React

Table des matières

Site, Github

1. EDI

- VS Code
- WebStorm

2. es5

a. Template simple avec CDNs

```
<html>

<head>
    <title>Welcome</title>
    <script src="https://unpkg.com/react@15/dist/react.min.js"></script>
    <script src="https://unpkg.com/react-dom@15/dist/react-
dom.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/babel-
core/5.8.34/browser.js"></script>
</head>

<body>
    <div id="app"></div>
    <script type="text/babel">

    </script>
</body>
</html>
```

b. Création de component

```
var MyHeader = React.createClass({
   render:function () {
       return (
           <header>Header</header>
       )
   }
});
```

Création du component principal et ajout de components

```
var App = React.createClass({
    render: function () {
        return (
                <div>
                    <MyHeader />
                    <MyContent />
                    <MyFooter />
                </div>
        )
    }
});

ReactDOM.render(<App />, document.getElementById('app'));
```

c. Props

PropTypes

Les props servent à échanger des informations entre component

Validation du type de données (exception levée si invalide)

Si on ne définit pas de « pageTitle » sur le component, la valeur par défaut est appliquée

Ne pas utiliser les « **arrow functions** » pour ne pas perdre le contexte

```
var MyHeader = React.createClass({
    propTypes: {
        pageTitle: React.PropTypes.string
    },
    getDefaultProps() {
        return {
            pageTitle: 'My default title'
        };
    },
    render:function () {
        return (
                <header>{this.props.pageTitle}</header>
        )
    }
});
```

```
var App = React.createClass({
    render: function () {
        return (
                <div>
                    <MyHeader pageTitle="My title"/>
                    <MyContent />
                    <MyFooter />
                </div>
        )
    }
});

ReactDOM.render(<App />, document.getElementById('app'));
```

La function « **getDefautlProps** » n'est disponible qu'avec « **React.createClass** » (pas avec un export de module es6/ typescript)

d. Event

```
var MyHeader = React.createClass({
    render:function () {
        function onHomeClick() {
            console.log('click!')
        }
        return (
                <header>
                    <h1>My header</h1>
                    <nav>
                        <a href="#" onClick={onHomeClick}>Home</a>
                    </nav>
                </header>
        )
    }
});
```

Avec props

```
var MyHeader = React.createClass({
    render:function () {
        return (
                <header>
                    <h1>My header</h1>
                    <nav>
                        <a href="#" onClick={this.props.onHomeClick}>Link</a>
                    </nav>
                </header>
        )
    }
});

// App component
var App = React.createClass({
    handleOnHomeClick: function(){
        console.log('click!!')
    },
    render: function () {
        return (
                <div>
                    <MyHeader onHomeClick={this.handleOnHomeClick}/>
                </div>
        )
    }
});
ReactDOM.render(<App />, document.getElementById('app'));
```

e. State

Permet de changer des variables durant l'exécution

```
var MyComponent = React.createClass({
    getInitialState:function () {
        return {
            age:20
        }
    },
    onIncrementAge: function(){
        this.setState({
            age: this.state.age + 1
        });
    },
    render: function(){
        return (
            <div>
                <p>The current age is : {this.state.age}</p>
                <button onClick={this.onIncrementAge.bind(this)}>Increment</button>
            </div>
        )
    }
});
```

Initial state avec la function « getInitialState »

Changement avec la function « setState »

3. es6

a. Avec « create-react-app »

Github

```
npm install -g create-react-app
```

```
create-react-app <project-name>
```

b. Avec un starter kit

Starter kit es6, avec TypeScript

c. Création de component

On range les components dans un dossier « components »

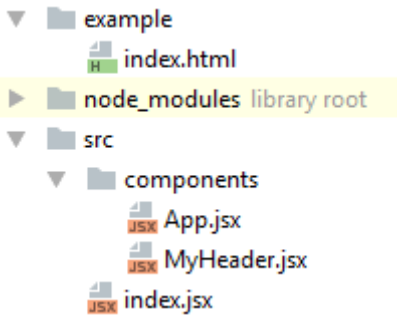
```
import React from 'react';
import { render } from 'react-dom';

export class MyHeader extends React.Component {
    render() {
        return (
            <header>My header</header>
        );
    }
}
```

Création du component principal et ajout de components

```
import React from 'react';
import { render } from 'react-dom';

import { MyHeader } from './MyHeader';

export class App extends React.Component {
    constructor(props){
        super(props);
    }
    render() {
        return (
            <div>
                <MyHeader />
            </div>

        );
    }
}
```

« index.jsx »

```
import React from 'react';
import { render } from 'react-dom';

import { App } from './components/App';

render(<App />, document.getElementById('app'));
```



« index.html »

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8">
    <title></title>
</head>

<body>
    <div id="app"></div>
    <script src="/dist/build.js"></script>
</body>

</html>
```

Configuration avec Webpack et Webpack dev server

Stateless component

```
import React from 'react';

export const User = (props) => {
    return (
        <div className="container">
            <p>User: {props.username}</p>
        </div>
    );
}
```

d. Props

```
import React from 'react';
import { render } from 'react-dom';

class MyHeader extends React.Component {
    render() {
        return (
            <header>{this.props.pageTitle}</header>
        );
    }
}
MyHeader.propTypes = {
    pageTitle: React.PropTypes.string
};
MyHeader.defaultProps = {
    pageTitle :'My default title'
};
export default MyHeader ;
```

Export avec « default » de manière à pouvoir renommer à l'import

e. Event

```
export class MyHeader extends React.Component {
    onHomeClick() {
        console.log('click!')
    }
    render() {
        return (
        <header>
            <h1>My header</h1>
            <nav>
                <a href="#" onClick={this.onHomeClick}>Home</a>
            </nav>
        </header>
        );
    }
}
```

Contexte

Pour les problèmes de contexte (si par exemple dans la fonction on a besoin d'accéder à « this »)

- Soit avec arrow function

```
<a href="#" onClick={() => this.onHomeClick()}>Home</a>
```

- Soit avec bind(this)

```
<a href="#" onClick={this.onHomeClick.bind(this)}>Home</a>
```

- Pour de **meilleures performances** on peut également définir dans le constructeur

```
 constructor(props) {
        super(props);

        this.onSearchChange = this.onSearchChange.bind(this);
        this.onSubmit = this.onSubmit.bind(this);
    }
```

… et dans le jsx

```
 render() {
        return (
            <div>
                <h1>Search For A Movie</h1>
                <input type="text" value={this.state.search} onChange={this.onSearchChange}
/>
                <input type="submit" value="Submit" onClick={this.onSubmit} />
            </div>
        );
    }
```

f. State

Permet de changer des variables durant l'exécution

```
import React from 'react';
import { render } from 'react-dom';

export class MyComponent extends React.Component {
    constructor(props){
        super();
        this.state = {
            age : 20
        };
    }
    onIncrementAge() {
        this.setState({
           age: this.state.age + 1
        });
    }
    render() {
        return (
            <div>
                <p>The current age is : {this.state.age}</p>
                <button onClick={() => this.onIncrementAge()}>Increment</button>
            </div>
        );
    }
}
```

Initial state dans constructeur

Changement avec la function setState

g. Context

Documentation

Parent

```
export class MyParent extends React.Component {
    getChildContext() {
        return { myParent: this };
    }
}
MyParent.childContextTypes = {
    myParent: React.PropTypes.instanceOf(MyParent)
};
```

Accéder au parent depuis le **Child**

```
export class MyChild extends React.Component {
    constructor(props, context) {
        super(props, context);

        // this.context.myParent
    }
}
MyChild.contextTypes = {
    myParent: React.PropTypes.instanceOf(MyParent)
};
```

Si on veut accéder au **child depuis le parent** on peut « **enregistrer** » celui-ci.

Parent

```
export class MyParent extends React.Component {
    getChildContext() {
        return { myParent: this };
    }
    register(child) {
        this.child = child;
    }
}
MyParent.childContextTypes = {
    myParent: React.PropTypes.instanceOf(MyParent)
};
```

Child

```
export class MyChild extends React.Component {
    constructor(props, context) {
        super(props, context);
        if (context.myParent) {context.myParent.register(this); }
    }
}
MyChild.contextTypes = {
    myParent: React.PropTypes.instanceOf(MyParent)
};
```

Un article parlant de comment communiquer entre components.

h. Refs

Documentation

Permet d'accéder à l'élément Html après le render. Exemple (avec CodeMirror)

```
import React from 'react';
import PropTypes from 'prop-types';
import CodeMirror from 'codemirror';
import  'codemirror/lib/codemirror.css';
import 'codemirror/mode/jsx/jsx';
export class CodeExample extends React.Component {
    componentDidMount() {
        CodeMirror(this.refs.refcode, { value: this.props.children, mode:
"jsx",  readOnly: true });
    }
    render() {
        return <pre><code ref="refcode" /></pre>;
    }
}
CodeExample.propTypes = {
    children: PropTypes.string.isRequired
};
```

4. class => className

Exemple

```
var App = React.createClass({
    render: function () {
        return (
                <div className="text-red">My App</div>
        )
    }
});
ReactDOM.render(<App />, document.getElementById('app'));
```

CSS

```
.text-red{
    color: red;
}
```

5. Style (avec {{}})

```
var App = React.createClass({
    propTypes: {
        color: React.PropTypes.string
    },
    render: function () {
        return (
            <div style={{color:this.props.color}}>My App</div>
        )
    }
});
ReactDOM.render(<App color="red" />, document.getElementById('app'));
```

6. Component Lifecycle

- **componentWillMount**: avant le render initial
- **componentDidMount** : après le render initial
- **componentWillReceiveProps** : quand le component reçoit de nouveaux props
- **shouldComponentUpdate** : avant le render après changement de state, etc. (on peut retourner false pour annuler le rendering)
- **componentWillUpdate**
- **componentDidUpdate**
- **componentWillUnmount** : avant de supprimer le component du DOM

7. React Router

Github

a. v3.0.x

```
npm i react-router -S
```

Exemple

```
export class Home extends React.Component {
    render() {
        return (
                <div><h1>Home</h1></div>
        )
    }
}
```

Création du **component principal** avec **liens** de navigation et affichage du contenu

```
import { render } from 'react-dom'
import { Router, Route, IndexRoute, Link, IndexLink, browserHistory } from 'react-router'
const ACTIVE = { color: 'red' }

export class App extends React.Component {
    render() {
        return (
                <div>
                    <nav>
                        <ul>
                            <li><Link to="/"
                                      activeStyle={ACTIVE}
                                      onlyActiveOnIndex>Home</Link></li>
                            <li><Link to="/about"
                                      activeStyle={ACTIVE}>About</Link></li>
                            <li><Link to="/products" activeStyle={ACTIVE}
                                      onlyActiveOnIndex>Product list</Link></li>
                            <li><Link to="/products/10" activeStyle={ACTIVE}
                                      onlyActiveOnIndex>Product details</Link></li>
                        </ul>
                    </nav>
                    <div>
                        {this.props.children}
                    </div>
                </div>
        )
    }
}
```

Affichage du contenu

Définition des **routes**

```
render((
        <Router history={browserHistory}>
            <Route path="/" component={App}>
                <IndexRoute component={Home} />
                <Route path="/about" component={About} />
                <Route path="products" component={ProductStart}>
                    <IndexRoute component={ProductList} />
                    <Route path=":id" component={ProductDetail} />
                </Route>
                <Route path="*" component={PageNotFound} />
            </Route>
        </Router>
), document.getElementById('root'));
```

Child routes

Passage de paramètre

```
export class ProductDetail extends React.Component {
    render() {
        return (
            <p>Product detail  {this.props.params.id}</p>
        );
    }
}
```

b. v4

Documentation

Navigation

```
import React from 'react';
import { NavLink } from 'react-router-dom';

const Header = () => {
    return (
        <nav>
            <NavLink to="/" activeClassName="active">Home</NavLink>
            <NavLink to="/about" activeClassName="active">About</NavLink>
        </nav>
    );
};
export default Header;
```

App

```
import React from 'react';
import { BrowserRouter, Switch, Route } from 'react-router-dom';
import { connect } from 'react-redux';

// components
import Header from './common/Header';
import HomePage from './home/HomePage';
import AboutPage from './about/AboutPage';

class App extends React.Component {
    render() {
        return (
            <BrowserRouter>
                <div>
                    <Header />
                    <div className="container">
                        <Switch>
                            <Route exact path="/" component={HomePage} />
                            <Route path="/about" component={AboutPage} />
                        </Switch>
                    </div>
                </div>
```

```
            </BrowserRouter>
        );
    }
}
export default App;
```

Index avec Redux

```
import React from 'react';
import { render } from 'react-dom';
import { Provider } from 'react-redux';
import store from './store/store';

import './styles/styles.css';

import App from './components/App';

render(
    <Provider store={store}>
        <App />
    </Provider>,
    document.getElementById('app')
);
```

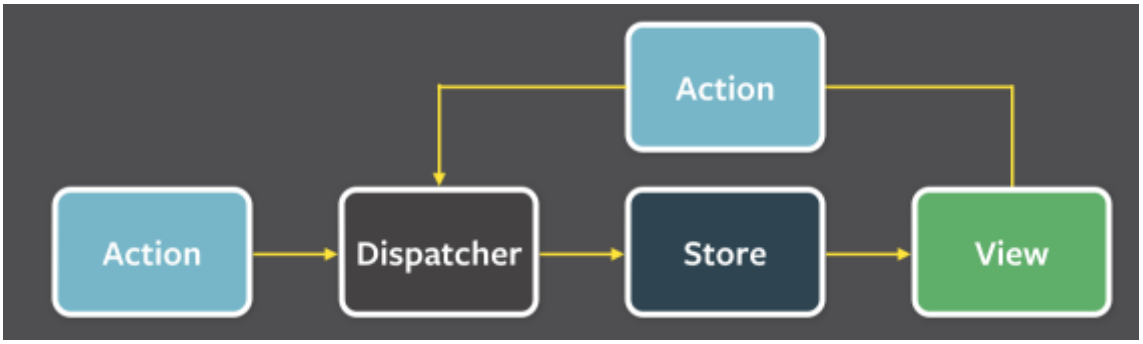
8. Flux

Documentation, Github

```
npm i flux -D
```

C'est un pattern. Plusieurs implémentations :

- Facebook Flux
- Alt
- Reflux
- Flummox
- Marty
- Fluxxor
- Redux
- Etc.



1 dispatcher qui dispatch les actions

1 ou plusieurs stores, contient les models, utilise EventEmitter

Dispatcher

```
import { Dispatcher } from 'flux';

export const  AppDispatcher  = new Dispatcher();
```

Constants

```
export const ActionTypes = {
    SEARCH_MOVIES: 'SEARCH_MOVIES',
    RECEIVE_MOVIE_RESULTS: 'RECEIVE_MOVIE_RESULTS'
};
```

Actions

```
import { ActionTypes } from '../constants/ActionTypes';
import { AppDispatcher } from '../dispatcher/AppDispatcher';
import { movieAPI } from '../api/MovieAPI';

class MovieActions {
    searchMovies(movie) {
        movieAPI.searchMovies(movie, (movies) => {
            AppDispatcher.dispatch({
                actionType: ActionTypes.SEARCH_MOVIES,
                movies
            });
        });
    }
    receiveMovieResults(movies) {
        AppDispatcher.dispatch({
            actionType: ActionTypes.RECEIVE_MOVIE_RESULTS,
            movies
        });
    }
}

export const movieActions = new MovieActions();
```

Api

```
import { movieActions } from '../actions/MovieActions';

export function sendRequest(url, onSuccess, onError) {
    const xhr = new XMLHttpRequest();
    xhr.open('GET', url, true);
    xhr.onreadystatechange = () => {
        if (xhr.readyState === 4) {
            onSuccess(xhr.responseText);
        }
    };
    xhr.send(null);
}

export class MovieAPI {
    searchMovies(title, onSuccess, onError) {
        sendRequest(`http://www.omdbapi.com/?s=${title}`, (response) => {
            let data = JSON.parse(response);
            movieActions.receiveMovieResults(data.Search);
        });
    }
```

```
}

export const movieAPI = new MovieAPI();
```

Store

```
import { AppDispatcher } from '../dispatcher/AppDispatcher';
import { ActionTypes } from '../constants/ActionTypes';
import EventEmitter from 'events';

let _movies = [];
const CHANGE_EVENT = 'change';

class MovieStore extends EventEmitter {
    constructor() {
        super();
    }

    setMovieResults(movies) {
        _movies = movies;
    }

    getMovieResults() {
        return _movies;
    }

    emitChange() {
        this.emit(CHANGE_EVENT);
    }

    addChangeListener(callback) {
        this.on(CHANGE_EVENT, callback);
    }

    removeChangeListener(callback) {
        this.removeListener(CHANGE_EVENT, callback);
    }
}

export const movieStore = new MovieStore();


AppDispatcher.register(function (action) {
    switch (action.actionType) {
        case ActionTypes.SEARCH_MOVIES:
            _movies = action.movies;
            movieStore.emitChange();
            break;
        case ActionTypes.RECEIVE_MOVIE_RESULTS:
            movieStore.setMovieResults(action.movies);
            movieStore.emit(CHANGE_EVENT);
            break;
    }
});
```

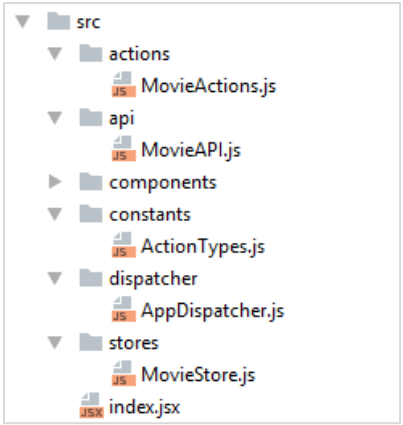
Abonnement au dispatcher

App

On s'abonne aux stores afin de re render à chaque changement d'état

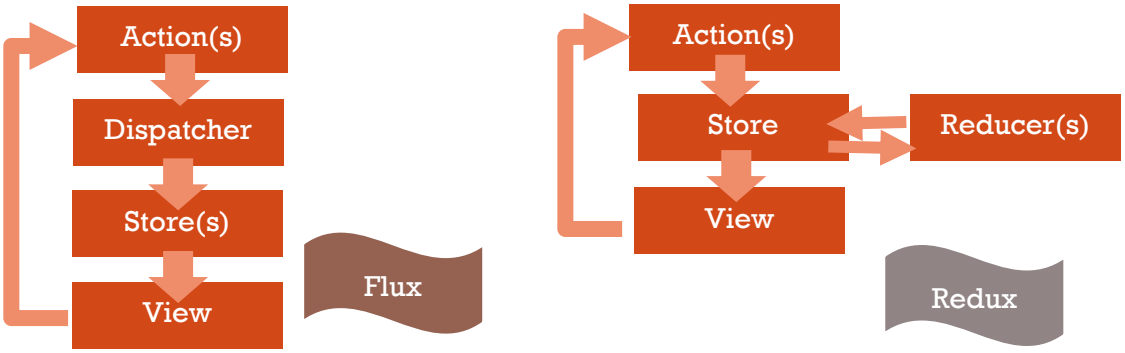
```
import React from 'react';
import { render } from 'react-dom';
```

```
import { MovieResults } from './MovieResults';
import { SearchMovies } from './SearchForm';
import { movieStore } from '../stores/MovieStore';

export function getAppState() {
    return {
        movies: movieStore.getMovieResults()
    };
}

export class App extends React.Component {
    constructor(props) {
        super(props);

        // initial state
        this.state = getAppState();
    }

    _onChange() {
        this.setState(getAppState());
        console.log('change', this.state);
    }

    componentDidMount() {
        movieStore.addChangeListener(() => this._onChange());
    }

    componentWillUnmount() {
        movieStore.removeChangeListener(() => this._onChange());
    }

    render() {
        console.log('render', this.state.movies);
        let movieResults = '';
        if (this.state.movies.length > 0) {
            movieResults = <MovieResults movies={this.state.movies} />
        }

        return (
            <div className="container">
                <SearchMovies />
                {movieResults}
            </div>
        );
    }
}
```



9. Redux

```
npm i redux -S
```



Imports

```
import { createStore, combineReducers, applyMiddleware } from 'redux';
```

On peut également utiliser un cdn

```
// https://cdnjs.cloudflare.com/ajax/libs/redux/3.6.0/redux.js"
 var createStore = Redux.createStore,
      combineReducers = Redux.combineReducers,
      applyMiddleware = Redux.applyMiddleware;
```

Reducers

```
const mathReducer = (state = {
    result: 1
}, action) => {
    switch (action.type) {
        case "ADD":
            return Object.assign({}, state, {
                result: state.result + action.payload
            });
            break;
        case "SUBSTRACT":
            return Object.assign({}, state, {
                result: state.result - action.payload
            });
            break;
    }
    return state; // return state
};
```

Initial state

On ne modifie pas le state (immutable) .On crée un nouvel objet (meilleures performances)

Exemple un autre reducer

```
const userReducer = (state = {
    name: 'Marie',
    age: 20
}, action) => {
    switch (action.type) {
        case "SET_NAME":
            return Object.assign({}, state, {
                name: action.payload
            })
            break;
        case "SET_AGE":
            return Object.assign({}, state, {
                age: action.payload
            })
            break;
    }
    return state; // return state
};
```

Store

1 seul store

```
const store = createStore(mathReducer);
store.subscribe(() => {
    console.log('store change', store.getState());
});
```

Si on a plusieurs reducers on les combine

```
const store = createStore(combineReducers({ mathReducer, userReducer }));
```

Simulation d'actions

```
store.dispatch({
    type: "ADD",
    payload: 10
});

store.dispatch({
    type: "SET_AGE",
    payload: 25
});
```

Avec React

Cours sur Egghead

React-redux fait le lien entre React et Redux

```
npm i react-redux -S
```

a. Connect

Appliqué en général sur un « container »

```
import React, { PropTypes } from 'react';
import { connect } from 'react-redux';
// components
import SearchForm from './SearchForm';
// actions
import { searchMovies } from '../../actions/movieActions';

class MoviePage extends React.Component {
    constructor(props) {
        super(props);
        // initial state
        this.state = {
            search: ''
        };

        this.onSearchChange = this.onSearchChange.bind(this);
        this.onSubmit = this.onSubmit.bind(this);
    }

    onSearchChange(event) {
        const search = event.target.value;
        this.setState({ search });
    }

    onSubmit() {
        this.props.dispatch(searchMovies(this.state.search));
    }
    render() {
        return (
            <div>
                <h1>Search For A Movie</h1>
                <input type="text" value={this.state.search} onChange={this.
onSearchChange} />
                <input type="submit" value="Submit" onClick={this.onSubmit}
/>
                {this.props.movies.map((movie, i) => {
                    return <div key={i}>{movie.Title}</div>;
                })}
            </div>
```

Si l'action retourne une promise, on pourra bind « dispatch » au résultat (then, catch)

```
        );
    }
}
const mapStateToProps = (state) => {
    return {
        movies: state.movies,
    };
};
export default connect(mapStateToProps)(MoviePage);
```

Ajoute les states aux props, ce qui permet d'y accéder ou de les passer aux components

Si on ne définit pas **mapDispatchToProps**, la function **dispatch** est **ajoutée** automatiquement aux **props**

Avec « mapDispatchToProps »

```
onSubmit() {
        this.props.searchMovies(this.state.search);
}
```

```
const mapStateToProps = (state) => {
    return {
        movies: state.movies,
    };
};
const mapDispatchToProps = (dispatch) => {
    return {
        searchMovies: (search) => {
            dispatch(searchMovies(search));
        }
    };
};
export default connect(mapStateToProps, mapDispatchToProps)(MoviePage);
```

Dispatch l'action

Avec « bindActionCreators »

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

On importe toutes les actions avec un alias

```
import * as movieActions from '../../actions/movieActions';
```

```
const mapStateToProps = (state) => {
    return {
        movies: state.movies,
    };
};
const mapDispatchToProps = (dispatch) => {
    return {
        actions: bindActionCreators(movieActions, dispatch)
    };
};
export default connect(mapStateToProps, mapDispatchToProps)(MoviePage);
```

b. Action

Action types : on y définit les **constantes** (évite les erreurs de syntaxe dans la saisie des types d'action)

```
export const SEARCH_MOVIES_SUCCESS = 'SEARCH_MOVIES_SUCCESS';
```

```
import * as types from './actionTypes';
import { movieAPI } from '../api/MovieAPI';

export function searchMoviesSuccess(movies) {
    return { type: types.SEARCH_MOVIES_SUCCESS, movies };
}

export function searchMovies(search) {
    /*  const movies = [
          { title: 'Movie 1' },
          { title: 'Movie 2' }
      ];
      return { type: types.SEARCH_MOVIES_SUCCESS, movies };*/

    // Async
    return (dispatch) => {
        return movieAPI.searchMovies(search).then((movies) => {
            dispatch(searchMoviesSuccess(movies));
        });
    };
}
```

Async avec « thunk »

```
npm i redux-thunk –S
```

Ajout aux middlewares du « store »

```
import { createStore, combineReducers, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';

import user from '../reducers/userReducer';
import movie from '../reducers/movieReducer';

export default createStore(
    combineReducers({ user, movie }),
    {},
    applyMiddleware(thunk)
);
```

Utilisation

Exemples d'actions

```
export function setName(name) {
    return (dispatch) => {
```

```
        setTimeout(function () {
            dispatch({
                type: 'SET_NAME',
                payload: name + '!!'
            });
        }, 2000);
    }
}
```

Avec une promise

```
export function setName(name) {
    return (dispatch) => {
        return new Promise((resolve) => {
                dispatch({
                    type: 'SET_NAME',
                    payload: name + '!!'
                });
        });
    }
}
```

Asyc avec « redux-promise-middleware »

```
npm i redux-promise-middleware -S
```

Ajout aux middlewares du « store »

```
import { createStore, combineReducers, applyMiddleware } from 'redux';
import promise from 'redux-promise-middleware';

import user from '../reducers/userReducer';
import movie from '../reducers/movieReducer';

export default createStore(
    combineReducers({ user, movie }),
    {},
    applyMiddleware(promise())
);
```

c. Reducers

```
import * as types from '../actions/actionTypes';

export default function courseReducer(state = [], action) {

  switch (action.type) {
    case types.SEARCH_MOVIES_SUCCESS:
      return action.movies;
    default:
      return state;
  }
}
```

On utilise Object.assign et l'opérateur spread quand besoin afin de recréer l'objet plutôt que de modifier le state (meilleures performances)

d. Store

```
import { createStore, combineReducers, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
// reducers
import movies from '../reducers/movieReducer';

export default createStore(
    combineReducers({ movies }),
    {},
    applyMiddleware(thunk)
);
```

e. Container vs presentation component

Exemple de page mieux structurée

```
import React, { PropTypes } from 'react';
import { connect } from 'react-redux';
import { bindActionCreators } from 'redux';
// components
import SearchForm from './SearchForm';
import MovieList from './MovieList';
// actions
import * as movieActions from '../../actions/movieActions';

class MoviePage extends React.Component {
    render() {
        return (
            <div>
                <SearchForm actions={this.props.actions} />
                <MovieList movies={this.props.movies} />
            </div>
        );
    }
}
MoviePage.propTypes = {
    actions:PropTypes.object.isRequired,
    movies: PropTypes.array.isRequired
};
const mapStateToProps = (state) => {
    return {
        movies: state.movies,
    };
};
const mapDispatchToProps = (dispatch) => {
    return {
        actions: bindActionCreators(movieActions, dispatch)
    };
};
export default connect(mapStateToProps, mapDispatchToProps)(MoviePage);
```

On a des components auxquels on passe les informations

La form

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

import { searchMovies } from '../../actions/movieActions';

class SearchForm extends React.Component {
    constructor(props) {
        super(props);
        // initial state
        this.state = {
            search: ''
        };

        this.onSearchChange = this.onSearchChange.bind(this);
        this.onSubmit = this.onSubmit.bind(this);
    }

    onSearchChange(event) {
        const search = event.target.value;
        this.setState({ search });
    }

    onSubmit(event) {
        event.preventDefault();
        this.props.actions.searchMovies(this.state.search);
    }

    render() {
        return (
            <form onSubmit={this.onSubmit}>
                <div className="input-group">
                    <input type="text" value={this.state.search} onChange={t
his.onSearchChange} className="form-
control" placeholder="Enter a movie title ..." />
                    <span className="input-group-btn">
                        <button value="Submit" className="btn btn-
default" type="button">Search Movies</button>
                    </span>
                </div>
            </form>
        );
    }
}

SearchForm.propTypes = {
    actions:PropTypes.object.isRequired
};
export default SearchForm;
```

MovieList

```
import React, { PropTypes } from 'react';
import Movie from './Movie';

const MovieList = ({movies}) => {
    return (
        <div>
            {
                movies.map((movie, i) => {
                    return (
                        <Movie movie={movie} key={i} />
                    );
                })
            }
        </div>
    );
};
MovieList.propTypes = {
    movies: PropTypes.array.isRequired
};
export default MovieList;
```

Movie

```
import React, { PropTypes } from 'react';
const Movie = ({movie}) => {
    const link = 'http://www.imdb.com/title/' + movie.imdbID;
    return (
        <div className="well">
            <div className="row">
                <div className="col-md-4">
                    <img className="thumbnail" src={movie.Poster} />
                </div>
                <div className="col-md-8">
                    <h4>{movie.Title}</h4>
                    <ul className="list-group">
                        <li className="list-group-item">Year Released: {movie.Year}</li>
                        <li className="list-group-item">IMDB ID: {movie.imdbID}</li>
                    </ul>
                    <a className="btn btn-primary" href={link}>View on IMDB</a>
                </div>
            </div>
        </div>
    );
};
Movie.propTypes = {
    movie: PropTypes.object.isRequired
};
export default Movie;
```

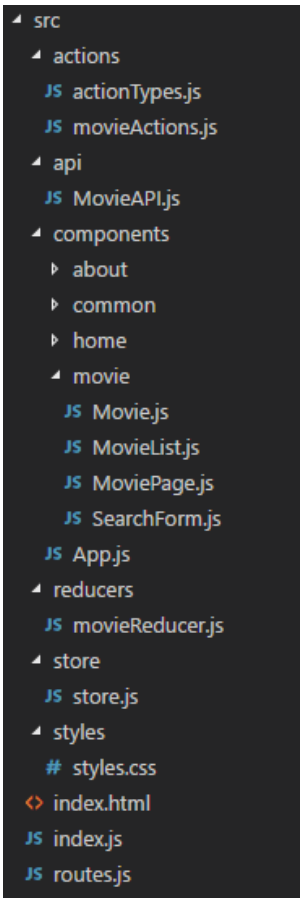
f. Provider

« attache » le component App au store

```
import React from 'react';
import { render } from 'react-dom';
import { Provider } from 'react-redux';
import { Router, browserHistory } from 'react-router';
import routes from './routes';
import store from './store/store';

import '../node_modules/bootstrap/dist/css/bootstrap.min.css';
import './styles/styles.css';

render(
    <Provider store={store}>
        <Router history={browserHistory} routes={routes} />
    </Provider>,
    document.getElementById('app')
);
```

Router ou App si on n'utilise pas de router

Organisation



10. Test

```
npm i react-addons-test-utils –D
npm i enzyme -D
```

Documentation Enzyme, tests avec redux, redux mock store

a. Jest

Documentation, assertions

Installation de Jest + utilitaires

```
npm i jest babel-jest react-test-renderer enzyme react-addons-test-utils -D
```

Configuration

.babelrc

```
{
  "presets": ["react","latest"]
}
```

NPM Script

```
"test": "jest"
```

package.json

```
"jest": {
    "testEnvironment": "node",
    "transform": {
      "^.+\\.(js|jsx)$": "<rootDir>/node_modules/babel-jest",
      "^.+\\.css$": "<rootDir>/config/jest/cssTransform.js"
    }
  }
```

« cssTransform.js » (dans un dossier « config/jest »)

Permet d'handle les imports de feuilles de styles dans le code (qui normalement seraient gérés par les loaders de Webpack)

```
'use strict';
module.exports = {
    process() {
        return 'module.exports = {};';
    },
    getCacheKey() {
        return 'cssTransform';
    },
};
```

eslint plugin

Documentation

Pour retirer les errors pour « expect »

```
npm i eslint-plugin-jest -D
```

Dans le fichier de configuration de eslint (« eslintrc.json » par exemple)

Ajouter le **plugin**

```
"plugins": [
        "jest"
],
```

Ajouter dans « **env** »

```
"env": {
        "jest/globals": true
},
```

Ajouter les **rules**

```
"rules": {
        "jest/no-disabled-tests": "warn",
        "jest/no-focused-tests": "error",
        "jest/no-identical-title": "error",
        "jest/valid-expect": "error"
    }
```

Création de tests avec Jest

On peut créer un dossier « __tests__ » ou définir les fichiers de tests dans le même dossier que les sources (« src/.. »)

Exemples

Test unitaire

```
import React from 'react';
import PropTypes from 'prop-types';

class MyComponent1 extends React.Component {
    constructor(props) {
        super(props);
    }
    getSum() {
        let { a, b } = this.props;
        return a + b;
    }
    render() {
        let sum = this.getSum();
        return <div>{sum}</div>;
    }
}
MyComponent1.propTypes = {
    a: PropTypes.number.isRequired,
    b: PropTypes.number.isRequired
};
export default MyComponent1;
```

MyComponent1.spec.js

```
import React from 'react';
import { shallow } from 'enzyme';

import MyComponent1 from './MyComponent1';

describe('MyComponent1', () => {

    test('getSum should return 6 with a=4 and b=2', () => {
        const wrapper = shallow(<MyComponent1 a={4} b={2} />);
        const result = wrapper.instance().getSum();
        expect(result).toEqual(6);
    });

});
```

Note : on pourrait aussi créer un helper avec seulement la function getSum

Snpashot

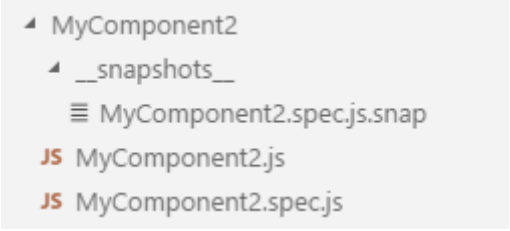
Crée une « photo » du html généré

```
import React from 'react';
import PropTypes from 'prop-types';

class MyComponent2 extends React.Component {
    constructor(props) {
        super(props);
    }
    render() {
        return <div>{this.props.message}</div>;
    }
}
MyComponent2.propTypes = {
    message: PropTypes.string.isRequired
};
export default MyComponent2;
```

« MyComponent2.spec.js »

```
import React from 'react';
import renderer from 'react-test-renderer';
import MyComponent2 from './MyComponent2';
import { shallow } from 'enzyme';

describe('MyComponent2', () => {
  test('snapshot demo', () => {
    const tree = renderer.create(<MyComponent2 message="My message"
/>).toJSON();
    expect(tree).toMatchSnapshot();
  });
});
```

Un dossier __snapshots__ est crée



Avec pour l'exemple :

```
// Jest Snapshot v1, https://goo.gl/fbAQLP

exports[`MyComponent2 snapshot demo 1`] = `
<div>
 My message
</div>
`
```

Test Async avec Jest

Pas besoin d'ajouter de function « done » comme avec Mocha par exemple. Exemple on simule un event avec enzyme

```
import React from 'react';
import { shallow } from 'enzyme';
import Checkbox from './Checkbox';

describe('Checkbox', () => {
    it('Should notify on value change', () => {
        let props = {
            name: 'my-field',
            onValueChange: (name, value) => {
                expect(name).toEqual('my-field');
                expect(value).toBeTruthy();
            }
        };
        const wrapper = shallow(<Checkbox  {...props} />);
        let input = wrapper.find('input');

        let event = { target: { checked: true } };
        input.simulate('change', event);
    });
});
```

b. Tester les components

Shallow, mount (full DOM rendering), render (HTML)

Enzyme permet de sélectionner plus facilement les éléments, presque comme jQuery, qu'avec seulement les test utils.

```
import { assert } from 'chai';
import React from 'react';
import { mount, shallow } from 'enzyme';
import PostForm from './PostForm';

function setup() {
    let props = {
        post: {
            title: 'Post 1',
            content: 'Content 1'
        },
        onTitleChange: () => { },
        onContentChange: () => { },
        onSubmit: () => { }
    };

    return shallow(<PostForm {...props} />);
}

describe('PostForm', () => {

    it('Should fill fields', () => {
        const wrapper = setup();
        // html element
        assert.equal(wrapper.find('h1').text(),'Add New Post');
        // input value
        assert.equal(wrapper.find('#title').props().value, 'Post 1');
    });

});
```

Exemple de form

```
import React from 'react';

const PostForm = ({ post, onTitleChange, onContentChange, onSubmit }) => {
    return (
        <form onSubmit={onSubmit}>
            <h1>Add New Post</h1>
            <div className="form-group">
                <label htmlFor="title">Title:</label>
                <input id="title" type="text" value={post.title} onChange={o
nTitleChange} />
            </div>
            <div className="form-group">
                <textarea value={post.content} onChange={onContentChange} />
            </div>
            <input type="submit" value="Submit" className="btn btn-
primary" />
        </form>
    );
}

PostForm.propTypes = {
    post: React.PropTypes.object.isRequired,
    onSubmit: React.PropTypes.func.isRequired,
    onTitleChange: React.PropTypes.func.isRequired,
    onContentChange: React.PropTypes.func.isRequired
};

export default PostForm;
```

Et le container

```
import React, { Component } from 'react';
import { connect } from 'react-redux';
import { browserHistory } from 'react-router';
import { bindActionCreators } from 'redux';
import { createPost } from '../../actions/postActions';
import PostForm from './PostForm';

class AddPostPage extends Component {
    constructor(props) {
        super(props);

        this.state = {
            post: {
                title: 'Post 3',
                content: 'Content 3'
            }
        };
```

```
        this.onSubmit = this.onSubmit.bind(this);
        this.onTitleChange = this.onTitleChange.bind(this);
        this.onContentChange = this.onContentChange.bind(this);
    }

    onTitleChange(event) {
        let post = this.state.post;
        post.title = event.target.value;
        this.setState({
            post
        });
    }

    onContentChange(event) {
        let post = this.state.post;
        post.content = event.target.value;
        this.setState({
            post
        });
    }

    onSubmit(event) {
        event.preventDefault();
        let post = this.state.post;
        // save
        this.props.dispatch(createPost(post)).then((result) => {
            browserHistory.push('/posts');
        });
    }

    render() {
        return (
            <PostForm
                post={this.state.post}
                onTitleChange={this.onTitleChange}
                onContentChange={this.onContentChange}
                onSubmit={this.onSubmit}
            />
        );
    }
}

function mapStateToProps(state, ownProps) {
    return {
        posts: state.posts
    };
}
export default connect(mapStateToProps)(AddPostPage);
```

c. Tester les actions

Synchrone

```
import { assert } from 'chai';
import * as postActions from './postActions';
import * as types from './actionTypes';

describe('Post Actions', () => {
    describe('createPostSuccess', () => {
        it('should create a CREATE_POST_SUCCESS action', () => {
            const post = { title: 'Post 1', content: 'Content 1' };
            const action = postActions.createPostSuccess(post);

            assert.deepEqual(action, {
                type: types.CREATE_POST_SUCCESS,
                post
            });
        });
    });
});
```

Asynchrone

Installer

```
npm i nock redux-mock-store -D
```

```
import { assert } from 'chai';
import * as postActions from './postActions';
import * as types from './actionTypes';

import thunk from 'redux-thunk';
import nock from 'nock';
import configureMockStore from 'redux-mock-store';

const middleware = [thunk];
const mockStore = configureMockStore(middleware);

describe('Get Actions', () => {
    afterEach(() => {
        nock.cleanAll();
    });

    it('should create LOAD_POSTS_SUCCESS when loading posts', (done) => {
        const store = mockStore({ posts: [] });

        store.dispatch(postActions.loadPosts())
            .then(() => {
                const actions = store.getActions();
                assert.equal(actions[0].type, types.LOAD_POSTS_SUCCESS);
                done();
```

```
            });
    });
});
```

d. Tester un reducer

```
import { assert } from 'chai';
import postReducer from './postReducer';
import * as postActions from '../actions/postActions';

describe('Post reducer', () => {
    it('should add post on CREATE_POST_SUCCESS', () => {
        const initialState = [
            { title: 'Post 1', content: 'Content 1' },
            { title: 'Post 2', content: 'Content 2' }
        ];

        // action
        const newPost = { title: 'Post 3', content: 'Content 3' };
        const action = postActions.createPostSuccess(newPost);

        // reducer + new state
        const newState = postReducer(initialState, action);

        assert.equal(newState.length, 3);
        assert.deepEqual(newState[2], newPost);
    });
});
```

Note : Avec Karma + PhantomJS installer

```
npm i babel-polyfill -D
```

Ajouter aux files de karma.conf.js

```
files: [
        'node_modules/babel-polyfill/browser.js',
         './src/**/*.spec.js'
],
```

e. Tester le store

```
import { assert } from 'chai';
import { createStore, combineReducers } from 'redux';
import posts from '../reducers/postReducer';
import * as postActions from '../actions/postActions';

describe('Store', function () {
    it('Should handle creating posts', function () {
        // arrange
        const store = createStore(combineReducers({
            posts
        }), {
                posts: []
            });
        const post = {
            title: 'Post 1',
            content: 'Content 1'
        };

        // act
        const action = postActions.createPostSuccess(post);
        store.dispatch(action);

        // assert
        const actual = store.getState().posts[0];
        assert.deepEqual(actual, post);
    });
});
```

11. Form et liste

```
class App extends Component {
  constructor(props) {
    super(props);
    this.state = {
      current: '',
      items: []
    };
    this.onChange = this.onChange.bind(this);
    this.onSubmit = this.onSubmit.bind(this);
  }
  onChange(event) {
    let value = event.target.value;
    this.setState({
      current: value
    });
  }
  onSubmit(event) {
    event.preventDefault();
    let items = this.state.items;
    let value = this.state.current;
    if (value && value !== '') {
      items.push(value);
      this.setState({
        current: '',
        items
      });
    }
  }
  render() {
    return (
      <div>
        <h2>Add an Item</h2>
        <form onSubmit={this.onSubmit}>
          <input type='text' value={this.state.current} onChange={this.onChange} />
          <input type='submit' value='Submit' />
        </form>
        <div>
          <ul>
            {this.state.items.map((item, i) => {
              return <li key={i}>{item}</li>
            })}
          </ul>
        </div>
      </div>
    );
  }
}
```

Note : dans le cas présent : le component App sera re render à chaque saisie dans le champ texte, et pas seulement qu'à la soumission du formulaire.

12. Animation

a. Avec JavaScript

Exemple : placer une référence sur un élément

```
<form onSubmit={this.onSubmit} ref='form'>

</form>
```

Attendre que le component soit mounted et récupérer la ref

```
componentDidMount() {
    this.form = ReactDOM.findDOMNode(this.refs.form);
}
```

On pourrait également directement récupérer l'élément pour « this » sans passer par une ref.

Puis quand on désire animer l'élément, exemple au submit

```
animate(this.form, 'bounce', () => {
});
```

Code **JavaScript pour l'animation**

```
const animationNames = {
  'animation': 'animationend',
  'WebkitAnimation': 'webkitAnimationEnd',
  'OAnimation': 'oAnimationEnd',
  'msAnimation': 'MSAnimationEnd',
};
function getEndAnimationName(el) {
  for (let name in animationNames) {
    if (el.style[name] !== undefined) {
      return animationNames[name];
    }
  }
}
function animate(el, cssClass, next) {
  function onEnd() {
    this.removeEventListener(endAnimationName, onEnd);
    this.classList.remove(cssClass);
    if (next) { next(); }
  }
  let endAnimationName = getEndAnimationName(el);
  if (endAnimationName) {
    el.addEventListener(endAnimationName, onEnd);
    el.classList.add(cssClass);
  }
  else if (next) { next(); }
}
```

b. Avec ReactCSSTransitionGroup

```
npm i react-addons-css-transition-group -S
```

Exemple sur une « todo » liste

```
<ul>
  <ReactCSSTransitionGroup transitionName="animation"
                           transitionEnterTimeout={1000}
                           transitionLeaveTimeout={1000}>
    {this.state.items.map((item, i) => { return
    <li key={i}>{item} <button onClick={()=> this.onDeleteItem(item, i)}>Delete</button></li>
    })}
  </ReactCSSTransitionGroup>
</ul>
```

Nom de « base » pour les classes CSS + ajouter les durées des animations enter et leave

Transition jouée en **entrant** (à l'ajout d'un item à la liste)

```
.animation-enter {
 opacity: 0.01;
 }
.animation-enter.animation-enter-active {
 opacity: 1;
 transition: opacity 1s ease-in;
 }
```

Transition jouée au **leave** (suppression d'un item de la liste dans exemple)

```
 .animation-leave {
 opacity: 1;
 }
.animation-leave.animation-leave-active {
 opacity: 0;
 transition: opacity 1s ease-in;
 }
```

OU avec **animations**

```
@keyframes fadeIn{
  from {
    opacity: 0;
  }
  to{
    opacity: 1;
  }
}

@keyframes fadeOut{
  from {
    opacity: 1;
  }
  to{
    opacity: 0;
  }
```

```
}

.animation-enter {
 opacity: 0.01;
 }
.animation-enter.animation-enter-active {
 opacity: 1;
 animation: 1000ms linear 0s fadeIn forwards;
 }

 .animation-leave {
 opacity: 1;
 }
.animation-leave.animation-leave-active {
 opacity: 0;
 animation: 1000ms linear 0s fadeOut forwards;
 }
```

On peut également utiliser **ReactTransitionGroup** wrappant un **component** qui ajoute des events au cycle de vie du component (« **componentWillEnter** », « **componentWillLeave** »). On peut utiliser alors JavaScript pour animer l'élément

Exemple