Router

Router React

Nous allons installer **React Router**, une librairie implémentant la gestion des routes. Cette librairie permettra de connecter une route à un Component donné.

En utilisant cette libraire, nous sommes dans le cadre d'une SAP (single application page). Il n'y a pas de requête HTTP côté client, de fait, tout se passe dans la page du navigateur et en Javascript.

```
npm install --save react-router-dom
```

Il faudra par la suite importer les modules suivants dans l'application :

```
import React from "react";
import { BrowserRouter as Router, Route, Link } from "react-router-dom";
```

Pour définir une route avec cette librairie, vous écrirez :

```
<!-- en HTML : <a href="/" >Home</a>
Et avec la librairie : -->
<link to="/" >Home</link>
```

Une route sera connectée à un composant donné de la manière suivante :

```
<Router>
    <Link to="/" >Home </Link>
    <Route exact path="/" component={Home} />
</Router>
```

Remarques :

L'attribut exact permet de faire correspondre uniquement l'url "/" avec cette route. Si vous ne mettez pas exact d'autres urls pourront correspondre également : "/a", "/a/b", ...

On met souvent exact pour l'url racine de l'application.

Plusieurs routes

Si vous avez plusieurs routes à gérer dans votre application, vous devez utiliser la balise switch. Lorsqu'il y a plusieurs routes à rendre la méthode switch utilisera la première route trouvée pour faire la correspondance :

```
<Router>
    <div>
    <ul>
        <li>
            <Link to="/">Home</Link>
```

```
            </li>
            <li>
                <Link to="/login">Login</Link>
            </li>
            <li>
                <Link to="/dashboard">Dashboard</Link>
            </li>
        </ul>
        <Switch>
            <Route exact path="/">
                <Home />
            </Route>
            <Route path="/login">
                <Login />
            </Route>
            <Route path="/dashboard">
                <Dashboard />
            </Route>
        </Switch>
        </div>
    </Router>
```

Gestion de paramètres dans une route

Vous pouvez passer des paramètres à une route. Dans ce cas utiliser la syntaxe ":param" pour désigner le paramètre variable dans l'url.

Par exemple si vous avez des posts avec leur id vous écrirez dans le code React :

```
{
    posts.map((post, i) => ( <Link to={`/post/${post.id}`}>{post.title}</Link> ) )
}
```

D'un autre côté vous connecterez chaque url à un pattern à déterminer avec un paramètre variable :

```
<Switch>
    <Route path="post/:id" component={<Post />} />
</Switch>
```

Pour récupérer le paramètre id dans l'url il faudra alors dans le contexte du Router utiliser la variable match dans les props :

```
const { url, path } = this.props.match;
```

La variable path designe le pattern /post/:id et url l'url concrète dans le code par exemple /post/12

Redirection & PrivateRoute

Vous pouvez protéger des routes avec React Router et mettre en place un système de redirection, c'est pratique lorsqu'on souhaite par exemple protéger un dashboard pour administrer des contenus.

```
<Router>
    <div>
    <ul>
        <li>
        <Link to="/dashboard">Dashboard</Link>
        </li>
    </ul>
    <Switch>
        <PrivateRoute path="/dashboard">
            <Dashboard />
        </PrivateRoute>
    </Switch>
    </div>
</Router>
```

Vous pouvez utiliser la syntaxe suivante également, la logique de la redirection et de la protection de la route "/dashboard" sera implémentée dans le composant PrivateRoute

```
<PrivateRoute path="/dashboard" />
```

Dans le composant PrivateRoute vous écrirez quelque chose comme suit, où la variable rest contiendra les informations liées au router. La balise Redirect du router permet pour sa part de rediriger la route vers "/login", notez que vous pouvez également définir l'origine avec from.

Pour que tout fonctionne correctement vous devez vous trouvez dans le contexte du Router.

```
const { ...rest } = this.props;

return (
    <Route
        { ...rest }
        render={ (rest) =>
        isLogged ? <Dashboard  { ...rest } /> :
        <Redirect
            to={{pathname : '/login', state : { from : "/"}}}
        />
        }
    />
)
```

Exercice avec des posts

Créez une application blog.

```
const POSTS = [
  { id: 16, title: "React JS", content: "Libraire ou Framework ?" },
  { id: 11, title: "React Native", content: "Mobile React" },
  { id: 100, title: "Angular", content: "Super ..." },
  { id: 7, title: "Symfony", content: "Framework expressif ..." },
  { id: 27, title: "MongoDB", content: "Base de données NoSQL" },
];
```

- Créez une page d'accueil avec un texte de votre choix.
- Créez un lien qui affiche tous les titres des posts
- Chaque titre de post est cliquable. Un lien de ce type affichera le titre et le contenu de l'article.

Exercice private routes & posts

Créez un squelette d'application private-posts sur votre machine.

Vous allez mettre en place un dashboard et une page home. Le dashboard contiendra les liens vers des posts. Pour y accéder vous devrez vous authentifiez à l'aide d'un formulaire.

Page principale

Cette page est simple vous affichez le menu principal, par défaut on arrive sur la page principale avec un message de bienvenu :

```
Home Dashboard

Bienvenu sur la page principale
```

Si vous essayez de vous connectez sur le dashboard la première fois vous serez alors redirigez vers la page de login. Et sinon, si vous êtes authentifié alors vous êtes redirigé vers la page en question.

Page de login

Cette page (composant Login) est constituée de deux champs respectivement un champ pour son email et un champ pour le password. Le menu principal reste présent sur cette page. Pour l'authentification utilisez un couple de valeurs email/password que vous placerez dans une constante dans le composant Login.

Utilisez le locastorage du navigateur pour enregistrer une variable token qui permettra de savoir si vous êtes authentifié ou non.

Vous devez également gérer les messages d'erreur si l'authentification échoue.

```
Home Dashboard

Authentifiez-vous

Email : []

Password : []

[Valider]
```

Page Dashboard

Sur cette page vous afficherez les titres de chaque post. Ces derniers seront cliquables et une fois cliqués dessus on affichera le titre ainsi que le contenu de

l'article dans le même composant Dashabord.

```
Home Dashboard

Bienvenu sur le Dashboard

React JS
React Native
Angular
Symfony
MongoDB

---- contenu de l'article

MongoDB

Base de données NoSQL
```

Pour vous aidez voici ci-dessous la constante contenant les posts que l'on souhaite afficher :

```
const posts = [
  { id: 16, title: "React JS", content: "Libraire ou Framework ?" },
  { id: 11, title: "React Native", content: "Mobile React" },
  { id: 100, title: "Angular", content: "Super ..." },
  { id: 7, title: "Symfony", content: "Framework expressif ..." },
  { id: 27, title: "MongoDB", content: "Base de données NoSQL" },
];
```