Routing avec React Router

Présentation

React Router (https://reactrouter.com) :

- Est une librairie permettant de naviguer entre des pages d'une application React
- La navigation se fait côté client
- En mode "Single Page App"
- Est utilisable en web (`react-router-dom`) ou en React Native (`react-router-native`)

Installation

```
$ npm install react-router-dom
```

Les types TypeScript sont inclus avec la librairie.

Les routeurs

React Router fournit plusieurs routeur adaptés à différentes situations :

`BrowserRouter` :	Routeur de base, utilisant le système d'historique du navigateur *URLs : /auth/login*
`HashRouter` :	**Non recommandé.** Utilise la partie "hash" de l'URL pour stocker les routes dans le cas où celles-ci ne devraient pas être envoyées au serveur. *URLs : /#/auth/login*
`MemoryRouter` :	Utile si besoin d'un contrôle total sur la gestion de l'historique.
`NativeRouter` :	Utile pour React Native.

Définir des routes

Une fois le routeur choisi, chaque route sera définie à l'aide du composant **`<Route />`** :

```
import { BrowserRouter, Route, Routes } from 'react-router-dom';

<BrowserRouter>
  <Routes>
    <Route path="/" element={<Home />} />
    <Route path="/test" element={<TestPage />} />
    <Route path="/test/:id" element={<TestPageWithId />} />
  </Routes>
</BrowserRouter>
```

Permets de passer un paramètre en URL

Définir des routes

`BrowserRouter` :	A placer à la racine de tous les composants.
`Routes` :	Définit un emplacement dont le contenu sera déterminé par le composant `<Route />` correspondant à l'URL actuelle. Plusieurs blocs `<Routes />` peuvent être définis afin de changer plusieurs parties d'une page en fonction de l'URL.
`Route` :	Définit un contenu devant être affiché pour une URL donnée.

Le composant Route

`path` :	Chemin pour lequel afficher l'élément donné. La notation `:param` permet de définir des paramètres dynamiques d'URL.
`element` :	Elément à afficher si l'attribut `path` correspond à l'URL actuelle.
`caseSensitive` :	**true** si la route doit être sensible à la casse.

Récupérer des paramètres d'URL

Si une route est définie avec un paramètre dans son URL, il est possible de récupérer la valeur du paramètre avec le hook **`useParams`** :

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

const TestPageWithId = () => {
  const { id } = useParams();

  return (
    <span>
      ID : {id}
    </span>
  );
};
```

Navigation

Une fois les routes définies, il est possible de naviguer entre elles via des liens :

```
import { Link } from 'react-router-dom';

<Link to="/test">Go to test !</Link>
// Ou :
<a href="/test">Go to test !</a>
```

Ou programmatiquement :

```
import { useNavigate } from 'react-router-dom';

const navigate = useNavigate();

navigate('/test');
navigate(-1); // Navigue à la page précédente
```

Obtenir l'URL actuelle

Si nécessaire, il est possible de récupérer l'URL actuelle :

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

const location = useLocation();
console.log(location.pathname); // URL actuelle
```

Ou de vérifier si l'URL actuelle correspond à une route spécifique :

```
import { useMatch } from 'react-router-dom';

// Non null si l'URL actuelle correspond à la route /test/:id
const match = useMatch('/test/:id');
```

Exercice

Exercice

Créez une page d'accueil qui :

- Affichera un message de bienvenue ainsi qu'une description de l'application
- Possèdera un bouton permettant de naviguer vers la page **/characters**

Créez la page **/characters** qui affichera la liste des personnages.

Exercice

Créez une page **/character/:id** :

- Elle affichera les informations détaillées d'un personnage ayant pour ID celui passé en paramètre
- L'utilisateur y sera redirigé lors d'un clic sur un personnage de la liste