- https://en.wikipedia.org/wiki/Single_responsibility_principle

Conclusion

- **Unit testing** allows to improve code quality / reduces number of bugs
- **Code coverage** detects useless code
- Unit tests **mustn’t** replace QA. Functional tests exist for this.

More sources

- Presentation + examples :

https://github.com/DanYellow/presentations/tree/master/unit-tests