텔레포트

Vue School의 무료 동영상 강의 보기

`<Teleport>` 는 컴포넌트의 템플릿 일부를 해당 컴포넌트의 DOM 계층 외부에 존재하는 DOM 노드로 "텔레포트"할 수 있게 해주는 내장 컴포넌트입니다.

기본 사용법

때때로 컴포넌트의 템플릿 일부가 논리적으로는 해당 컴포넌트에 속하지만, 시각적으로는 DOM의 다른 위치, 심지어 Vue 애플리케이션 외부에 표시되어야 할 때가 있습니다.

이의 가장 일반적인 예는 전체 화면 모달을 만들 때입니다. 이상적으로는 모달의 버튼과 모달 자체의 코드를 동일한 싱글 파일 컴포넌트 내에 작성하고 싶습니다. 둘 다 모달의 열림/닫힘 상태와 관련이 있기 때문입니다. 하지만 이렇게 하면 모달이 버튼과 함께 렌더링되어 애플리케이션의 DOM 계층에 깊이 중첩됩니다. 이는 CSS로 모달의 위치를 지정할 때 까다로운 문제를 일으킬 수 있습니다.

다음 HTML 구조를 살펴보세요.

template

```
<div class="outer">
  <h3>Vue 텔레포트 예제</h3>
  <div>
    <MyModal />
  </div>
</div>
```

그리고 `<MyModal>` 의 구현은 다음과 같습니다:

vue

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

```
</script>

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

  <div v-if="open" class="modal">
    <p>모달에서 인사합니다!</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>
```

이 컴포넌트에는 모달을 여는 `<button>` 과, 모달의 내용을 담고 스스로 닫을 수 있는 버튼이 있는 `.modal` 클래스를 가진 `<div>` 가 포함되어 있습니다.

이 컴포넌트를 초기 HTML 구조 내에서 사용할 때 다음과 같은 잠재적 문제가 있습니다:

`position: fixed` 는 조상 요소 중에 `transform` , `perspective` 또는 `filter` 속성이 설정되어 있지 않을 때만 뷰포트 기준으로 요소를 배치합니다. 예를 들어, 조상 `<div class="outer">` 에 CSS transform으로 애니메이션을 주려고 한다면, 모달 레이아웃이 깨질 수 있습니다!

모달의 `z-index` 는 포함하는 요소에 의해 제한됩니다. `<div class="outer">` 를 덮는 다른 요소가 더 높은 `z-index` 를 가지고 있다면, 그 요소가 모달을 가릴 수 있습니다.

`<Teleport>` 는 중첩된 DOM 구조에서 벗어날 수 있는 깔끔한 방법을 제공합니다. `<MyModal>` 을 `<Teleport>` 를 사용하도록 수정해봅시다:

template

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

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

`<Teleport>` 의 `to` 대상은 CSS 선택자 문자열 또는 실제 DOM 노드를 기대합니다. 여기서는 Vue에게 **"이 템플릿 조각을 `body` 태그로 텔레포트하라"**라고 말하는 셈입니다.

모달 열기

`<Teleport>` 와 `<Transition>` 을 결합하여 애니메이션 모달을 만들 수 있습니다 - 예제 보기.

> **TIP**
>
> 텔레포트의 `to` 대상은 `<Teleport>` 컴포넌트가 마운트될 때 이미 DOM에 존재해야 합니다. 이상적으로는 전체 Vue 애플리케이션 외부의 요소여야 합니다. 만약 Vue가 렌더링한 다른 요소를 대상으로 한다면, 해당 요소가 `<Teleport>` 보다 먼저 마운트되었는지 확인해야 합니다.

컴포넌트와 함께 사용하기

`<Teleport>` 는 렌더링된 DOM 구조만 변경할 뿐, 컴포넌트의 논리적 계층에는 영향을 주지 않습니다. 즉, `<Teleport>` 가 컴포넌트를 포함하고 있다면, 그 컴포넌트는 여전히 `<Teleport>` 를 포함한 부모 컴포넌트의 논리적 자식으로 남아 있습니다. props 전달과 이벤트 발생은 동일하게 동작합니다.

이것은 또한 부모 컴포넌트로부터의 주입이 정상적으로 동작하며, 자식 컴포넌트가 실제 내용이 이동된 위치가 아니라 Vue Devtools에서 부모 컴포넌트 아래에 중첩되어 표시됨을 의미합니다.

텔레포트 비활성화하기

경우에 따라 `<Teleport>` 를 조건부로 비활성화하고 싶을 수 있습니다. 예를 들어, 데스크톱에서는 오버레이로, 모바일에서는 인라인으로 컴포넌트를 렌더링하고 싶을 수 있습니다.
`<Teleport>` 는 동적으로 토글할 수 있는 `disabled` prop을 지원합니다:

template

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

이제 `isMobile` 값을 동적으로 업데이트할 수 있습니다.

동일한 대상에 여러 텔레포트 사용하기

일반적인 사용 사례는 재사용 가능한 `<Modal>` 컴포넌트로, 동시에 여러 인스턴스가 활성화될 수 있는 경우입니다. 이런 시나리오에서는 여러 `<Teleport>` 컴포넌트가 동일한 대상 요소에 콘텐츠를 마운트할 수 있습니다. 순서는 단순히 append 방식으로, 나중에 마운트된 것이 앞선 것 뒤에 위치하지만 모두 대상 요소 내에 있게 됩니다.

다음과 같이 사용하면:

template

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

렌더링 결과는 다음과 같습니다:

html

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

지연된 텔레포트 3.5+

Vue 3.5 이상에서는 `defer` prop을 사용하여 텔레포트의 대상 해석을 애플리케이션의 다른 부분이 마운트될 때까지 지연할 수 있습니다. 이를 통해 텔레포트가 Vue에 의해 렌더링되지만 컴포넌트 트리의 더 뒤쪽에 위치한 컨테이너 요소를 대상으로 할 수 있습니다:

template

```
<Teleport defer to="#late-div">...</Teleport>

<!-- 템플릿의 더 뒤쪽 어딘가에 -->
<div id="late-div"></div>
```

대상 요소는 텔레포트와 동일한 마운트/업데이트 틱 내에 렌더링되어야 한다는 점에 유의하세요. 즉, `<div>` 가 1초 뒤에야 마운트된다면 텔레포트는 여전히 오류를 보고합니다. defer는 `mounted` 라이프사이클 훅과 유사하게 동작합니다.

GitHub에서 이 페이지 편집

Previous
KeepAlive

Next
Suspense