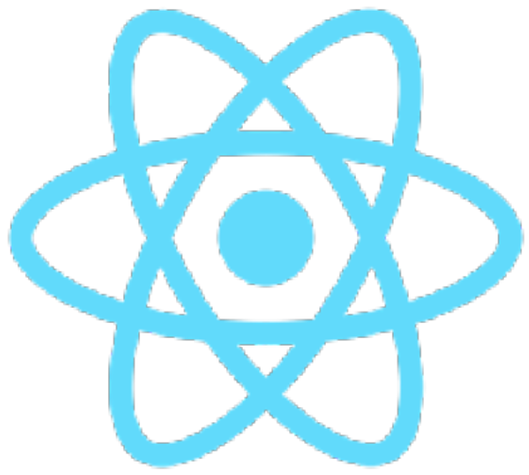
Javascript / React

Composants / Découpage



Correction

Client Reddit

Composant global

```
class App extends Component {
    state = {
        subreddit: 'reactjs',
    };

    render() {
        return (
            <div className="App">
                <Subreddit subreddit={this.state.subreddit} />
            </div>
        );
    }
}
```

Création du composant

```
export default class Subreddit extends Component {
    static propTypes = {
        subreddit: PropTypes.string.isRequired,
    };

    state = {
        threads: []
    };
    //…
}
```

Récupération des données

```
async componentWillMount() {
    const response = await fetch('https://
api.reddit.com/r/' + subreddit);
    const json     = await response.json();

    this.setState({threads: json.data.children});
}
```

Affichage

```
render() {
    return (
        <div>
            <h2>/r/{this.props.subreddit}</h2>
            <div>
                {this.state.threads.map(thread => (
                    <Thread key={thread.data.id} thread={thread.data} />
                ))}
            </div>
        </div>
    );
}
```

Affichage

```
const Thread = ({thread}) ⇒ (
    <div>
        <h3>{thread.title}</h3>
        <!—— … ——>
    </div>
);
```

Redux

Gérer l’état global de l’application

Regrouper toutes les données dans un unique objet

Plus d’utilisation du « state » des composants

Ou presque

Global state

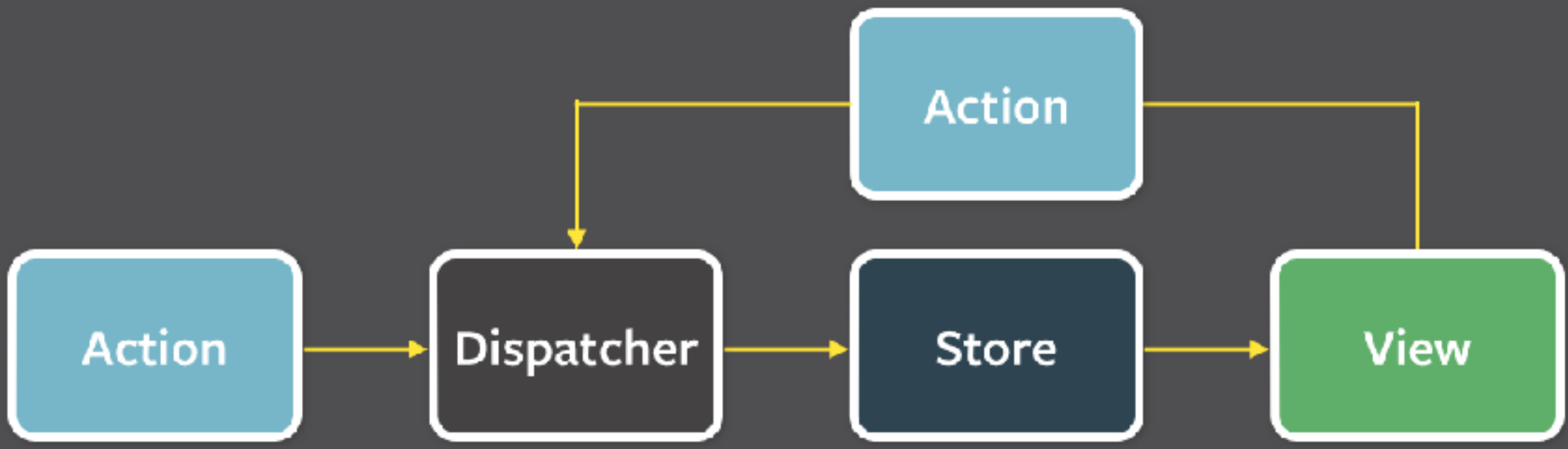
```
{
    visibilityFilter: 'SHOW_ALL',
    todos:                [
        {
            text:       'Consider using Redux',
            completed: true,
        },
        {
            text:       'Keep all state in a single tree',
            completed: false
        }
    ]
}
```

Le state redux est « immuable »…

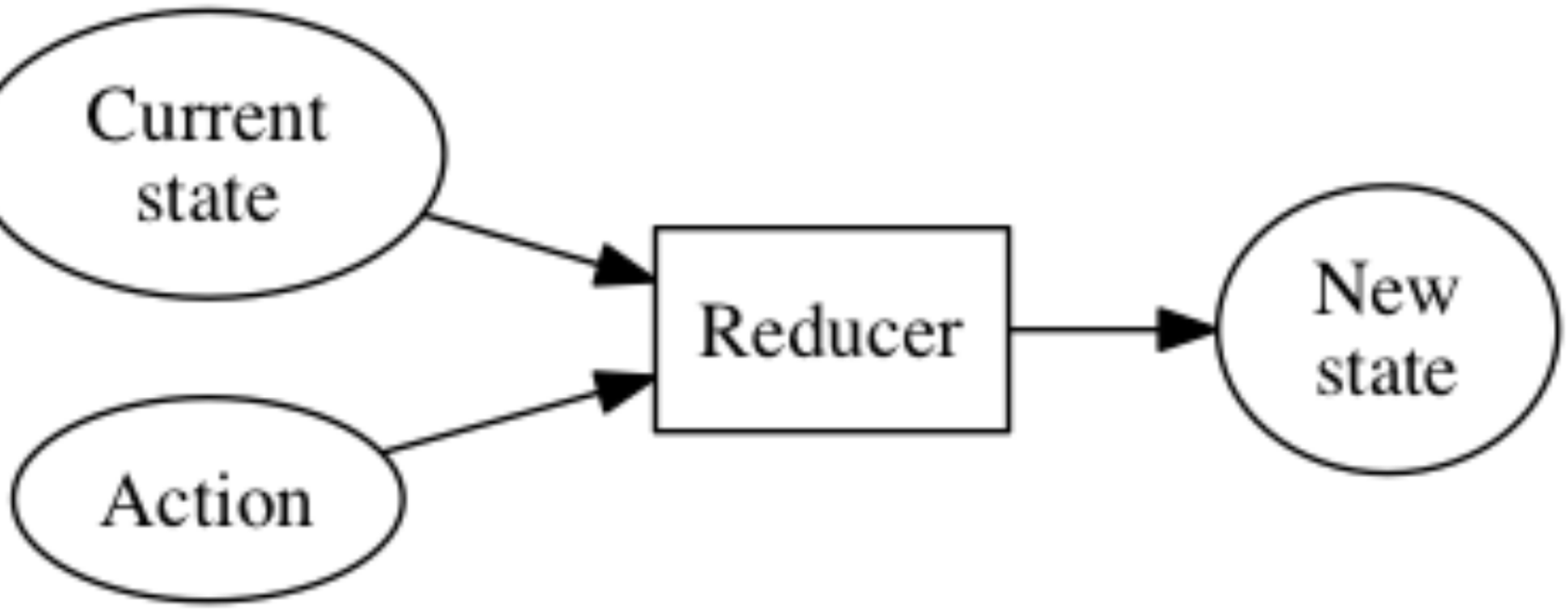
On ne le modifie pas, on le remplace

…et les changements sont fait par des fonctions pures

Les réducteurs


Action
Action
Dispatcher
Store
View

Pour générer un nouveau « state », les réducteurs utilisent le state actuel et une « action »


Current state
Action
Reducer
New state

Exemple :

Reducers

```
const state = { myNumber: 1 };

const reducer = function(state, action) {
    if (action.type === 'add') {
        return {myNumber: myNumber + action.value};
    }
    if (action.type === 'sub') {
        return {myNumber: myNumber - action.value};
    }

    return state;
}

const newState = reducer(state, {type: 'add', value: 5});
```

Exemples concrets

Reducers

```
const state = { myNumber: 1 };

const reducer = function(state, action) {
    if (action.type === 'add') {
        return {myNumber: myNumber + action.value};
    }
    if (action.type === 'sub') {
        return {myNumber: myNumber - action.value};
    }

    return state;
}

const newState = reducer(state, {type: 'add', value: 5});
```

Détail des fonctions redux

createStore

```
createStore(reducer, [preloadedState], [enhancer])
```

```
import {createStore} from 'redux';
const store = createStore(reducer, applyMiddleware( ...middleware));
```

createStore

```
combineReducers(reducers)
```

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

export default combineReducers({
    todos,
    // …
});
```

connect

```
connect([mapStateToProps], [mapDispatchToProps],
[mergeProps], [options])
```

```
const mapStateToProps = state ⟹ ({todos: state.todos});

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

Créer une application redux

Créer une application redux

```
$ create-react-app redux-todo
$ cd redux-todo
$ yarn add redux redux-logger
react-redux
```