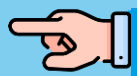
Manejo de eventos y reutilización de componentes

Evento DOM

{desafío} latam_

Implementar interacción en los elementos de una interfaz web utilizando manejo de eventos Vue para dar solución a un requerimiento y componentes reutilizables utilizando el framework Vue para el desarrollo de una aplicación web mantenible en el tiempo.

- 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

{desafío} latam_

¿Qué aprenderás en esta sesión?

- *Reconoce los principales eventos manejables de un DOM.*
- *Enlaza una función a un evento para dar interacción a un elemento de acuerdo a lo solicitado.*
- *Utiliza modificadores de eventos para dar solución a problemáticas comunes en una vista.*
- *Aplica estilos condicionados a un componente en base a un evento para resolver un problema planteado.*

¿Cómo asignamos eventos a elementos del DOM usando JavaScript puro?

Eventos en Js puro

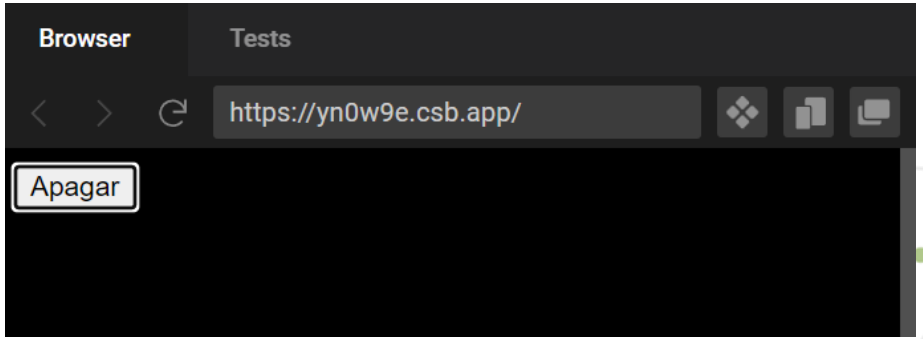
addEventListener

Para agregar eventos con JavaScript puro podemos utilizar el **addEventListener**.

```
<button>Apagar</button>

<script>
 const btn = document.querySelector("button");

 btn.addEventListener("click", () => {
   document.body.style.background = "black";
 });
</script>
```



En Vue Js la asignación de eventos se trabaja de forma diferente y es lo que veremos en esta clase

{desafío} latam_

/* Eventos */

Eventos

Eventos en el DOM

Las interacciones en una página web se interpretan a nivel de código como **eventos**. Existen muchos tipos de eventos disponibles en JavaScript. De los más populares y utilizados tenemos los siguientes:

Mouse
mouseover
mouseleave
mousemove
click
dbclick

Inputs/Keyboard
change
blur
focus
keypress

Otros
scroll
load

Eventos

Uso de eventos en Vue

En Vue Js, para asignarle un evento a un elemento debemos usar el arroba(@) seguido del nombre del evento.

@<nombre del evento>

Los eventos finalmente son gatilladores de funciones y la manera de declarar qué queremos que suceda cuando ocurra un evento es escribiendo el nombre del método como valor:

```
<etiqueta @<nombre del evento>="<nombre del método>" >  </etiqueta>
```

Eventos

methods

Ahora, **¿Dónde definimos el método que queremos ejecutar al momento que el evento suceda?**

Para declarar métodos en un componente debemos **agregar otro atributo** en el objeto que exportamos dentro del script.

- Este atributo se llama **methods**.

Pongamos a prueba los métodos y los eventos en Vue haciendo aparecer una ventana emergente al hacer click en un elemento.

Ejercicio guiado 1

{desafío} latam_

Eventos
methods

1. Agrega el atributo **methods** y dentro agrega un método **saludar** qué muestre una ventana emergente diciendo **Hello vue Js**

```
<script>
export default {
 name: "App",

 methods: {
   saludar() {
     alert("Hello Vue Js");
   },
 },

};
</script>
```

Eventos
methods

2. Agrega en el template un título que con un evento click que llame a un método saludar

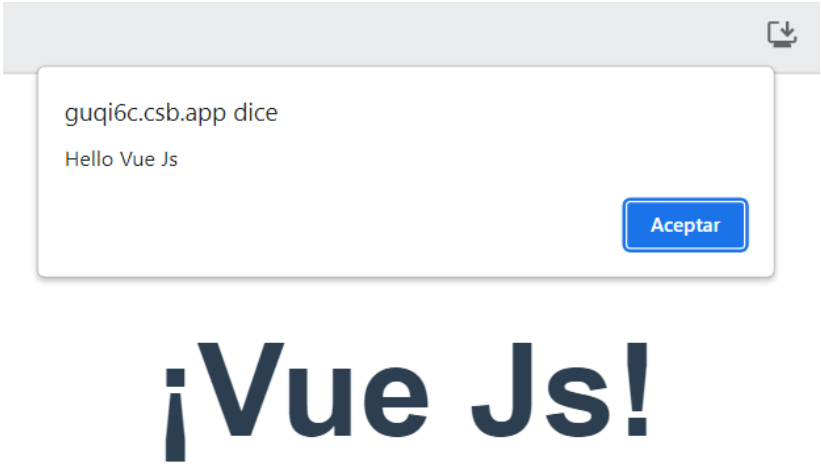
```
<h1 @click="saludar" >¡Vue Js!</h1>
```

Eventos
methods

Si probamos el ejemplo anterior podemos ver que al hacer un click en el título que teníamos aparece la ventana emergente diciendo **Hello Vue Js**:

```
<template>
  <div id="app">
    <h1 @click="saludar">¡Vue Js!</h1>
  </div>
</template>

<script>
export default {
  name: "App",
  methods: {
    saludar() {
      alert("Hello Vue Js");
    },
  },
};
</script>
```



Ejercicio propuesto 1

Ejercicio propuesto

¡Sigamos practicando!

De acuerdo al ejercicio guiado 1:

- Crea un método que imprima por consola tu nombre.
- El método debe ser invocado a través del evento ***dbclick*** en una imágen ubicada en el template.

Eventos

methods

- Dentro de los métodos tenemos acceso al operador **this.**
- Usando este operador podemos **acceder a las variables del estado** e incluso a **otros métodos**.
- Veamos cómo funciona con una pequeña aplicación de tipo TODO

Ejercicio guiado 2

Eventos
methods

1. Agrega un **input** y un **button** para escribir una tarea a agregar

```
<input/>
<button>agregar</button>
```

1. En el **input** agrega una directiva que enlace el input a una variable **nuevaTarea**

```
<input v-model="nuevaTarea" />
<button>agregar</button>
```

Eventos
methods

3. En el **button** agrega un evento **click** que llame a un método **agregarTarea**

```
<input v-model="nuevaTarea" />
<button @click="agregarTarea">agregar</button>
```

4. Agrega una lista desordenada cuyos elementos li se rendericen por la iteración de una variable en el estado llamada **tareas**

```
<ul>
    <li v-for="tarea in tareas">{{ tarea }}</li>
</ul>
```

Eventos
methods

5. Ahora crea 2 variables en el estado llamadas **nuevaTarea** y **tareas**

```
data() {
 return {
   nuevaTarea: "",
   tareas: [],
 };
},
```

Eventos
methods

6. Finalmente agrega el método **agregarTarea** que agregue la nueva tarea al arreglo

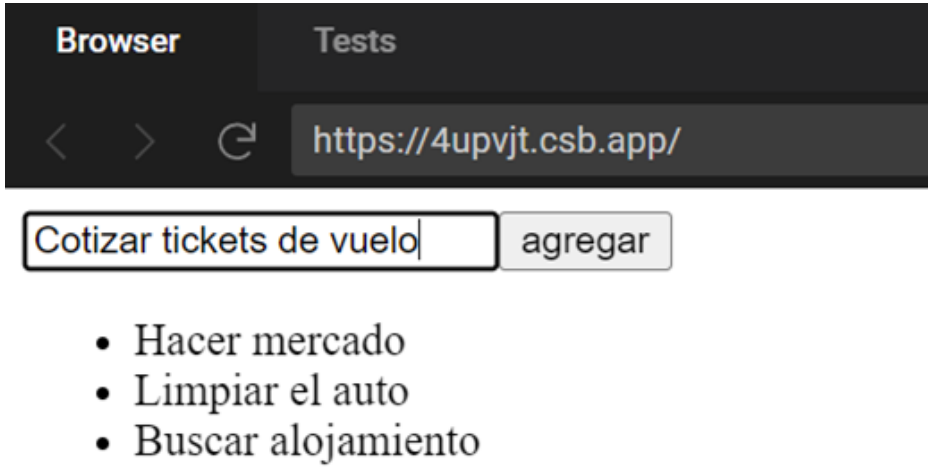
```
methods: {
 agregarTarea() {
   const nuevaTarea = this.nuevaTarea;

   this.tareas.push(nuevaTarea);

   this.nuevaTarea = "";
 },
},
```

Eventos
methods

Ahora prueba el TODO agregando algunas tareas



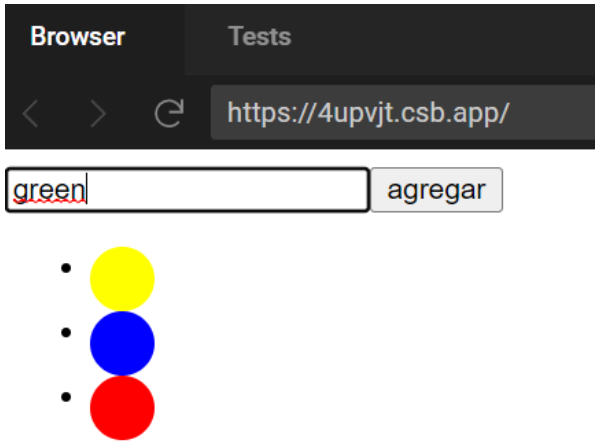
Ejercicio propuesto 2

Ejercicio propuesto

¡Sigamos practicando!

De acuerdo al ejercicio guiado 2:

- Crea una pequeña aplicación que agregue círculos de colores definidos en un input



{desafío} latam_

/* Modificadores */

Modificadores
de eventos

Los modificadores son extensiones que podemos agregar en los eventos para agregarles una configuración predeterminada.

Los diferentes modificadores que tenemos disponible son:

- .stop
- .prevent
- .capture
- .self
- .once
- .passive

Modificadores
de eventos

Por ejemplo, en ocasiones que necesitemos ocupar la etiqueta **form**, debemos recordar **neutralizar el evento submit** para evitar que recargue la página.

Para esto podemos ocupar el modificador .**prevent** en el evento **@submit** de un formulario.

{desafío} latam_

Ejercicio guiado 3

Modificadores
de eventos

1. Agrega un formulario en el template con un botón **enviar**

```
<form>
  <button>enviar</button>
</form>
```

1. Agregale el evento **@submit.prevent** al formulario para llamar un método **send**

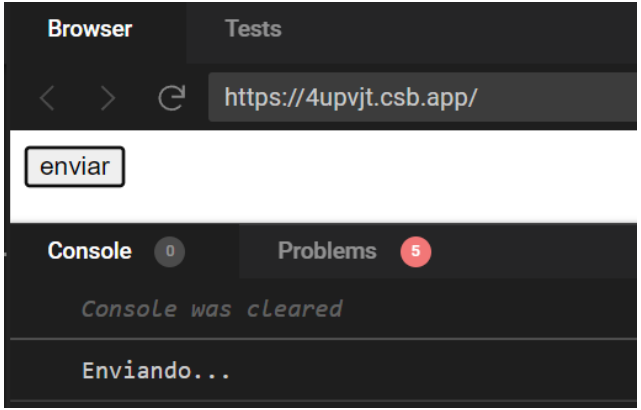
```
<form @submit.prevent="send">
 <button>enviar</button>
</form>
```

Modificadores

modificadores de eventos

3. Agrega el método **send** en el componente que muestre por consola un “Enviando…”

```
<script>
export default {
 methods: {
   send() {
     console.log("Enviando...");
   },
 },
};
</script>
```



Vue Js nos sigue ahorrando
muchísimo código y
tiempo de desarrollo

¿Qué te ha parecido hasta ahora?

Modificadores
de teclas

También tenemos los modificadores de teclas que acompañan el evento **@keyup** y nos permite realizar una acción detectando una tecla en específico

Este tipo de modificador los usaremos en los inputs y etiquetas de formulario.

Entre los principales modificadores tenemos:

- .enter
- .tab
- .delete
- .up
- .down
- .left
- .right
- .esc
- .space

Ejercicio guiado 4

{desafío}
latam_

Modificadores
de teclas

1. Agrega un **input** cuyo **fontSize** sea definido en el estado y utilice los modificadores **.up** y **.down** para incrementar o disminuir el tamaño de fuente

```
<input
 :style="{ fontSize: fontSize + 'px' }"
 @keyup.up="incrementar"
 @keyup.down="disminuir"
/>
```

Modificadores
de teclas

2. Agrega la variable **fontSize** en el estado

```
data() {
 return {
   fontSize: 16,
 };
},
```

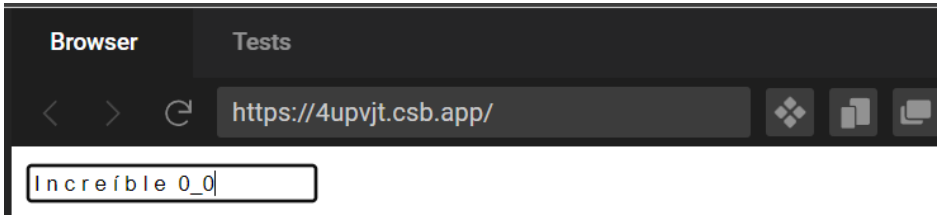
Modificadores
de teclas

3. Agrega los métodos **incrementar** y **disminuir**

```
methods: {
 incrementar() {
   this.fontSize++;
 },
 disminuir() {
   this.fontSize--;
 },
},
```

Modificadores
de teclas

Ahora prueba el ejercicio presionando las flechas arriba y abajo y viendo como incrementa y disminuye el tamaño de fuente dentro del input



{desafío} latam_

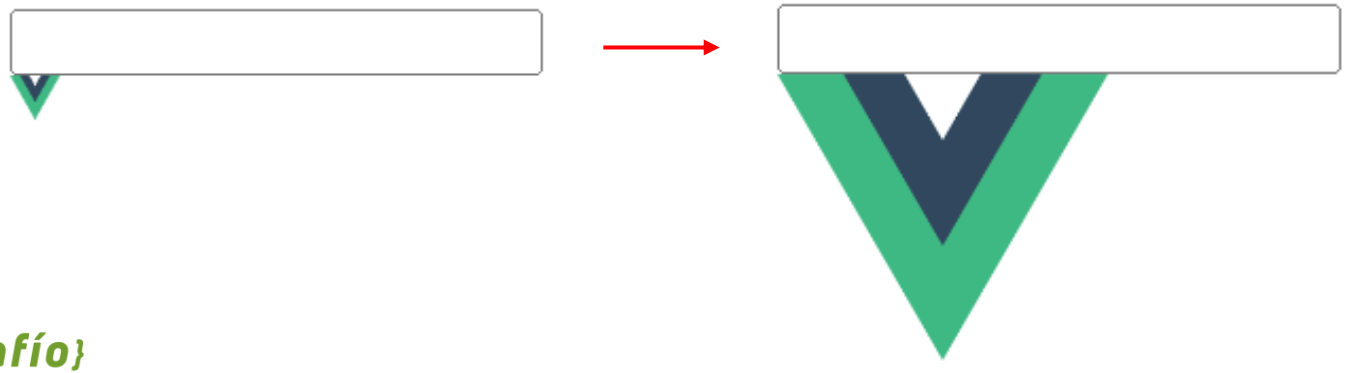
Ejercicio propuesto 3

Ejercicio propuesto

¡Sigamos practicando!

De acuerdo al ejercicio guiado 4:

- Incrementa o disminuye el tamaño de una imagen usando los modificadores desde un input



{desafío} latam_

Debate con tus compañeros

¿Cuál modificador te gustó más?
¿Por qué?

Próxima sesión...

- *Describe el concepto de reutilización de componentes distinguiendo sus beneficios para el desarrollo de una aplicación web mantenible.*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales