텔레포트

<Teleport> 는 컴포넌트 템플릿의 일부를 해당 컴포넌트의 DOM 계층 외부의 DOM 노드로 "이동"할 수 있게 해주는 빌트인 컴포넌트입니다.

기본 사용법

때때로 우리는 다음과 같은 시나리오에 직면할 수 있습니다. 컴포넌트 템플릿의 일부가 논리적으로 여기에 속하지만, 시각적인 관점에서 Vue 앱 외부의 다른 DOM 어딘가에 표시되어야 합니다.

가장 일반적인 예는 전체 화면 모달을 구현할 때입니다. 이상적으로는 모달의 버튼과 모달 자체가 동일한 컴포넌트 내에 있기를 원합니다. 둘 다 모달의 열기/닫기 상태와 관련이 있기 때문입니다. 그러나 이는 모달이 버튼과 함께 렌더링되고 앱의 DOM 계층 구조에 깊숙이 중첩되어 있음을 의미합니다. 이것은 CSS를 통해 모달을 배치할 때 몇 가지 까다로운 문제를 일으킬 수 있습니다.

다음 HTML 구조를 고려하십시오.

template

```
<div class="outer">
  <h3>Vue Teleport Example</h3>
  <div>
    <MyModal />
  </div>
</div>
```

다음은 <MyModal> 의 구현입니다:

vue

```
<script setup>
import { ref } from 'vue'

const open = ref(false)
</script>

<template>
  <button @click="open = true">모달 열기</button>

  <div v-if="open" class="modal">
    <p>짜자잔~ 모달입니다!</p>
    <button @click="open = false">닫기</button>
  </div>
</template>

<style scoped>
.modal {
  position: fixed;
  z-index: 999;
  top: 20%;
  left: 50%;
  width: 300px;
  margin-left: -150px;
}
</style>
```

컴포넌트에는 모달의 열기를 트리거하는 <button> 과 .modal 클래스가 있는 가 포함되어 있으며, 여기에는 모달의 컨텐츠와 닫기 버튼이 포함됩니다.

초기 HTML 구조 내에서 이 컴포넌트를 사용할 때, 여러 가지 잠재적인 문제가 있습니다:

position: fixed 는 부모 엘리먼트에 transform , perspective 또는 filter 속성이 설정되지 않은 경우에만 뷰포트를 기준으로 엘리먼트를 배치합니다. 예를 들어 CSS 트랜지션으로 조상 에 애니메이션을 적용하려는 경우 모달 레이아웃이 깨집니다!

모달의 z-index 는 모달을 포함하는 엘리먼트에 의해 제한됩니다. 와 겹치고 z-index 가 더 높은 또 다른 엘리먼트가 있으면 모달을 덮을 것입니다.

<Teleport> 는 중첩된 DOM 구조에서 벗어날 수 있도록 하여 이러한 문제를 해결할 수 있는 깔끔한 방법을 제공합니다. <Teleport> 를 사용하도록 <MyModal> 을 수정해 보겠습니다:

template

```
<button @click="open = true">모달 열기</button>

<Teleport to="body">
  <div v-if="open" class="modal">
    <p>짜자잔~ 모달입니다!</p>
    <button @click="open = false">닫기</button>
  </div>
</Teleport>
```

<Teleport> 의 to 대상은 CSS 셀렉터 문자열 또는 실제 DOM 노드여야 합니다. 여기서 우리는 Vue에게 이 템플릿 조각을 **body** 태그로 **이동**하도록 지시합니다.

아래 버튼을 클릭하고 브라우저의 개발자도구를 통해 <body> 태그를 확인해 봅시다:

모달 열기

<Teleport> 를 **<Transition>** 과 결합하여 애니메이션 모달을 만들 수 있습니다. 예제를 참고하세요.

TIP

<Teleport> 컴포넌트가 to 대상으로 이동하여 마운트 되기 전, 대상은 이미 DOM에 있어야 합니다. 이상적으로는 전체 Vue 앱 외부의 엘리먼트여야 합니다. Vue에서 렌더링한 다른 엘리먼트를 대상으로 하는 경우, <Teleport> 전에 해당 엘리먼트가 마운트되었는지 확인해야 합니다.

컴포넌트와 함께 사용하기

<Teleport> 는 렌더링된 DOM 구조만 변경하며 컴포넌트의 논리적 계층 구조에는 영향을 주지 않습니다. 즉, <Teleport> 에 컴포넌트가 포함되어 있으면, 해당 컴포넌트는 논리적으로 <Teleport> 를 포함하는 부모 컴포넌트의 자식으로 유지됩니다. Props 전달() 및 이벤트 발신(emit)은 계속 동일한 방식으로 작동합니다.

이것은 부모 컴포넌트로부터 주입되어 예상대로 작동함을 의미합니다. 또한 자식 컴포넌트는 실제 컨텐츠가 이동한 위치에 배치되지만, Vue 개발자도구에서는 부모 컴포넌트 내부에 위치합니다.

텔레포트 비활성화

경우에 따라 <Teleport> 를 조건부로 비활성화 할 수 있습니다. 예를 들어 컴포넌트를 데스크톱용 오버레이로 렌더링하지만 모바일에서는 인라인으로 렌더링할 수 있습니다. <Teleport> 는 동적으로 토글할 수 있는 disabled prop을 지원합니다:

template

```
<Teleport :disabled="isMobile">
  ...
</Teleport>
```

미디어 쿼리 변경을 감지하여 isMobile 상태를 동적으로 업데이트할 수 있습니다.

같은 대상에 여러 텔레포트 사용

일반적인 사용 사례는 여러 인스턴스가 동시에 활성화될 가능성이 있는 재사용 가능한 <Modal> 컴포넌트입니다. 이러한 종류의 시나리오에서는 여러 <Teleport> 컴포넌트의 컨텐츠를 동일한 대상 엘리먼트에 탑재할 수 있습니다. 순서는 단순히 추가한 순서대로 이므로, <Teleport> 컴포넌트 마운트는 대상 엘리먼트 내 이전 마운트 다음에 위치합니다.

다음과 같이 사용할 때:

template

```
<Teleport to="#modals">
  <div>A</div>
</Teleport>
<Teleport to="#modals">
  <div>B</div>
</Teleport>
```

렌더링된 결과는 다음과 같습니다:

html

```
<div id="modals">
  <div>A</div>
  <div>B</div>
</div>
```

관련 문서

- **<Teleport> API** 참고
- **SSR**에서 텔레포트 핸들링