컴포넌트 v-model

기본 사용법

`v-model` 은 컴포넌트에서 양방향 바인딩을 구현하는 데 사용할 수 있습니다.

Vue 3.4부터, 이를 달성하는 권장 방법은 `defineModel()` 매크로를 사용하는 것입니다:

Child.vue

```vue
<script setup>
const model = defineModel()

function update() {
  model.value++
}
</script>

<template>
  <div>부모에 바인딩된 v-model 값: {{ model }}</div>
  <button @click="update">증가</button>
</template>
```

```vue-html [Parent.vue]

부모는 `v-model`로 값을 바인딩할 수 있습니다:

```vue-html [Parent.vue]
<Child v-model="countModel" />
```

`defineModel()` 이 반환하는 값은 ref입니다. 이 ref는 다른 ref와 마찬가지로 접근하고 변경할 수 있지만, 부모 값과 로컬 값 사이의 양방향 바인딩 역할을 합니다:

- `.value` 는 부모 `v-model` 에 바인딩된 값과 동기화됩니다;
```

즉, 이 ref를 네이티브 input 요소에 `v-model` 로 바인딩할 수도 있으므로, 네이티브 input 요소를 감싸면서 동일한 `v-model` 사용법을 제공하는 것이 간단해집니다:

vue

```
<script setup>
const model = defineModel()
</script>

<template>
  <input v-model="model" />
</template>
```

플레이그라운드에서 직접 해보기

내부 동작 방식

`defineModel` 은 편의 매크로입니다. 컴파일러는 이를 다음과 같이 확장합니다:

로컬 ref의 값과 동기화되는 `modelValue` 라는 prop;
로컬 ref의 값이 변경될 때 발생하는 `update:modelValue` 라는 이벤트.

아래는 3.4 이전에 동일한 자식 컴포넌트를 구현하는 방법입니다:

Child.vue

vue

```
<script setup>
const props = defineProps(['modelValue'])
const emit = defineEmits(['update:modelValue'])
</script>

<template>
  <input
    :value="props.modelValue"
    @input="emit('update:modelValue', $event.target.value)"
  />
</template>
```

그런 다음, 부모 컴포넌트에서 `v-model="foo"` 는 다음과 같이 컴파일됩니다:

Parent.vue

template

```
<Child
  :modelValue="foo"
  @update:modelValue="$event => (foo = $event)"
/>
```

다.

`defineModel` 이 prop을 선언하기 때문에, prop 옵션을 `defineModel` 에 전달하여 선언할 수 있습니다:

```js
// v-model을 필수로 만들기
const model = defineModel({ required: true })

// 기본값 제공
const model = defineModel({ default: 0 })
```

> ⚠ WARNING
>
> `defineModel` prop에 `default` 값을 지정하고 부모 컴포넌트에서 이 prop에 값을 제공하지 않으면, 부모와 자식 컴포넌트 간에 동기화가 깨질 수 있습니다. 아래 예시에서, 부모의 `myRef` 는 undefined이지만, 자식의 `model` 은 1입니다:
>
> Child.vue
>
> ```vue
> <script setup>
> const model = defineModel({ default: 1 })
> </script>
> ```
>
> Parent.vue
>
> ```vue
> <script setup>
> const myRef = ref()
> </script>
>
> <template>
>   <Child v-model="myRef"></Child>
> </template>
> ```

`v-model` 인자

컴포넌트의 `v-model` 은 인자도 받을 수 있습니다:

```template
<MyComponent v-model:title="bookTitle" />
```

수 있습니다:

MyComponent.vue

```vue
<script setup>
const title = defineModel('title')
</script>

<template>
  <input type="text" v-model="title" />
</template>
```

플레이그라운드에서 직접 해보기

prop 옵션도 필요하다면, 모델 이름 뒤에 전달해야 합니다:

```js
const title = defineModel('title', { required: true })
```

▶ 3.4 이전 사용법

다중 `v-model` 바인딩

`v-model` 인자에서 배운 대로, 특정 prop과 이벤트를 지정하는 기능을 활용하여, 이제 하나의 컴포넌트 인스턴스에 여러 개의 `v-model` 바인딩을 만들 수 있습니다.

각 `v-model` 은 별도의 prop과 동기화되며, 컴포넌트에 추가 옵션이 필요하지 않습니다:

```template
<UserName
  v-model:first-name="first"
  v-model:last-name="last"
/>
```

```vue
<script setup>
const firstName = defineModel('firstName')
const lastName = defineModel('lastName')
</script>

<template>
  <input type="text" v-model="firstName" />
  <input type="text" v-model="lastName" />
</template>
```

▶ 3.4 이전 사용법

`v-model` 수식어 처리

폼 입력 바인딩에 대해 배울 때, `v-model` 에는 내장 수식어 - `.trim` , `.number` , `.lazy` 가 있다는 것을 보았습니다. 경우에 따라, 커스텀 입력 컴포넌트의 `v-model` 도 커스텀 수식어를 지원하길 원할 수 있습니다.

예시로, `v-model` 바인딩으로 전달된 문자열의 첫 글자를 대문자로 만드는 커스텀 수식어 `capitalize` 를 만들어봅시다:

template

```
<MyComponent v-model.capitalize="myText" />
```

컴포넌트 `v-model` 에 추가된 수식어는 자식 컴포넌트에서 `defineModel()` 반환값을 구조 분해 할당하여 접근할 수 있습니다:

vue

```
<script setup>
const [model, modifiers] = defineModel()

console.log(modifiers) // { capitalize: true }
</script>

<template>
  <input type="text" v-model="model" />
</template>
```

수식어에 따라 값을 읽거나 쓸 때 조건부로 조정하려면, `defineModel()` 에 `get` 과 `set` 옵션을 전달할 수 있습니다. 이 두 옵션은 모델 ref의 get/set 시 값을 받아 변환된 값을 반환해야 합니다. 아래는 `set` 옵션을 사용해 `capitalize` 수식어를 구현하는 방법입니다:

vue

```
<script setup>
const [model, modifiers] = defineModel({
  set(value) {
    if (modifiers.capitalize) {
      return value.charAt(0).toUpperCase() + value.slice(1)
    }
    return value
  }
})
</script>
```

```
<template>
  <input type="text" v-model="model" />
</template>
```

플레이그라운드에서 직접 해보기

▶ 3.4 이전 사용법

인자가 있는 `v-model` 의 수식어

다음은 여러 인자와 각각 다른 수식어를 가진 다중 `v-model` 에서 수식어를 사용하는 또 다른 예시입니다:

template

```
<UserName
  v-model:first-name.capitalize="first"
  v-model:last-name.uppercase="last"
/>
```

vue

```
<script setup>
const [firstName, firstNameModifiers] = defineModel('firstName')
const [lastName, lastNameModifiers] = defineModel('lastName')

console.log(firstNameModifiers) // { capitalize: true }
console.log(lastNameModifiers) // { uppercase: true }
</script>
```

▶ 3.4 이전 사용법

GitHub에서 이 페이지 편집