

13. Gestion du routage

13.1. 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.

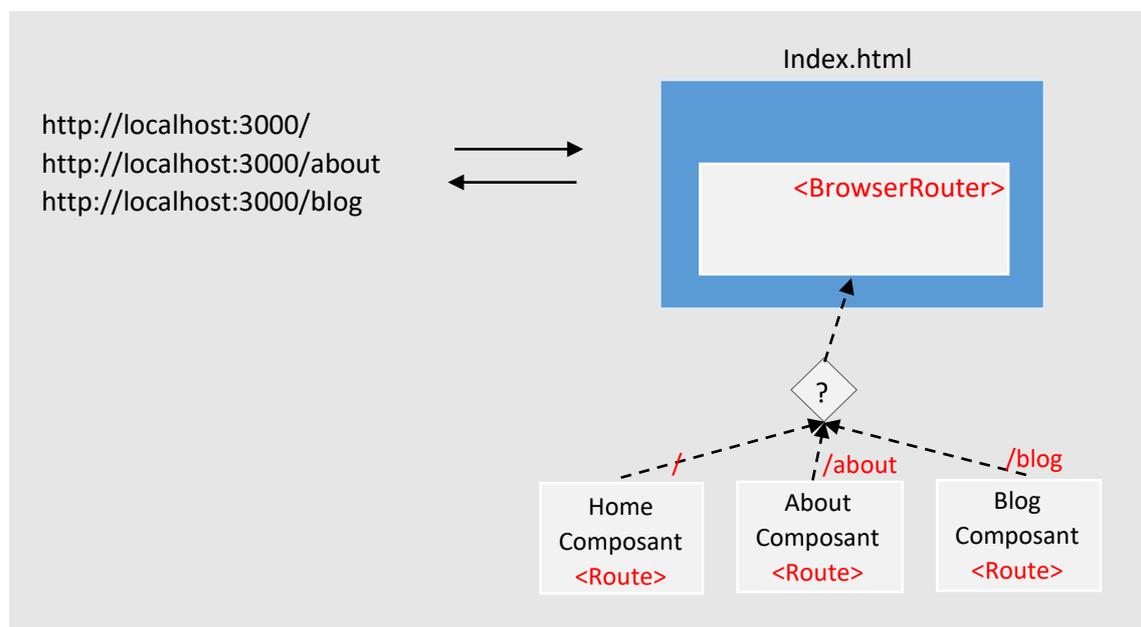
L'idée de Router (Routeur) est tellement utile car vous travaillez avec React, une bibliothèque Javascript pour programmer des applications d'une seule page (Single Page Application). Afin de développer une application React il vous faut écrire plusieurs Component mais il a besoin d'un seul fichier afin de servir des utilisateurs, c'est index.html.

Par défaut, React vient sans routage. Et pour l'activer, nous devons ajouter une bibliothèque nommée react-router.

Pour l'installer, vous devrez exécuter la commande suivante dans votre terminal :

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

Le React Router vous permet de définir des URL dynamiques et de sélectionner un Component approprié pour render (afficher) sur le navigateur d'utilisateur en correspondance à chaque URL.



13.2. <BrowserRouter> vs <HashRouter>

Le React Router vous fournit deux composants tels que <BrowserRouter> & <HashRouter>. Ces deux composants sont différents dans le type de URL qu'ils créent et synchronisent avec.

```
// <BrowserRouter>  
http://example.com/about  
// <HashRouter>  
http://example.com/#/about
```



<BrowserRouter> est utilisé plus couramment, il utilise le History API incluse dans HTML5 pour surveiller l'historique de votre routeur tandis que <HashRouter> utilise le hash de l'URL (window.location.hash) pour tout mémoriser. Si vous avez l'intention de soutenir des anciens navigateurs, vous devez être scellé contre le <HashRouter> ou créer une application React à l'aide du routeur côté client. <HashRouter> est un choix raisonnable.

13.3. <Route>

Le composant <Route> définit une relation (mapping) entre une URL et un Component. Cela signifie que lorsque l'utilisateur visite une URL sur le navigateur, un Component correspondant doit être rendu sur l'interface.

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

L'attribut exact est utilisé dans le <Route> afin de dire que ce <Route> ne fonctionne que si la URL sur le navigateur correspond absolument à la valeur de son attribut path.

13.4. Exemple :

Créer un projet 13-router et installer react-router

```
npx create-react-app 13-routage
cd 13-routage
install --save react-router-dom
```

Supprime tous les contenus des deux fichiers App.css & App.js, nous allons écrire le code de ces deux fichiers.

App.css

```
.main-route-place {
  border: 1px solid #bb8fce;
  margin: 10px;
  padding: 5px;
}
```

App.js

```
import React, { Component } from 'react';
import './App.css';
import { Routes, Route, Link } from 'react-router-dom';
class App extends Component {
  render() {
    return (
      <div>
        <ul>
          <li>
```



```
        <Link to="/">Home</Link>
      </li>
      <li>
        <Link to="/about">About</Link>
      </li>
      <li>
        <Link to="/blog">Blog</Link>
      </li>
    </ul>
    <div className="main-route-place">
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="/about" element={<About />} />
        <Route path="/blog" element={<Blog />} />
      </Routes>
    </div>
  </div>
);
}
}
class Home extends React.Component {
  render() {
    return (
      <div>
        <h2>Home</h2>
      </div>
    );
  }
}
class About extends React.Component {
  render() {
    return (
      <div>
        <h2>About</h2>
      </div>
    );
  }
}
class Blog extends React.Component {
  render() {
    return (
      <div>
        <h2>Blog</h2>
      </div>
    );
  }
}
export default App;
```



Index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import './index.css';
import App from './App';
import { BrowserRouter } from "react-router-dom";
const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App />
  </BrowserRouter>
);
```

Résultat :

