Form Input Bindings

In Vue, we use v-model to create two-way data binding between form inputs and reactive state.

Vue automatically updates the data when the user types or interacts with the form — and also updates the form when the data changes.

1. Text Input (v-model)

Binds a single-line text input to a string value.

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

const name = ref('')
</script>

<template>
  <input v-model="name" placeholder="Enter your name" />
  <p>Your name is: {{ name }}</p>
</template>
```

2. Textarea

Binds multi-line input to a string value.

```
<script setup>
const message = ref('')
</script>

<template>
  <textarea v-model="message" placeholder="Write something..."></textarea>
  <p>Message: {{ message }}</p>
</template>
```

3. Checkbox (Single Boolean)

Binds a checkbox to a boolean true/false.

```
<script setup>
const isChecked = ref(false)
</script>

<template>
```

```
  <input type="checkbox" v-model="isChecked" />
  <p>Checkbox is {{ isChecked ? 'checked' : 'unchecked' }}</p>
</template>
```

4. Checkbox (Multiple Values - Array)

Binds multiple checkboxes to an array of selected values.

```
<script setup>
const selectedFruits = ref([])
</script>

<template>
  <label><input type="checkbox" value="Apple" v-model="selectedFruits" />
Apple</label>
  <label><input type="checkbox" value="Banana" v-model="selectedFruits" />
Banana</label>
  <label><input type="checkbox" value="Mango" v-model="selectedFruits" />
Mango</label>
  <p>Selected Fruits: {{ selectedFruits }}</p>
</template>
```

5. Radio Buttons

Binds multiple radio buttons to a single value.

```
<script setup>
const gender = ref('')
</script>

<template>
  <label><input type="radio" value="Male" v-model="gender" /> Male</label>
  <label><input type="radio" value="Female" v-model="gender" /> Female</label>
  <p>Gender: {{ gender }}</p>
</template>
```

6. Select (Single Value)

Binds a dropdown to a single selected value.

```
<script setup>
const selectedColor = ref('')
</script>

<template>
  <select v-model="selectedColor">
```

```
    <option disabled value="">Please select a color</option>
    <option>Red</option>
    <option>Green</option>
    <option>Blue</option>
  </select>
  <p>Color: {{ selectedColor }}</p>
</template>
```

7. Select (Multiple Values)

```
<script setup>
const selectedColors = ref([])
</script>

<template>
  <select v-model="selectedColors" multiple>
    <option>Red</option>
    <option>Green</option>
    <option>Blue</option>
  </select>
  <p>Selected Colors: {{ selectedColors }}</p>
</template>
```

Binds a multi-select dropdown to an array.

Modifiers with v-model

Vue provides modifiers that change how input data is bound.

1. .lazy Modifier

Normally, v-model updates the data immediately when the user types (input event). .lazy changes that it updates only when the input loses focus (blur event).

Reduces unnecessary updates while typing (good for performance). Useful when you only care about the final value after user finishes editing.

Ideal when you're updating backend or doing validation on blur (not on every keystroke).

```
<input v-model.lazy="username" />
```

2. .number Modifier

Automatically converts string input into a number.

HTML inputs always return strings, even for number fields. If you need actual numbers (for math, calculations, or type-checking), .number ensures correct data type.

Without .number, age would be "25" (string). With .number, age is 25 (number).

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

3. .trim Modifier

Removes leading and trailing whitespace from the user's input.

Prevents accidental spaces from messing up validations or searches. Especially important for usernames, emails, passwords — spaces can break logic.

If the user types " joshi124@gmail.com " → it becomes "joshi124@gmail.com" automatically. This saves time from doing email.value.trim() manually in your methods.

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