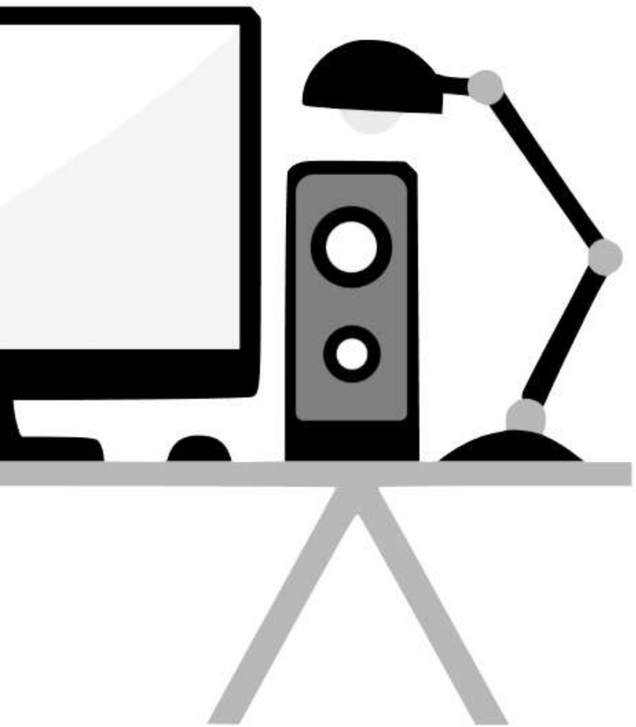
BINDING DE FORMULARIOS

Lección 03

Competencia General

Implementar un formulario de datos interactivo utilizando form binding de Vue para dar solución a un requerimiento.

Contenidos



Binding de formularios

1. Qué es binding
2. Qué significa Two way binding
3. Uso básico: input, textarea, checkbox, radio, select
4. Vinculando valores: radio, checkbox, option, select

Qué es Binding

El ***binding*** en Vue se refiere a la acción de **vincular los datos de la aplicación con los elementos de la interfaz**. Gracias a este mecanismo, cuando una variable en el código JavaScript cambia, la vista se actualiza de forma automática, sin necesidad de manipular el DOM manualmente.

En Vue, esta vinculación puede ser de **una sola dirección** (unidireccional) o de **dos direcciones** (bidireccional). El **binding unidireccional** ocurre cuando el valor de una variable fluye desde la lógica de la aplicación hacia la vista, por ejemplo, cuando mostramos un dato con {{ variable }} o cuando utilizamos la directiva v-bind para asignar dinámicamente valores a los atributos HTML.

El **binding bidireccional**, que explicaremos más adelante en esta lección, permite que el valor del dato se actualice tanto desde JavaScript como desde la interacción del usuario con la interfaz, lo que resulta especialmente útil en formularios.

Cómo funciona v-bind

Para comprender mejor el binding, primero pensemos en cómo se define una imagen en **HTML estático**. Cuando escribimos:

```
<img src="https://ejemplo.com/javascript/logo.svg" alt="Logo de Javascript">
```

El navegador muestra la imagen correctamente, pero si quisiéramos cambiar la ruta o el texto alternativo, tendríamos que editar el HTML manualmente.

En cambio, con Vue podemos separar la estructura de la interfaz de los datos mediante la directiva v-bind. Esta directiva permite que los valores provengan de variables de JavaScript, logrando así un HTML dinámico. Por ejemplo:

```
<template>
  <img :src="image" :alt="text">
</template>

<script>
export default {
  data() {
    return {
      image: "https://ejemplo.com/javascript/logo.svg",
      text: "Logo de Javascript"
    }
  }
}
</script>
```

En este caso, el atributo src de la imagen ya no tiene un valor fijo, sino que depende de la variable image. Si su valor cambia en JavaScript, la imagen mostrada en la interfaz también cambiará. Esta separación entre datos y estructura es la base para comprender el **two-way binding**, que veremos en los formularios con v-model.

sustantiva
CON•SENTIDO

Qué significa Two way binding

El ***two-way data binding*** o **enlace bidireccional** es una de las características más útiles de Vue cuando trabajamos con formularios. Este mecanismo permite que los datos **fluyan en ambas direcciones**:

- **Del JavaScript al DOM**: Los valores definidos en la aplicación se reflejan automáticamente en los elementos de la interfaz.
- **Del DOM al JavaScript**: Cuando el usuario interactúa con un input, el valor de la variable también se actualiza sin escribir código adicional.

Gracias a esto, cualquier cambio en el formulario se refleja inmediatamente en los datos, y cualquier cambio en los datos se refleja automáticamente en la vista.

Two-Way Binding con v-model

El **two-way binding** enlaza un campo del formulario con una variable de tu app **en ambos sentidos**: lo que escribes en el input actualiza el dato, y si el dato cambia desde JavaScript, el input también se actualiza. En Vue se logra con **v-model**.

En este ejemplo, la variable **message** está conectada al <input> con v-model.

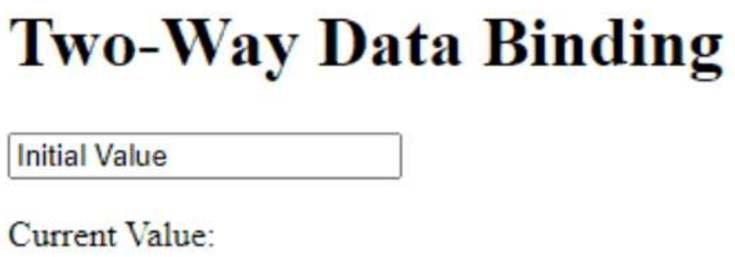
- Escribir en el input actualiza el párrafo de inmediato.
- Cambiar el valor de message desde JavaScript también modifica el input.

```
<div id="app">
  <h2>Two-Way Data Binding</h2>
  <input v-model="message" />
  <p>Current Value: {{ message }}</p>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const message = ref('Initial Value')
    return { message }
  }
}).mount('#app')
</script>
```

sustantiva
CON•SENTIDO

Two-Way Binding con v-model

```
<div id="app">
  <h2>Two-Way Data Binding</h2>
  <input v-model="message" />
  <p>Current Value: {{ message }}</p>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const message = ref('Initial Value')
    return { message }
  }
}).mount('#app')
</script>
```



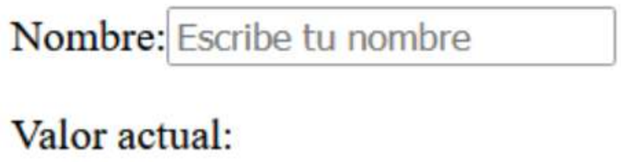
Input con v-model

El ***input* de texto** es el campo más común de un formulario.

Esa vinculación es **bidireccional**: lo que se escribe en el input actualiza el dato y, si el dato cambia en JavaScript, el input se actualiza sin tocar el DOM manualmente.

Ten en cuenta que **placeholder no es un valor**, solo un texto guía; el valor inicial proviene de la variable vinculada.

```
<div id="app">
  <label>Nombre:</label>
  <input v-model="nombre" placeholder="Escribe tu nombre">
  <p>Valor actual: {{ nombre }}</p>
</div>
<script src="(Vue 3 CDN)"></script>

<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const nombre = ref('') // valor inicial
    return { nombre }
  }
}).mount('#app')
</script>
```



Input con v-model

Modificadores útiles

A veces necesitas ajustar cómo se captura el valor. v-model ofrece **modificadores** que te ahorran código:

- **.trim**: elimina espacios al inicio y al final cuando el usuario escribe.
- **.number**: convierte el valor a número (útil con type="number").
- **.lazy**: actualiza el dato **al perder el foco** (blur) o al enviar, no en cada pulsación.

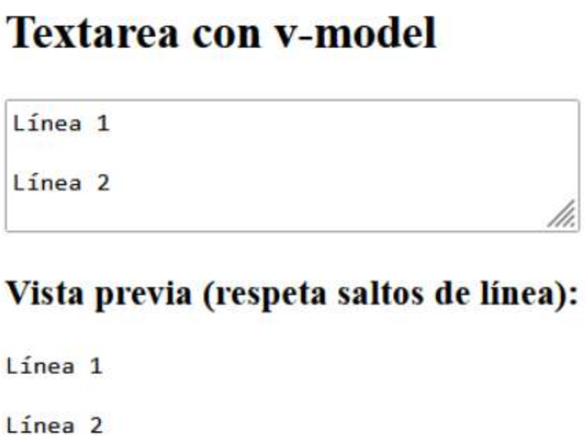
```
<!-- Quita espacios extra -->
<input v-model.trim="usuario">

<!-- Convierte a número (con type="number") -->
<input type="number" v-model.number="edad">

<!-- Actualiza al salir del campo -->
<input v-model.lazy="email" type="email">
```

Textarea con v-model

textarea se usa para **texto multilínea**. En Vue, lo enlazamos con una variable mediante **v-model**, obteniendo un **two-way binding**: lo que escribe el usuario actualiza el dato, y si el dato cambia desde JavaScript, el contenido del textarea también se sincroniza.

A diferencia de un <p>, los saltos de línea (\n) del *textarea* no se ven como saltos en HTML normal. Para mostrarlos en la vista puedes usar un <pre> (que respeta el formato) o aplicar white-space: pre-line por CSS.

```
<div id="app">
  <h2>Textarea con v-model</h2>
  <textarea v-model="bio" rows="4" cols="40" placeholder="Escribe varias líneas...">
  </textarea>
  <h3>Vista previa (respeta saltos de línea):</h3>
  <pre>{{ bio }}</pre>
</div>
<script src="Vue 3 CDN"></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const bio = ref('Línea 1\nLínea 2')
    return { bio }
  }
}).mount('#app')
</script>
```



Checkbox con v-model

Un **checkbox** puede representar un valor **independiente** (marcado / desmarcado) o formar parte de un **grupo** donde el usuario selecciona varias opciones.
En Vue, v-model maneja ambos escenarios:

- **Caso booleano (uno solo):** v-model enlaza con un **Boolean**.

- **Múltiples selecciones:** varios <input type="checkbox"> con el **mismo v-model** construyen un **arreglo** con los value marcados.

- **Valores personalizados:** con :true-value y :false-value puedes guardar algo distinto a true/false (por ejemplo, "noche"/"día").

Por ejemplo, tenemos el siguiente código:

Checkbox con v-model

```html
<div id="app">
  <h2>Checkbox con v-model</h2>

  <!-- 1) Booleano -->
  <h3>1) Caso booleano</h3>
  <label>
    <input type="checkbox" v-model="acepto">
    Acepto los términos
  </label>
  <p>acepto: {{ acepto }}</p>
  <hr>

  <!-- 2) Múltiples selecciones (array) -->
  <h3>2) Múltiples selecciones</h3>
  <label><input type="checkbox" v-model="preferencias" value="condiciones"> Condiciones</label>
  <label><input type="checkbox" v-model="preferencias" value="privacidad"> Privacidad</label>
  <label><input type="checkbox" v-model="preferencias" value="novedades"> Novedades</label>
  <p>preferencias: {{ preferencias }}</p>
  <hr>

  <!-- 3) Valores personalizados -->
  <h3>3) Valores personalizados (true/false)</h3>
  <label>
    <input
      type="checkbox"
      v-model="turno"
      :true-value="'noche'"
      :false-value="'dia'"
    >
    Trabajar en turno noche
  </label>
  <p>turno: {{ turno }}</p>
</div>
```

Checkbox con v-model

1) Caso booleano

☐ Acepto los términos

acepto: false

---

2) Múltiples selecciones

☑ Condiciones ☐ Privacidad ☑ Novedades

preferencias: ["condiciones", "novedades"]

---

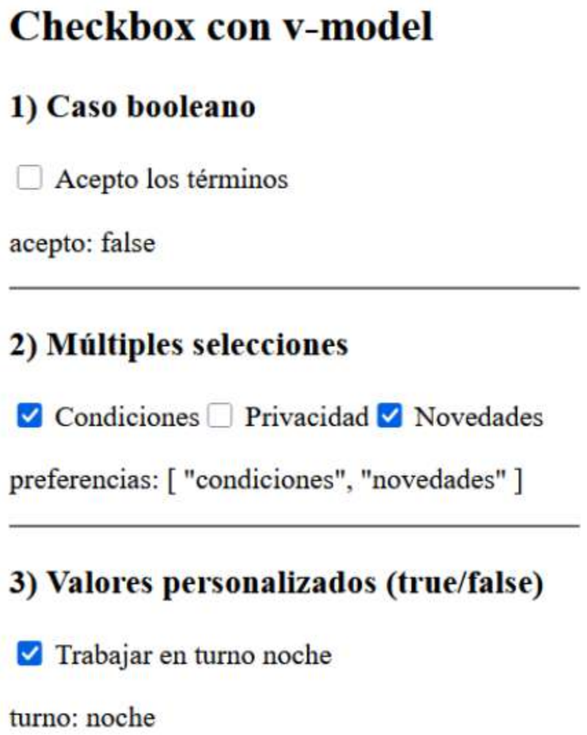
3) Valores personalizados (true/false)

☑ Trabajar en turno noche

turno: noche

Checkbox con v-model

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const acepto = ref(false)
    const preferencias = ref([])
    const turno = ref('dia')
    return { acepto, preferencias, turno }
  }
}).mount('#app')
</script>
```



Radiobutton con v-model

Un **radiobutton** permite seleccionar **solo una opción** dentro de un grupo. En HTML puro, verificar cuál está seleccionado requiere recorrer los elementos y leer la propiedad checked.

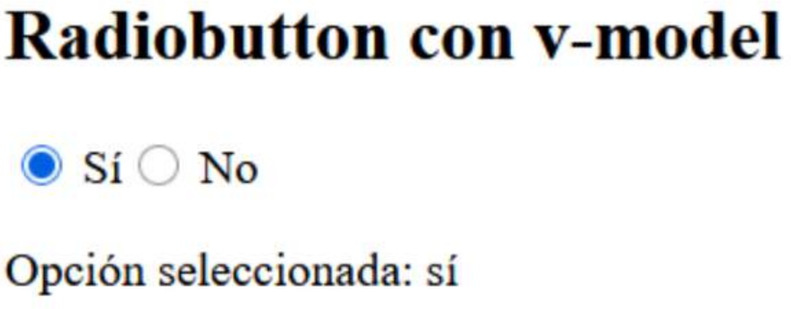
En Vue, el manejo se simplifica gracias a **v-model**:

- Todos los radios del mismo grupo comparten el **mismo v-model**.
- La variable vinculada siempre contendrá el **value del radio seleccionado**.
- No necesitas recorrer los radios ni usar getElementsByName.

Por ejemplo, tenemos el siguiente código:

Radiobutton con v-model

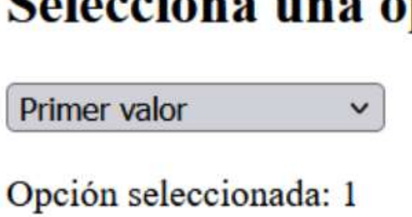
```html
<div id="app">
  <h2>Radiobutton con v-model</h2>

  <label><input type="radio" v-model="respuesta" value="sí"> Sí</label>
  <label><input type="radio" v-model="respuesta" value="no"> No</label>

  <p>Opción seleccionada: {{ respuesta }}</p>
</div>

<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const respuesta = ref('sí') // valor inicial
    return { respuesta }
  }
}).mount('#app')
</script>
```



Select con v-model

Los elementos <select> permiten a los usuarios elegir una opción de una lista desplegable.

En JavaScript puro, acceder al valor seleccionado requiere múltiples pasos (acceder al DOM, usar selectedIndex, etc.). Con Vue 3, el proceso se simplifica enormemente gracias a v-model, que enlaza directamente el valor seleccionado con una variable. Por ejemplo:




```
<div id="app">
  <h2>Selecciona una opción</h2>
  <select v-model="opcionSeleccionada">
    <option disabled value="">Seleccione una opción</option>
    <option value="1">Primer valor</option>
    <option value="2">Segundo valor</option>
    <option value="3">Tercer valor</option>
  </select>
  <p>Opción seleccionada: {{ opcionSeleccionada }}</p>
</div>

<script src=""Vue 3 CDN""></script>
<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const opcionSeleccionada = ref("")
    return { opcionSeleccionada }
  }
}).mount('#app')
</script>
```

sustantiva

CON•SENTIDO