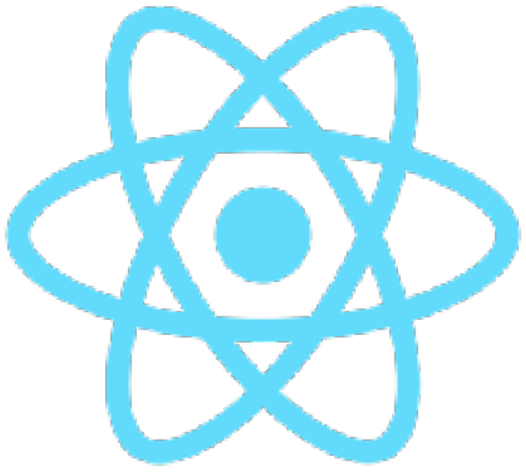
Javascript / React

Redux



Redux

Rappels et compléments

State : état (données) de l'application

```
const state = {
    blog: {
        posts: [
            {id: 1, title: 'Article 1', text: '…'},
            {id: 2, title: 'Article 2', text: '…'},
            {id: 3, title: 'Article 3', text: '…'},
            {id: 4, title: 'Article 4', text: '…'},
            {id: 5, title: 'Article 5', text: '…'},
            {id: 6, title: 'Article 6', text: '…'},
        ]
    }
};
```

State : Immuable

```
// NON !
state.blog.post[0].title = 'Test';
```

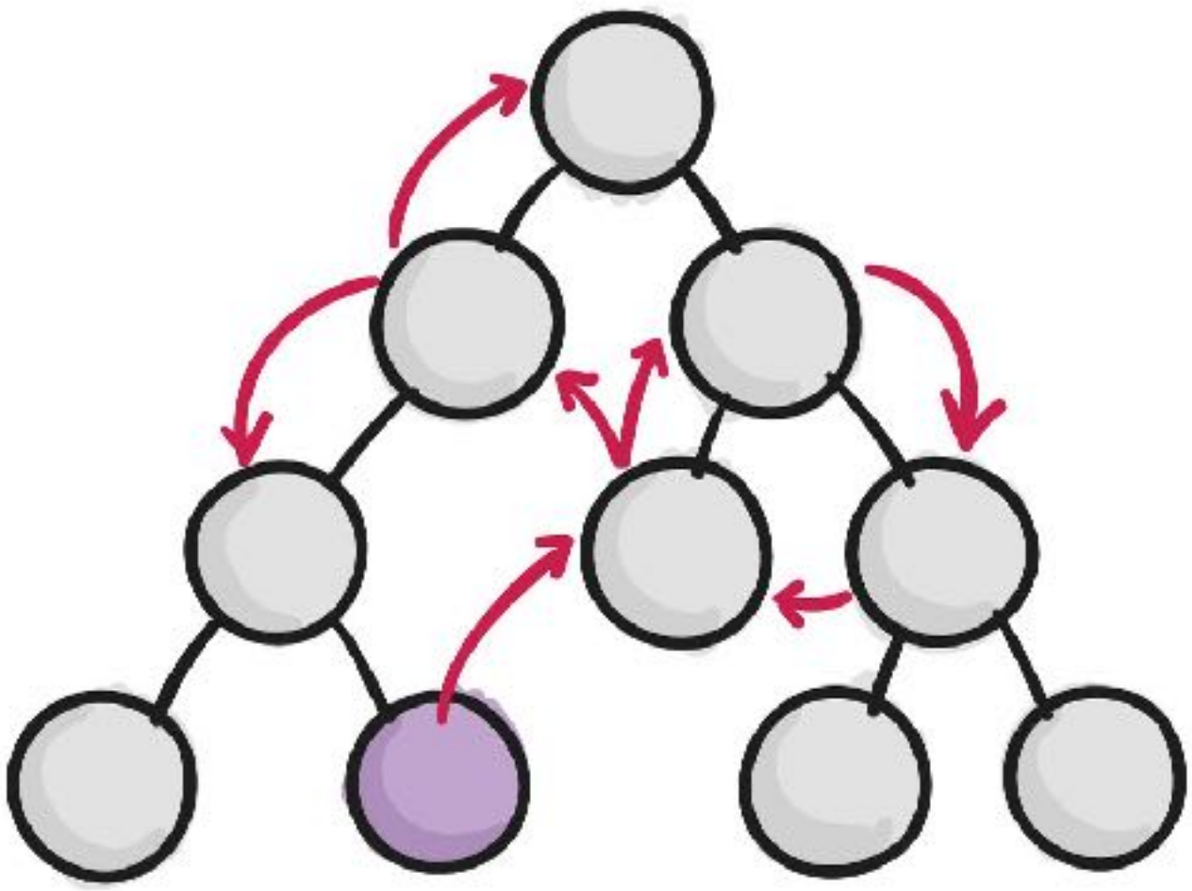
Les composants récupèrent leurs données par « Props »

Ces props sont issues du « Global state »

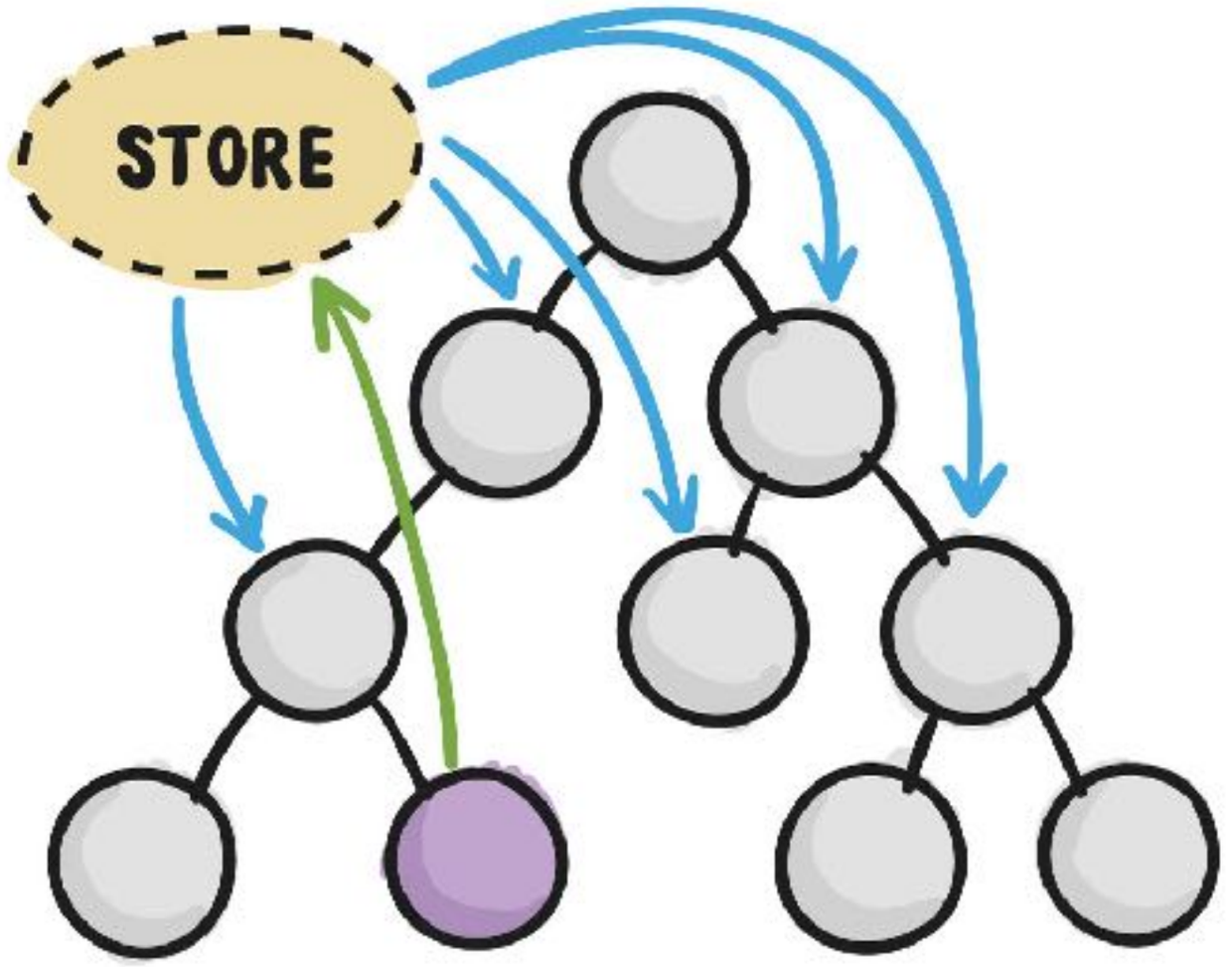
State => props

```
class BlogPosts extends Component {
    render() {
        return (
            <ul>
                {this.props.posts.map(function (post) {
                    return <li>{post.title}</li>;
                })}
            </ul>
        );
    }
}

const mapStateToProps = function (state) {
    return {posts: state.blog.posts};
};

export default connect(mapStateToProps)(BlogPosts);
```

WITHOUT REDUX



WITH REDUX



COMPONENT INITIATING CHANGE

Le composant est mis à jour automatiquement

Lors du changement de ses props, et donc, du state.

State immutable

Les « réducteurs » génèrent un nouveau « state » lors d’actions.

State immutable

Les réducteurs définissent également l’état par défaut

State : état initial

```
const defaultState = [
    {id: 1, title: 'Article 1', text: '…'},
    {id: 2, title: 'Article 2', text: '…'},
    //…
];

const posts = function (state = defaultState, action)
{
    // …

    return state;
};
```

Actions

Les actions sont les « événements » contenant les données pour modifier le state.

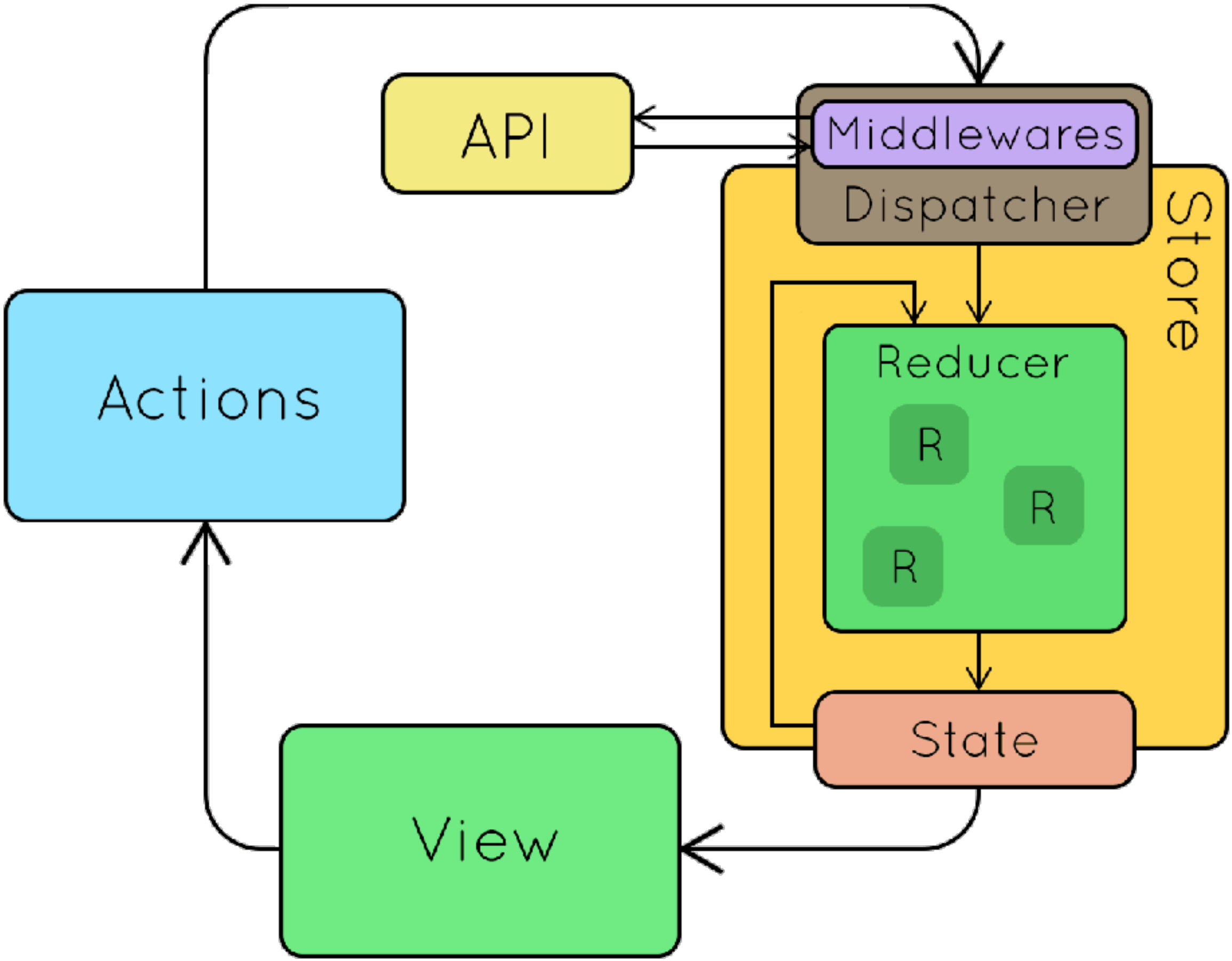
Elles sont lancées par les composants

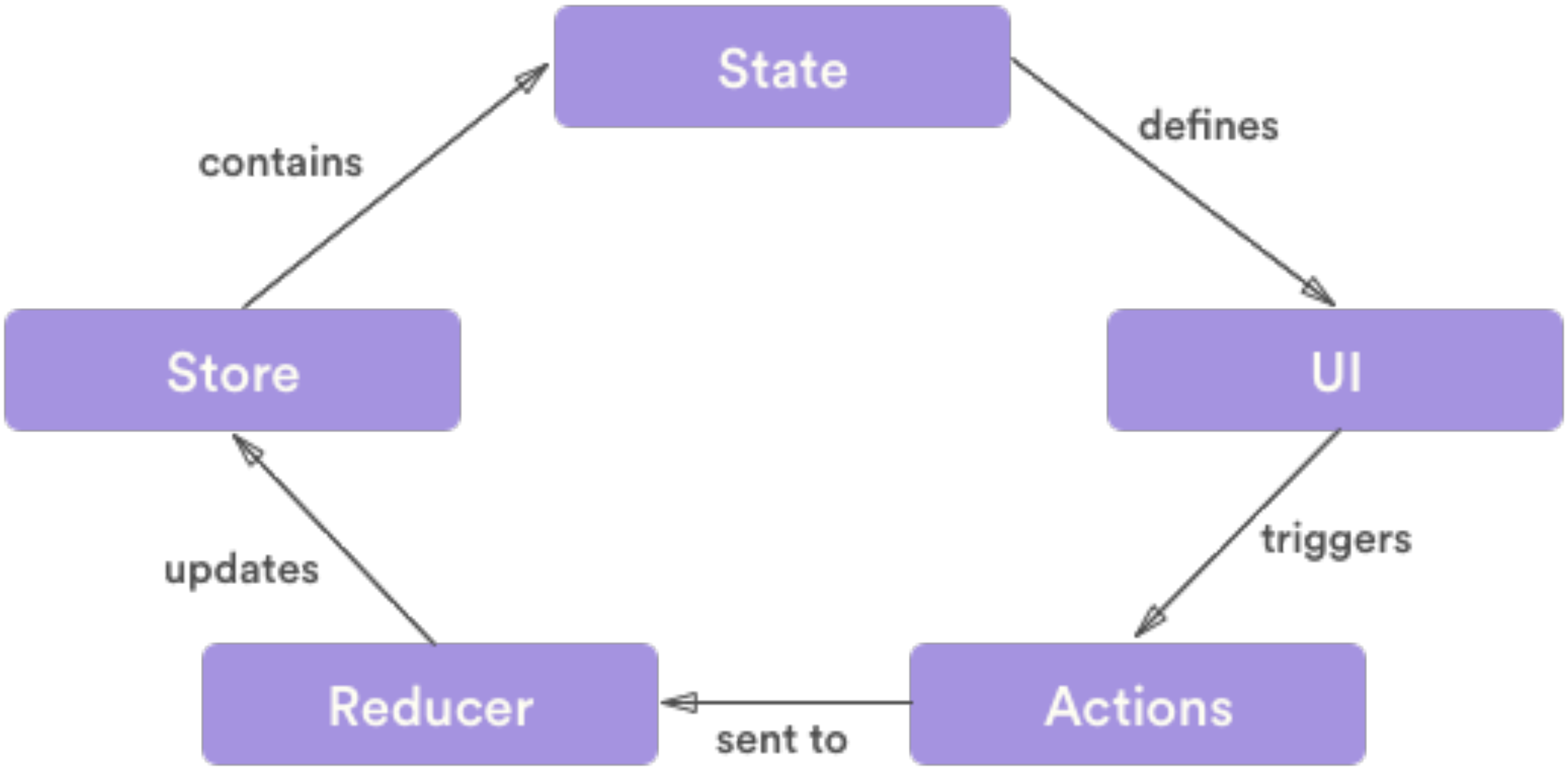
Actions / Réducteurs

```
// Lancer une action
this.props.dispatch({
    type: 'CREATE_BLOG_POST',
    post: {id: 7, title: 'Article 7', text: '…'}
});

// L'utiliser pour modifier le state avec un réducteur
const posts = function(state = defaultState, action) {
    if (action.type === 'CREATE_BLOG_POST') {
        return [ ... state, action.post];
    }

    return state;
};
```

Actions

```
class Form extends Component {
    static propTypes = {dispatch: PropTypes.func.isRequired};

    render() {
        return (
            <form onSubmit={(event) ⇒ {
                this.props.dispatch({
                    type: '…',
                    post: {
                        title: e.target.elements[0].value,
                        title: '…'
                    },
                });
            }}>
                <input type="text" />
                <button type="submit">Ajouter</button>
            </form>
        );
    }
}
export default connect(/* … */)(Form);
```


API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View


State
contains
defines
Store
UI
updates
triggers
Reducer
sent to
Actions

TP

Application de TODO

Todo

- Lister les tâches à faire (fonction « connect »)
- Pouvoir créer des « tâches à faire » (fonction « dispatch » et réducteur
- Pouvoir les marquer comme « fait »
- Les stocker en LocalStorage