ReactJS

Philippe Depouilly

March 2019

mathrice

ReactJS l'arme de guerre pour décupler l'expérience utilisateur...

- ReactJS est un framework moderne de développement d'interfaces Web modernes et riches... il permet de développer rapidement des composants qu'il suffit ensuite d'imbriquer... mais...
- ReactJS est un peu compliqué à mettre en oeuvre...
- Pourquoi ?

mathrice

ReactJS l'arme de guerre pour décupler l'expérience utilisateur...

- ReactJS est un ~~framework~~ outil moderne de développement d'interfaces Web modernes et riches... il permet de développer rapidement des composantes qu'il suffit ensuite d'imbriquer... mais...
- ReactJS tout seul n'est pas rigolo... car ça revient à écrire du du HTML en JavaScript...
- Mais le JavaScript standard c'est trop simple... alors écrivons en ES6... qu'il faudra "transpiler" en JavaScript standard...
- Alors JSX c'est tellement mieux pour simplifier mon code... c'est une syntaxe compacte de JavaScript qu'il faudra aussi "transpiler" en JavaScript
- Donc il faut utiliser Babel pour convertir le JSX en JavaScript... et ES6 en standard...
- Qu'il est plus sympa d'utiliser un environnement de développement comme Webpack (Aïe...)

ReactJS l'arme de guerre pour décupler l'expérience utilisateur...

- ReactJS est un outil moderne de développement d'interfaces Web modernes et riches... il permet de développer rapidement des composantes qu'il suffit ensuite d'imbriquer... mais...
- Il m'a fallu installer 67 Mo de modules npm pour afficher $< div > Hello\ World < div >$ sur mon navigateur
- Un fichier package.json, webpack.config.js, une arborescence pour découper tout cela en trois fichier js...
- index.html, app/main.js app/App.js package.json webpack.config.js

Mais je m'en suis sorti...

mathrice

ReactJS, alors c'est quoi...

Pour la suite de cette présentation j'ai repris in extenso l'excellent article de Batiste Donaux :
`https://www.baptiste-donaux.fr/react-redux-concept/`

mathrice

ReactJS, alors c'est quoi...

Avec ReactJS vous pouvez fabriquer des composant sous la forme

```
import React from "react"; // Syntaxe ES6

class App extends React.Component {
  render() {
    // Syntaxe JSX
    return <div id="main-element" className="title">
    Hello World !</div>
  }
}

export default App;
```

mathrice

ReactJS, alors c'est quoi...

Au lieu de

```
const React = require("react");

class App extends React.Component {
render() {
    return React.createElement(
              "div",
              {id: 'main-element', className: "title"},
              "Hello, world!"
            )
    }
}

export default App;
```

ReactJS, alors c'est quoi...

Et voici le JavaScript qui va injecter le composant React dans le DOM HTML

```
import React from "react";
import ReactDom from "react-dom";

import App from "./App";

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

mathrice

ReactJS, un exemple de composant de liste

Fichier list.jsx

```
const React = require("react"),
    Item = require("./item.jsx");
class List extends React.Component {
    render() {
        return <ul>
            {this.props.items.map((item) => <Item
                content={item.content}
            />)}
        </ul>;
    }
}
module.exports = List;
```

mathrice

ReactJS, un exemple de composant de liste

Fichier item.jsx

```
const React = require("react");
class Item extends React.Component {
    render() {
        return <li>
            {this.props.content}
        </li>;
    }
}
module.exports = Item;
```

mathrice

ReactJS, un exemple de composant de liste

fichier App.js

```
const React = require("react"),
    ReactDOM = require("react-dom"),
    List = require("./list.jsx"),
    items = [{
        content: "Foo"
    }, {
        content: "Bar"
    }, {
        content: "Baz"
    }];
ReactDOM.render(
    <List
        items={items}
    />,
    document.getElementById("container")
```

ReactJS sans redux c'est dommage...

redux est un middelware permettant de gérer l'état des objets React (les données d'un composant), de les modifier et de permettre à un composant de connaître l'état d'un autre composant (ses données).
redux utilise un mécanisme de "store"

mathrice

Redux, intégration de redux dans ReactJS

```
const List = require("./components/list.jsx"),
    Provider = require("react-redux").Provider,
    React = require("react"),
    ReactDOM = require("react-dom"),
    reducers = require("./redux/reducers.js"),
    Redux = require("redux");
let store = null;
module.exports = () => {
    store = Redux.createStore(
        reducers,
        {
            items: []
        }
    );
    ReactDOM.render(
        <Provider store={store} >
            <List/>
        </Provider>,
```

Redux, instrumentation des l'objet List

```
const actions = require("./redux/actions.js"),
    Item = require("./item.jsx"),
    React = require("react"),
    ReactRedux = require("react-redux");
class List extends React.Component {
    render() {
        return <ul>
            {this.props.items.map((item) => <Item
                content={item.content}
                deleteItem={this.props.deleteItem.bind(null, ite
            />)}
        </ul>;
    }
}
module.exports = ReactRedux.connect(
    (state = {}) => state,
    (dispatch, props) => Object.assign({}, props, {
        deleteItem: actions.deleteItem.bind(null, dispatch)
```

Redux, instrumentation des l'objet Item

```
const React = require("react");
class Item extends React.Component {
    render() {
        return <li>
            {this.props.content}
            <a onClick={this.props.deleteItem}>
                Supprimer
            </a>
        </li>;
    }
}
module.exports = Item;
```

mathrice

Redux, action deleteItem

Fichier redux/actions.js

```
function deleteItem(dispatch, item) {
    dispatch({
        type: "REMOVE_ITEM",
        item: item
    });
}
module.exports = {
    deleteItem: deleteItem
};
```

mathrice

Redux, le reducers qui gère la source des données

Fichier redux/reducers.js

```
function reducer(state, action) {
    const newState = Object.assign({}, state);
    switch (action.type) {
        case "REMOVE_ITEM":
            const index = newState.items.indexOf(action.item);
            if (index !== -1) {
                newState.items.splice(index, 1);
            }
            break;
        default:
            return state;
    }
    return newState;
}
module.exports = reducer;
```

mathrice

Les fichiers utiles...

```
npm init
# repondre oui...
npm install --save @babel/preset-react @babel/preset-env \
webpack webpack-cli babel-loader @babel/core react react-dom
```

Et le fichier package.json est prêt et les modules sont dans node_modules

mathrice

Les fichiers utiles...

Fichier webpack.config.js

```
module.exports = {
    entry: "./app/main.js",
    output: {
        path: __dirname + "./",
        filename: "index.js"
    },
    devServer: { // configuration du server permettant le live-reload
        inline: true,
        port: 8080
    },
    module: {
        rules: [
            {
                test: /\.js$/,
                exclude: /node_modules/,
                loader: 'babel-loader',
                query: {
                    presets: ['@babel/preset-env','@babel/react']
                }
            }
        ]
    }
}
```

Utiliser webpack-dev-server pour un serveur http sur le port 8080 pour afficher localement son site

ReactJS

Des Questions ?