## TEXT CLASS REVIEW

## TEMAS A TRATAR EN EL CUE:

- Escuchando eventos.
- Métodos para controlar eventos.
- Modificadores de eventos.

### ESCUCHANDO EVENTOS:

Podemos usar la directiva v-on para escuchar eventos del DOM, y correr algún Método JavaScript cuando éste sea gatillado.

### PROPIEDAD V-ON:

```
<button v-on:click="counter +=1">Suma 1</button>
```

Con ésta podemos escuchar eventos generados por el usuario. En este caso, el evento “clic” ha sido ligado al dato “counter”, por lo que al requerir un solo dato, simplemente se ejecuta la acción, que sería sumar 1 a la variable counter.

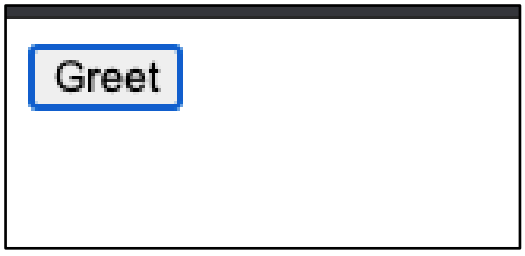
```
<template>
  <div>
    <button v-on:click="counter +=1">Suma 1</button>
    <p>El boton de arriba ha sido clickeado {{counter}}</p>
  </div>
</template>
<script>
export default {
  name:"ListenEvents",
  data: function(){
    return{
      counter: 0,
    }
  },
}
</script>
```

**MÉTODOS PARA CONTROLAR EVENTOS.**

Al generar un evento, algunas veces la lógica que necesitamos, puede ser más compleja que solo cambiar un dato. Para estos casos, la propiedad “v-on”, nos permite ligar un método al evento, dando la posibilidad de desarrollar la lógica necesaria para responder a éste.

```
<template>
  <div>
  <!-- `greet` es el nombre del metodo definido -->
  <button v-on:click="greet">Greet</button>
</div>
</template>
<script>
export default {
  data: function(){
    return {
      name:"Vue.js"
    }
  },
  methods:{
    greet: function(event){
      alert("Hola"+ this.name + "!")
      'event es el evento Nativo del DOM'
      if(event){
        alert(event.target.tagName);
      }
    }
  }
}
</script>
```

Si observamos el método “greet”, está asociado al evento v-on. Al ser de esta manera, podemos recibirlo como argumento. En este caso, “event” es el evento nativo de JavaScript, y el botón asociado al método “greet” abrirá 2 alertas, una con un saludo, y otra con el nombre del tag.




**MODIFICADORES DE EVENTOS:**

Éstos nos servirán para poder cambiar cierto comportamiento por default de algún elemento Html, por ejemplo:

- Cambiar comportamiento de un <a> link, cuando es clickeado.
- Frenar la actualización por defecto, luego de ejecutar un submit en un formulario.

Para aquellas necesidades podríamos utilizar event.preventDefault(), o event.stopPropagation(), pero Vue nos provee modificadores para los v-on.

**MODIFICADOR STOP:**

Se refiere a stopPropagation, lo que significa que al momento de gatillar un evento, éste no se propague a otro elemento.

**MODIFICADOR PREVENT:**

Viene del event.preventDefault(). Vue nos proporciona un atajo para poder usarla, este prevent, que previene el comportamiento por default. Por ejemplo:

- Un formulario, recarga la página al ser submit.
- Un link <a> no redirigirá al href cuando se haga clic sobre él.

**MODIFICAR CAPTURE:**

Para poder entender qué hace **capture,** primero debemos asociar que al hacer clic se genera un evento, y esto ocurre en dos fases:

1. Capturing.
2. Bubbling.

Cuando gatillamos un evento, en la primera fase llamada “capturing”, se ejecuta desde el div más lejano, hacia el más cercano que lo gatilla; luego pasamos al estado bubling, que se propaga desde el elemento que gatilló el evento, hacia el más lejano.

**MODIFICADOR SELF:**

Self maneja solo eventos propios, no de componentes hijos.