



# Plantillas y rendering en Vue

Renderización

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

- Unidad 1: Introducción a Componentes Web y Vue Js
- Unidad 2: Binding de formularios
- Unidad 3: Templates y rendering en Vue
- Unidad 4: Manejo de eventos y reutilización de componentes
- Unidad 5: Consumo de datos desde una API REST



Te encuentras aquí



## ¿Qué aprenderás en esta sesión?

- *Utiliza renderización condicional para condicionar el despliegue de un valor o variable de acuerdo a los requerimientos.*
- *Utiliza renderización de lista para el despliegue de una estructura de elementos múltiples que dé solución a un problema.*

¿Cómo modificarías los  
estilos de un elemento  
HTML con Js puro?

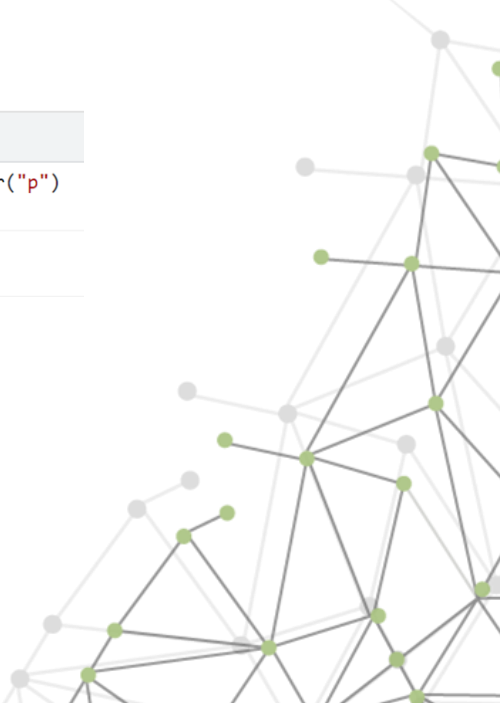


# Atributo style

## Modificar estilos con js

Para modificar los estilos de un elemento HTML con Js debemos acceder al atributo style del elemento y asignarle valores a los diferentes atributos de CSS

Lorem ipsum dolor sit amet  
consectetur, adipisicing elit. Quam  
debitis architecto velit reiciendis  
adipisci, itaque corporis molestiae,  
quis, accusamus expedita voluptates.  
Illo ullam ut architecto accusamus  
officiis at, quasi corrupti!



```
top Filter
1 Issue: 1
> const parrafo = document.querySelector("p")
< undefined
> parrafo.style.color = "red"
< 'red'
> |
```

# Atributo style

## Modificar estilos con js

El objeto **style** contiene todas las propiedades de CSS.

En JavaScript, los atributos que escribimos en css cuyos nombres están separados por un guión (-), son expresados en camelCase

font-size === fontSize

```
▼ CSSStyleDeclaration {0: 'color', acc
  0: "color"
  accentColor: ""
  additiveSymbols: ""
  alignContent: ""
  alignItems: ""
  alignSelf: ""
  alignmentBaseline: ""
  all: ""
  animation: ""
  animationDelay: ""
  animationDirection: ""
  animationDuration: ""
  animationFillMode: ""
  animationIterationCount: ""
  animationName: ""
  animationPlayState: ""
  animationTimingFunction: ""
  appRegion: ""
  appearance: ""
  ascentOverride: ""
  aspectRatio: ""
  backdropFilter: ""
  backfaceVisibility: ""
  background: ""
  backgroundAttachment: ""
  backgroundBlendMode: ""
  backgroundClip: ""
  backgroundColor: ""
  backgroundImage: ""
```

```
color: "red"
colorInterpolation: ""
colorInterpolationFilters: ""
colorRendering: ""
colorScheme: ""
columnCount: ""
columnFill: ""
columnGap: ""
columnRule: ""
columnRuleColor: ""
columnRuleStyle: ""
columnRuleWidth: ""
columnSpan: ""
columnWidth: ""
columns: ""
contain: ""
containIntrinsicBlockSize: ""
containIntrinsicHeight: ""
containIntrinsicInlineSize: ""
containIntrinsicSize: ""
containIntrinsicWidth: ""
container: ""
containerName: ""
containerType: ""
content: ""
contentVisibility: ""
counterIncrement: ""
counterReset: ""
counterSet: ""
cursor: ""
cx: ""
```

¿Qué instrucción escribirías  
para asignarle sombra a un  
párrafo desde js?



**`/* Renderización condicional*/`**

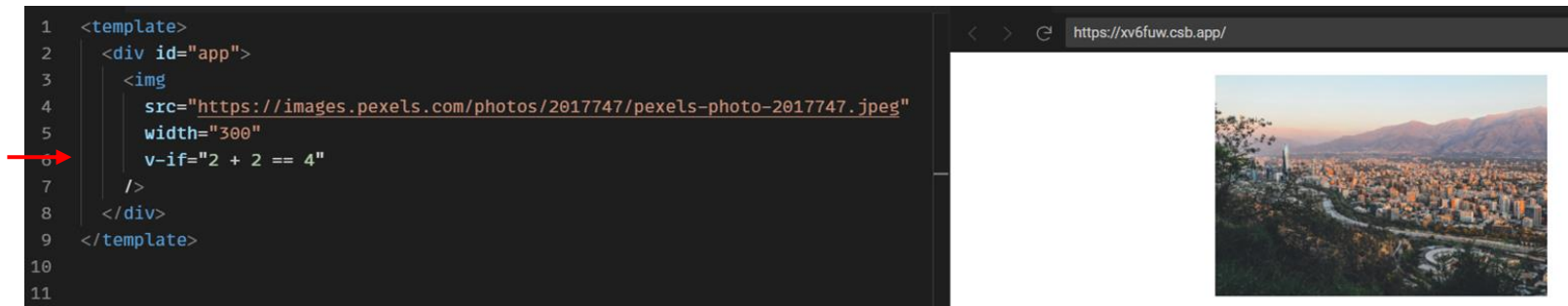


# Renderización condicional

## ¿Qué es?

Existe una directiva que nos permite decidir si un elemento se renderiza o no.

Esta directiva es **v-if**



El valor de esta directiva debe ser una expresión o un valor booleano

# Renderización condicional

¿Qué es?

De manera que si le entregamos una expresión **falsa**, el elemento no se renderizará.

```
1 <template>
2 <div id="app">
3   
8 </div>
9 </template>
```

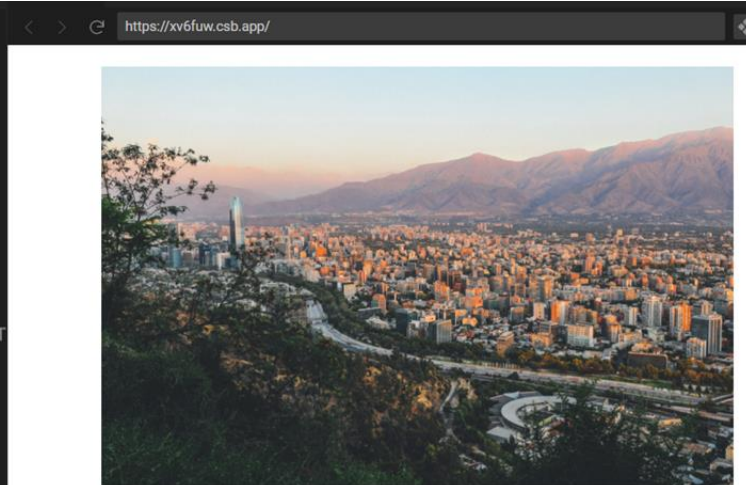
< > ↻ <https://xv6fuw.csb.app/>

# Renderización condicional

¿Qué es?

Podemos asignar como valor una variable del estado y funcionará de la misma manera:

```
1 <template>
2   <div id="app">
3     
8   </div>
9 </template>
10
11 <script>
12 export default {
13   name: "App",
14   data() {
15     return {
16       → show: true,
17     };
18   },
19 };
20 </script>
```



{desafío}  
latam\_

Ahora lo ves,  
ahora no los ves



# Renderización condicional

## Ejercicio

Vamos a crear juntos una aplicación Vue Js que contenga un input de tipo checkbox que le permita al usuario renderizar o no un elemento HTML

Tests

https://xv6fuw.csb.app/

Mostrar ☒

Harry Houdini

Tests

https://xv6fuw.csb.app/

Mostrar ☐



**/\* Diferencia entre v-if y v-show \*/**

# Renderización condicional

## *v-show*

Existe otra directiva llamada **v-show** cuyo comportamiento es parecido al **v-if** pero se diferencian técnicamente

*v-show* !== *v-if*



# Renderización condicional

*v-show* !== *v-if*

En el caso del **v-if** la etiqueta HTML se puede encontrar en el inspector de elementos.

Mostrar 

Harry Houdini

```
▼ <div id="app">  
  " Mostrar "  
  <input type="checkbox">  
  <p>Harry Houdini</p>  
</div>
```



# Renderización condicional

*v-show* !== *v-if*

Si el **v-if** recibe una expresión o un valor **false**, el elemento no se renderizará, es decir que ni siquiera existirá entre en el árbol HTML interpretado por el navegador.

Mostrar ☐

```
▼<div id="app">  
  " Mostrar "  
  <input type="checkbox">  
</div>
```


# Renderización condicional

*v-show* !== *v-if*

En cambio, el **v-show** en vez de evitar la renderización del elemento, lo que hace es aplicar un **display: none**

Mostrar ☐

```
▼ <div id="app">
  " Mostrar "
  <input type="checkbox">
  <p style="display: none;">Harry Houdini</p>
</div>
```



# Renderización condicional

## *v-else*

Otra directiva relacionada a la renderización condicional es el **v-else**

Esta directiva en particular solo se utiliza en conjunto con el **v-if**

```
<div id="app">
  Mostrar
  <input v-model="show" type="checkbox" />

  <p v-if="show">Harry Houdini</p>
  <p v-else>Algo salió mal con el truco de magia :/</p>
</div>
```

<https://xv6fuw.csb.app/>

Mostrar ☐

Algo salió mal con el truco de magia :/

¿En qué casos utilizarías  
el v-if y en qué caso  
preferirías el v-show?

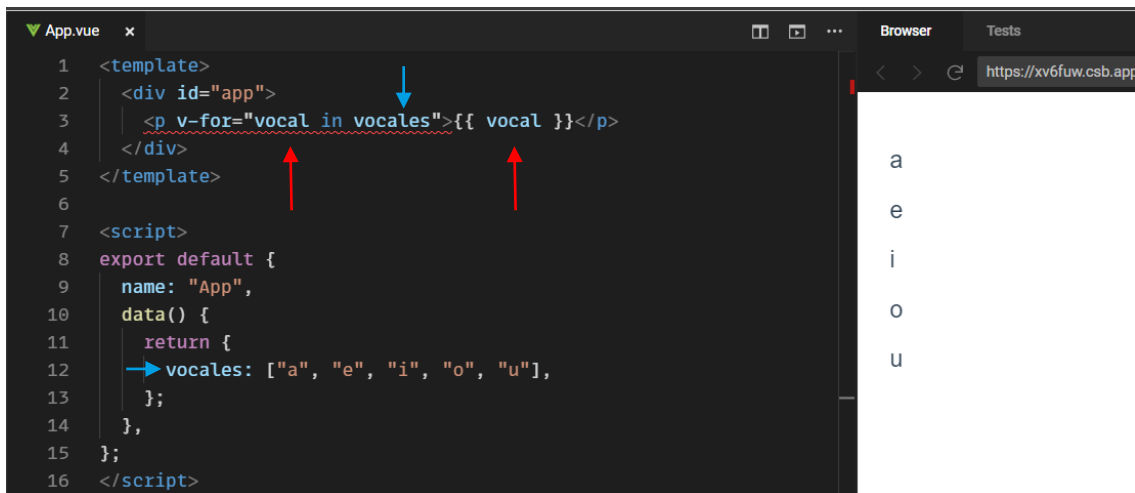


***/\* Renderización de una lista \*/***

# Renderización de una lista

## *v-for*

Cuando necesitamos reutilizar un mismo elemento, grupo de elementos o en sí un componente, podemos ocupar la directiva **v-for**.



```
1 <template>
2   <div id="app">
3     <p v-for="vocal in vocales">{{ vocal }}</p>
4   </div>
5 </template>
6
7 <script>
8   export default {
9     name: "App",
10    data() {
11      return {
12        →vocal: ["a", "e", "i", "o", "u"],
13      };
14    },
15  };
16 </script>
```

a  
e  
i  
o  
u

# Renderización de una lista

## *v-for*

Cuando necesitamos reutilizar un mismo elemento, grupo de elementos o en sí un componente, podemos ocupar la directiva **v-for**.



```
1 <template>
2   <div id="app">
3     <p>Días de la semana</p>
4     <ul>
5       <li v-for="dia in diasDeLaSemana">{{ dia }}</li>
6     </ul>
7   </div>
8 </template>
9
10 <script>
11 export default {
12   name: "App",
13   data() {
14     return {
15       diasDeLaSemana: [
16         "Lunes",
17         "Martes",
18         "Miércoles",
19         "Jueves",
20         "Viernes",
21         "Sábado",
22         "Domingo",
23       ],
24     };
25   },
26 };
27 </script>
```

Días de la semana

- Lunes
- Martes
- Miércoles
- Jueves
- Viernes
- Sábado
- Domingo

# Renderización de una lista

## *v-for*

Es posible que en tu editor de código aparezca este subrayado indicando que algo falta o está mal con el **v-for**.

```
<p v-for="vocal in vocales">{{ vocal }}</p>
```

No hay por qué preocuparse, por ahora lo dejaremos así y lo veremos cómo corregirlo más adelante.



# Renderización de una lista

## *v-for*

La anatomía que usaremos en esta directiva es la siguiente

Parámetro que contiene  
el valor en cada iteración



```
<p v-for="vocal in vocales">{{ vocal }}</p>
```



Arreglo u objeto en  
el estado a recorrer

# Ejercicio propuesto



## Ejercicio propuesto

Crea un selector cuyas opciones sean iteraciones de un arreglo con el v-for.



¿Cuál directiva  
te ha gustado más  
hasta ahora y por qué?





## Próxima sesión...

- *Guía de ejercicios.*

**{desafío}**  
**latam\_**

*Academia de  
talentos digitales*

