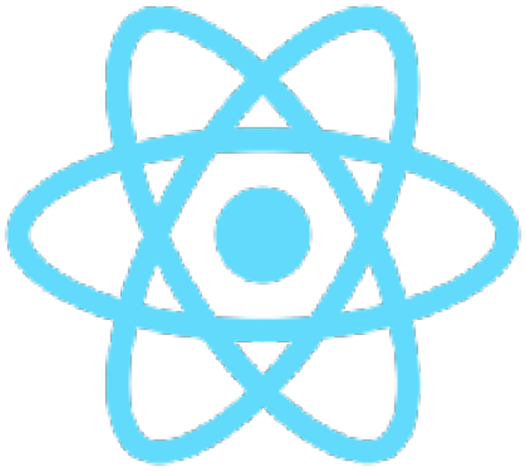
Javascript / React

Redux



Correction

reducer.js

```
import {combineReducers} from 'redux';

const todos = (state = initialState, action) ⇒ {
    // …

    return state;
};

export default combineReducers({ todos });
```

Réducteur « todo »

```
const todos = (state = [], action) ⇒ {
    if ('add_todo' === action.type) {
        return [...state, {title: action.title, done: false}]
    }
    if ('do_todo' === action.type) {
        return  [...state].map(todo ⇒ {
            if (todo.title === action.todo.title) {
                return {...todo, done: true};
            }

            return todo;
        })
    }

    return state;
};
```

Liste des « todo »

```
class TodoList extends Component {
    static propTypes = { /* … */ };

    render() {
        return (
            <ul>
                {this.props.todos
                    .sort((left, right) ⇒ left.done)
                    .map((todo, i) ⇒ (
                        <Todo key={'todo-' + i} {...todo} />
                    ))}
            </ul>
        );
    }
}

const mapStateToProps = function(state) {
    return {todos: state.todos}
};
export default connect(mapStateToProps)(TodoList);
```

Affichage d’un « todo »

```
class Todo extends Component {
    render() {
        return (
            <li style={this.props.done ? {background: 'grey'} : {}}>
                <div>
                    <h4>
                        {this.props.title}
                        <button onClick={this.props.onClick}>Faire</button>
                    </h4>
                </div>
            </li>
        );
    }
}

export default connect(
    state => ({}),
    (dispatch, ownProps) => ({
        onClick: () => dispatch({type: 'do_todo', todo}),
    })
)(Todo);
```

Asynchrone

Asynchrone

Problème :

Les réducteurs ne peuvent pas provoquer d’effets de bord

Asynchrone

Rappel :

Réducteur = fonction pure.

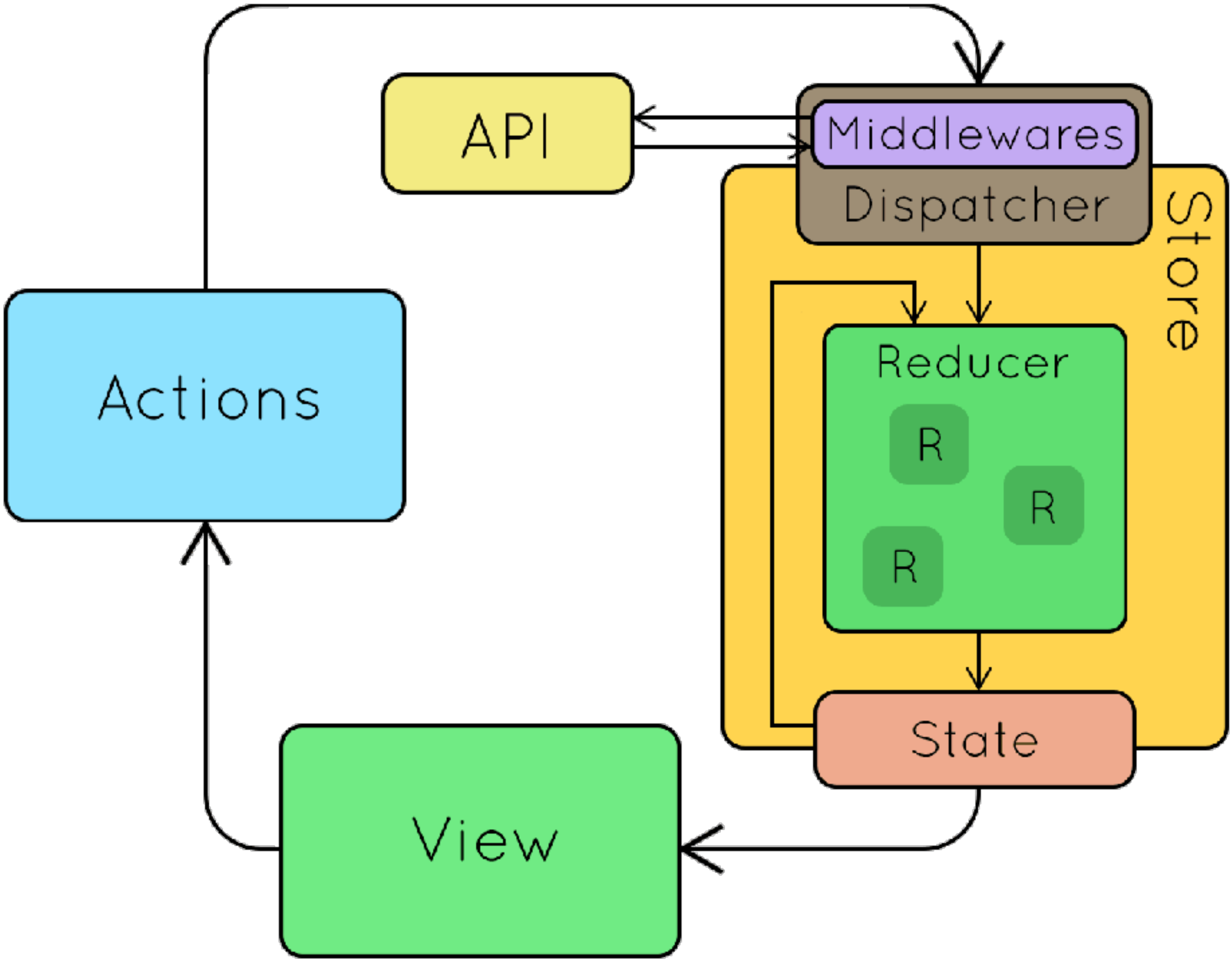
La valeur de retour est toujours la même pour des arguments donnés.

Asynchrone

Impossible donc de lancer une requête HTTP depuis un réducteur.

Asynchrone

Solution : Middleware


API
Middlewares
Dispatcher
Store
Actions
Reducer
R
R
R
State
View

Asynchrone

Action creators
redux-thunk

Action creator

Fonction qui retourne la structure d’une action

Action creator

```
// Avant
dispatch({type: 'get_todos'});

// Après
// actions.js
const getTodos = function() {
    return { type: 'get_todos' };
};

dispatch(getTodos());
```

Action creator

Dans le cas d’une action
« classique » (= synchrone) retourne
un simple objet

Action creator

Dans le cas d’une action asynchrone…

Retourne une fonction.

Action creator

```
export const RECEIVE_TODOS = 'RECEIVE_TODOS';
export const RECEIVE_TODOS_ERROR = 'RECEIVE_TODOS_ERROR';
export const requestTodos = () => async dispatch => {
    try {
        const res  = await fetch(
            'https://todo-api-s3im.herokuapp.com/todos'
        );
        const json = await res.json();

        dispatch({
            type: RECEIVE_TODOS,
            todos: json['hydra:member']
        });
    } catch (error) {
        dispatch({type: RECEIVE_TODOS_ERROR, error});
    }
};
```

Réducteur

```
const initialState = {
    todos: [],
};

const todos = (state = initialState, action) ⟹ {
    switch (action.type) {
        case RECEIVE_TODOS:
            return { ... state, todos: action.todos};
        default:
            return state;
    }
};
```

Thunk

```
$ yarn add redux-thunk

// Ou

$ npm install --save redux-thunk

// App.js
const middleware = [thunk];
if (process.env.NODE_ENV !== 'production') {
    middleware.push(createLogger());
}
```

Demander le contenu

```
componentWillMount() {
    this.props.dispatch(requestTodos());
}
```

Fetch

```
fetch(
    'https://todo-api-s3im.herokuapp.com' + todo['@id'],
    {
        method:  'PUT',
        headers: {
            'Accept':       'application/ld+json',
            'Content-Type': 'application/ld+json',
        },
        body:    JSON.stringify({
            // …
        }),
    },
);
```