React Router DOM V6

React router dom V6 xuất hiện sử dụng các tính năng tốt nhất từ các phiên bản trước đã kết thúc một thập kỉ định tuyến phía client, nó tương thích với react từ 16.8 trở lên

Cài đặt thư viện

```
npm install react-router-dom@6
```

Cấu hình Routes

Import component BrowserRouter và bọc vào Component App

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

```
<BrowserRouter>
    <App />
</BrowserRouter>
```

Khai báo các path tương ứng với các Component, sử dụng Component Routes và Route

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

```
<Routes>
      <Route path='/' element={<Dashboard />} />
      <Route path='/orders' element={<Orders />} />
      <Route path='/products' element={<Products />} />
      <Route path='/customers' element={<Customers />} />
</Routes>
```

Thêm liên kết bằng cách dùng Component Link

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

```
<ul>
      <li>
        <Link to='/' className='nav-link'>Dashboard</Link>
      </li>
      <li>
        <Link to='/orders' className='nav-link'>Orders</Link>
      </li>
      <li>
        <Link to='/products' className='nav-link'>Products</Link>
      </li>
      <li>
        <Link to='/customers' className='nav-link'>Customers</Link>
      </li>
</ul>
```

Active Link

Làm thế nào để biết được liên kết nào đang được xem, thường sẽ dùng class **active**, **current** hoặc css inline

Active link với tên class là active

Nếu class tên là active thì easy rồi chúng ta chỉ việc thay Link thành **NavLink**

```
<NavLink className='nav-link' to='/orders'>Orders</NavLink>

// html
<a aria-current="page" class="nav-link active" href="/orders">Orders</a>
```

Active link với tên class khác

Nếu chúng ta muốn tên class khi được active khác đi như activated hay current-page thì viết kiểu function như này

```
import { NavLink } from 'react-router-dom'

const Sidebar = () => {
  const navLinkClass = ({ isActive }) => {
    return isActive ? 'nav-link activated' : 'nav-link'
  }
  return (
    <ul>
```

```
      <li>
        <NavLink to='/' className={navLinkClass}>Dashboard</NavLink>
      </li>
    </ul>
  )
}
```

Active link với style

Nếu không muốn trạng thái active viết theo kiểu class mà viết kiểu style thì làm như này

```
import { NavLink } from 'react-router-dom'

const Sidebar = () => {
  const navLinkStyle = ({ isActive }) => ({
    color: isActive ? '#fff' : '',
    backgroundColor: isActive ? '#0d6efd' : ''
  })

  return (
    <ul>
      <li>
        <NavLink to='/' className='nav-link' style={navLinkStyle}>
          Dashboard
        </NavLink>
      </li>
    </ul>
  )
}
```

Custom active link

Vì lý do nào đó mà chúng ta không thể đặt class active hay style active vào chính thẻ a mà phải đặt ở thẻ cha của nó như thẻ li chẳng hạn

```
<li>
  <a href="/" class="active">Dashboard</a>
</li>

// chuyển sang như này
<li class="active">
  <a href="/">Dashboard</a>
</li>
```

Sau đây mình hướng dẫn tạo **Custom Active Link** sử dụng **useMatch** và **useResolvedPath** trong react router

```
import { Link, useMatch, useResolvedPath } from 'react-router-dom'

const Sidebar = () => {
  const CustomLink = ({ children, to, ...props }) => {
    const resolved = useResolvedPath(to)
    const match = useMatch({ path: resolved.pathname, end: true })
    return (
      <li className={match ? 'active' : ''}>
        <Link to={to} {...props}>
          {children}
        </Link>
      </li>
    )
  }

  return (
    <ul>
      <CustomLink to='/'>Dashboard</CustomLink>
      <CustomLink to='/orders'>Orders</CustomLink>
      <CustomLink to='/products'>Products</CustomLink>
      <CustomLink to='/customers'>Customers</CustomLink>
    </ul>
  )
}

export default Sidebar
```

Tự động chuyển trang với Navigate

useNavigate với 1 tham số

Đoạn code dưới khi click vào button thì sẽ chuyển đến trang `orders`

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

const Dashboard = () => {
  const navigate = useNavigate()

  return (
    <button onClick={() => navigate('orders')}>
      Go to Orders
    </button>
  )
}
```

useNavigate với history

Trường hợp bạn muốn sử dụng `go`, `goBack`, `goForward` trong lịch sử trình duyệt.

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

export default function App() {
  const navigate = useNavigate()

  return (
    <>
      <button onClick={() => navigate(-2)}>Go 2 pages back</button>
      <button onClick={() => navigate(-1)}>Go back</button>
      <button onClick={() => navigate(1)}>Go forward</button>
      <button onClick={() => navigate(2)}>Go 2 pages forward</button>
    </>
  )
}
```

useNavigate với replace true

Sử dụng tham số thứ hai của `navigate` để chỉ thay đổi URL chứ không muốn URL đó lưu lại trong lịch sử trình duyệt. Kiểu như tại trang A đi tới trang B, tại trang B chúng ta click back trên trình duyệt thì sẽ không quay trở lại trang A nữa.

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

const Dashboard = () => {
  const navigate = useNavigate()
  return (
    <button onClick={() => navigate('orders', { replace: true })}>
      Go to Orders
    </button>
  )
}
```

useNavigate với passing data

- Tình huống đặt ra là tại `componentA` khi navigate chuyển trang sang `componentB` sẽ kèm thêm một id là 6996
- Tại `componentB` sẽ nhận được data thông qua `useLocation`

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

const componentA = () => {
  const navigate = useNavigate()
  return (
    <button onClick={() => navigate("orders", { state: { id: "6996" } })}>
      Send ID to componentB
    </button>
```

```
  )
}
```

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

const ComponentB = () => {
  const location = useLocation()
  return <h1>ID nhận từ componentA: {location.state?.id}</h1>
}
```

Component Navigate

Sử dụng component Navigate để chuyển trang trong phần return ở function

```
import { Navigate } from 'react-router-dom'
import Dashboard from './Dashboard'

const PrivateRoutes = () => {
  const isAuth = true

  return isAuth ? <Dashboard /> : <Navigate to='/login' />
}
```

Not Found Routes - 404

```
<Routes>
  <Route path='/' element={<Dashboard />} />
  <Route path='*' element={<NotFound />} />
</Routes>
```

Nested Routes

Nested Routes để lồng component con vào trong component cha.

```
<Routes>
  <Route path='/' element={<Dashboard />} />
  <Route path='/products' element={<Products />}>
    <Route path='laptop' element={<Laptop />} />
    <Route path='desktop' element={<Desktop />} />
  </Route>
</Routes>
```

```
import { Outlet } from 'react-router-dom'

const Products = () => (
  <>
    <h1>Products</h1>
    <Outlet />
  </>
)
```

- Nếu truy cập vào **localhost:3000/products** sẽ load component `Products`
- Nếu truy cập vào **localhost:3000/products/laptop** sẽ load `Products` bên trong có component `Laptop`
- Nếu truy cập vào **localhost:3000/products/desktop** sẽ load `Products` bên trong có component `Desktop`
- Lưu ý tại component cha là `Products` ta sử dụng Outlet để xác định vị trí hiển thị component con khi route trùng khớp

Index routes

- Như ví dụ ở trên thì khi truy cập vào **localhost:3000/products** sẽ load component `Products` và không hiển thị conponent con nào cả
- Điều chúng ta mong muốn là vẫn URL như vậy nhưng hiển thị một conponent con nào đó ở ngay component cha
- Để làm được điều này ta sử dụng **index** và truyền conponent con muốn được hiển thị

Ví dụ: nếu truy cập vào `localhost:3000/products` sẽ load component `Products` bên trong có component `BestSeller`

```
<Routes>
  <Route path='/products' element={<Products />}>
    <Route index element={<BestSeller />} />
    <Route path='laptop' element={<Laptop />} />
    <Route path='desktop' element={<Desktop />} />
  </Route>
</Routes>
```

Dynamic routes

```
<Routes>
  <Route path='/' element={<Dashboard />} />
  <Route path='/courses' element={<Courses />} />
  <Route path='/courses/:courseId' element={<CourseDetail />} />
</Routes>
```

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

const CustomerDetail = () => {
  const params = useParams()
  return <h2>Chi tiết khóa học: {params.courseId}</h2>
}
```

- Dynamic routes giúp chúng ta giải quyết các routes động, routes thay đổi theo một cấu trúc đã được định nghĩa sẵn.
- Ví dụ ta có URL theo kiểu `courses/html`, `courses/css`, `courses/javascript` thì có thể mô hình hóa nó thành `courses/:courseId`
- Tại `CourseDetail` ta sử dụng hook useParams để lấy tham số trên URL.
- Nếu ta định nghĩa route là `courses/:courseId` thì lúc lấy giá trị cũng lấy đúng tên là `params.courseId`

Dynamic routes với các route cố định

```
<Routes>
  <Route path='/courses' element={<Courses />} />
  <Route path='/courses/:courseId' element={<CourseDetail />} />
  <Route path='/courses/add-course' element={<AddCourse />} />
  <Route path='/courses/edit-course' element={<EditCourse />} />
</Routes>
```

- Khi sử dụng Dynamic routes ta có URL theo kiểu `courses/html`, `courses/css`, `courses/javascript` thì đã mô hình hóa nó thành `courses/:courseId`
- Trong một vài trường hợp route cố định như `courses/add-course` hay `courses/edit-course` thì ta sẽ khai báo route để bắt các trường hợp này

Multiple dynamic routes

```
<Routes>
  <Route path='/courses' element={<Courses />} />
  <Route path='/courses/:courseType/' element={<CourseType />} />
  <Route path='/courses/:courseType/:courseId' element={<CourseDetail />} />
</Routes>
```

Đoạn code trên áp dụng với trường hợp nhiều URL Parameters

Search params

```
import { useSearchParams } from 'react-router-dom'

let [searchParams, setSearchParams] = useSearchParams()
```

Hook useSearchParams được sử dụng để đọc và sửa đổi query string trên URL, hook này trả về một mảng gồm hai giá trị: tham số tìm kiếm và một hàm để thay đổi.

Để thay đổi searchParams

```
<button onClick={() => setSearchParams({product: 'laptop'})}>
  Laptop
</button>
<button onClick={() => setSearchParams({product: 'laptop', stock: 'in-stock'})}>
  Còn hàng
</button>
<button onClick={() => setSearchParams({})}>
  Clear
</button>
```

- Khi click vào button Laptop => `https://example.com/?product=laptop`
- Khi click vào button Còn hàng => `https://example.com/?product=laptop&stock=in-stock`
- Khi click vào button Clear => `https://example.com`

Để đọc searchParams

```
const productType = searchParams.get('product'); // laptop
const stock = searchParams.get('stock'); // in-stock
```

Protected routes

Giả sử ứng dụng của chúng ta có hai phần: **public** và **private**.

- Phần public thì ai cũng có thể truy cập được như trang chủ, tin tức...
- Phần private thì phải đăng nhập vào mới xem được như trang cá nhân, trang account...

Về hành vi đối với người dùng

- Nếu đăng nhập rồi thì truy cập được tất cả các link của public hay private...
- Nếu chưa thì chỉ truy cập được các trang public, nếu người dùng vẫn cố truy cập vào các trang private thì ta điều hướng họ sang trang login hay trang nào thì tùy nghiệp vụ.

Bước 1: Nhóm các route cần bảo vệ vào trong PrivateRoutes

```
<Routes>
  <Route path='/' element={<Home />} />
  <Route path='/news' element={<News />} />
  <Route element={<PrivateRoutes />}>
    <Route path='/personal' element={<Personal />} />
    <Route path='/account' element={<Account />} />
  </Route>
</Routes>
```

Bước 2: Trong PrivateRoutes sẽ kiểm tra login hay chưa để xử lý

```
import { Navigate, Outlet } from 'react-router-dom'

const PrivateRoutes = () => {
  const isAuth = true

  return isAuth ? <Outlet /> : <Navigate to='/login' />
}

export default PrivateRoutes
```

Lazy loading

Lazy Loading là một kỹ thuật để tối ưu hoá ứng dụng, khi chuyển trang ứng dụng sẽ chỉ load những component cần thiết, điều này giúp trang web chuyển động mượt mà, nhanh chóng, tăng trải nghiệm người dùng.

```
import React from 'react'
import { Routes, Route } from 'react-router-dom'
import Dashboard from './components/Dashboard'
const LazyAbout = React.lazy(() => import('./components/About'))

export default function App() {
  return (
    <Routes>
      <Route path='/' element={<Dashboard />} />
      <Route
        path='/about'
        element={
          <React.Suspense fallback={<div>Loading...</div>}>
            <LazyAbout />
          </React.Suspense>
        }
      />
    </Routes>
  )
}
```

Để áp dụng kĩ thuật **Lazy Loading** chúng ta cần:

- `Dynamic import trong React`
- `React.lazy()`
- `React.Suspense()`

Route Objects

Hook useRoutes cho phép bạn xác định các tuyến đường dưới dạng object thuần javascript thay cho `<Routes>` và `<Route>`. Đây chỉ là một phong cách tùy chọn cho những người không muốn sử dụng JSX.

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

export default function App() {
  return (
    <Routes>
      <Route path='/' element={<Dashboard />} />
      <Route path='/products' element={<Products />}>
        <Route index element={<BestSeller />} />
        <Route path='laptop' element={<Laptop />} />
        <Route path='desktop' element={<Desktop />} />
```

```
      </Route>
      <Route path='*' element={<NoMatch />} />
    </Routes>
  )
}
```

```
import { useRoutes } from 'react-router-dom'

export default function App() {
  let routes = [
    { path: '/', element: <Dashboard /> },
    {
      path: '/products',
      element: <Products />,
      children: [
        { index: true, element: <BestSeller /> },
        { path: 'laptop', element: <Laptop /> },
        { path: 'desktop', element: <Desktop /> }
      ]
    },
    { path: '*', element: <NoMatch /> }
  ]

  let element = useRoutes(routes)

  return element
}
```

Scroll to top

Trong thực tế chúng ta hay gặp phải trường hợp đang ở trang A và cuộn đến cuối trang này, khi chuyển sang trang B thì thanh cuộn vẫn ở dưới gây chút khó chịu.

Để fix vấn đề này chúng ta có thể tạo 1 component `scrollToTop` và thêm nó vào trong đoạn `config route`

```
<div>
  <ScrollToTop />
  <Routes>
    <Route path='/' element={<Dashboard />} />
    <Route path='/orders' element={<Orders />} />
    <Route path='/products' element={<Products />} />
  </Routes>
</div>
```

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

```
const ScrollToTop = () => {
  const { pathname } = useLocation()

  useEffect(() => {
    window && window.scrollTo(0, 0)
  }, [pathname])

  return null
}

export default ScrollToTop
```

Can't setState on unmounted component

- Khi chúng ta đang ở trang Home và lấy dữ liệu từ API
- Dữ liệu chưa GET xong thì người dùng chuyển sang trang About
- Component Home lúc đó đã bị unmount
- Sau đó dữ liệu từ API được trả về và setState()
- Component Home không còn để update

```
async function handleSubmit() {
  setPending(true)
  await post('/someapi') // component might unmount while we're waiting
  setPending(false)
}
```

```
let isMountedRef = useRef(false)
useEffect(() => {
  isMountedRef.current = true
  return () => {
    isMountedRef.current = false
  }
}, [])

async function handleSubmit() {
  setPending(true)
  await post('/someapi')
  if (isMountedRef.current) {
    setPending(false)
  }
}
```