라우팅

클라이언트 측 라우팅과 서버 측 라우팅 비교

서버 측 라우팅은 사용자가 방문하는 URL 경로에 기반하여 서버가 응답을 보내는 것을 의미합니다. 전통적인 서버 렌더링 웹 앱에서 링크를 클릭하면, 브라우저는 서버로부터 HTML 응답을 받고 새 HTML로 전체 페이지를 다시 로드합니다.

그러나 싱글 페이지 애플리케이션(SPA)에서는 클라이언트 측 자바스크립트가 탐색을 가로채고 새 데이터를 동적으로 가져와 전체 페이지를 다시 로드하지 않고 현재 페이지를 업데이트할 수 있습니다. 이는 일반적으로 사용자가 오랜 시간 동안 많은 상호 작용을 수행해야 하는 실제 "애플리케이션"과 유사한 사용 사례에서 보다 빠른 사용자 경험을 제공합니다.

이러한 SPA에서 '라우팅'은 브라우저의 클라이언트 측에서 수행됩니다. 클라이언트 측 라우터는 히스토리 **API** 또는 **hashchange** 이벤트와 같은 브라우저 API를 사용하여 애플리케이션의 렌더링된 View를 관리할 책임이 있습니다.

공식 라우터

대부분의 싱글 페이지 앱(SPA)의 경우 공식적으로 지원되는 **vue-router** 라이브러리를 사용하는 것이 좋습니다. 자세한 내용은 vue-router의 문서를 참조하십시오.

간단한 라우팅 구성하기

매우 간단한 라우팅만 필요하여 모든 기능을 갖춘 라우터 라이브러리를 포함하지 않으려면 동적인 컴포넌트를 사용하고, 브라우저 **hashchange** 이벤트를 수신하거나 **History API**를 사용하여 현재 컴포넌트 상태를 업데이트할 수 있습니다.

다음은 기본적인 예입니다:

vue

```vue
<script setup>
import { ref, computed } from 'vue'
import Home from './Home.vue'
import About from './About.vue'
import NotFound from './NotFound.vue'

const routes = {
  '/': Home,
  '/about': About
}

const currentPath = ref(window.location.hash)

window.addEventListener('hashchange', () => {
  currentPath.value = window.location.hash
})

const currentView = computed(() => {
  return routes[currentPath.value.slice(1) || '/'] || NotFound
})
</script>

<template>
  <a href="#/">Home</a> |
  <a href="#/about">About</a> |
  <a href="#/non-existent-path">잘못된 링크</a>
```

```
    <component :is="currentView" />
  </template>
```

온라인 연습장으로 실행하기