【Next.js】Link 元件

當然可以！

□ Next.js 中的 `<Link>` 元件完整說明（針對 App Router）

□ 什麼是 `<Link>`？

`<Link>` 是 Next.js 提供的**內部導頁元件**，用來在不同頁面之間切換（而不會重新載入整個網頁）。

它的作用類似於 HTML 的 `<a>` 標籤，但它是：

> □ 加上 **預先載入（prefetch）**
> □ 使用 **Client-side navigation（客戶端跳轉）**
> □ 避免整頁重新載入，更快更流暢

□ 基本用法

```
import Link from 'next/link'

export default function HomePage() {
  return (
    <div>
      <h1>首頁</h1>
      <Link href="/about">前往 About 頁面</Link>
    </div>
  );
}
```

□ 對應效果：

```
<a href="/about">前往 About 頁面</a>
```

但點擊時會走 Next.js 的路由系統，不會整頁 reload。

□ 為什麼不用 `<a href="/about">`？

因為 `<a>` 是瀏覽器原生跳轉：

- □ 整頁重新載入
- □ 丟失 React 狀態
- □ 破壞 Single Page App 流暢性

而 `<Link>` 則使用 **client-side routing**：

- ⚡ 更快，不會重載頁面
- □ 保留 layout 與資料
- □ 支援 App Router 的預先載入

□ 支援的 props（常見）

Prop	說明
`href`	□ 必填，跳轉目標
`prefetch`	是否預先載入（預設 `true`）
`replace`	不加入瀏覽器歷史紀錄（像 `location.replace`）
`scroll`	是否自動捲動到頂部（預設 `true`）
`locale`	多語系路由時指定語言

□ 帶參數用法

```
<Link href="/product/123">商品詳情</Link>
```

對應 `app/product/[id]/page.tsx`

□ 包含其他元素（像按鈕）

```
<Link href="/about">
  <button>了解更多</button>
</Link>
```

□ 搭配動態路由（含物件參數）

```
<Link href={{ pathname: '/product/[id]', query: { id: '123' } }}>
  查看商品
</Link>
```

□ 注意事項

規則	說明
`<Link>` 只能用於內部路由	外部網站請用 `<a href="https://...">`
不需要手動加 `as`	App Router 不需要像舊版 Pages Router 那樣加 `as`
可以搭配 `usePathname()`	用來比對當前路徑，製作「選單高亮」

□ 實務建議

使用場景	建議寫法
切換內部頁面	用 `<Link href="/about">`
導向外部網站	用 `<a href="https://xx">`
高亮當前路徑	搭配 `usePathname()`

□ 小結

功能	`<Link>`
路由切換	□ Client-side navigation
預先載入	□ 支援 prefetch
整頁 reload	□ 不會 reload，保持 SPA 流暢性
搭配 layout	□ Layout 不會重新渲染

在 Next.js 中，**Active Link（目前頁面導覽高亮）** 是一個常見的 UI 需求。當使用者正在某個頁面時，對應的導覽列連結（nav bar）應該加上「目前頁面樣式」來提示使用者。

□ Active Link 是什麼？

就是讓當前頁面的導覽項目加上特殊樣式，例如：

```
<a href="/about" class="active">About</a>
```

讓使用者知道「我現在在這個頁面」。

□ Next.js 怎麼做 Active Link？

Next.js **並不自帶** active class 功能，需要我們手動處理。但 App Router 搭配 `usePathname()` **非常方便。**

□ 使用 `usePathname()` 判斷當前路徑

□ 範例：`ActiveLink.tsx` 元件

```
// components/ActiveLink.tsx
"use client";

import Link from "next/link";
import { usePathname } from "next/navigation";
import clsx from "clsx"; // optional, 幫助管理 class

type Props = {
  href: string;
  children: React.ReactNode;
};

export default function ActiveLink({ href, children }: Props) {
  const pathname = usePathname();
  const isActive = pathname === href;

  return (
    <Link
      href={href}
      className={clsx("px-4 py-2", {
        "text-blue-600 font-bold underline": isActive,
        "text-gray-600": !isActive,
      })}
    >
      {children}
    </Link>
```

```
  );
}
```

使用方式

```
// app/layout.tsx 或其他 Nav 元件中
import ActiveLink from "@/components/ActiveLink";

export default function NavBar() {
  return (
    <nav className="flex gap-4">
      <ActiveLink href="/">Home</ActiveLink>
      <ActiveLink href="/about">About</ActiveLink>
      <ActiveLink href="/products">Products</ActiveLink>
    </nav>
  );
}
```

小技巧

- 路徑可以使用 `startsWith()` 判斷「巢狀頁面」也高亮
  ```
  const isActive = pathname.startsWith(href)
  ```
- 可以抽出 `activeClassName` 與 `className` 讓元件更彈性

顯示效果

當你瀏覽 `/products` 頁面時：

```
<a href="/products" class="text-blue-600 font-bold underline">
```

若你使用 Tailwind，可用 `clsx` 管理 class：

```
pnpm add clsx
# or
npm install clsx
```

小結

工具	用途
`usePathname()`	取得目前 URL 路徑
`clsx`	根據條件切換 class
自訂 `ActiveLink`	建立可重用的 active 判斷元件

修訂版本 #2
★由 treeman 建立於 25 ﷺ 2025 16:48:18
由 treeman 更新於 31 ﷺ 2025 17:51:11