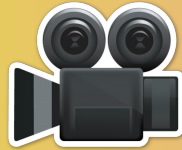




Clase 3. Vue JS

Directivas condicionales, estructurales y de atributo

RECORDÁ PONER A GRABAR LA CLASE





OBJETIVOS DE LA CLASE

- Ejecutar directivas condicionales, estructurales y de atributos para formar vista dinámica a nivel de contenido y estilos.

CRONOGRAMA DEL CURSO

Clase 2



Proyecto CDN y primeras directivas



PROYECTO CON
HTML, CSS Y JS



USO DE DATA BINDING



IMPLEMENTAR VUE CDN UN
CONTADOR

Clase 3



Directivas estructurales, condicionales y de atributo



OCULTAR Y MOSTRAR UN
BOTÓN



TABLA CON DIRECTIVAS

Clase 4



Componentes



CREANDO COMPONENTES

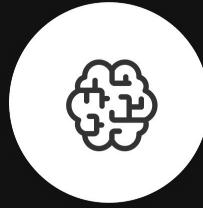


PASAJES DE PARÁMETROS



PROYECTO VUE CDN

DIRECTIVAS CONDICIONALES



¡PARA PENSAR!

*La clase anterior aprendimos los tipos de directivas.
¿Recuerdas cuáles son las directivas condicionales?*

CONTESTA LA ENCUESTA DE ZOOM



DIRECTIVAS CONDICIONALES

Las directivas condicionales nos permiten establecer condiciones que se aplican directamente sobre nuestras etiquetas HTML. Y cuando de condiciones hablamos, nos referimos a los clásicos condicionales de **JS** o de cualquier otro lenguaje de programación.

Veamos cómo Vue simplifica este tipo de sentencias.



CONDICIONAL V-IF

CONDICIONAL V-IF



Con el condicional `v-if` podemos mostrar o no elementos en un documento HTML dependiendo del valor de una variable o del resultado de una expresión.

v-if actúa tal como funciona un if convencional en JS, con la diferencia de que lo usamos directamente como un atributo más de un tag HTML.



CONDICIONAL V-IF

```
var app = new Vue({  
  el: '#app',  
  data: {  
    mostrarSaludo: false,
```

Del lado HTML definimos un bloque `<div>` el cual mostrará su contenido solo si el resultado de la expresión `v-if`, es verdadero.

Podemos, por ejemplo, definir una variable con un valor **booleano** para plantear la condición `v-if` en base a éste.

```
<div id="app">  
  <div v-if="mostrarSaludo">  
    <p> Bienvenido, {{ nombreCompleto }}</p>  
  </div>  
  <p>{{ totalNot }} notificaciones  
  pendientes.</p>
```



DÓNDE APLICAR V-IF

El condicional `v-if` es aplicable a cualquier elemento HTML: `<h1>`, `<p>`, ``, incluso dentro de las etiquetas de un componente personalizado.

```
<mi-componente v-if="condicion">  
  <div>...</div>  
</mi-componente>
```

También ten presente que, al aplicarlo en un `<div>` o `<contenedor>`, los tags definidos de forma anidada en estos se visualizarán u ocultarán de acuerdo al resultado de la expresión evaluada.

CONDICIONAL V-ELSE

CONDICIONAL V-ELSE

Y como no hay **if** sin **else**, tampoco hay **v-if** sin **v-else**.

Este último nos permite plantear un camino alternativo a **v-if** en el caso que la condición de este no se cumpla. Al utilizar **v-else**, podemos plantear un análisis de condición desde las más básicas hasta aquellas más complejas que integren operadores **and** u **or**, entre otros.



CONDICIONAL V-ELSE

Un claro ejemplo que combina **v-if** con **v-else**, es evaluar si un usuario está logueado o no, y con ello mostrarle contenido o invitarlo a identificarse.

```
index.html M  app.js M x
js > app.js > [e] app > data
1  var app = new Vue({
2    el: '#app',
3    data: {
4
5      userName: 'JOBS, Steve',
6      totalNot: 21,
7      nombreCompleto: "Lee Pace",
8      message: 'Hola Coders!',
9      a: 21,
10     b: 3,
11     htmlCard: '<div class="card black white-text"></div>',
12     popcorn: {
13       portada: "images/movies/jobs.jpg",
14       descripcion: "Biopic de Steve Jobs basada en su libro biográfico."
15     }
16   }
17 }
```

CONDICIONAL V-ELSE-IF

CONDICIONAL V-ELSE-IF

A su vez, `v-if` y `v-else` pueden tener una tercera opción de la mano de `v-else-if`. En el hipotético caso que la condición de `v-if` no se cumpla como así tampoco la condición de `v-else`, tenemos la posibilidad de plantear una opción adicional mediante esta alternativa, tal como ocurre en cualquier otro lenguaje de programación que integra el clásico `else-if`.

CONDICIONAL V-SHOW

CONDICIONAL V-SHOW



El condicional `v-show` tiene un comportamiento similar al del condicional `v-if` , aunque este último no genera los tags HTML si no se cumple la condición que está evaluando. Por parte de `v-show`, este crea el tag HTML y lo oculta mediante el estilo CSS `display:none;` en el caso que no deba mostrarlo.

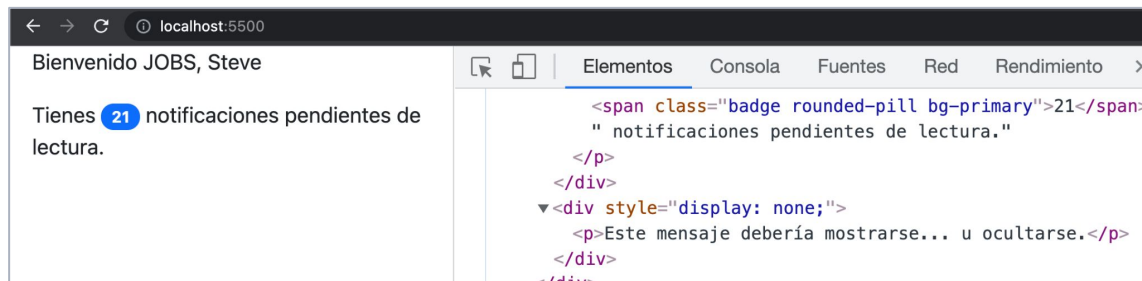


CONDICIONAL V-SHOW

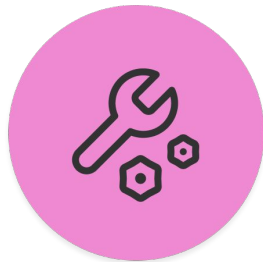
```
mostrarDiv: false, //Código JavaScript

<div v-show="mostrarDiv"> <!-- Código HTML -->
  <p>Este mensaje debería mostrarse... u ocultarse.</p>
</div>
```

Diseñando un ejemplo similar al representado 🖱️ podrás evaluar el resultado de este código en **Herramientas para desarrollador > Elementos**.



¡VAMOS A PRACTICAR LO VISTO!

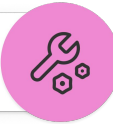


BOTONES CON CONDICIONALES

Integra elementos HTML con interactividad a partir de un condicional.

Tiempo estimado: **10 minutos**

CODER HOUSE



BOTONES CON CONDICIONALES

Genera una aplicación Vue que contenga un elemento HTML
button y otro del tipo paragraph.

Deberás diseñar un algoritmo en Vue CDN el cual, al pulsarlo,
deberá mostrar u ocultar el elemento HTML paragraph.

Tiempo estimado: **10 minutos**



BREAK

¡5/10 MINUTOS Y VOLVEMOS!

BUGLE V-FOR

BUCLE V-FOR

Como habrás podido dilucidar por su nombre, el bucle **v-for** es un condicional que permite **iterar sobre un array de elementos** u objetos del tipo JSON. Durante el proceso de iteración, se toma cada elemento individual y se representa en pantalla.

👁️ Veamos a continuación un ejemplo más claro de ello.



BUCLE V-FOR

```
data: {  
  mostrarDiv: false,  
  estoyLogueado: true,  
  userName: 'JOBS, Steve',  
  totalNot: 21,  
  tareas: [{id: 1, desc: "Revisar hardware faltante"},  
            {id: 2, desc: "Validar user testing"},  
            {id: 3, desc: "Verificar versión del S.O."},  
            {id: 4, desc: "Instalar S.O. en SD-Card"},  
            {id: 5, desc: "Realizar downgrade de S.O."},  
            {id: 6, desc: "Instalar Office beta"}]  
},
```

→ Partamos de un **array de objetos** como podría ser una **lista de tareas**.

La misma contiene N cantidad de elementos que pueden aumentar o decrecer según necesidad, y a los cuales buscamos reflejar en una lista visible.



BUCLE V-FOR

Ctrl + / Bootstrap v5.1

Basic example

The most basic list group is an unordered list with list items and the proper classes. Build upon it with the options that follow, or with your own CSS as needed.

An item

A second item

A third item

A fourth item

And a fifth one

On this page

- Basic example
- Active items
- Disabled items
- Links and buttons
- Flush
- Numbered
- Horizontal
- Contextual classes
- With badges
- Custom content
- Checkboxes and radios
- Sass
- Variables
- Mixins
- Less

```
<ul class="list-group">
  <li class="list-group-item">An item</li>
```

Copy

→ Aquí es donde el bucle **v-for** permite iterar sobre la misma para renderizar el total de sus elementos, sin tener que indicar cuántas veces debemos repetir el proceso.

→ Sumemos Bootstrap al ejemplo: utilicemos el componente [list-group basic](#) para listar elementos de la lista.



BUCLE V-FOR

```
...  
<ul class="list-group">  
  <li class="list-group-item">  
    ...  
  </li>  
</ul>  
...
```

→ Agregamos la lista desordenada al documento HTML y, dentro de está, un solo elemento ``.

→ Agregamos en este último elemento el bucle `v-for`, quien se ocupará de repetirlo una vez por cada elemento de la lista de tareas.



BUCLE V-FOR

Con este bucle, recorreremos el array de objetos `tareas`.

→ Para generar la iteración sobre este, agregamos un paréntesis donde definimos dos parámetros: `item` e `i`.

```
...  
<ul class="list-group">  
  <li class="list-group-item" v-for="(item, i) in tareas" :key="i">  
    ...  
  </li>  
</ul>  
...
```



BUCLE V-FOR

item es el nombre de la variable que asume el valor de cada elemento individual, que contiene el array de objetos.

i es la variable que inicializa el contador del bucle. El mismo se inicia en **0** (*cero*) e incrementa **1** dígito por cada iteración. Así, hasta finalizar el recorrido de todo el array.

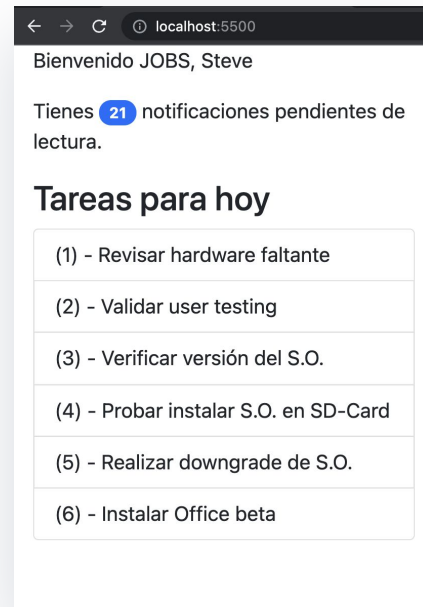
```
<li class="list-group-item" v-for="(item, i) in tareas" :key="i">  
  ... <!-- Aquí reflejamos cada tarea a mostrar -->  
</li>
```

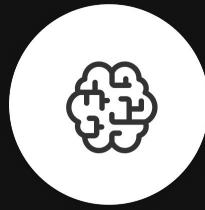


BUCLE V-FOR: resultado

```
<li class="list-group-item" v-for="(item, i) in tareas"
:key="i">
  ({{ item.id }}) - {{ item.desc }}
</li>
```

→ Dentro de los tags `` definimos mediante *double moustache* el o los elementos del array que deseamos renderizar en pantalla. Así llegamos al resultado esperado.





¡PARA PENSAR!

¿Es posible anidar bucles v-for?

CONTESTA EN EL CHAT DE ZOOM



ANIDAR BUCLES V-FOR

¡Así es! **Vanilla JavaScript** permite anidar los ciclos **for** tantas veces sea necesario, sin quitar el ojo de la performance de la aplicación web.

Y como Vue JS es un framework basado en el lenguaje JS, por herencia, también permite anidar ciclos v-for, para renderizar elementos HTML complejos en pantalla.



DIRECTIVAS DE ATRIBUTOS CSS

DIRECTIVAS DE ATRIBUTOS CSS

Además de aplicar distintos niveles de lógica a nuestras aplicaciones web, Vue JS también permite manejar desde su framework todo lo concerniente a las directivas de atributos CSS, ya sea **aplicando propiedades y valores**, como también determinando **qué clases CSS utilizar**.

Veamos cómo sacar provecho de esto.

DIRECTIVA CLASS

DIRECTIVA CLASS

Vue integra una directiva la cual permite manejar diferentes clases CSS. Estas pueden ser creadas dentro de un array de objetos JSON para luego reflejarlas en el elemento HTML utilizando la directiva `:class`, en lugar de invocar al atributo `class` propio de cada elemento HTML.



DIRECTIVA CLASS

En **data** declaramos un objeto del tipo JSON, al cual le definimos propiedades y valores. Cada propiedad corresponde a una clase CSS (*del framework en uso o creada por nosotros*) y el valor debe referir a un booleano.

```
data: {  
  ...,  
  cssClasses: {  
    "bg-warning": true,  
    "text-secondary": true,  
  },  
  ...  
}
```



DIRECTIVA CLASS

```
"bg-warning": true
```

Como una propiedad de objeto JSON no puede contener guiones en su nombre, debemos **encerrarla entre comillas dobles** para que Vue la interprete como tal.

El valor booleano es simplemente para **indicarle si debe estar** (**true**), o no (**false**), presente en el conjunto de clases CSS de dicho elemento HTML.



DIRECTIVA CLASS

```
<div :class="cssClasses">  
  <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Consequatur  
sequi nesciunt, dolores voluptatum porro dignissimos, maiores quas vero hic at  
fugiat sunt optio esse praesentium? Hic totam porro pariaturn voluptas.</p>  
</div>
```

Una vez definido el objeto JSON con las clases CSS necesarias, nos queda incorporar en el elemento HTML en cuestión, la directiva `:class` con el objeto JSON que contiene el set de clases a aplicar.



DIRECTIVA CLASS



Así conseguimos reflejar todas las clases CSS en un elemento HTML utilizando la directiva class de Vue JS.

Como ventaja, podemos destacar la posibilidad de disponer de las clases CSS en un objeto JSON para poder manejar más fácilmente su interacción y actualización para con los documentos HTML.

DIRECTIVA STYLE



DIRECTIVA STYLE

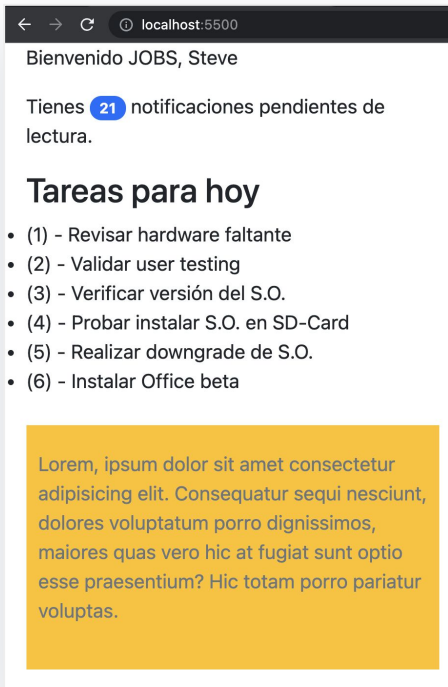
Vue JS también da soporte a la integración de una directiva **:style**, que permite aplicar estilos CSS directamente sobre los elementos HTML.

En este caso, integramos un array de elementos CSS definidos por **propiedad-valor**, tal como vemos en el siguiente ejemplo de código.

```
<div :class="cssClasses" :style="{ 'padding': '20px 10px' }">  
  <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit  
  ...
```



DIRECTIVA STYLE



Finalmente, conseguimos el resultado esperado, tal como podemos apreciar en la captura de pantalla. Si bien es posible manejar clases y estilos a través de las directivas homónimas, y hasta combinar ambas opciones, siempre es recomendable concentrar los estilos CSS en una clase, para luego aplicarlo en el o los elementos HTML.

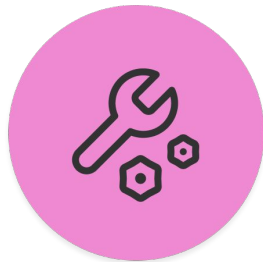


TABLA DINÁMICA CON DIRECTIVAS

Integra elementos HTML con interactividad a partir de un condicional.

Tiempo estimado: **20 minutos**

CODER HOUSE



TABLA DINÁMICA CON DIRECTIVAS

Afiancemos las directivas Vue aprendidas hasta ahora creando una tabla dinámica en la vista HTML que represente la información contenida en un array de Objetos.

A su vez, dicha tabla deberá contener estilos o clases CSS que permitan diferenciar las filas pares de las impares, aplicando un color gris a las primeras y blanco en las segundas.

Tiempo estimado: **20 minutos**

CODER HOUSE

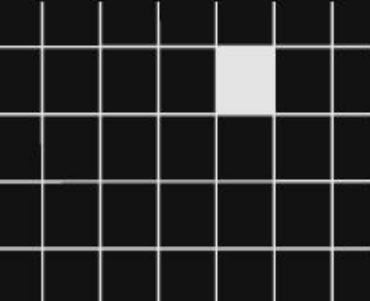
¿PREGUNTAS?





¡MUCHAS GRACIAS!

Resumen de lo visto en clase hoy:

- Condicionales: v-show v-if v-else-if v-else.
 - Estructurales: v-for.
 - Directivas de atributos CSS: :style :class
- 



OPINA Y VALORA ESTA CLASE

#DEMOCRATIZANDO LA EDUCACIÓN

CODER HOUSE