第 5 章 - Link 元件 & useRouter

Link 元件

<Link>元件擴充了a標記，提供了客戶端的瀏覽功能，以及預先載入(Prefetching)功能，意即在訪問前先預先在背景中預先載入路由的資料，由客戶端進行快取。

> 註: Link 元件的預先載入(Prefetching)功能雖然是預設啟動的，但只有正式營運時才會啟動。

Link 元件的使用

導入next/link模組，使用 Link 元件取代 a 標記:

```
import Link from 'next/link'

export default function Page() {
  return <Link href="/dashboard">Dashboard</Link>
}
```

href 屬性是必要的，也可以使用物件類型，例如:

```
// Navigate to /about?name=test
<Link
  href={{
    pathname: '/about',
    query: { name: 'test' },
  }}
>
  About
</Link>
```

關於 href 的屬性值，如果使用物件值，是標準的 URL 物件，可以參考Node.js 的 URL 模組文件

屬性

- href: **必要屬性**，可以是字串(String)或物件類型(Object)，
- replace: 預設是false，布林類型(Boolean)
- prefetch: 預設為true，布林類型(Boolean)
- scroll: 預設為true，到下一頁連結後自動捲動到最上面，除非有必要才會設為false
- passHref: 預設是false，可以把 href 的連結功能傳遞給包住的元件中，通常用來整合其它元件用。使用時直接加上即可。
- legacyBehavior: 預設是false，也是整合包裹住的元件用。

範例:

```
import Link from 'next/link'
import styled from 'styled-components'

// 這裡是用styled-components來建立<a>標記的自訂元件
const RedLink = styled.a`
  color: red;
`

export default function NavLink({ href, name }) {
  return (
    <Link href={href} passHref legacyBehavior>
      <RedLink>{name}</RedLink>
    </Link>
  )
}
```

> 註: 更資詳細資訊參考官網文件

useRouter 勾子

`useRouter` 提供了另一種導覽的需求，可以在程式碼的邏輯中進行導覽控制。`router.push(href)`相當於上述的 Link 元件連結到一特定連結過去。

範例:

```
import { useRouter } from 'next/router'

function ActiveLink({ children, href }) {
  const router = useRouter()

  const style = {
    marginRight: 10,
    color: router.asPath === href ? 'red' : 'black',
  }

  const handleClick = (e) => {
    e.preventDefault()
    router.push(href)
  }

  return (
    <a href={href} onClick={handleClick} style={style}>
      {children}
    </a>
  )
}
```

router 物件主要由下列兩個屬性組成:

- `pathname`: 字串(String)值 ，目前要導覽去的路由檔案
- `query`: 物件(Object)值，查詢字串剖析成的物件值，預設是`{}`

通常會應用的方法如下:

- `router.push(url, as, options)`: 如上面的範例
- `router.back()`: 回上頁，與`window.history.back()`相同
- `router.reload()`: 重新整理此頁面，與`window.location.reload()`相同

`query`所得到的物件值，會使用在動態路由中的自動轉化為此物件的屬性值，實際應用請參考"路由"章節。

> 註: 更多詳細的選項或方法屬性參考官網文件

其它進階應用

保持查詢字串的連結

保持下一頁，或是連結後繼續有同樣的查詢字串(query string)，要用物件參數的方式來傳遞，參考來源:

擴充 Link 元件:

```
// components/retain-query-link.js
import Link from 'next/link'
import { useRouter } from 'next/router'

export default function RetainQueryLink({ href, ...props }) {
  // 1. use useRouter hook to get access to the current query params
  const router = useRouter()

  // 2. get the pathname
  const pathname = typeof href === 'object' ? href.pathname : href

  // 3. get the query from props
  const query =
    typeof href === 'object' && typeof href.query === 'object' ?
href.query : {}

  return (
    <Link
      {...props}
      href={{
        pathname: pathname,
        // combine router.query and query props
        query: {
          ...router.query,
          ...query,
        },
      }}
    />
  )
}
```

或

```
const router = useRouter()
router.push({
  pathname: '/signin',
  query: router.query,
})
```

使用的範例如下:

```
// pages/users.js
import RetainQueryLink from '@/components/common/retain-query-link'
import Link from 'next/link'
import { useRouter } from 'next/router'
import { useEffect, useState } from 'react'

export default function UsersPage() {
  const pages = [1, 2, 3, 4, 5]
  const [pathNow, setPathNow] = useState('')

  const router = useRouter()

  console.log(router.query)

  useEffect(() => {
    setPathNow(router.asPath)
  }, [router.asPath])

  return (
    <div>
      Users
      <div>
        <span>Pagination</span>
        <pre>Location:{pathNow}</pre>
        <hr />
        <div>Retain Query Link</div>
        {pages.map((page) => {
          return (
            <RetainQueryLink
              key={page}
              href={{
                query: { page },
              }}
            >
              {` ${page} `}
            </RetainQueryLink>
          )
        })}
        <hr />
        <div>Link</div>
```

```
          {pages.map((page) => {
            return (
              <Link
                key={page}
                href={{
                  query: { page },
                }}
              >
                {` ${page} `}
              </Link>
            )
          })}
        </div>
      </div>
    )
  }
```