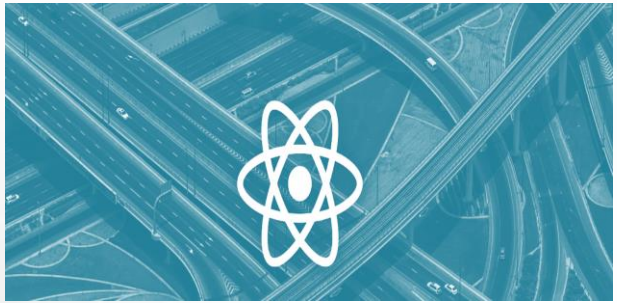
Chapitre 6 : Le routage



Qu'est-ce que le routage?

- Le routage est la capacité d'afficher différentes pages à l'utilisateur.
- Cela signifie qu'il donne la possibilité de naviguer entre les différentes parties d'une application en entrant une URL ou en cliquant sur un élément.
- React est une bibliothèque JavaScript qui permet de créer des interfaces utilisateur. On peut également l'utiliser pour créer des applications multipages à l'aide de React Router.
- Par défaut, React vient sans routage. Et pour l'activer, nous devons ajouter une bibliothèque nommée **react-router**.
- Il s'agit d'une bibliothèque tierce qui permet le routage dans les applications React.

Qu'est-ce que React Router ?

- Le React Router est une bibliothèque qui permet de gérer les routes dans une application Web, en utilisant le routage dynamique.
- Le routage dynamique a lieu lors du rendu de l'application sur la machine, contrairement à l'ancienne architecture de routage où le routage est géré dans une configuration en dehors d'une application en cours d'exécution.
- Le routeur React implémente une approche de routage basée sur les composants.
- Il fournit différents composants de routage en fonction des besoins de l'application et de la plate-forme.
- Cela rend l'interface de l'application synchrone avec l'URL du navigateur.
- Le React Router permet de définir des URL dynamiques et de sélectionner un Component approprié pour « render » (afficher) sur le navigateur de l’utilisateur en correspondance à chaque URL.

Mise en place des routes

- Tout d'abord on doit installer React Router à partir du registre public npm avec npm ou yarn.
- Puisque il s'agit d'une application Web, on va utiliser **react-router-dom** :

npm install --save react-router-dom

- Ensuite, il faut utiliser les imports suivants pour mettre en place les routes :

import { BrowserRouter, Route, Link } from "react-router-dom";

- Le paquet React Router a un composant pratique appelé BrowserRouter.
- On doit d'abord l'importer depuis react-router-dom afin de pouvoir utiliser le routage dans l'application.

Configuration d'un routeur

- React Router fournit deux composants qui sont <BrowserRouter> et <HashRouter>. Ces deux composants sont différents du type de l'URL qu'ils créeront et synchronisent.
- <BrowserRouter> est le plus utilisé, il se base sur History API comprenant dans HTML5 afin de surveiller l'histoire du routage. Il doit contenir tous les éléments de l'application où le routage est nécessaire.
- Cela signifie que si on a besoin du routage dans l'ensemble de l'application, on doit envelopper le composant le plus haut avec BrowserRouter.

```
<BrowserRouter>
  <Route path="/" component={Home}/>
  <Route path="/about" component={About}/>
  <Route path="/topics" component={Topics}/>
</BrowserRouter>
```

- Le composant **<Route>** définit un mapping entre un URL et un Component.
- Cela signifie que lorsqu'un utilisateur accède par un URL sur le navigateur, un Component correspondant sera rendu sur l'interface.

Configuration d'un routeur

- On peut renommer BrowserRouter as Router comme ci-dessous, ça rend tout simplement les représentations plus lisibles :

```
import {Route, Switch, Link, BrowserRouter as Router} from 'react-router-dom';
export default function App() {
 return (
  <Router>
   <nav>
     <ul>
      <li>          <a href="/">Home</a>          </li>
    </ul>
   </nav>
<Route path="/" component={Home} />
 </Router>
 ) }
```

Ajout de liens 1/2

- Pour obtenir toute la puissance de React Router, on doit avoir plusieurs pages et liens avec lesquels sont attachés.
- Les liens sont représentés par **Link**.

```
import {Route, Switch, Link, BrowserRouter as Router} from 'react-router-dom';
export default function App() {
 return (
  <Router>
   <nav>
     <ul> <li> <Link to="/">Home</Link> </li>
          <li> <Link to="/about">About</Link> </li>      </ul>
   </nav>
    <Route path="/" exact component={Home} />
    <Route path="/about" component={About} />
 </Router>
 ) }
```

Ajout de liens 2/2

- Au lieu d'utiliser « a href », React Router utilise Link et to pour pouvoir basculer entre les pages sans recharger la page.
- Ainsi on pourra accéder à différentes parties de l'application via des liens.
- Seulement, le composant Home est toujours affiché même si on passe à d'autres pages.
- La raison est que le React Router vérifie si le path défini commence par / si c'est le cas, il rendra le composant.
- Et ici, le premier itinéraire commence par / donc Home sera affiché.
- Mais on peut toujours changer le comportement par défaut en ajoutant un nouveau accessoire à Route: la propriété **exact**.

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

- En mettant à jour l'itinéraire Home avec exact, maintenant, il ne sera rendu que s'il correspond au chemin complet.

Switch

- En enveloppant les routes avec Switch on précisera à React Router de ne charger qu'une seule route à la fois.

```
import {Route, Switch, Link, BrowserRouter as Router} from 'react-router-dom';
export default function App() {
 return (
  <Router>
   <nav>  <ul>  <li> <Link to="/">Home</Link> </li>
                <li> <Link to="/about">About</Link> </li>     </ul>      </nav>
<Switch>
    <Route path="/" exact component={Home} />
    <Route path="/about" component={About} />
</Switch>
 </Router>
 ) }
```

Passage des paramètres entre les routes

- Pour passer des paramètres entre les routes, il faut déclarer une nouvelle constante (exemple name) qui sera passée en paramètre à une autre page (à la page About, par exemple).
- Avec cela, on doit mettre à jour la route About, en ajustant son chemin pour pouvoir recevoir le nom en tant que paramètre path="/about/:name".
- Maintenant, le paramètre sera reçu sous forme de props dans le composant About. On le récupère avec **props.match.params.name**

```
import {Route, Switch, Link, BrowserRouter as Router} from 'react-router-dom';
export default function App() {  const name = "Mohamed";
 return (    <Router>
                    <nav>   <ul>   <li> <Link to="/">Home</Link> </li>
                                  <li>  <Link to={`/about/${name}`}>About</Link> </li>      </ul>      </nav>
<Switch>
     <Route path="/" exact component={Home} />
     <Route path="/about/:name" component={About} />
</Switch>
 </Router>    ) }
```

Redirection vers une autre page

- Le React Router a un autre composant nommé Redirect qui permet à rediriger l'utilisateur vers une autre page.

```
import {  BrowserRouter as Router,  Route,  Link,  Switch,  Redirect} from "react-router-dom";
export default function App() {
 return (
<Router>  <nav>   <ul>   <li> <Link to="/">Home</Link> </li>
                                    <li> <Link to ="/about"> About</Link> </li>
                                    <li>  <Link to="/topics">Topics</Link> </li>       </ul> </nav>
<Switch>
     <Route path="/" exact component={Home} />
     <Route path="/about/:name" component={About} />
     <Route path="/topics" component={Topics}/>
     <Redirect from="/about" to="/topics" />
</Switch>
</Router>
 ) }
```

Redirection vers la page 404

- Pour rediriger l'utilisateur, vers une page 404, on peut créer un composant pour afficher la page, mais aussi on peut juste afficher un message avec render.
- Le nouvel itinéraire sans path interceptera toutes les routes qui n'existent pas et redirigera l'utilisateur vers la page 404.

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

export default function App() {
  return (
<Router>  <nav>   <ul>   <li> <Link to="/">Home</Link> </li>
                                        <li> <Link to ="/about"> About</Link> </li>
                    </ul> </nav>
<Switch>
      <Route path="/" exact component={Home} />
      <Route path="/about/:name" component={About} />
       <Route render={() => <h1>404: page not found</h1>} />
</Switch>
</Router>
```

Les Hooks du Routeur 1/3

- Les Hooks du routeur sont :
  - useHistory
  - useParams
  - useLocation
- Le hook **useHistory** donne accès à l'instance d'historique sans la retirer des props reçus.

```
import { useHistory } from "react-router-dom";
const Contact = () => {
  const { history } = useHistory();
  return (
    <div>
      <h1>Contact</h1>
      <button onClick={() => history.push("/")}>Go to home</button>
    </div>
  )
}
```

Les Hooks du Routeur 2/3

- Le Hook **useParams** permet d'accéder au paramètre passé dans l'URL sans utiliser l'objet props.

```
import { useParams } from "react-router-dom";

const Contact = () => {
  const { name } = useParams();
 return (
   <div>
     <h1>Welcome {name}</h1>
   </div>
 )
}
```

Les Hooks du Routeur 3/3

- Le Hook **useLocation** renvoie l'objet de localisation qui représente l'URL actuelle.

```
import { useLocation } from "react-router-dom";

const Contact = () => {
  const { pathname } = useLocation();
 return (
   <div>
       <h1>Contact</h1>
      <p>Current URL: {pathname}</p>
  </div>
 )
}
```