템플릿 ref

Vue의 선언적 렌더링 모델은 대부분의 직접적인 DOM 조작을 추상화해주지만, 여전히 기본 DOM 요소에 직접 접근해야 하는 경우가 있을 수 있습니다. 이를 위해 특별한 `ref` 속성을 사용할 수 있습니다:

```template
<input ref="input">
```

`ref` 는 `v-for` 챕터에서 다룬 `key` 속성과 유사한 특별한 속성입니다. 이를 통해 마운트된 후 특정 DOM 요소나 자식 컴포넌트 인스턴스에 직접 참조를 얻을 수 있습니다. 예를 들어, 컴포넌트가 마운트될 때 프로그래밍적으로 input에 포커스를 주거나, 요소에 3rd party 라이브러리를 초기화할 때 유용할 수 있습니다.

---

ref 접근하기

Composition API에서 참조를 얻으려면 `useTemplateRef()` 3.5+ 헬퍼를 사용할 수 있습니다:

```vue
<script setup>
import { useTemplateRef, onMounted } from 'vue'

// 첫 번째 인자는 템플릿의 ref 값과 일치해야 합니다
const input = useTemplateRef('my-input')

onMounted(() => {
  input.value.focus()
})
</script>

<template>
  <input ref="my-input" />
</template>
```

컴포넌트에 따라 `input.value` 의 타입을 자동으로 추론합니다.

▶ 3.5 이전 버전에서의 사용법

ref는 **컴포넌트가 마운트된 후에만** 접근할 수 있다는 점에 유의하세요. 템플릿 표현식에서 `input` 에 접근하려고 하면, 첫 렌더에서는 `null` 이 됩니다. 이는 요소가 첫 렌더 이후에야 존재하기 때문입니다!

템플릿 ref의 변화를 감시하려는 경우, ref가 `null` 값을 가질 수 있는 상황을 반드시 고려해야 합니다:

```js
watchEffect(() => {
  if (input.value) {
    input.value.focus()
  } else {
    // 아직 마운트되지 않았거나, 요소가 언마운트됨 (예: v-if에 의해)
  }
})
```

참고: 템플릿 ref 타입 지정 TS

컴포넌트에 ref 사용하기

> 이 섹션은 컴포넌트에 대한 지식을 전제로 합니다. 필요하다면 건너뛰고 나중에 다시 오세요.

`ref` 는 자식 컴포넌트에도 사용할 수 있습니다. 이 경우 참조는 컴포넌트 인스턴스가 됩니다:

```vue
<script setup>
import { useTemplateRef, onMounted } from 'vue'
import Child from './Child.vue'

const childRef = useTemplateRef('child')

onMounted(() => {
  // childRef.value는 <Child />의 인스턴스를 가집니다
})
</script>

<template>
  <Child ref="child" />
</template>
```

▶ 3.5 이전 버전에서의 사용법

넌트의 `this` 와 동일합니다. 즉, 부모 컴포넌트는 자식 컴포넌트의 모든 속성과 메서드에 완전히 접근할 수 있습니다. 이는 부모와 자식 간에 강하게 결합된 구현 세부사항을 쉽게 만들 수 있으므로, 컴포넌트 ref는 반드시 필요할 때만 사용해야 합니다. 대부분의 경우, 표준 props와 emit 인터페이스를 사용하여 부모/자식 상호작용을 구현하는 것이 좋습니다.

예외적으로, `<script setup>` 을 사용하는 컴포넌트는 **기본적으로 비공개**입니다: 부모 컴포넌트가 `<script setup>` 을 사용하는 자식 컴포넌트를 참조할 경우, 자식 컴포넌트가 `defineExpose` 매크로를 사용해 공개 인터페이스를 노출하지 않는 한 아무것도 접근할 수 없습니다:

vue

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

const a = 1
const b = ref(2)

// defineExpose와 같은 컴파일러 매크로는 import가 필요하지 않습니다
defineExpose({
  a,
  b
})
</script>
```

부모가 템플릿 ref를 통해 이 컴포넌트의 인스턴스를 얻으면, 반환된 인스턴스는 `{ a: number, b: number }` 형태가 됩니다 (ref는 일반 인스턴스처럼 자동으로 언래핑됩니다).

defineExpose는 반드시 await 연산 이전에 호출되어야 합니다. 그렇지 않으면 await 이후에 노출된 속성과 메서드는 접근할 수 없습니다.

참고: 컴포넌트 템플릿 ref 타입 지정 TS

`v-for` 내부의 ref

> v3.5 이상 필요

`v-for` 내부에서 `ref` 를 사용할 때, 해당 ref는 Array 값을 가져야 하며, 마운트 후 요소들로 채워집니다:

vue

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

const list = ref([
  /* ... */
])
```

```
const itemRefs = useTemplateRef('items')

onMounted(() => console.log(itemRefs.value))
</script>

<template>
  <ul>
    <li v-for="item in list" ref="items">
      {{ item }}
    </li>
  </ul>
</template>
```

Playground에서 실행해보기

▶ 3.5 이전 버전에서의 사용법

ref 배열은 소스 배열과 **동일한 순서를 보장하지 않는다**는 점에 유의해야 합니다.

함수 ref

문자열 키 대신, `ref` 속성은 함수에 바인딩할 수도 있습니다. 이 함수는 각 컴포넌트 업데이트 시 호출되며, 요소 참조를 어디에 저장할지 완전히 자유롭게 결정할 수 있습니다. 함수는 첫 번째 인자로 요소 참조를 받습니다:

```template
<input :ref="(el) => { /* el을 속성이나 ref에 할당 */ }">
```

함수 대신 ref 이름 문자열을 전달하는 것이 아니라, 동적 `:ref` 바인딩을 사용하고 있습니다. 요소가 언마운트될 때 인자는 `null` 이 됩니다. 물론, 인라인 함수 대신 메서드를 사용할 수도 있습니다.

GitHub에서 이 페이지 편집