폼 입력 바인딩

Vue School의 무료 동영상 강의 보기

프론트엔드에서 폼을 다룰 때, 우리는 종종 폼 입력 요소의 상태를 JavaScript의 해당 상태와 동기화해야 합니다. 값을 바인딩하고 변경 이벤트 리스너를 수동으로 연결하는 것은 번거로울 수 있습니다:

template

```
<input
  :value="text"
  @input="event => text = event.target.value">
```

`v-model` 디렉티브를 사용하면 위의 코드를 다음과 같이 간단하게 만들 수 있습니다:

template

```
<input v-model="text">
```

또한, `v-model` 은 다양한 타입의 입력, `<textarea>` , `<select>` 요소에도 사용할 수 있습니다. 사용되는 요소에 따라 자동으로 다른 DOM 속성과 이벤트 쌍으로 확장됩니다:

- 텍스트 타입의 `<input>` 과 `<textarea>` 요소는 `value` 속성과 `input` 이벤트를 사용합니다.
- `<input type="checkbox">` 와 `<input type="radio">` 는 `checked` 속성과 `change` 이벤트를 사용합니다.
- `<select>` 는 `value` 를 prop으로, `change` 를 이벤트로 사용합니다.

> ⓘ 참고
>
> `v-model` 은 폼 요소에 있는 초기 `value` , `checked` 또는 `selected` 속성을 무시합니다. 항상 현재 바인딩된 JavaScript 상태를 진실의 원천으로 간주합니다. 초기 값은 반응성 API를 사용하여 JavaScript 쪽에서 선언해야 합니다.

기본 사용법

텍스트

```template
<p>메시지: {{ message }}</p>
<input v-model="message" placeholder="수정하세요" />
```

메시지:

수정하세요

플레이그라운드에서 실행해보기

> 참고
>
> IME (중국어, 일본어, 한국어 등)이 필요한 언어의 경우, IME 조합 중에는 `v-model` 이 업데이트되지 않는다는 것을 알 수 있습니다. 이러한 업데이트에도 반응하고 싶다면, `v-model` 대신 직접 `input` 이벤트 리스너와 `value` 바인딩을 사용하세요.

여러 줄 텍스트

```template
<span>여러 줄 메시지:</span>
<p style="white-space: pre-line;">{{ message }}</p>
<textarea v-model="message" placeholder="여러 줄을 추가하세요"></textarea>
```

여러 줄 메시지:

여러 줄을 추가하세요

플레이그라운드에서 실행해보기

`<textarea>` 내부에서 보간(interpolation)은 동작하지 않습니다. 대신 `v-model` 을 사용하세요.

```template
<!-- 잘못된 예 -->
<textarea>{{ text }}</textarea>
```

```
<!-- 줄바꿈 예 -->
<textarea v-model="text"></textarea>
```

체크박스

단일 체크박스, 불리언 값:

template

```
<input type="checkbox" id="checkbox" v-model="checked" />
<label for="checkbox">{{ checked }}</label>
```

☐ false

플레이그라운드에서 실행해보기

여러 개의 체크박스를 동일한 배열 또는 Set 값에 바인딩할 수도 있습니다:

js

```
const checkedNames = ref([])
```

template

```
<div>선택된 이름: {{ checkedNames }}</div>

<input type="checkbox" id="jack" value="Jack" v-model="checkedNames" />
<label for="jack">Jack</label>

<input type="checkbox" id="john" value="John" v-model="checkedNames" />
<label for="john">John</label>

<input type="checkbox" id="mike" value="Mike" v-model="checkedNames" />
<label for="mike">Mike</label>
```

선택된 이름: []

☐ Jack ☐ John ☐ Mike

이 경우, `checkedNames` 배열에는 현재 체크된 박스의 값들이 항상 포함됩니다.

플레이그라운드에서 실행해보기

라디오

```
<input type="radio" id="one" value="One" v-model="picked" />
<label for="one">One</label>

<input type="radio" id="two" value="Two" v-model="picked" />
<label for="two">Two</label>
```

선택됨:

○ One ○ Two

▶ 플레이그라운드에서 실행해보기

셀렉트

단일 선택:

template

```
<div>선택됨: {{ selected }}</div>

<select v-model="selected">
  <option disabled value="">하나를 선택하세요</option>
  <option>A</option>
  <option>B</option>
  <option>C</option>
</select>
```

선택됨:

하나를 선택하세요

▶ 플레이그라운드에서 실행해보기

ⓘ 참고

`v-model` 표현식의 초기 값이 옵션 중 어떤 것과도 일치하지 않으면, `<select>` 요소는 "선택되지 않음" 상태로 렌더링됩니다. iOS에서는 이 경우 사용자가 첫 번째 항목을 선택할 수 없는데, 이는 iOS가 이 경우 change 이벤트를 발생시키지 않기 때문입니다. 따라서 위 예시처럼 값이 비어 있는 비활성화된 옵션을 제공하는 것이 좋습니다.

다중 선택(배열에 바인딩):

```
<select v-model="selected" multiple>
  <option>A</option>
  <option>B</option>
  <option>C</option>
</select>
```

선택됨: []

A
B
C

플레이그라운드에서 실행해보기

셀렉트 옵션은 `v-for` 로 동적으로 렌더링할 수 있습니다:

```js
const selected = ref('A')

const options = ref([
  { text: 'One', value: 'A' },
  { text: 'Two', value: 'B' },
  { text: 'Three', value: 'C' }
])
```

```template
<select v-model="selected">
  <option v-for="option in options" :value="option.value">
    {{ option.text }}
  </option>
</select>

<div>선택됨: {{ selected }}</div>
```

선택됨: A

One

플레이그라운드에서 실행해보기

값 바인딩

스의 경우 불리언)입니다:

```template
<!-- 체크되면 `picked`는 문자열 "a"가 됩니다 -->
<input type="radio" v-model="picked" value="a" />

<!-- `toggle`은 true 또는 false입니다 -->
<input type="checkbox" v-model="toggle" />

<!-- 첫 번째 옵션이 선택되면 `selected`는 문자열 "abc"가 됩니다 -->
<select v-model="selected">
  <option value="abc">ABC</option>
</select>
```

하지만 때로는 현재 활성 인스턴스의 동적 속성에 값을 바인딩하고 싶을 수 있습니다. 이럴 때는 `v-bind` 를 사용할 수 있습니다. 또한, `v-bind` 를 사용하면 입력 값을 문자열이 아닌 값에 바인딩할 수 있습니다.

체크박스

```template
<input
  type="checkbox"
  v-model="toggle"
  true-value="yes"
  false-value="no" />
```

`true-value` 와 `false-value` 는 `v-model` 에서만 동작하는 Vue 전용 속성입니다. 여기서 체크박스가 체크되면 `toggle` 속성의 값이 `'yes'` 로, 체크 해제되면 `'no'` 로 설정됩니다. 동적 값에 바인딩하려면 `v-bind` 를 사용할 수도 있습니다:

```template
<input
  type="checkbox"
  v-model="toggle"
  :true-value="dynamicTrueValue"
  :false-value="dynamicFalseValue" />
```

> ⓘ 팁
>
> `true-value` 와 `false-value` 속성은 입력의 `value` 속성에 영향을 주지 않습니다. 브라우저는 체크되지 않은 박스를 폼 제출에 포함하지 않기 때문입니다. 폼에서 두 값 중 하나(예: "yes" 또는 "no")가 반드시 제출되도록 하려면, 라디오 입력을 사용하세요.

```template
<input type="radio" v-model="pick" :value="first" />
<input type="radio" v-model="pick" :value="second" />
```

첫 번째 라디오 입력이 체크되면 `pick` 은 `first` 의 값으로, 두 번째가 체크되면 `second` 의 값으로 설정됩니다.

셀렉트 옵션

```template
<select v-model="selected">
  <!-- 인라인 객체 리터럴 -->
  <option :value="{ number: 123 }">123</option>
</select>
```

`v-model` 은 문자열이 아닌 값의 바인딩도 지원합니다! 위 예시에서 옵션이 선택되면, `selected` 는 `{ number: 123 }` 객체 리터럴 값으로 설정됩니다.

수식어

`.lazy`

기본적으로, `v-model` 은 각 `input` 이벤트 후에 입력과 데이터를 동기화합니다(위에서 언급한 IME 조합 제외). 대신 `change` 이벤트 후에 동기화하려면 `lazy` 수식어를 추가할 수 있습니다:

```template
<!-- "input" 대신 "change" 후에 동기화됨 -->
<input v-model.lazy="msg" />
```

`.number`

사용자 입력을 자동으로 숫자 타입으로 변환하려면, `v-model` 이 적용된 입력에 `number` 수식어를 추가할 수 있습니다:

```template
<input v-model.number="age" />
```

값이 `parseFloat()` 로 파싱될 수 없으면, 원래(문자열) 값이 대신 사용됩니다. 특히 입력이 비어 있으면(예: 사용자가 입력 필드를 지운 경우), 빈 문자열이 반환됩니다. 이 동작은 DOM 속성

입력에 `type="number"` 가 있으면 `number` 수식어가 자동으로 적용됩니다.

`.trim`

사용자 입력의 공백을 자동으로 제거하려면, `v-model` 이 적용된 입력에 `trim` 수식어를 추가할 수 있습니다:

```template
<input v-model.trim="msg" />
```

컴포넌트에서의 `v-model`

> Vue의 컴포넌트에 익숙하지 않다면, 이 부분은 지금은 건너뛰어도 됩니다.

HTML의 내장 입력 타입만으로는 항상 요구사항을 충족할 수 없습니다. 다행히도, Vue 컴포넌트를 사용하면 완전히 커스텀된 동작을 가진 재사용 가능한 입력을 만들 수 있습니다. 이러한 입력도 `v-model` 과 함께 동작합니다! 더 자세한 내용은 컴포넌트 가이드의 v-model과 함께 사용하기를 참고하세요.