



KeepAlive

`<KeepAlive>` 는 여러 컴포넌트 간을 동적으로 전환할 때 컴포넌트 인스턴스를 조건부로 캐시할 수 있게 해주는 내장 컴포넌트입니다.

기본 사용법

컴포넌트 기본 챕터에서 동적 컴포넌트에 대한 문법을 소개한 바 있습니다. `<component>` 특수 엘리먼트를 사용하는 방식입니다:

```
<component :is="activeComponent" />
```

template

기본적으로 활성화된 컴포넌트 인스턴스는 전환 시 언마운트됩니다. 이로 인해 해당 컴포넌트가 가지고 있던 변경된 상태는 모두 사라집니다. 다시 이 컴포넌트를 표시하면, 오직 초기 상태만을 가진 새로운 인스턴스가 생성됩니다.

아래 예시에서는 상태를 가진 두 개의 컴포넌트가 있습니다. A는 카운터를, B는 `v-model` 을 통해 입력값과 동기화된 메시지를 가지고 있습니다. 둘 중 하나의 상태를 변경한 뒤, 다른 컴포넌트로 전환했다가 다시 돌아와 보세요:

☒ A ☐ B

Current component: A

Count: 0

+

다시 전환해 돌아오면, 이전에 변경했던 상태가 초기화된 것을 확인할 수 있습니다.

전환 시마다 새로운 컴포넌트 인스턴스를 생성하는 것은 일반적으로 유용한 동작이지만, 이 경우에는 두 컴포넌트 인스턴스를 비활성 상태에서도 보존하고 싶을 수 있습니다. 이 문제를 해결하려면 동적 컴포넌트를 `<KeepAlive>` 내장 컴포넌트로 감싸주면 됩니다:



```
<KeepAlive>
  <component :is="activeComponent" />
</KeepAlive>
```

이제 컴포넌트 전환 시에도 상태가 유지됩니다:

☒ A ☐ B

Current component: A

Count: 0

+

▶ Playground에서 직접 실행해보기

❗ TIP

DOM 내 템플릿에서 사용할 때는 `<keep-alive>` 로 참조해야 합니다.

포함 / 제외

기본적으로 `<KeepAlive>` 는 내부의 모든 컴포넌트 인스턴스를 캐시합니다. `include` 와 `exclude` prop을 통해 이 동작을 커스터마이징할 수 있습니다. 두 prop 모두 쉼표로 구분된 문자열, RegExp , 또는 이 타입들을 포함하는 배열이 될 수 있습니다:

```
<!-- 쉼표로 구분된 문자열 -->
<KeepAlive include="a,b">
  <component :is="view" />
</KeepAlive>

<!-- 정규식 (v-bind 사용) -->
<KeepAlive :include="/a|b/">
  <component :is="view" />
</KeepAlive>

<!-- 배열 (v-bind 사용) -->
<KeepAlive :include="['a', 'b']">
  <component :is="view" />
</KeepAlive>
```

template



포넨트는 반드시 `name` 옵션을 명시적으로 선언해야 합니다.

TIP

3.2.34 버전부터는 `<script setup>` 을 사용하는 단일 파일 컴포넌트의 경우 파일명을 기반으로 `name` 옵션이 자동으로 추론되므로, 직접 이름을 선언할 필요가 없습니다.

최대 캐시 인스턴스 수

`max` prop을 통해 캐시할 수 있는 컴포넌트 인스턴스의 최대 개수를 제한할 수 있습니다. `max` 가 지정되면, `<KeepAlive>` 는 **LRU** 캐시처럼 동작합니다. 캐시된 인스턴스의 수가 지정한 최대 개수를 초과하려고 하면, 가장 오랫동안 접근하지 않은 캐시 인스턴스가 파괴되어 새로운 인스턴스를 위한 공간을 만듭니다.

```
<KeepAlive :max="10">
  <component :is="activeComponent" />
</KeepAlive>
```

template

캐시된 인스턴스의 생명주기

컴포넌트 인스턴스가 DOM에서 제거되지만 `<KeepAlive>` 로 캐시된 컴포넌트 트리의 일부라면, 언마운트되는 대신 **비활성화(deactivated)** 상태로 전환됩니다. 캐시된 트리의 일부로 DOM에 다시 삽입되면 **활성화(activated)** 됩니다.

캐시된 컴포넌트는 `onActivated()` 와 `onDeactivated()` 를 사용해 이 두 상태에 대한 생명주기 혹은 등록할 수 있습니다:

```
<script setup>
import { onActivated, onDeactivated } from 'vue'

onActivated(() => {
  // 최초 마운트 시 호출됨
  // 그리고 캐시에서 다시 삽입될 때마다 호출됨
})

onDeactivated(() => {
  // DOM에서 캐시로 이동할 때 호출됨
  // 그리고 언마운트될 때도 호출됨
})
```

vue



다음 사항에 유의하세요:

`onActivated` 는 마운트 시에도 호출되며, `onDeactivated` 는 언마운트 시에도 호출됩니다.

두 혹 모두 `<KeepAlive>` 로 캐시된 루트 컴포넌트뿐만 아니라, 캐시된 트리 내의 하위 컴포넌트에도 적용됩니다.

관련 문서

`<KeepAlive>` [API 레퍼런스](#)

 [GitHub에서 이 페이지 편집](#)

[< Previous](#)

TransitionGroup

[Next >](#)

Teleport