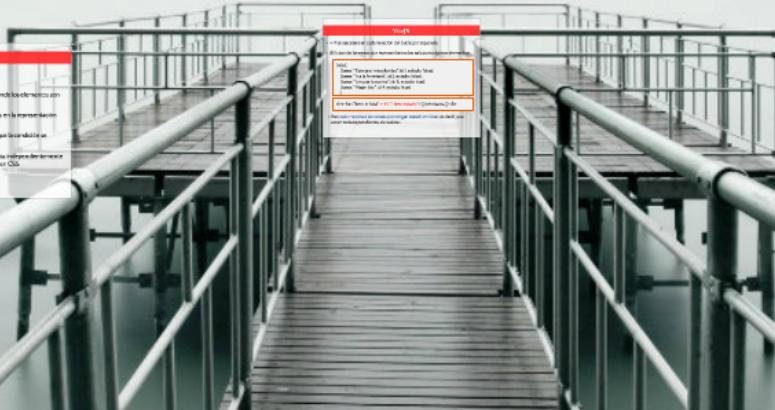


ADM

Dudas?



ADM

Dudas?



Representación Condicional

- Podemos representar elementos y contenidos condicionalmente en función de atributos de datos o declaraciones de JavaScript
- Estos incluyen **v-if**, para mostrar un contenedor si una declaración equivale a verdadero, y **v-else**, para mostrar una alternativa.

v-if

- La directiva **v-if** se usa para representar condicionalmente un bloque.
- El bloque solo **se representará** si la expresión de la directiva **devuelve un valor verdadero**.

VueJS

- En la lógica deberíamos tener:

```
var app= new Vue({  
  el: "#contenedor",  
  data: {  
    mensajito: "Mi super App en Vue",  
    mostrar:true  
  }  
})
```

index.html

```
<p v-if="mostrar">Texto que se mostrará si la variable tiene  
valor TRUE</p>
```

- Se puede usar **v-if** si queremos cambiar más de un elemento con **<template>**, este envoltorio una vez compilado no se verá en el código final.

VueJS

index

```
<template v-if="mostrar">
  <p>Esto se mostrará si la variable mostrar es true</p>
  <p>Puedo tener mas estructura de código</p>
</template>
```

consola

```
▶ <div>...</div>
  <p>Esto se verá si la variable mostrar es true</p>
  <p>Puedo tener mas estructura de código</p>
</div>
```

- Esos párrafos no están dentro de ninguna etiqueta html. (el último div, es el div de cierre donde se está montando vue)

VueJS

- **v-if** no solo funciona con valores booleanos verdadero / falso. Puede verificar si una propiedad de datos es igual a una cadena específica:

index.html

```
<p v-if="selected == 'si' ">Se ve !</p>
```

app.js

```
data: {  
    mensajito : "Mi super App en Vue",  
    mostrar : true,  
    selected : "si"  
}  
})
```

- El atributo **v-if** acepta operadores de JavaScript, por lo que puede verificar que no sea igual, mayor o menor que.



VueJS

index.html

```
<p v-if="valor >= 9">El numero es mayor o igual a 9</p>
<p v-if="valor === 9">El numero es igual a 9</p>
<p v-if="valor != 10">El numero distinto a 10</p>
```

app.js

```
data: {
    valor : 9
}
```

v-else

- Permite hacer un elemento alternativo basado en lo contrario de la declaración v-if.
- Si eso da como resultado verdadero, se mostrará el primer elemento; de lo contrario, el elemento que contiene v-else lo hará.

VueJS

- **v-else** no tiene valor y se coloca dentro de la etiqueta del elemento.

```
<p v-if="mostrar">Texto que se mostrará si la variable tiene valor TRUE</p>
<p v-else>Texto que se mostrará si la variable tiene valor FALSE</p>
```

- El elemento con **v-else** debe seguir directamente al que contiene **v-if**; de lo contrario, la consola arrojará error.

✖ ▶ [Vue warn]: Error compiling template:

v-else used on element <p> without corresponding v-if.

```
6 |          <p v-if="mostrar">Texto que se mostrará si la variable tiene valor TRUE</p>
7 |          <p>Parrafo</p>
8 |          <p v-else="">Texto que se mostrará si la variable tiene valor FALSE</p>
9 |          ^^^^^^^^^^
10 |        </div>
(found in <Root>)
```

v-else-if

- Si tenemos que encadenar condiciones podemos usar la directiva v-else-if.
- Debe seguir inmediatamente a un elemento v-if o v-else-if.

app.js

```
data: {evaluar: 10}
```

index.html

```
<p v-if="evaluar == 1">Es igual a 1</p>
<p v-else-if="evaluar == 5">Es igual a 5</p>
<p v-else-if="evaluar == 15">Es igual a 15</p>
<p v-else-if="evaluar == 10">Es igual a : {{evaluar}}</p>
<p v-else>No coincide, el valor es : {{evaluar}}</p>
```

v-if vs v-show

- v-if es una renderización condicional “real” donde los elementos son destruidos y recreados durante la alternancia.
- v-if también es diferido: si la condición es falsa en la representación inicial, no se hará nada.
- El bloque condicional no se procesará hasta que la condición se convierta en true por primera vez.
- Con v-show el elemento siempre se representa independientemente de la condición inicial, con alternancia basada en CSS.

VueJS

- La documentación sugiere v-show para cambiar algo muy a menudo, y elegir v-if si es poco probable que la condición cambie en el tiempo de ejecución.

Iterar Arrays

v-for

- Permite representar una lista de elementos basada en una matriz.
- Requiere una sintaxis especial en forma de **item in items**.
- Los **items** son la matriz de datos y el **item** es un alias para el elemento de matriz que se está iterando:

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

```
data:{  
  peliculas:[ "Mortal Kombat",  
             "Sherlock Holmes",  
             "Kung Fusion" ]  
}
```

Iterar Array de Objetos

```
juegos:[  
    {nombre: "Super Mario Bros", anio: 1985, calificacion: 9 },  
    {nombre: "Mortal Kombat", anio: 1992, calificacion:10 },  
    {nombre: "Street Fighter", anio :1987, calificacion: 8}  
]
```

```
<ul>  
    <li v-for="item in juegos">{{item.nombre}} : {{item.anio}}</li>  
</ul>
```

- Por cada propiedad del objeto que se quiera mostrar se debe aclarar en la sintaxis del html.

VueJS

- Se puede usar una etiqueta <template> con v-for para renderizar un bloque de varios elementos.

index

```
<ul>
  <template v-for="item in juegos">
    <li>{{ item.nombre }}</li>
    <li>Calificacion : {{item.calificacion}}</li>
  </template>
</ul>
```

consola

```
▼ <ul>
  <li>Super Mario Bros</li>
  <li>Calificacion : 8</li>
  <li>Mortal Kombat</li>
  <li>Calificacion : 9</li>
  <li>Street Fighter</li>
  <li>Calificacion : 7</li>
</ul>
```

VueJS

- Dentro de los bloques **v-for** tenemos acceso completo a las propiedades del ámbito principal.
- v-for admite un segundo argumento opcional para el **índice** del elemento actual.

```
<ul>
  <li v-for="(item, index) in juegos">Indice: {{index}}, {{item.nombre}} : {{item.anio}}</li>
</ul>
```

- 
- Indice: 0, Super Mario Bros : 1985
 - Indice: 1, Mortal Kombat : 1992
 - Indice: 2, Street Fighter : 1987

VueJS

- Puede usar **of** como delimitador en lugar de **in**, de modo que esté más cerca de la sintaxis de JavaScript para los iteradores:

```
<ul>
  <li v-for="item of juegos">{{item.nombre}} : {{item.anio}}</li>
</ul>
```

Iterar Objetos

```
obj: {
  animal: "gato", raza: "Persa", edad:4
}
```

VueJS

```
<ul>
  <li v-for="value of obj">{{value}} </li>
</ul>
```

- gato
- Persa
- 4

- En caso de querer mostrar también las propiedades de los objetos se puede pasar como segundo argumento **key**.

```
<ul>
  <li v-for="(valor, key) of obj">{{key}} : {{valor}} </li>
</ul>
```

- Clave : animal, Valor: gato
- Clave : raza, Valor: Persa
- Clave : edad, Valor: 4

VueJS

- Además podríamos pasarle un tercer argumento para mostrar el índice del elemento

```
<ul>
  <li v-for="(valor, key, index) of obj"> {{index}} --> Clave : {{key}}, Valor: {{valor}} </li>
</ul>
```

- 0 --> Clave : animal, Valor: gato
- 1 --> Clave : raza, Valor: Persa
- 2 --> Clave : edad, Valor: 4

- Al iterar sobre un objeto, el orden se basa en el orden de enumeración de claves. (Puede que no sea consistente en todas las implementaciones del motor de JavaScript.)

VueJS

- Podemos sugerirle a Vue un **atributo key único** para cada elemento para que pueda rastrear la identidad de cada nodo y reordenar y reutilizar los elementos existentes.
- Tenemos que usar **v-bind** para enlazarlo **v-bind:key**
- Se recomienda que se use el atributo **key** al usar **v-for** para aprovechar el **rendimiento de vue**

```
lista:[  
    {tarea: "Comprar más plantas ", id:1},  
    {tarea: "Ir a la ferretería", id:2},  
    {tarea: "Limpiar la cocina", id: 3},  
    {tarea: "Hacer bici", id:4}  
]
```

VueJS

```
<ul>
  <li v-for="item in lista" :key="item.id" > {{item.tarea}} </li>
</ul>
```

- Comprar más plantas
- Ir a la ferretería
- Limpiar la cocina
- Hacer bici

v-for en combinación con v-if

- La documentacion oficial no recomienda usar v-if y v-for
- Cuando existen en el mismo elemento, **v-for tiene prioridad más alta que v-if.**

VueJS

- v-if se ejecutará en cada iteración del bucle por separado.
- Útil cuando tenemos que representar nodos solo para algunos elementos.

```
lista:[
```

```
  {tarea: "Comprar más plantas", id:1, estado: false},  
  {tarea: "Ir a la ferretería", id:2, estado: false},  
  {tarea: "Limpiar la cocina", id: 3, estado: true},  
  {tarea: "Hacer bici", id:4, estado: true}
```

```
]
```

```
<li v-for="item in lista" v-if="! item.estado"> {{ item.tarea }}</li>
```

- Esto **solo mostrará las tareas que tengan estado en false**, es decir, que están todavía pendientes de realizar.



Dudas?

ADM

Dudas?

