React.js avanzado

Patricio López Juri

Contenidos

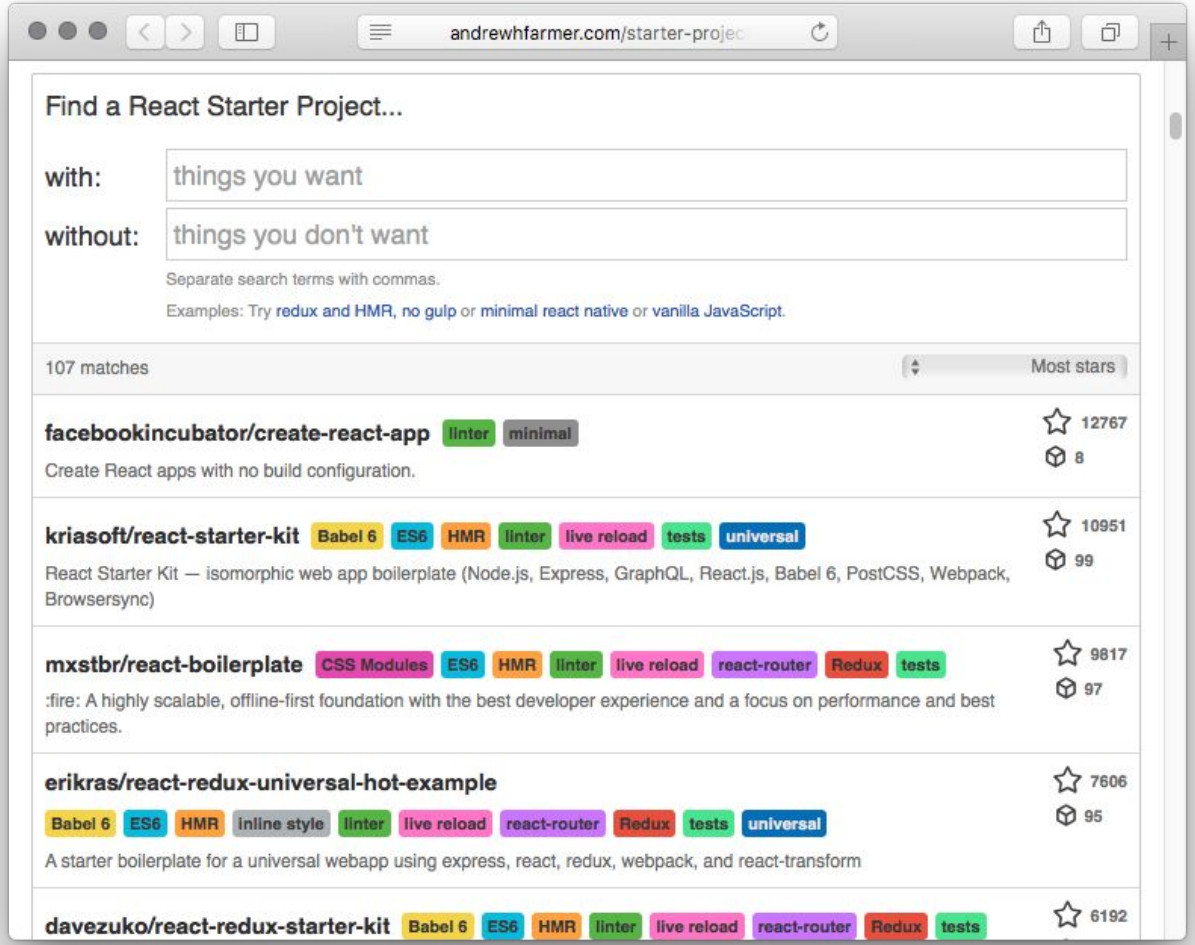
- Generar una app en React.js
- Integrar con algún Framework UI (como Bootstrap)
- Estilo avanzado
- Sacando provecho de ES7+
- Patrón HOC (High Order Components)
- ¿Patrón MVC?
- Flux
- Redux

Generar una app en React.js

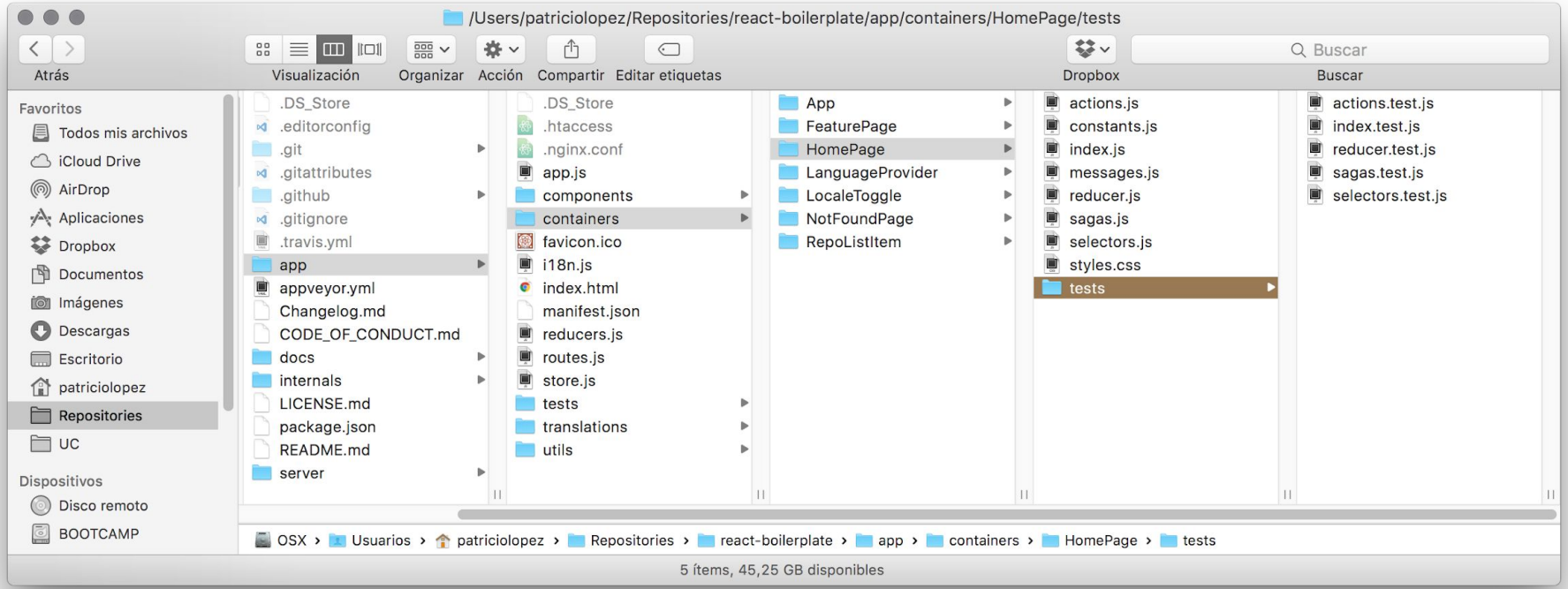
Generar una app en React.js

Antiguamente (22 Julio) había que configurar Webpack, Babel y muchas cosas a mano.

- Una opción era tomar uno de los **miles** de *boilerplates* o *templates.*
  - http://andrewhfarmer.com/starter-project/


andrewhfarmer.com/starter-projec
Find a React Starter Project...
with:
things you want
without:
things you don't want
Separate search terms with commas.
Examples: Try redux and HMR, no gulp or minimal react native or vanilla JavaScript.
107 matches
Most stars
facebookincubator/create-react-app
linter
minimal
Create React apps with no build configuration.
12767
8
kriasoft/react-starter-kit
Babel 6
ES6
HMR
linter
live reload
tests
universal
React Starter Kit — isomorphic web app boilerplate (Node.js, Express, GraphQL, React.js, Babel 6, PostCSS, Webpack, Browsersync)
10951
99
mxstbr/react-boilerplate
CSS Modules
ES6
HMR
linter
live reload
react-router
Redux
tests
:fire: A highly scalable, offline-first foundation with the best developer experience and a focus on performance and best practices.
9817
97
erikras/react-redux-universal-hot-example
Babel 6
ES6
HMR
inline style
linter
live reload
react-router
Redux
tests
universal
A starter boilerplate for a universal webapp using express, react, redux, webpack, and react-transform
7606
95
davezuko/react-redux-starter-kit
Babel 6
ES6
HMR
linter
live reload
react-router
Redux
tests
6192

Todo esto para tener un proyecto tan simple como...

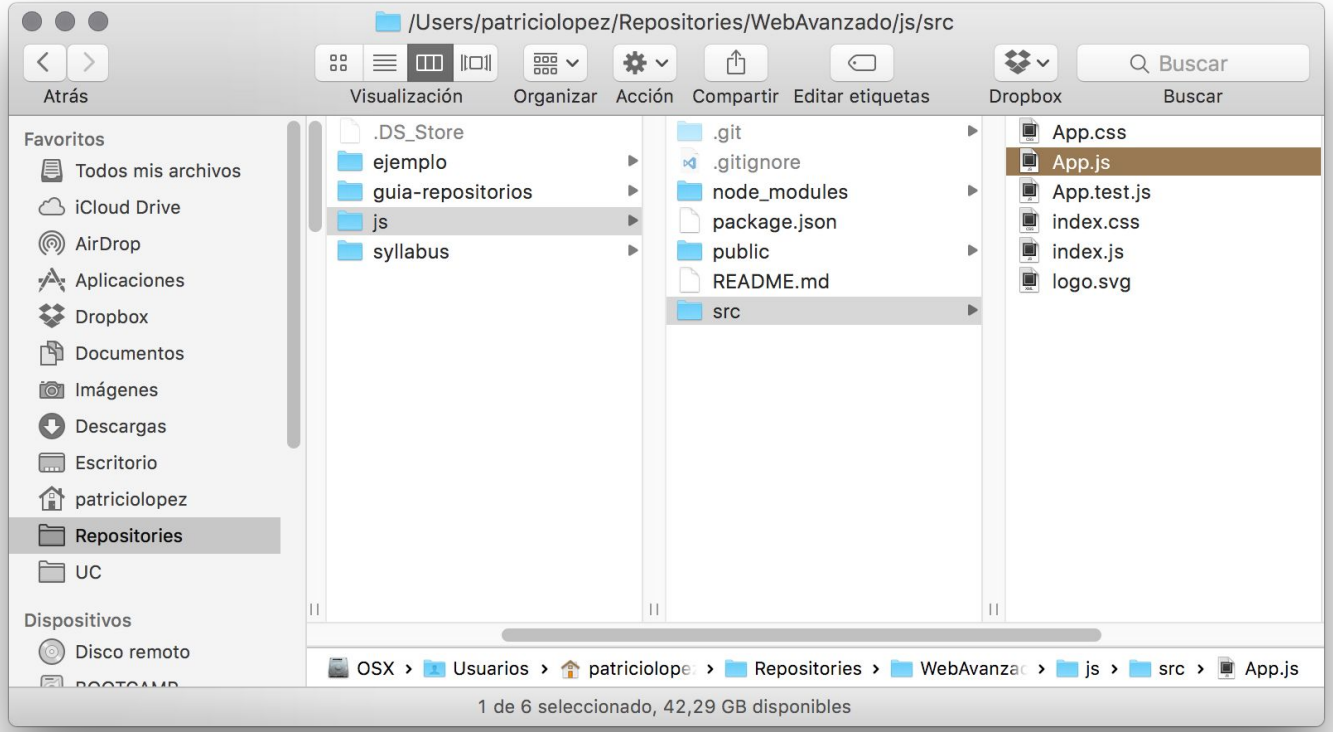


Facebook creó un generador oficial

https://github.com/facebookincubator/create-react-app

```
npm install -g create-react-app
create-react-app web-avanzado
cd web-avanzado/
npm start
```

Ahora sí es más simple



Integrar con Bootstrap

Cargar CSS externo

Por debajo, *create-react-app* tiene configurado *webpack* para **cargar desde el Javascript archivos *.css***

```
npm install --save bootstrap
```

```
import React, { Component } from 'react';

import 'bootstrap/dist/css/bootstrap.css';

class HelloWorld extends Component {
  render() {
    return (
      <button className="btn btn-primary">
        Hola
      </button>
    );
  }
}
```

Como librerías

- react-bootstrap: https://github.com/react-bootstrap/react-bootstrap
- elemental-ui: http://elemental-ui.com/
- material-ui: http://www.material-ui.com/

… y muchas más

¿CSS es cosa del pasado?

```
import React, { Component } from 'react';

class HelloWorld extends Component {
  render() {
    return (
      <h1 style={styles.title}>
        Hola
      </h1>
    );
  }
}
```

```
const styles = {
  title: {
    color: 'red',
    fontSize: 18,
  },
};
```

¿CSS es cosa del pasado?

Si puedo declarar el estilo en JS tengo **más control**, **mejor encapsulamiento** y puedo aplicar **lógica más compleja a las vistas**.

inline-styles tiene limitantes

Hay cosas de CSS que no puedo expresar en JS.

Por ejemplo en CSS:

```
@media (max-width: 600px) {
  .facet_sidebar {
    display: none;
  }
}
```

```
a:hover {
  color: #FF00FF;
}
```

Hay librerías para eso:

https://github.com/FormidableLabs/radium

```
const style = {
  width: '25%',

  '@media (min-width: 320px)': {
    width: '100%',

    ':hover': {
      background: 'white'
    },
  },
};
```

Aprovechen ES7+

Stateful

```
var React = require('React');

var Message = React.createClass({
  change: function(e) {
    this.setState({ text: e.target.value });
  },
  render: function() {
    return <input value={this.state.text} onChange={this.change} />;
  },
};
```

```
import React, { Component } from 'react';

class Message extends Component {
  constructor(props) {
    super(props);
    this.change = this.change.bind(this);
  }
  change(e) {
    this.setState({ text: e.target.value });
  }
  render() {
    return <input value={this.state.text} onChange={this.change} />
  }
}
```

```
import React, { Component } from 'react';

class Message extends Component {
  change = (e) => this.setState({ text: e.target.value })
  render = () => (
    <input value={this.state.text} onChange={this.change} />
  )
}
```

```
import React, { Component } from 'react';

class Message extends Component {
  state = {
    text: 'initial',
  }
  change = (e) => this.setState({ text: e.target.value })
  render = () => (
    <input value={this.state.text} onChange={this.change} />
  )
}
```

```
import React, { Component, PropTypes } from 'react';

class Message extends Component {
  static propTypes = {
    interval: PropTypes.number,
  }
  state = {
    text: 'initial',
  }
  change = (e) => this.setState({ text: e.target.value })
  render = () => (
    <input value={this.state.text} onChange={this.change} />
  )
}
```

Aprovechen ES7+

Stateless

```
function Button(props) {
  return (
    <button>
      {props.text}
    </button>
  );
}
```

```
const Button = ({ text, ...props }) => (
  <button {...props}>
    {text}
  </button>;
);
```

HOC

High Order Components

HOC

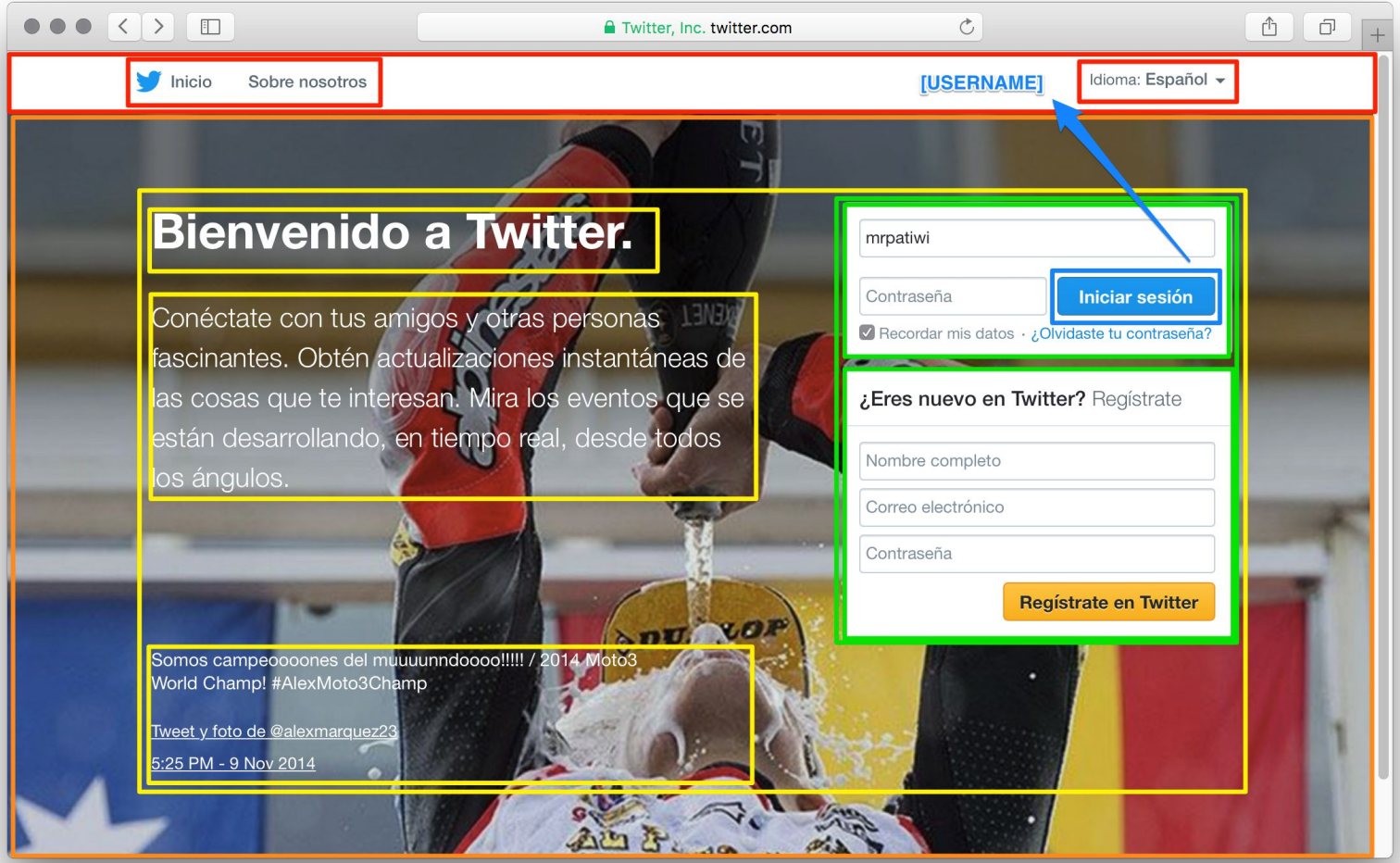
- Patrón decorador
- Envolver un componente con una función
- Reemplazo a los obsoletos *mixins*

Ejemplo: decorador que modifica el color del componente:

```
import React, { Component } from 'react';

function colorized(Comp) {
  const override = {
    color: 'red',
  };
  return class Decorated extends Component {
    render() {
      const { style, ...props } = this.props;
      return (
        <Comp style={{ ...style, ...override }} {...props} />
      );
    }
  }
}
```

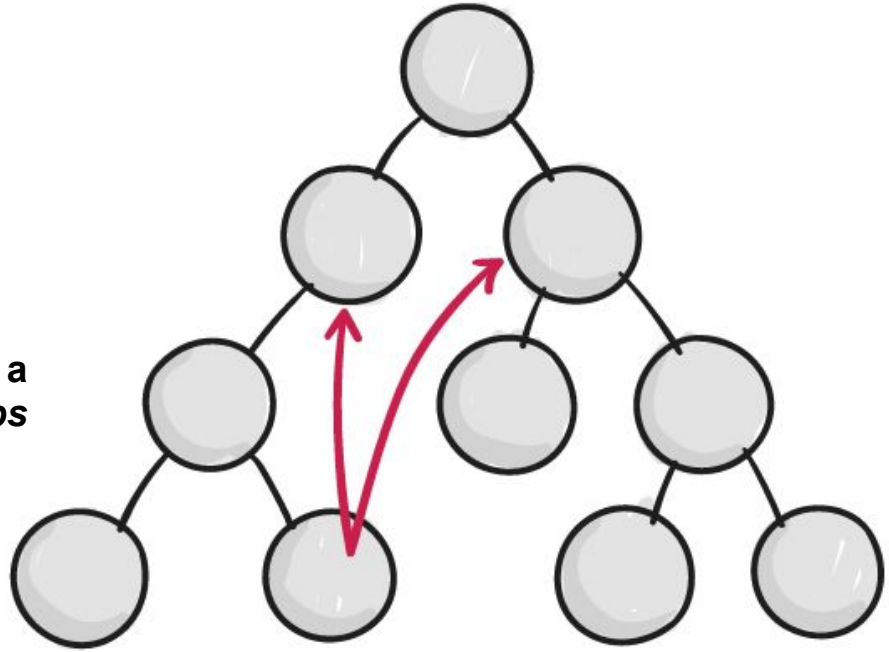
```
export class Message extends Component {
  // ...
}

export default colorized(Message);
```

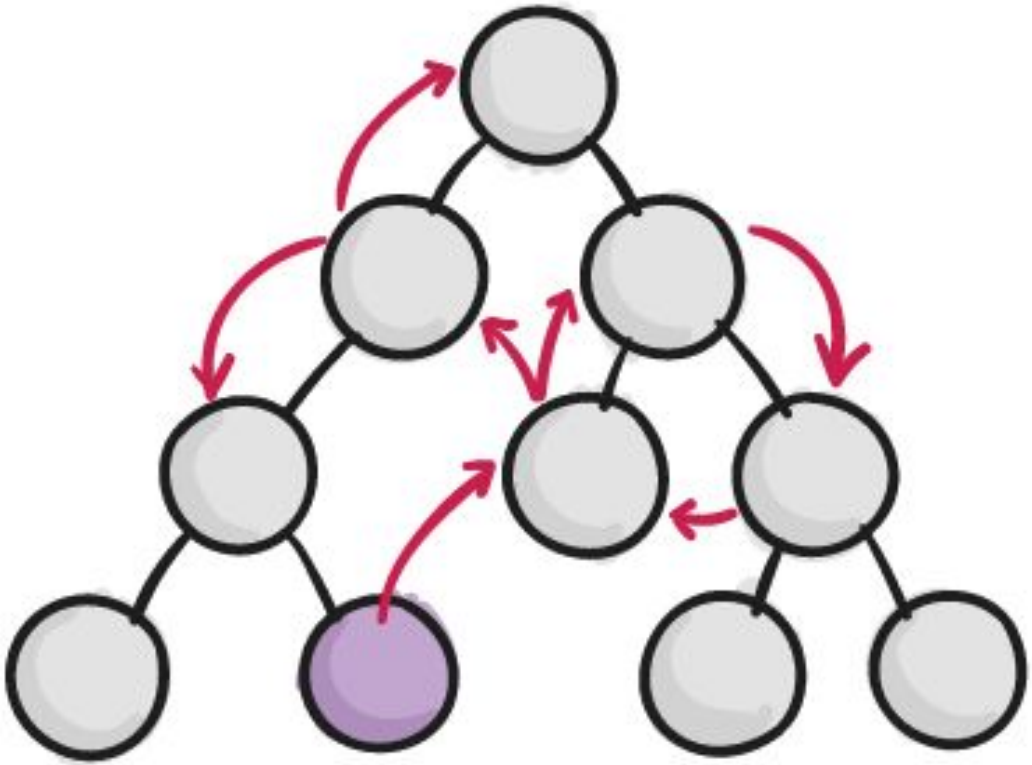
¿Cómo hacemos que un componente hable con otro más lejano?


Twitter, Inc. twitter.com
Inicio
Sobre nosotros
[USERNAME]
Idioma: Español
Bienvenido a Twitter.
Conéctate con tus amigos y otras personas fascinantes. Obtén actualizaciones instantáneas de las cosas que te interesan. Mira los eventos que se están desarrollando, en tiempo real, desde todos los ángulos.
Somos campeoooones del muuuunndoooo!!!!! / 2014 Moto3 World Champ! #AlexMoto3Champ
Tweet y foto de @alexmarquez23
5:25 PM - 9 Nov 2014
mrpatiwi
Contraseña
Iniciar sesión
Recordar mis datos · ¿Olvidaste tu contraseña?
¿Eres nuevo en Twitter? Regístrate
Nombre completo
Correo electrónico
Contraseña
Regístrate en Twitter

¿Cómo hacemos que un componente hable con otro más lejano?

Pasamos *callbacks* y valores a través de las *props*



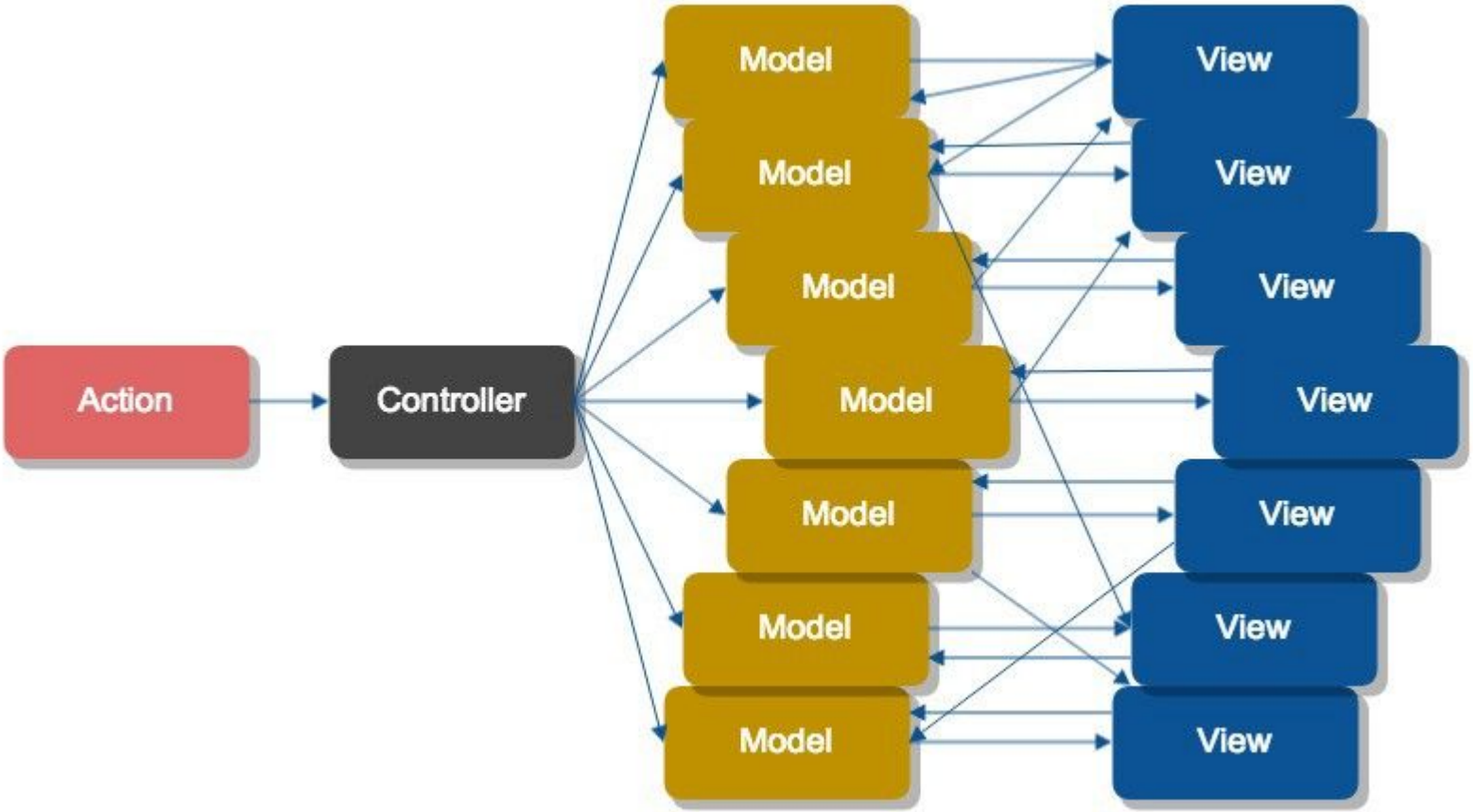


Fuente: https://css-tricks.com/learning-react-redux/

Patrón arquitectónico MVC

“A ver,

¿quiénes de aquí conocen MVC?”



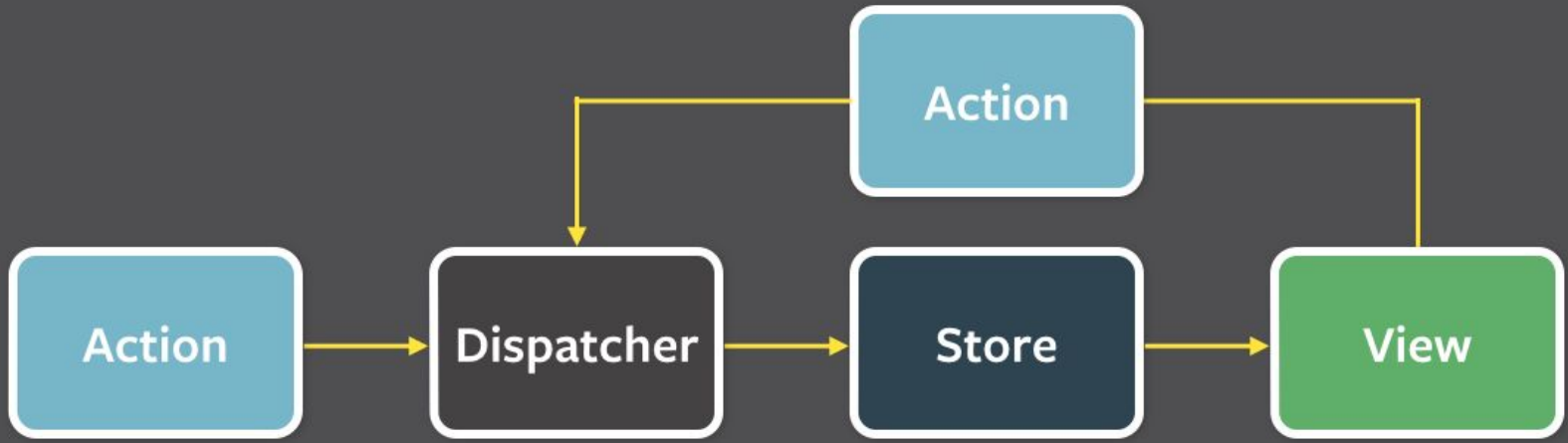
Fuente: *http://slidedeck.io/jchristianhall/redux-101*

Flux

Flux

- Patrón arquitectónico propuesto por Facebook para usarlo con React.js
- Similar a lo que vimos en Angular2 y sus *directivas*.
- **Funciona en base a eventos.**

Fuente: https://facebook.github.io/flux/docs/overview.html



```
import EventEmitter from 'events';

const CHANGE_EVENT = 'change';

class Store extends EventEmitter {
  emitChange() {
    this.emit(CHANGE_EVENT);
  }
  addChangeListener(callback) {
    this.on(CHANGE_EVENT, callback);
  }
  removeChangeListener(callback) {
    this.removeListener(CHANGE_EVENT, callback);
  }
}

export default new Store();
```

```
import React, { Component } from 'react';
import store from './store';

class List extends Component {
  componentDidMount() {
    store.addChangeListener(this.change);
  }
  componentWillUnmount() {
     store.removeListener(this.change);
  }
  change(args) {
    // ...
  }
  // ...
}
```

Otras implementaciones

- Alt.js
- Flummox
- Fluxxor
- Fluxible
- Lux
- Reflux

… etc

¿Qué está pasando aquí?

Todas estas siguen el paradigma OOP

… y al final todas son iguales:

- Stateful
- Alto acoplamiento entre *stores*
- Flux acoplado a las vistas (componentes)

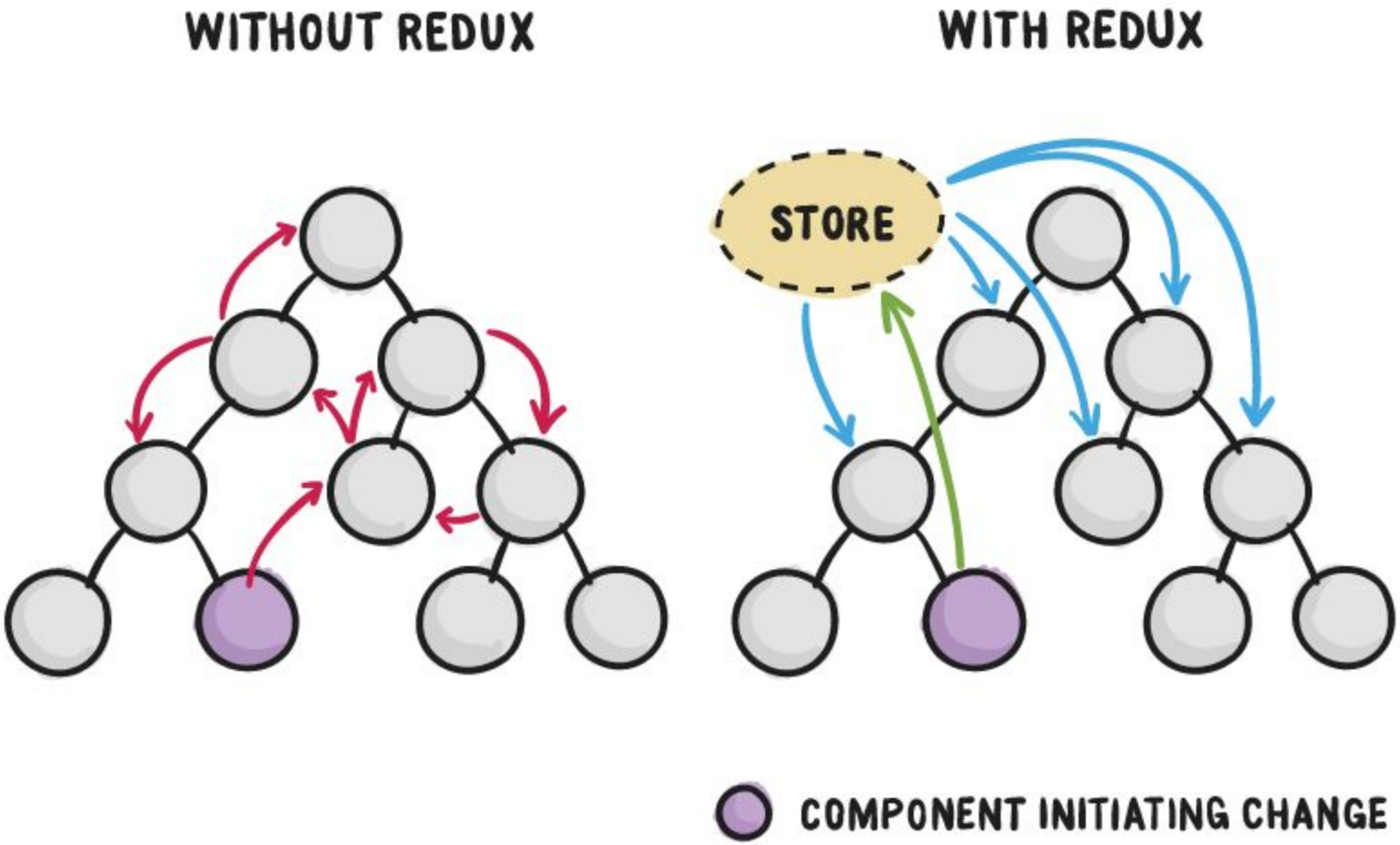
Redux

Un *approach* funcional

Redux

- Conversa los conceptos de *Store*, *Dispatcher* y *Action*.
- Hace uso de las ***props*** de los componentes en vez suscripciones a eventos.
- No los acopla las vistas a la implementación.
- La *store* es **la única fuente de verdad** y contiene el estado global de la app.
  - Es fácil de testear
- **El estado global es *inmutable*.**
- La *store* está compuestas por *reducers.*
  - La única manera de modificar el estado global es con llamadas a *reducers.*
  - Los *reducers* están compuestos por *reducers.*
  - Los *reducers* deben ser funciones puras.

(No está acoplado a React, se puede usar en cualquier ambiente JS)


WITHOUT REDUX
WITH REDUX
STORE
COMPONENT INITIATING CHANGE

Getting started

Instalamos Redux:

```
npm install --save redux react-redux
```

Constantes

```
const ADD_TODO = 'ADD_TODO';
const REMOVE_TODO = 'REMOVE_TODO';
```

Reducers

```
function todos(state = [], action) {
  switch (action.type) {
  case ADD_TODO:
    return [...state, action.text];
  case REMOVE_TODO:
    return [
      ...state.slice(0, action.index),
      ...state.slice(action.index + 1),
    ];
  default:
    return state;
  }
}
```

Reducers

```
const reducers = combineReducers({
  todos,
});
```

Store

```
const store = createStore(reducers);
```

Actions

```
const addTodoAction = (text) => ({
  type: ADD_TODO,
  text,
});
const removeTodoAction = (index) => ({
  type: REMOVE_TODO,
  index,
});
```

Conectarlos

```
class App extends Component {
  // ...
}

const mapDispatchToProps = (dispatch) => ({
  add(text) {
    dispatch(addTodoAction(text));
  },
});

const mapStateToProps = (state) => ({
  todos: state.todos,
});

export default connect(mapStateToProps, mapDispatchToProps)(App);
```

Lanzar app

```
ReactDOM.render(
  <Provider store={store}>
    <ReduxApp />
  </Provider>,
  document.getElementById('root')
);
```

DEMO

https://github.com/IIC3585-2016-2/react-redux

Dan Abramov Follow
Librarian. Team @reactjs. Created Redux, React Hot Loader, React DnD. Personal opinions.
Sep 19 · 3 min read

You Might Not Need Redux

People often choose Redux before they need it. “What if our app doesn’t scale without it?” Later, developers frown at the indirection Redux introduced to their code. “Why do I have to touch three files to get a simple feature working?” Why indeed!

People blame Redux, React, functional programming, immutability, and many other things for their woes, and I understand them. It is natural to compare Redux to an approach that doesn’t require “boilerplate” code to update the state, and to conclude that Redux is just complicated. In a way it is, and by design so.

Redux offers a tradeoff. It asks you to: