React Router使用說明

註：本章中以React Router v5.1以上為主，更多相關資訊可到以下網站觀看 - React Router官方網站

註：React Router目前版本(v5.x)區分為給網站應用React開發使用的react-router-dom，以及給手機應用React Native開發使用的react-router-native

註：網站應用使用的導覽方式是路由的概念，但手機開發的導覽方式實際上是堆疊的結構，兩者不太一樣。

安裝 react-router-dom 模組

在終端機裡，對應專案的根目錄，輸入以下的指令(選擇其中一種即可，如果已經有安裝 yarn 建議使用上面這個):

```
yarn add react-router-dom
```

或

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

React Router使用基礎說明

應用的最外層元件必須是Router元件

Router元件直接由React Router模組導入，一般都是使用BrowserRouter作為Router元件。Router元件必需位於你的應用的最外層(最上層的元件)，例如下面的最基本的套用範例：

以下為新語法(v5.x) - 推薦使用:

```
import { BrowserRouter as Router, Route, Link, Switch } from "react-router-dom"
import React from 'react'
import Home from './pages/Home'
import About from './pages/About'

function BasicExample() {
  return (
    <Router>
       <>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>

        <Switch>
          <Route path="/about">
            <About />
          </Route>
          <Route exact path="/">
            <Home />
          </Route>
        </Switch>
       </>
    </Router>
  )
}
```

另一點要注意的是，Router元件有一個要求，是只能有一個子元素在裡面，所以如果你定義了兩個以上的子元素，要記得先用div或React.Fragment(<>...</>)先包含起來，例如以上面的程式碼。

> 註：由於最上層元件要求一定要是Router元件，與一些其它有類似要求的套件例如Redux可能會有整合上的問題，這部份的解決方案請參考官方相關的文件。

用Route元件建立路由表

Route元件是使用來建立路由的對照表，這個元件的可設置屬性很多，可以應用於各種應用情況。通常位置都是在你的應用的最上層元件的Router元件的最下面一段JSX碼，請見上一節的範例。

Switch元件通常會包裹Route元件，因為路由表的使用方式是用對照符合(match)的，Switch會從上到下尋找最近的一個，也只會使用一個。以下面的兩個例子來說明有使用Switch元件和沒使用Switch元件的差異：

```
<Switch>
    <Route path="/about">
        <About />
    </Route>
    <Route path="/about/contact">
        <Contact />
    </Route>
</Switch>
```

上面這個例子，如果網址是 `/about` 則出現About元件的內容，但如果網址是 `/about/contact` 則出現Contact元件的內容。

```
<Route path="/about">
    <About />
</Route>
<Route path="/about/contact">
    <Contact />
</Route>
```

上面這個例子，如果，如果網址是 `/about` 則出現About元件的內容，但如果網址是 `/about/contact` ，則會出現About與Contact元件兩者的內容。

由上面兩個例子可知道，使用與不使用Switch元件會有不同的呈現結果，開發者可以視使用情況來決定。

另外，Router會依照目前輸入的網址去比對路由表中的設定路徑，在某些情況下必須要配合 exact 屬性，這是"精準的"意思，代表要求要精準地符合路徑再套用這個元件或css樣式等等。

以Link取代a

> 注意：使用 `a` 與 `href` 有可能會導致頁面刷新，元件會重新回恢初始狀態，導致應用程式的運作失常，所以請儘可能用Link元件

`a` 元素與 `href` 屬性是網站應用中的連結網頁用元素，Link元件是React Router中用來取代a的元件。

原本的連結應該是像下面這樣：

```
<a href="/about">關於我們</a>
```

改為Link元素後會像下面這樣：

```
<Link to="/about">關於我們</Link>
```

Link元件中除了可以像a元素中，使用id、title、className等屬性外，to屬性中可以使用物件的定義方式，來定義這個連結路徑的參數值、hash值、state值，例如以下的範例：

```
<Link
  to={{
    pathname: "/courses",
    search: "?sort=name",
    hash: "#the-hash",
    state: { fromDashboard: true }
  }}
/>
```

React Router三個重要的屬性值

使用React Router後會綁入使用的元件的props，以下三個相關的屬性值：

- match: 主要要得到不同路徑的參數值
- history: 對於瀏覽器書籤、前後移動的處理方法
- location: 目前所在的位置(物件值)

withRouter方法

> 註：函式型/類別型元件通用的方式

太深層的子元件得不到React Router的屬性值時使用。要綁入props三個屬性值(match, history, loaction)必定要加上。

範例:

```
import React, { useState, useEffect } from 'react'
import { Link, Switch, withRouter } from 'react-router-dom'

function Product(props) {
  console.log(props)
  return (
    <>
      <h1>Product</h1>
      <h3>{props.match.params.id}</h3>
    </>
  )
}

export default withRouter(Product)
```

Hooks(勾子)

只能用於函式型元件。v5.1以上才能使用，請參考Hooks文字內容。

useHistory

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

function HomeButton() {
  let history = useHistory();

  function handleClick() {
    history.push("/home");
  }

  return (
    <button type="button" onClick={handleClick}>
      Go home
    </button>
  );
}
```

useLocation

```
import React from "react";
import ReactDOM from "react-dom";
import {
  BrowserRouter as Router,
  Switch,
  useLocation
} from "react-router-dom";

function usePageViews() {
  let location = useLocation();
  React.useEffect(() => {
    ga.send(["pageview", location.pathname]);
  }, [location]);
```

```
}

function App() {
  usePageViews();
  return <Switch>...</Switch>;
}

ReactDOM.render(
  <Router>
    <App />
  </Router>,
  node
);
```

useParams

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

function BlogPost() {
  let { slug } = useParams();
  return <div>Now showing post {slug}</div>;
}

ReactDOM.render(
  <Router>
    <Switch>
      <Route exact path="/">
        <HomePage />
      </Route>
      <Route path="/blog/:slug">
        <BlogPost />
      </Route>
    </Switch>
  </Router>,
  node
);
```

useRouteMatch

用於匹配路由之用，請參考useRouteMatch

React Bootstrap導覽選單項目點亮(active)議題

但如果是使用在選單項目的連結，因為會有active(被點按到時的特定css)，會改用 `NavLink` 元件，這個元件是特別針對像選單項目這種導覽所設計的，多出了幾個點按到時的特別屬性，例如：

- activeClassName：被點按進入套用的css類別
- activeStyle：被點按進入套用的css樣式
- isActive：決定被點按與否的函式

```
import {
  Navbar,
  Nav,
  Form,
  FormControl,
  Button,
  NavDropdown,
} from 'react-bootstrap'
// 選單連結要使用NavLink取代Link
import { NavLink } from 'react-router-dom'
```

```
//...

 <Nav className="mr-auto">
        {/* 把Nav.Link作為NavLink來使用 */}
        {/* 一定要加上exact，不然首頁會一直點亮(active) */}
        <Nav.Link as={NavLink} to="/" exact>
          首頁
        </Nav.Link>
        <Nav.Link as={NavLink} to="/todo">
          待辦事項
        </Nav.Link>
        <Nav.Link as={NavLink} to="/product">
          產品
        </Nav.Link>
      </Nav>
```

參考：舊語法(v4)

```
import { BrowserRouter as Router, Route, Link, Switch } from "react-router-dom"
import React from 'react'
import Home from './pages/Home'
import About from './pages/About'

function BasicExample() {
  return (
    <Router>
       <>
        <Link to="/">Home</Link>
        <Link to="/about">About</Link>

        <Switch>
            <Route exact path="/" component={Home} />
            <Route path="/about" component={About} />
        </Switch>
       </>
    </Router>
  )
}
```

參考：靜態路由表

另一種針對複雜的、具有規模的應用的路由表設定方式，是先使用一個物件陣列先定義好，然後再用迴圈或 map 方法輸出到同一位置，範例如下：

```
const routes = [
  {
    path: "/sandwiches",
    component: Sandwiches
  },
  {
    path: "/tacos",
    component: Tacos,
    routes: [
      {
        path: "/tacos/bus",
        component: Bus
      },
      {
        path: "/tacos/cart",
        component: Cart
      }
    ]
  }
];
```

要使用上面這個預先定義的路由表，可以自己撰寫一個元件來套用，或是使用react-router-config這個靜態路由的工具模組來協助套用。