TP : Navigation avec React Router

Objectif : Transformer votre application en une "SPA" (Single Page Application) capable de naviguer entre plusieurs vues sans recharger la page.

Concepts clés : BrowserRouter, Routes, Route, Link.

Installation requise : npm install react-router-dom

1. Installation et Configuration

Avant de commencer, vous devez installer la bibliothèque de routage officielle pour le web.

Action : Dans votre terminal (dossier client), lancez :

npm install react-router-dom

2. Le cerveau de la navigation

Le routeur doit envelopper toute votre application pour surveiller l'URL et décider quoi afficher.

Fichier : src/main.tsx

Action : Observez l'import ci-dessous et utilisez le composant pour entourer <App />.

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

ReactDOM.createRoot(document.getElementById('root')!).render(
  <React.StrictMode>
    {/* TODO : Enveloppez App avec le routeur importé */}
    <___>
      <App />
    </___>
  </React.StrictMode>
);
```

3. L'aiguillage des Routes

C'est ici que vous définissez la logique : "Si l'URL est ceci, alors affiche ce composant".

Fichier : src/App.tsx

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

// Composants de test
const Home = () => <h2>Page d'accueil</h2>;
const Library = () => <h2>Ma Bibliothèque</h2>;

function App() {
  return (
    <div>
      <nav style={{ padding: '20px', background: '#eee' }}>
        {/* TODO : Utilisez le composant Link pour naviguer sans rechargement */}
        <___ to="/">Accueil</___> | <___ to="/library">Livres</___>
      </nav>

      <main style={{ padding: '20px' }}>
        {/* TODO : Utilisez Routes comme conteneur et Route pour chaque chemin */}
        <Routes>
          <Route path="/" element={<___ />} />
          <Route path="/library" element={<___ />} />

          {/* CHALLENGE : Comment gérer une erreur 404 (URL inconnue) ? */}
          <Route path="*" element={<h2>Erreur 404 : Page introuvable</h2>} />
        </Routes>
      </main>
    </div>
  );
}
```

4. Exercice de réflexion

1. **Le rechargement :** Observez l'icône de rafraîchissement de votre navigateur quand vous passez d'Accueil à Livres. Que remarquez-vous par rapport à un site classique ?
2. **L'emplacement du menu :** Pourquoi avons-nous placé la balise <nav> en dehors du composant <Routes> ?
3. **Le State :** Si vous aviez un compteur (useState) dans votre App.tsx, est-il réinitialisé quand vous changez de page ? Pourquoi ?

Astuces pour réussir

- **BrowserRouter** : Doit être utilisé une seule fois, tout en haut de l'application.
- **Link** : C'est lui qui demande au routeur de changer l'URL sans recharger le HTML.
- **Path** : L'attribut path="/" correspond à la racine de votre site.