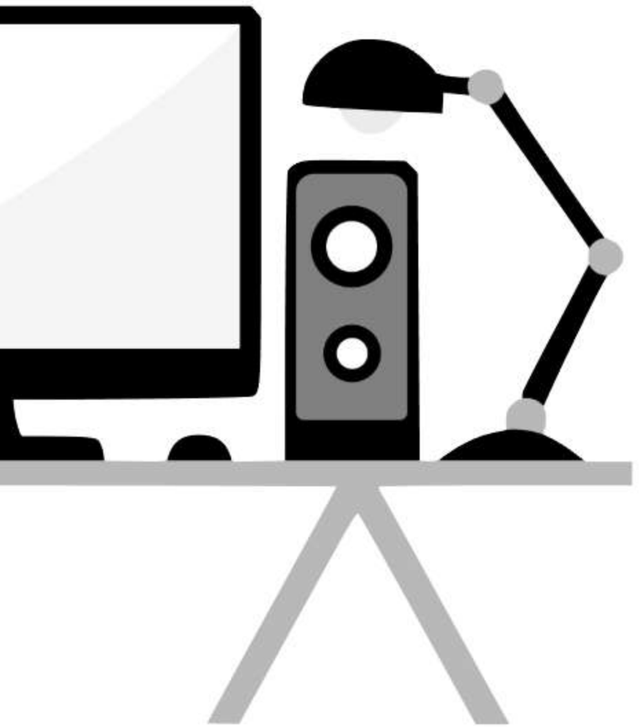
Templates y Rendering en Vue

Lección 02

Competencia General

Utilizar la sintaxis de templates de Vuex para el despliegue de valores y variables que den solución a un requerimiento

Contenidos



Templates y rendering en Vue

1. Templates y rendering en Vue
2. Interpolaciones
3. Directivas
4. Tipos de directivas y sus usos
5. Atributos
6. Modificadores
7. Alias
8. Modo Abreviado
9. Renderización Condicional
10. Diferencia entre v-if y v-show
11. Renderización de una Lista

Templates y rendering en Vue

Vue.js utiliza una sintaxis de template basada en HTML que permite vincular de forma declarativa el DOM a los datos de la aplicación.

Las templates de Vue son HTML válidas y compatibles con el DOM estándar, por lo que pueden analizarse con navegadores y herramientas de desarrollo.

En Vue 3, los templates siguen siendo el enfoque principal para definir la interfaz de usuario, aunque también es posible usar funciones de renderizado (render functions) o JSX cuando se requiere mayor flexibilidad.

```
<div id="app">
  <h1>{{ titulo }}</h1>
  <p>{{ descripcion }}</p>
</div>

<script>
const { createApp } = Vue
createApp({
  data: () => ({
    titulo: 'Bienvenido a Vue 3',
    descripcion: 'Contenido enlazado de forma
                  declarativa'
  })
}).mount('#app')
</script>
```

Templates y rendering en Vue

Vue compila las templates en funciones de renderizado optimizadas que trabajan sobre un DOM Virtual.

Gracias a la reactividad, solo se actualizan los elementos necesarios cuando cambian los datos.

Vue 3 optimiza automáticamente el renderizado detectando elementos que no cambian y eliminando código innecesario para que la aplicación sea más rápida.

```
<div id="app">
  <button @click="contador++">Haz clic ({{ contador }})</button>
</div>

<script>
const { createApp, ref } = Vue
createApp({
  setup() {
    const contador = ref(0)
    return { contador }
  }
}).mount('#app')
</script>
```

Interpolaciones

La forma más básica de **enlace de datos en Vue** es la **interpolación de texto** usando la sintaxis de **mustaches** (dobles llaves {{ }}).

```
<span>Mensaje: {{ msg }}</span>
```

La etiqueta {{ msg }} se reemplaza con el valor de la propiedad **msg** y se actualizará automáticamente cada vez que el dato cambie.

Si necesitas que el contenido se muestre **una sola vez y no se actualice más**, puedes usar la directiva **v-once**, que afecta a todo el nodo donde se aplica.

```
<span v-once>Esto nunca cambiará: {{ msg }}</span>
```

Interpolaciones

Por defecto, los **mustaches** muestran el contenido como **texto plano** y no interpretan etiquetas HTML.

```
<p>Mostrando texto plano: {{ rawHtml }}</p>
```

Si necesitas renderizar el contenido como **HTML real**, debes usar la directiva **v-html**:

```
<p>Mostrando HTML: <span v-html="rawHtml"></span></p>
```

El contenido del se reemplazará por el valor de la propiedad **rawHtml**, interpretando sus etiquetas HTML. En este caso, no se pueden usar enlaces de datos dentro del HTML generado.

Ten en cuenta que v-html no debe usarse para crear plantillas dinámicas, ya que Vue no es un motor de plantillas basado en cadenas. Si necesitas reutilizar partes de la interfaz, lo recomendable es usar componentes.

Directivas

Las **directivas** son atributos especiales en Vue que comienzan con el prefijo **v-**.

- Su valor debe ser una **expresión de JavaScript** (excepto en el caso de v-for, que se verá más adelante).

- Su función es **aplicar efectos reactivos al DOM** cuando cambia el valor de la expresión asociada.

```
<p v-if="seen">Now you see me</p>
```

En este caso, la directiva **v-if** insertará o eliminará el elemento <p> del DOM dependiendo del valor de la propiedad **seen**.

Tipos de directivas y sus usos

Las **directivas** son atributos especiales en Vue que permiten aplicar comportamiento reactivo al DOM. Existen varios **tipos de directivas**, cada una con un propósito diferente.

1. **Directivas condicionales**
   Permiten mostrar u ocultar elementos dependiendo de una condición.

```
<p v-if="mostrar">  Se muestra solo si 'mostrar' es verdadero</p>
<p v-show="mostrar" >  Se muestra con CSS si 'mostrar' es verdadero</p>
```

2. **Directivas de listas**
   Se utilizan para renderizar listas o elementos repetidos.

```
<li v-for="item in items">{{ item }}</li>
```

Tipos de directivas y sus usos

3. Directivas de atributos

Permiten enlazar dinámicamente valores a atributos HTML.

```
<img v-bind:src="imagenUrl">
```

4. Directivas de eventos

Escuchan eventos del DOM y ejecutan métodos.

```
<button v-on:click="hacerAlgo">Haz clic</button>
```

5. Directivas especiales

Algunas directivas tienen comportamientos únicos:

- v-model: enlaza datos de forma bidireccional (inputs, selects).
- v-html: renderiza HTML dinámico.
- v-once: renderiza el contenido solo una vez.

Atributos

Los **mustaches no pueden usarse dentro de los atributos HTML**. Para estos casos se debe utilizar la directiva **v-bind**.

```
<div v-bind:id="dynamicId"></div>
```

En el caso de los **atributos booleanos**, donde su mera presencia significa true, v-bind se comporta de forma especial:

```
<button v-bind:disabled="isButtonDisabled">Button</button>
```

Si isButtonDisabled es null, undefined o false, el atributo disabled **no se incluirá en el botón**.

sustantiva
CON•SENTIDO

Usando Expresiones JavaScript

Hasta ahora solo hemos vinculado datos usando propiedades simples en nuestras templates, pero **Vue permite usar expresiones de JavaScript** dentro de los enlaces de datos (mustaches o directivas como v-bind).

Ejemplos:

- `{{ number + 1 }}` → Muestra el valor de number sumado en 1.
- `{{ ok ? 'YES' : 'NO' }}` → Muestra "YES" si ok es verdadero, de lo contrario "NO".
- `{{ message.split('').reverse().join('') }}` → Muestra el texto de message invertido.
- `<div v-bind:id="'list-' + id"></div>` → Genera un atributo id dinámico como list-1, list-2, etc.

Argumentos

Argumentos

Algunas directivas pueden recibir un **argumento**, indicado con **dos puntos (:)** después del nombre de la directiva. Este argumento le indica a la directiva qué atributo o evento debe manejar.

Ejemplo con v-bind (actualiza atributos HTML):

```
<a v-bind:href="url">Enlace</a>
```

En este caso, el argumento href le indica a la directiva v-bind que debe vincular el atributo href del elemento con el valor de la propiedad url.

Ejemplo con v-on (escucha eventos del DOM):

```
<a v-on:click="doSomething">Haz clic</a>
```

Aquí el argumento click le indica a la directiva v-on que debe escuchar el evento click y ejecutar el método doSomething cuando ocurra.

Modificadores

Los **modificadores** son sufijos con punto (.) que alteran el comportamiento de una directiva.

Ejemplo con v-on:

```
<form v-on:submit.prevent="onSubmit">...</form>
```

El modificador .prevent llama a event.preventDefault() y evita el envío normal del formulario.

Otros ejemplos:

- v-on:click.stop="accion" → Detiene la propagación del evento click.
- v-model.trim="nombre" → Elimina espacios en blanco al inicio y fin del valor.

Modo abreviado

El prefijo v- sirve como una **señal visual** para identificar los atributos de Vue dentro de un template. Esto es útil cuando se agrega comportamiento dinámico a un HTML ya existente, porque hace más claro qué partes son de Vue.

Sin embargo, cuando se construye una SPA (Single Page Application) completa con Vue, el prefijo v- puede sentirse innecesariamente largo en algunas directivas muy utilizadas.

Por esta razón, Vue proporciona **abreviaturas** para dos directivas muy comunes:

v-bind:

```
<!-- Sintaxis completa -->
<a v-bind:href="url">...</a>

<!-- Sintaxis abreviada -->
<a :href="url">...</a>
```

v-on:

```
<!-- Sintaxis completa -->
<a v-on:click="doSomething">...</a>

<!-- Sintaxis abreviada -->
<a @click="doSomething">...</a>
```

Renderización condicional

La **renderización condicional** permite mostrar u ocultar elementos en el DOM dependiendo de una condición booleana.

Ejemplo básico con v-if:

```
<p v-if="activo">Este texto se muestra solo si 'activo' es verdadero</p>
```

También se pueden usar v-else y v-else-if para manejar varias condiciones:

```
<p v-if="estado === 'cargando'">Cargando...</p>

<p v-else-if="estado === 'listo'">Contenido cargado</p>

<p v-else>Error al cargar</p>
```

Diferencia entre v-if y v-show

En Vue existen **dos formas de mostrar u ocultar elementos** de manera condicional: v-if y v-show. Ambas directivas parecen similares, pero funcionan de manera diferente.

¿Por qué hay dos opciones?

Porque cada una resuelve situaciones distintas en términos de **rendimiento** y **frecuencia de cambio**.

Diferencia entre v-if y v-show

v-if

- Añade o elimina el elemento del **DOM real** según la condición
- Si la condición es false, el elemento **no existe en el DOM**.
- Tiene un mayor costo si la condición cambia frecuentemente, porque destruye y vuelve a crear el elemento.

```
<p v-if="mostrar">Este párrafo existe solo si 'mostrar' es verdadero</p>
```

v-show

- No elimina el elemento del DOM, simplemente alterna la propiedad CSS display.
- El elemento siempre existe en el DOM, solo que puede estar oculto.
- Es más rápido si necesitas mostrar u ocultar el elemento muchas veces.

```
<p v-show="mostrar">Este párrafo siempre está en el DOM</p>
```

Renderización de una Lista

En muchas aplicaciones es necesario **mostrar múltiples elementos** a partir de un arreglo u objeto. Vue proporciona la directiva **v-for** para renderizar listas de forma sencilla y reactiva.

Cuando usas v-for, Vue **crea una copia del elemento de la plantilla por cada elemento de la lista**. Si el contenido de la lista cambia, Vue actualizará el DOM automáticamente.

Ejemplo básico con un arreglo:

```
<ul>
  <li v-for="item in items">{{ item }}</li>
</ul>
```

Si items = ['Manzana', 'Pera', 'Plátano'], el resultado será:

- Manzana
- Pera
- Plátano

Renderización de una Lista

Uso de alias para el índice:

Además del elemento, puedes acceder al **índice actual de la iteración**. Esto es útil para mostrar la posición de cada elemento en la lista.

```
<li v-for="(item, index) in items">
  {{ index }} - {{ item }}
</li>
```

Resultado:

- 0 - Manzana
- 1 - Pera
- 2 - Plátano

Iteración sobre objetos:

v-for también permite recorrer **propiedades de un objeto**, accediendo tanto a la clave como al valor.

```
<li v-for="(valor, clave, i) in
usuarios" :key = "clave">
  {{ clave}} - {{ valor}}
</li>
```

Si usuario = { nombre: 'Ana', edad: 25 }, el resultado será:

- nombre: Ana
- edad: 25

Renderización de una Lista

Cuando Vue renderiza listas, necesita **distinguir cada elemento** para optimizar el rendimiento y evitar inconsistencias en el DOM. Por eso se recomienda usar el atributo **:key**.

```
<li v-for="item in items" :key="item.id">
  {{ item.nombre }}
</li>
```

- Cada clave (:key) debe ser **única por elemento** (generalmente un id).
- Si no usas una clave única, Vue puede cometer errores al reutilizar nodos del DOM cuando los datos cambien.
- Evita usar el índice (index) como clave si los elementos pueden cambiar de orden o eliminarse.

Alias en v-for

Cuando usamos la directiva v-for para iterar listas, Vue permite definir **alias** (nombres de variables) para cada elemento y su índice.

Ejemplo básico:

```
<li v-for="item in items">{{ item }}</li>
```

Aquí **item** es el alias que representa cada elemento de items.

También es posible definir un alias para el índice:

```
<li v-for="(item, index) in items">{{ index }} - {{ item }}</li>
```

Alias para propiedades de un objeto:

```
<div v-for="(valor, clave) in objeto">{{ clave }}: {{ valor }}</div>
```

Los alias son útiles para escribir el código de forma más legible y acceder fácilmente a los datos durante la iteración.


sustantiva
CON•SENTIDO