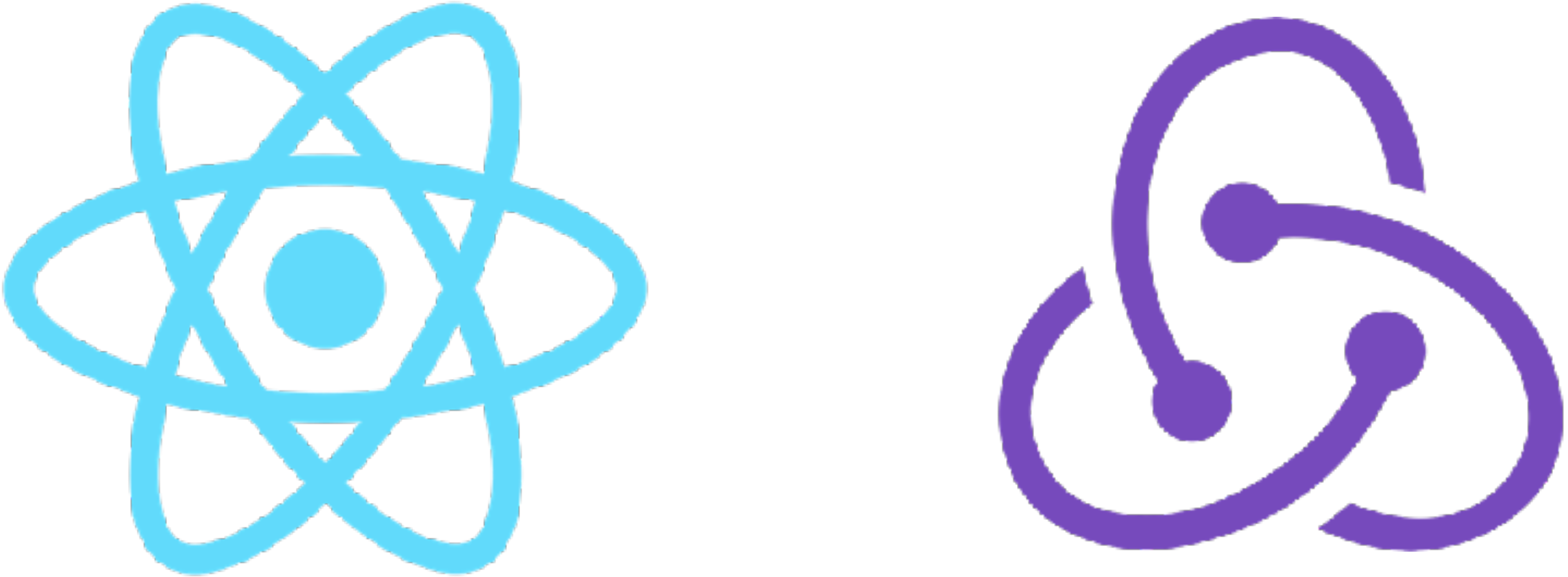


React + Redux

Créé par Anders Hejlsberg chez Microsoft

Support de l’ECMAScript 2015 et du TSX

Commence à être adopté par la majorité

webpack
Microsoft
JET BRAINS

Typescript

```
interface Person {
    firstName: string;
    lastName: string;
    age: number;
    job?: Job;
}


interface Job {
    name: string;
    location?: string;
}
```

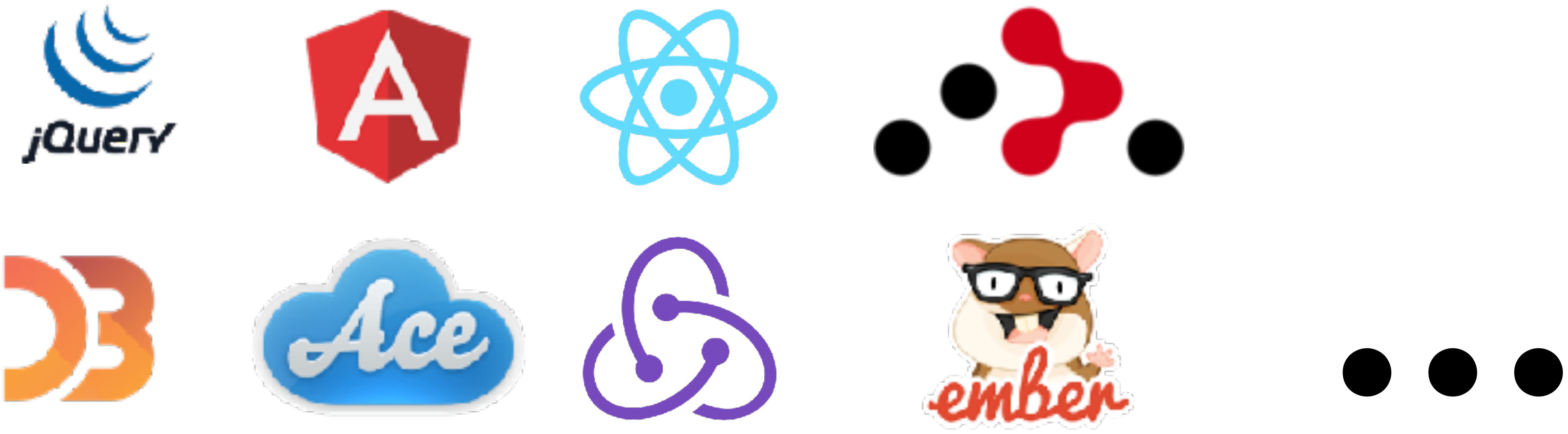
Typescript

```
interface Person {
    firstName: string;
    lastName: string;
    age: number;
    job?: Job;
}


interface Job {
    name: string;
    location?: string;
}
```

Typescript

Maintenu par la communauté

Beaucoup de @types déjà définis, presque 4000 différents



DefininitelyTyped: http://definitelytyped.org/

OKAY, THEN
LET'S BEGIN
memegenerator.net

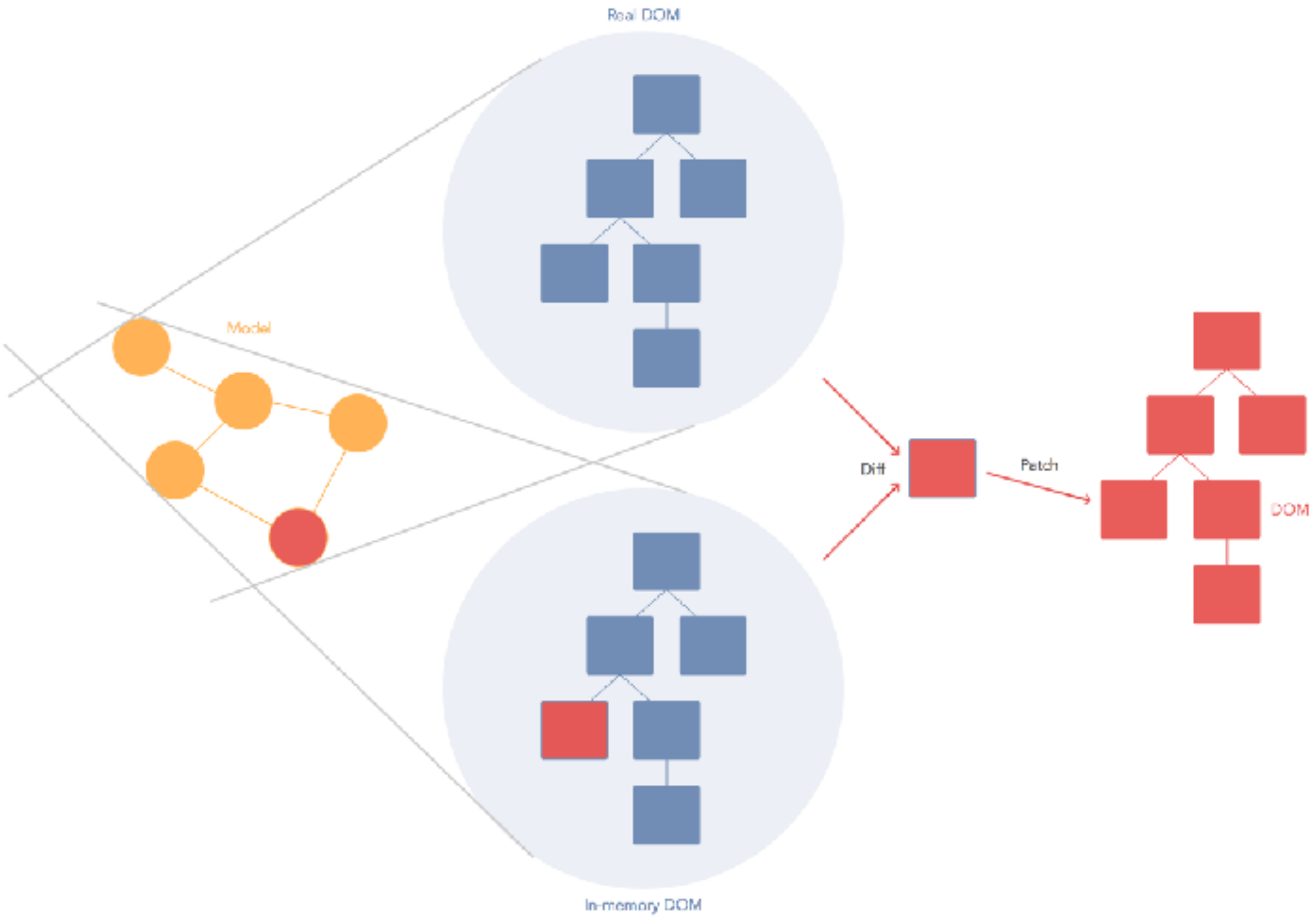
React

Développé par Facebook depuis 2013

Librairie JavaScript pour gérer les interfaces.

- **Virtual DOM**
- **Gère seulement l’interface**
- **Flux de données à sens unique**

Virtual DOM



Source: https://auth0.com/blog/face-off-virtual-dom-vs-incremental-dom-vs-glimmer/

Virtual DOM

Source: https://auth0.com/blog/face-off-virtual-dom-vs-incremental-dom-vs-glimmer/

Pourquoi le Virtual DOM ?

Mise à jour seulement du composant nécessaire

Mise à jour rapide du DOM

Algorithme trouvant le nombre minimum d’opérations à faire pour mettre à jour le DOM

Pourquoi le Virtual DOM ?

Mise à jour seulement du composant nécessaire

Mise à jour rapide du DOM

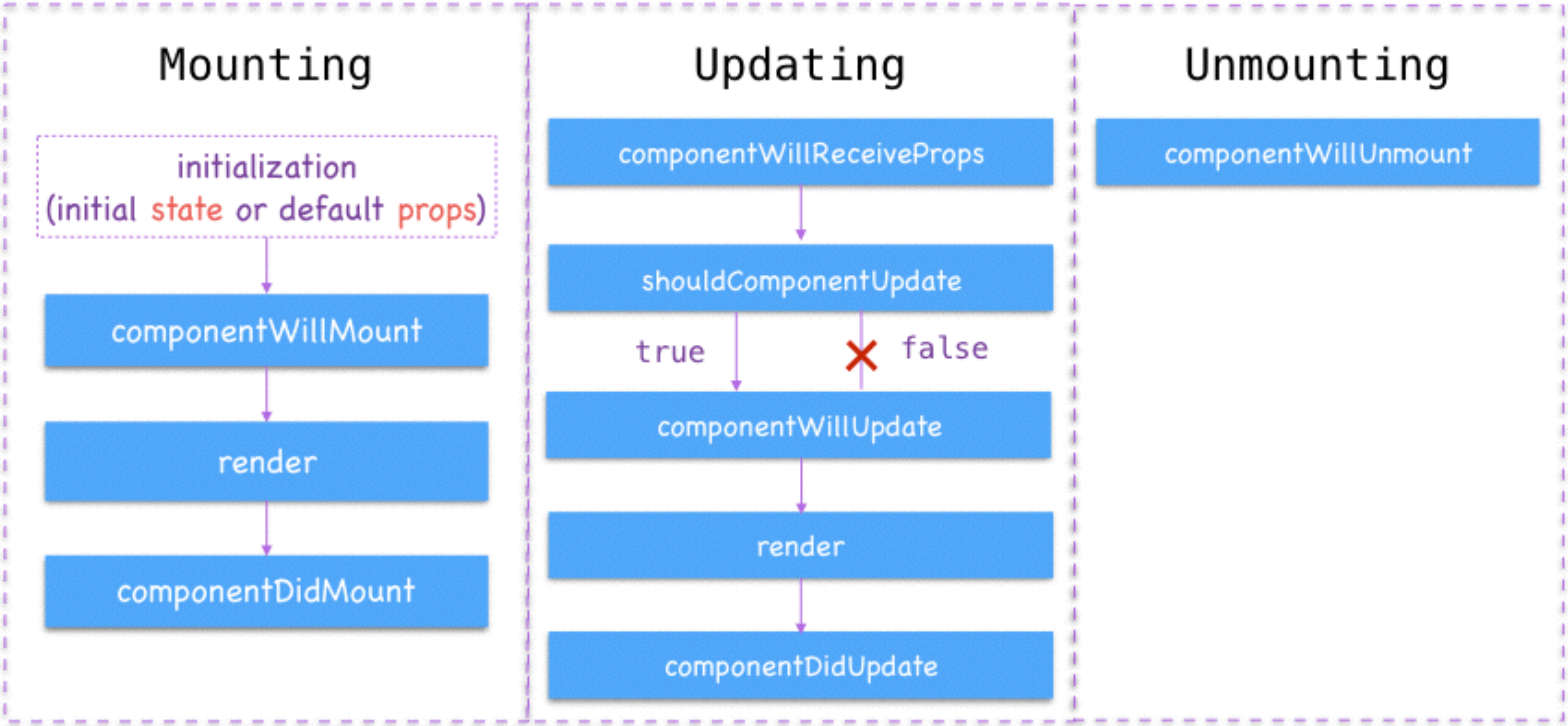
Algorithme trouvant le nombre minimum d'opérations à faire pour mettre à jour le DOM

APPROVED

React basics

TRY IT NOW!

Cycle de vie d’un component



Source: https://www.codevoila.com/post/57/reactjs-tutorial-react-component-lifecycle

Immutabilité

Changement de données avec mutation:

```
let person: Person = { firstName: 'Bryan', lastName: 'Johnson', age: 70 };
person.age = 73;
```

Changement de données sans mutation:

```
const person: Person = { firstName: 'Bryan', lastName: 'Johnson', age: 70 };
const newPerson: Person = { ...person, age: 73 };
```

Bénéfices:

- Facilité de faire / défaire
- Suivre plus facilement des changements
- Permettre de savoir quand re-render dans React

Webpack Configuration

```
…
module: {
    rules: [
        {
            test: /^(?!.*\.spec\.tsx?$).*\.tsx?$/,
            loader: 'awesome-typescript-loader',
            exclude: /node_modules/,
        },
        {
            test: /\.js$/,
            enforce: "pre",
            loader: "source-map-loader"
        },
        {
            test: /\.scss/,
            use: ExtractTextPlugin.extract({
                use: ['css-loader', 'sass-loader']
            })
        },
        {
            test: /\.(ttf|eot|svg|woff|woff2)(\?v=[0-9]\.[0-9]\.[0-9])?$/,
            use: {
                loader: 'url-loader?limit=1000'
            }
        },
        {
            test: /\.(jpe?g|png|gif|svg)$/i,
            use: [
                'url-loader?limit=10000',
                'img-loader'
            ]
        }
    ]
},
…
```

webpack.config.js

React

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
    <body>
        <div id="app"></div>
        <script src="dist/bundle.js"></script>
    </body>
</html>
```

index.html

Render un component

```tsx
import * as React from 'react';
import * as ReactDOM from 'react-dom';

const app = document.querySelector('#app');

ReactDOM.render(
    <div>Hello sir !</div>
, app);
```

index.tsx

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello sir !</div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Créer un sous component

Hello.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';

export default class Hello extends React.Component {

    render() {
        return (
            <div>Hello sir !!</div>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Hello from 'Hello';

const app = document.querySelector('#app');
ReactDOM.render(<Hello/>, app);
```

Créer un sous component

Hello.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';

export default class Hello extends React.Component {

    render() {
        return (
            <div>Hello sir !!</div>
        );
    }
}
```

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import Hello from 'Hello';

const app = document.querySelector('#app');
ReactDOM.render(<Hello/>, app);
```

index.tsx

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello sir !</div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Paramètre(s) d’un component

Hello.tsx

```
import * as React from 'react';
import { Person } from './index';


interface Props {
    person: Person;
}

export default class Hello extends React.Component<Props> {

    render() {
        const { person } = this.props;

        return (
            <div key={person.firstName + person.lastName}>
               Hello { person.firstName } { person.lastName }!!
            </div>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from ‘react-dom';
import * as tsTypes from './index.d';
import Hello from 'Hello';

const app = document.querySelector('#app');
const bryanJonhson: Person = { firstName: 'Bryan', lastName: 'Jonhson', age: 70 };

ReactDOM.render(<Hello person={bryanJonhson}/>, app);
```

Hello.tsx

```
import * as React from 'react';
import { Person } from './index';


interface Props {
    person: Person;
}

export default class Hello extends React.Component<Props> {

    render() {
        const { person } = this.props;

        return (
            <div key={person.firstName + person.lastName}>
                Hello { person.firstName } { person.lastName }!!
            </div>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from ‘react-dom';
import * as tsTypes from './index.d';
import Hello from 'Hello';

const app = document.querySelector('#app');
const bryanJonhson: Person = { firstName: 'Bryan', lastName: 'Jonhson', age: 70 };

ReactDOM.render(<Hello person={bryanJonhson}/>, app);

export = tsTypes;
```

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello Bryan Johnson!!</div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Affichage d’une liste

HelloPeople.tsx

```
import * as React from 'react';
import { Person } from './index';

interface Props {
    people: Person[];
}

export default class HelloPeople extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => <div key={person.firstName + person.lastName}>
                        Hello { person.firstName } { person.lastName }!!
                    </div>)
                }
            </>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeople from './HelloPeople';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70 },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 },
];

ReactDOM.render(<HelloPeople people={people}/>, app);

export = tsTypes;
```

HelloPeople.tsx

```tsx
import * as React from 'react';
import { Person } from './index';

interface Props {
    people: Person[];
}

export default class HelloPeople extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => <div key={person.firstName + person.lastName}>
                        Hello { person.firstName } { person.lastName }!!
                    </div>)
                }
            </>
        );
    }
}
```

index.tsx

```tsx
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeople from './HelloPeople';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70 },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 },
];

ReactDOM.render(<HelloPeople people={people}/>, app);

export = tsTypes;
```

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello Bryan Johnson!!</div>
    <div>Hello Michael Jackson!!</div>
    <div>Hello Noel Gallagher!!</div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Interfaces

```
interface Person {
    firstName: string;
    lastName: string;
    age: number;
    job?: Job;
}


interface Job {
    name: string;
    location?: string;
}
```

Affichage conditionnel

HelloPeopleJob.tsx

```
import * as React from 'react';
import {Action} from "redux";
import {Person} from "./index";

interface Props {
    people: Person[];
}

export default class HelloPeopleJob extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => {
                        if (person.job && person.job.name !== '')
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName } the { person.job.name }!!
                                </div>
                            );
                        else
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                </div>
                            );
                    })
                }
            </>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeopleJob from './HelloPeopleJob';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];

ReactDOM.render(<HelloPeopleJob people={people}/>, app);
```

```
import * as React from 'react';
import {Action} from "redux";
import {Person} from "./index";

interface Props {
    people: Person[];
}

export default class HelloPeopleJob extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => {
                        if (person.job && person.job.name !== '')
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName } the { person.job.name }!!
                                </div>
                            );
                        else
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                </div>
                            );
                    })
                }
            </>
        );
    }
}
```

HelloPeopleJob.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeopleJob from './HelloPeopleJob';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];

ReactDOM.render(<HelloPeopleJob people={people}/>, app);

export = tsTypes;
```

index.tsx

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello Bryan Johnson the singer!!</div>
    <div>Hello Michael Jackson!!</div>
    <div>Hello Noel Gallagher the guitarist!!</div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Intéractions

onClick, onHover, onChange, onMouseLeave…

HelloPeopleJob_2.tsx

```
import * as React from 'react';
import {Person} from "./index";

interface Props {
    people: Person[];
}

export default class HelloPeopleJob_2 extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => {
                        if (person.job && person.job.name !== '')
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                    <button onClick={() => alert(`My job is ${person.job.name}`)}>
                                        JOB
                                    </button>
                                </div>
                            );
                        else
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                </div>
                            );
                    })
                }
            </>
        );
    }
}
```

index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeopleJob_2 from './HelloPeopleJob_2';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];
```

```
import * as React from 'react';
import {Person} from "./index";

interface Props {
    people: Person[];
}

export default class HelloPeopleJob_2 extends React.Component<Props> {

    render() {
        const { people } = this.props;

        return (
            <>
                {
                    people.map(person => {
                        if (person.job && person.job.name !== '')
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                    <button onClick={() => alert(`My job is ${person.job.name}`)}>
                                        JOB
                                    </button>
                                </div>
                            );
                        else
                            return (
                                <div key={person.firstName + person.lastName}>
                                    Hello { person.firstName } { person.lastName }!!
                                </div>
                            );
                    })
                }
            </>
        );
    }
}
```

HelloPeopleJob_2.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import * as tsTypes from './index.d';
import HelloPeopleJob_2 from './HelloPeopleJob_2';

const app = document.querySelector('#app');

const people: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];

ReactDOM.render(<HelloPeopleJob_2 people={people}/>, app);

export = tsTypes;
```

index.tsx

```html
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>
        Hello Bryan Johnson!!
        <button>JOB</button>
    </div>
    <div>Hello Michael Jackson!!</div>
    <div>
        Hello Noel Gallagher!!
        <button>JOB</button>
    </div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

Router

index.tsx

```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import HelloPeopleJob_2 from "./HelloPeopleJob_2";
import { Switch, Route } from 'react-router-dom';
import * as tsTypes from './index.d';
import history from './history';
import {Router} from "react-router";

const app = document.querySelector('#app');

const people: tsTypes.Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];

const renderMergedProps = (component, ...props) => {
    const finalProps = {...props};
    return (
        React.createElement(component, finalProps)
    );
};

const PropsRoute = ({ component, ...props }) => {
    return (
        <Route {...props} render={routeProps => {
            return renderMergedProps(component, routeProps, rest);
        }}/>
    );
};

ReactDOM.render(
    <Router history={history}>
        <React.Fragment>
            <Switch>
                <PropsRoute exact path='/' component={HelloPeopleJob_2} people={people}/>
            </Switch>
        </React.Fragment>
    </Router>, app,
);

export = tsTypes;
```

history.ts

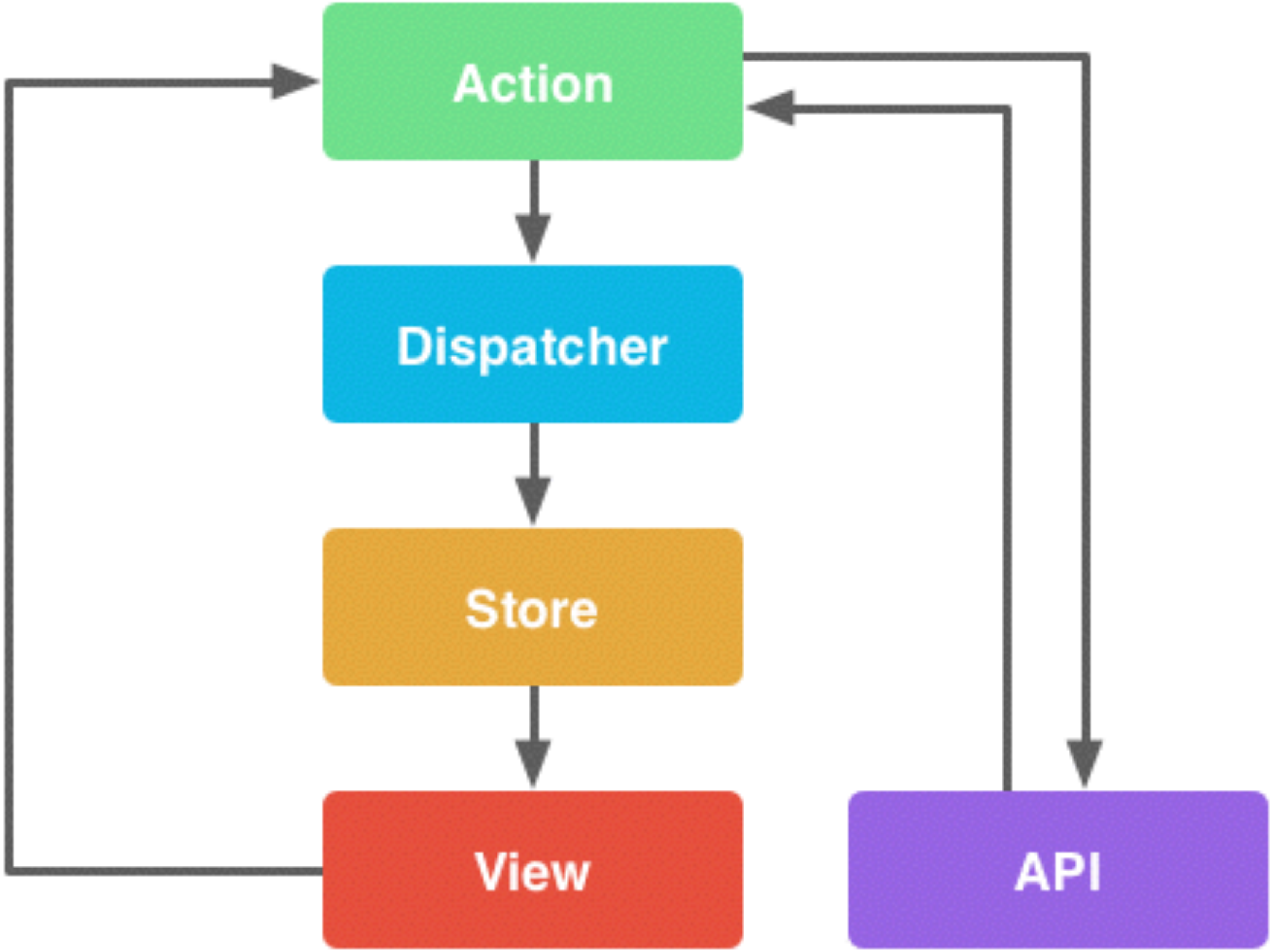
```
import createHistory from 'history/createBrowserHistory';

export default createHistory();
```

Gestion des données

Difficulté à le faire avec seulement React pure

Imbrication de données trop importante

Flux

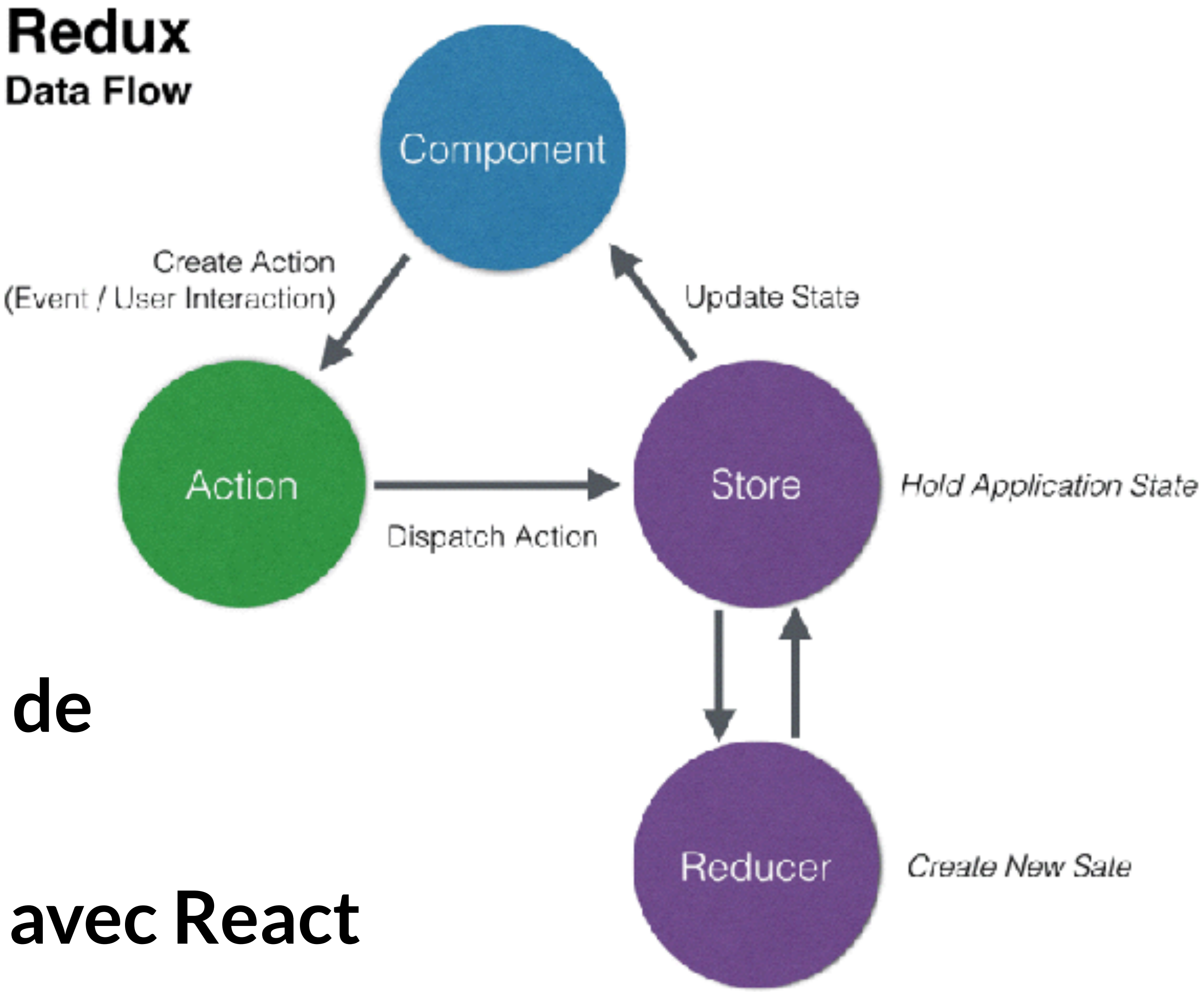


Source: https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture

Flux

Source: https://scotch.io/tutorials/getting-to-know-flux-the-react-js-architecture

Redux



Une implémentation de l'architecture Flux

Fonctionne très bien avec React

Source: https://codingthesmartway.com/learn-redux-introduction-to-state-management-with-react/

Redux

Trois principes fondamentaux:

- Single source of truth: le store
- State is read-only: émettre une action
- Changes are made with pure functions: écrire des reducers pures

Redux

Trois p

- Sing
- Stat
- Char ers pures

Interfaces

```
export interface RootState {
    population: PopulationState;
}


export interface PopulationState {
    people: Person[];
    newPeople: Person;
}


export interface PopulationActions {
    addPerson: (person: Person) => void;
    initPeople: (people: Person[]) => void;
    updateNewPeopleField: (field: string, value: any) => void;
}
```

Action

```
const acti = {
    type:
    name:
};
```

Action

```
const action: Action = {
    type: 'NEW_NAME',
    name: 'Johny',
};
```

```
const action: Action = {
    type: ‘NEW_AGE',
    age: 43,
};
```

```
const action: Action = {
    type: 'ADD_PERSON',
    person: {
        firstName: 'Johny',
        lastName: 'Hallyday',
        job: {
            name: 'singer'
        }
    },
};
```

Action creator

```
import { createAction } from "redux-actions";
import { ActionCreator } from "redux";
import { Person } from "../index";

const addPerson: ActionCreator<void> = (person: Person) => dispatch => {
    const action = person ? createAction<Person>('ADD_PERSON')(person) : createAction('ADD_PERSON')();
    return dispatch(action);
};

const initPeople: ActionCreator<void> = (people: Person[]) => dispatch => {
    const action = people ? createAction<Person[]>('INIT_PEOPLE')(people) : createAction('INIT_PEOPLE')();
    return dispatch(action);
};

const updateNewPeopleField: ActionCreator<void> = (field: string, value: any) => dispatch => {
    const action = createAction<any>('UPDATE_NEW_PEOPLE_FIELD')({field, value});
    return dispatch(action);
};


export default {
    addPerson,
    initPeople,
    updateNewPeopleField,
}
```

populationActionCreators.tsx

```
import {Person} from "./index";

export const initialPeople: Person[] = [
    { firstName: 'Bryan', lastName: 'Jonhson', age: 70, job: { name: 'singer' } },
    { firstName: 'Michael', lastName: 'Jackson', age: 51 },
    { firstName: 'Noel', lastName: 'Gallagher', age: 50 , job: { name: 'guitarist' }},
];
```

constants.ts

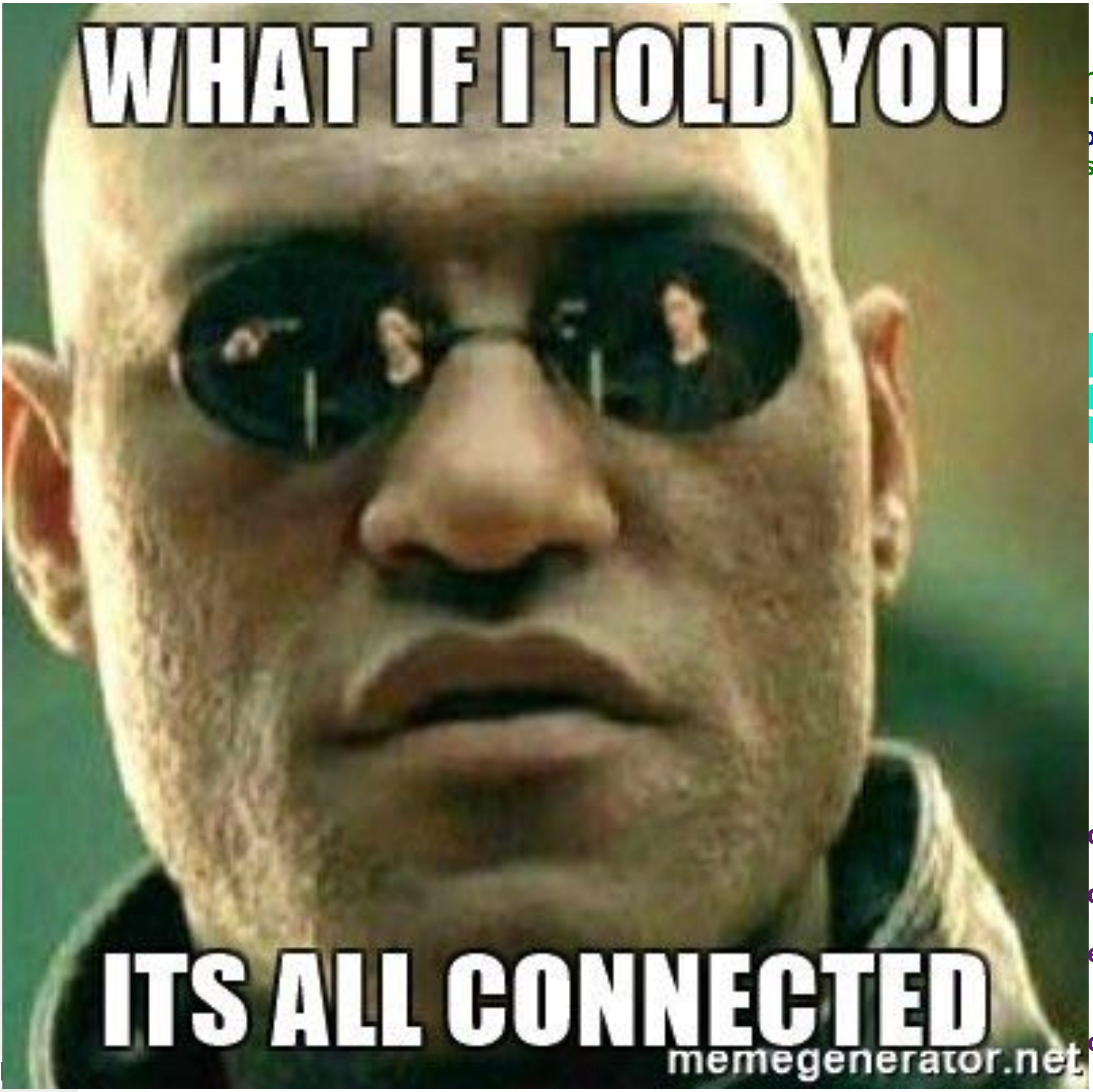
Reducer

populationReducer.tsx

```
import { PopulationState } from "../index";

const defaultState: PopulationState = {
    people: [],
    newPeople: {
        firstName: '',
        lastName: '',
        age: 0,
    },
};

export default function population(state = defaultState, action) {
    switch (action.type) {

        case 'INIT_PEOPLE':
            return {
                ...state,
                people: […action.payload],
            };

        case 'ADD_PERSON':
            if (state.people.length > 0) {
                return {
                    ...state,
                    people: [...state.people, action.payload],
                };
            } else {
                return {
                    ...state,
                    people: [action.payload],
                };
            }

        case 'UPDATE_NEW_PEOPLE_FIELD':
            return {
                ...state,
                newPeople: {
                    ...state.newPeople,
                    [action.payload.field]: action.payload.value
                }
            };

        default:
            return state;
    }
}
```

Component connecté

```
interface Props {
    people: Person[];
    newPeople: Person;
    populationActions: P
}

class HelloPeopleJobRedu

    componentDidMount()
        this.props.popul
    }

    render() {
        const { newPeopl

        return (
            <>
                {
                    peop




                    })
                }
                <div>
                    <inp

                    <inp

                    <inp

                    <but



                    </button>
                </div>
            </>
        );
    }
}
```



```
 RootState} from "./index";
redux";
';
om './actions/populationActionsCreator';
stants";
```

oPeopleRedux.tsx

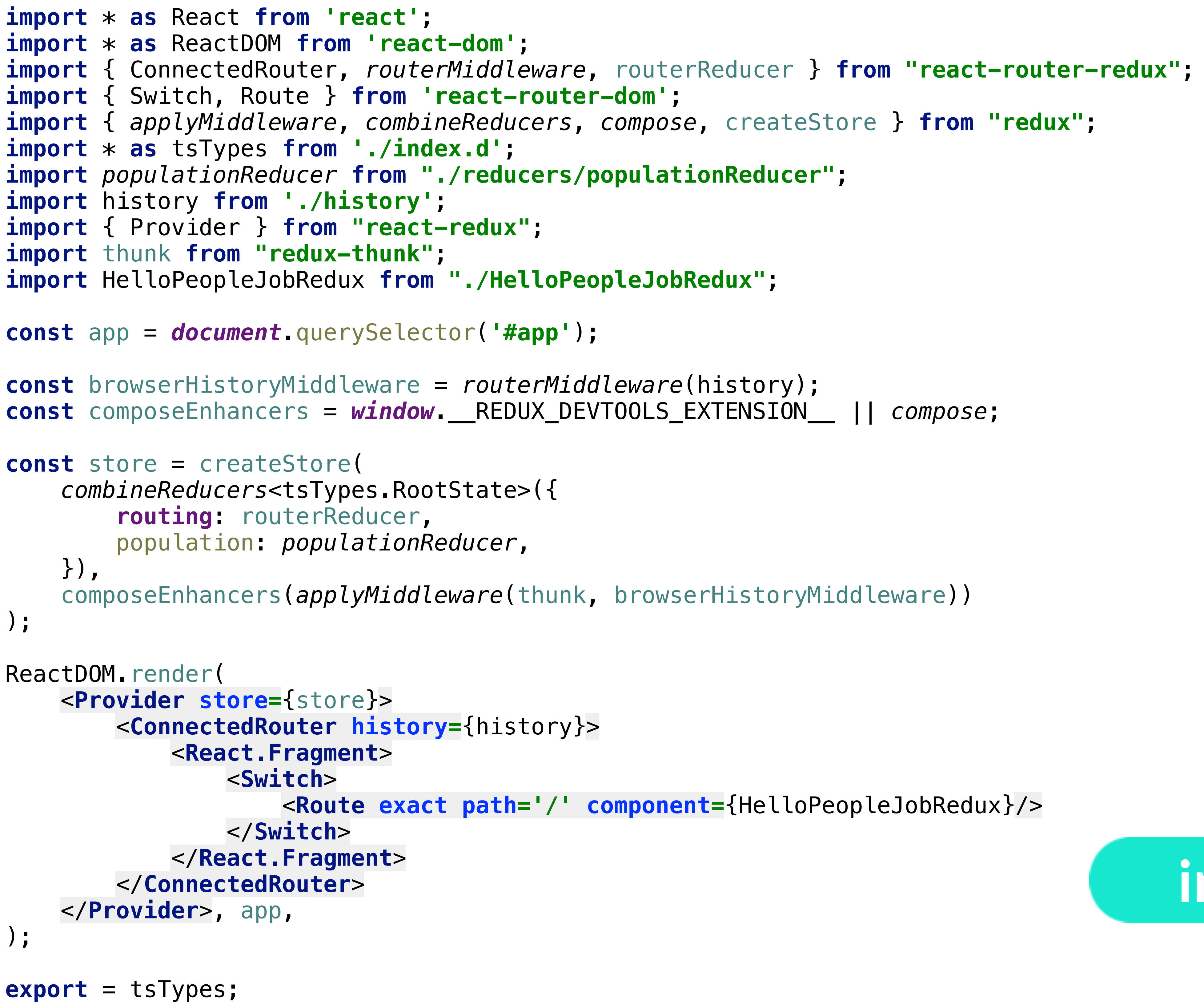
```
d('firstName', event.target.value)}

d('lastName', event.target.value)}

eld('age', event.target.value)}


ople.lastName === ''}>
```

HelloPeopleRedux.tsx

```
import * as React from 'react';
import {Person, PopulationActions, RootState} from "./index";
import {bindActionCreators} from "redux";
import {connect} from "react-redux";
import populationActionsCreator from './actions/populationActionsCreator';
import {initialPeople} from "./constants";
```

```
interface Props {
    people: Person[];
    newPeople: Person;
    populationActions: PopulationActions;
}

class HelloPeopleJobRedux extends React.Component<Props> {

    componentDidMount() {
        this.props.populationActions.initPeople(initialPeople);
    }

    render() {
        const { newPeople, people, populationActions } = this.props;

        return (
            <>
                {
                    people.map(person => {
                        return (
                            <div key={person.firstName + person.lastName}>
                                Hello { person.firstName } { person.lastName }!!
                            </div>
                        );
                    })
                }
                <div>
                    <input type="text" onChange={event => populationActions.updateNewPeopleField('firstName', event.target.value)}
                           value={newPeople.firstName}/>
                    <input type="text" onChange={event => populationActions.updateNewPeopleField('lastName', event.target.value)}
                           value={newPeople.lastName}/>
                    <input type="number" onChange={event => populationActions.updateNewPeopleField('age', event.target.value)}
                           value={newPeople.age}/>
                    <button onClick={() => populationActions.addPerson(newPeople)}
                            disabled={newPeople.age <= 0 && newPeople.firstName === '' && newPeople.lastName === ''}>
                        New person
                    </button>
                </div>
            </>
        );
    }
}

const mapStateToProps = (state: RootState) => ({
    people: state.population.people,
    newPeople: state.population.newPeople,
});

const mapDispatchToProps = (dispatch) => ({
    populationActions: bindActionCreators(populationActionsCreator, dispatch),
});

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

Initialisation avec Redux



```
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ConnectedRouter, routerMiddleware, routerReducer } from "react-router-redux";
import { Switch, Route } from 'react-router-dom';
import { applyMiddleware, combineReducers, compose, createStore } from "redux";
import * as tsTypes from './index.d';
import populationReducer from "./reducers/populationReducer";
import history from './history';
import { Provider } from "react-redux";
import thunk from "redux-thunk";
import HelloPeopleJobRedux from "./HelloPeopleJobRedux";

const app = document.querySelector('#app');

const browserHistoryMiddleware = routerMiddleware(history);
const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION__ || compose;

const store = createStore(
    combineReducers<tsTypes.RootState>({
        routing: routerReducer,
        population: populationReducer,
    }),
    composeEnhancers(applyMiddleware(thunk, browserHistoryMiddleware))
);

ReactDOM.render(
    <Provider store={store}>
        <ConnectedRouter history={history}>
            <React.Fragment>
                <Switch>
                    <Route exact path='/' component={HelloPeopleJobRedux}/>
                </Switch>
            </React.Fragment>
        </ConnectedRouter>
    </Provider>, app,
);

export = tsTypes;
```

index.tsx

```tsx
import * as React from 'react';
import * as ReactDOM from 'react-dom';
import { ConnectedRouter, routerMiddleware, routerReducer } from "react-router-redux";
import { Switch, Route } from 'react-router-dom';
import { applyMiddleware, combineReducers, compose, createStore } from "redux";
import * as tsTypes from './index.d';
import populationReducer from "./reducers/populationReducer";
import history from './history';
import { Provider } from "react-redux";
import thunk from "redux-thunk";
import HelloPeopleJobRedux from "./HelloPeopleJobRedux";

const app = document.querySelector('#app');

const browserHistoryMiddleware = routerMiddleware(history);
const composeEnhancers = window.__REDUX_DEVTOOLS_EXTENSION__ || compose;

const store = createStore(
    combineReducers<tsTypes.RootState>({
        routing: routerReducer,
        population: populationReducer,
    }),
    composeEnhancers(applyMiddleware(thunk, browserHistoryMiddleware))
);

ReactDOM.render(
    <Provider store={store}>
        <ConnectedRouter history={history}>
            <React.Fragment>
                <Switch>
                    <Route exact path='/' component={HelloPeopleJobRedux}/>
                </Switch>
            </React.Fragment>
        </ConnectedRouter>
    </Provider>, app,
);

export = tsTypes;
```

index.tsx

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <title>Example</title>
</head>
<body>
<div id="app">
    <div>Hello Bryan Johnson!!</div>
    <div>Hello Michael Jackson!!</div>
    <div>Hello Noel Gallagher!!</div>
    <div>
        <input type="text" value=""/>
        <input type="text" value=""/>
        <input type="number" value=""/>
        <button disabled="true">New person</button>
    </div>
</div>
<script src="dist/bundle.js"></script>
</body>
</html>
```

WAKE UP...
IT'S DEMO TIME!
memegenerator.net

Pourquoi React + Redux ?

- **Components réutilisables**
- **Performance**
- **Léger**
- **Actions prévisibles / suivies**

Project source: https://github.com/squelix/react-redux-talk