템플릿 참조

Vue의 선언적 렌더링 모델은 개발자가 직접 DOM에 접근해야 해야하는 대부분을 추상화하지만, 개발자가 DOM 엘리먼트에 직접 접근해야 하는 경우가 여전히 있을 수 있습니다. 이러한 필요성을 충족시키기 위해 `ref` 라는 특별한 속성을 사용할 수 있습니다:

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

`ref` 는 `v-for` 장에서 언급한 `key` 속성과 유사한 특수 속성입니다. 이를 통해 마운트된 특정 DOM 엘리먼트 또는 자식 컴포넌트 인스턴스에 직접적인 참조를 얻을 수 있습니다. 예를 들어 컴포넌트 마운트의 인풋에 초점을 맞추거나 엘리먼트에서 타사 라이브러리를 초기화하려는 경우, 프로그래밍 방식으로 조작하기 편리합니다.

ref로 접근하기

Composition API를 사용하여 참조를 얻으려면, 템플릿 ref 속성 값과 일치하는 이름으로 ref를 선언해야 합니다:

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

// 엘리먼트 참조를 위해 ref를 선언하십시오.
// 이름은 템플릿 ref 값과 일치해야 합니다.
const input = ref(null)

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

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

`<script setup>` 을 사용하지 않는 경우, `setup()` 에서 참조를 반환해야 합니다:

js
```
export default {
  setup() {
    const input = ref(null)
    // ...
    return {
      input
    }
  }
}
```

컴포넌트가 마운트된 **이후에만 ref에 접근할 수 있습니다.** 템플릿 표현식에서 `input` 에 접근하려고 하면, 첫 번째 렌더링에서는 `null` 이 됩니다. 이는 요소가 첫 번째 렌더링 이후에야 존재하기 때문입니다!

템플릿 ref의 변경 사항을 감시하려는 경우 ref에 `null` 값이 있는 경우를 고려해야 합니다:

js
```
watchEffect(() => {
  if (input.value) {
    input.value.focus()
  } else {
    // v-if에 의해 아직 마운트 되지 않았거나, 마운트 해제된 경우
```

```
  }
})
```

참고: 템플릿 ref에 타입 지정하기

v-for 내부에서 ref 사용하기

버전이 v3.2.25 이상 이어야 합니다.

ref 가 v-for 내부에서 사용되면, 해당 ref는 마운트 후 엘리먼트로 채워지므로 배열 값이어야 합니다:

vue

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

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

const itemRefs = ref([])

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

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

온라인 연습장으로 실행하기

ref 배열은 소스 배열과 **동일한 순서를 보장하지 않습니다**.

함수로 참조하기

문자열 키 대신 ref 속성을 함수에 바인딩할 수도 있습니다. 이 함수는 각 컴포넌트 업데이트 시 호출되며, 엘리먼트 참조를 저장할 위치에 대한 완전한 유연성을 제공합니다. 이 함수는 첫 번째 인자로 엘리먼트 참조를 받습니다:

template

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

:ref 와 같이 동적 바인딩으로 사용하는 경우에는 ref의 이름 대신 함수를 전달할 수 있습니다. 엘리먼트가 마운트 해제되는 경우 인자는 null 입니다. 물론 인라인 함수 대신 메서드를 사용할 수 있습니다.

컴포넌트에 ref 사용하기

이 섹션에서는 컴포넌트에 대한 지식이 있다고 가정하므로, 건너뛰고 나중에 읽어도 됩니다.

ref 는 자식 컴포넌트에 사용할 수도 있습니다. 이 경우 ref는 컴포넌트 인스턴스를 참조합니다:

vue

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

const child = ref(null)

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

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

자식 컴포넌트가 옵션 API를 사용하거나 `<script setup>` 을 사용하지 않는 경우, 참조된 인스턴스는 자식 컴포넌트의 `this` 와 동일합니다. 즉, 부모 컴포넌트는 자식 컴포넌트의 모든 속성과 메서드에 대한 접근 권한을 갖습니다. 이렇게 하면 상·하위(부모·자식) 간에 밀접하게 결합한 구현 세부 정보를 쉽게 만들 수 있으므로, 컴포넌트 참조는 반드시 필요할 때만 사용해야 합니다. 대부분의 경우 우선적으로 표준 props를 사용하여 상·하위 상호 작용을 구현하고 인터페이스를 내보내야 합니다.

여기서 예외는 `<script setup>` 을 사용하는 컴포넌트가 **기본적으로 비공개**라는 점입니다. `<script setup>` 을 사용하는 자식 컴포넌트를 부모 컴포넌트에서 참조하려는 경우, 자식 컴포넌트가 `defineExpose` 컴파일러 매크로를 사용하여 선택한 인터페이스를 노출하지 않는 한 그 무엇에도 접근할 수 없습니다. (역자주: 컴파일러 매크로이기 때문에 개발 환경 설정에 따라 lint 에러나 경고가 나올수 있습니다)

vue

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

const a = 1
const b = ref(2)

// 컴파일러 매크로, 예를 들어 defineExpose 는 import 할 필요가 없습니다
defineExpose({
  a,
  b
})
</script>
```

상위에서 템플릿 참조를 통해 이 컴포넌트의 인스턴스를 가져오면, 검색된 인스턴스는 `{ a: number, b: number }` 와 같습니다(일반 인스턴스에서처럼 ref는 자동으로 래핑 해제됨).

참고: 컴포넌트 템플릿 **ref**에 타입 지정하기