v-bind es lo mismo que usar solo los dos puntos

v-html es cuando tenes en data() un string con html por ej frase=’<h1>Hola wach<h1>’ si después a esto lo queres poner en el html te lo va a imprimir todo, entonces si hace en EL HTML:

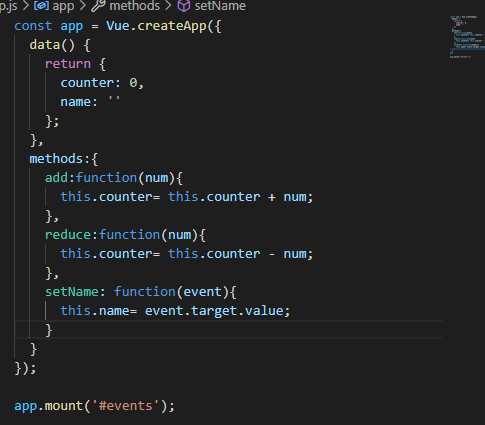
<p v-html="outputGoal()"></p> te lo va a imprimir sin las etiquetas

Para tomar el contenido de un input y ponerlo en una variable:

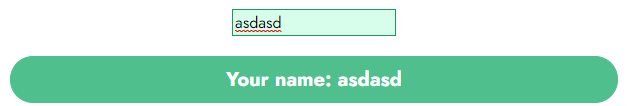
      <input type="text" @input="setName">

      <p>Your name: {{name}}</p>

Llamamos a la function desde el evento input



Creamos la variable name, y en la función el event es del browser y va guardando los cambios que se hacen en el input. Entonces ahí le asignamos a name lo que se va anotando en el input



Si tenes que pasar otro argumento tenes que hacer así:

      <input type="text" @input="setName($event, 'Lopez')">

      <p>Your name: {{name}}</p>

-----------------------------------------------------

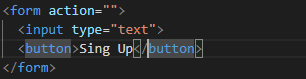
    setName: function(event, lastName){

      this.name= event.target.value + ' ' + lastName;

    }

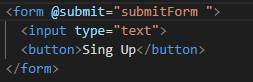
Events modifiers:

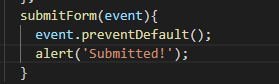
Cuando hacemos un formulario así y apretamos el botón la pagina se vuelve a recargar, xq esta así predefinido:



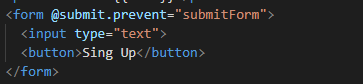
Y cuando la pagina se recarga se pierden todos los otros datos

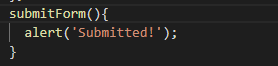
Podemos arreglarlo así:





Donde decimos que no se ejecute lo que es por default. Sino mas fácil y cumple la misma función:



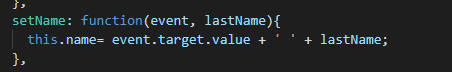


También podemos modificar el click, podemos poner que se active con el click derecho o del medio

Haciendo @click.right o @click.middle

Para hacer que se tome el valor de un input y se muestre con el enter hacemos además del @input un @keyup con otra función:



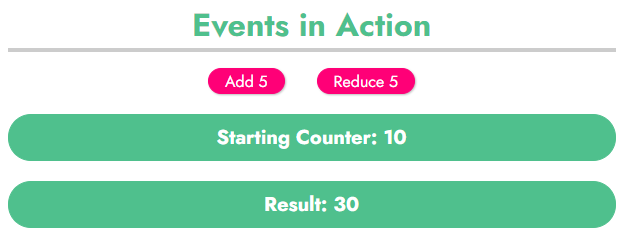




Entonces con el input le vamos cambiando el valor y con el el keyup lo pintamos abajo.

v-once sirve para solo evaluar la interpolación una vez, entonces después si se cambia el valor de la variable no va a cambiar en pantalla:

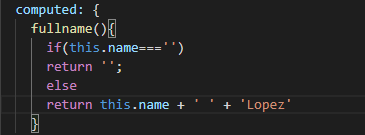




v-model es un shortcut para v-bind:value y v-bind:input envia los datos del parámetro y los devuelve va en las dos direcciones si no me equivoco. Es un enlace bidireccional

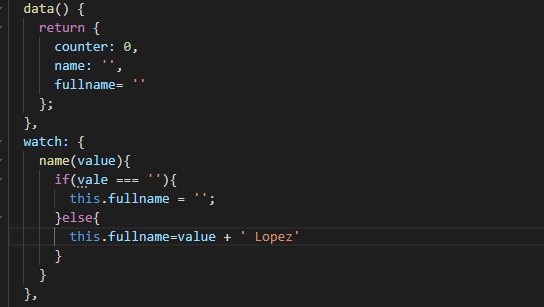
En las computadas nosotros hicimos un ejemplo que modifica una de las variables entonces retorna algo, y cuando lo llamamos lo hacemos sin () . **Las computed no se usan como funciones sino como variables**. Cuando la llamas se fija si hay una variable que se llame fullname y si no hay se fija en computed. Solo se ejecuta si una de las dependencias cambia en este caso name





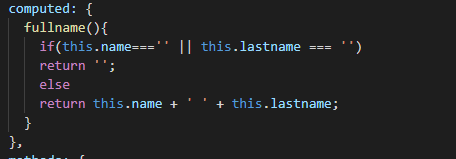
Watchers: Son funciones que les podes decir a Vue que ejecute, **cuando una de sus dependencias cambia**. Se puede usar Watchers en vez de computed properties.

Este ejemplo funciona igual que el anterior el método se tiene que llamar igual a la propiedad y el value le da el valor actual de esa propiedad. Se lo usa dentro de watch:



El tema con esto es que si tenes que usar por ej, dos propiedades y modificarlas se hace muy largo el código, porque lo tenes que hacer con cada una de las propiedades

Entonces es mucho mas fácil en computed porque podes llamar a propiedades directamente



Para cambiar clases de estilos con Vue:

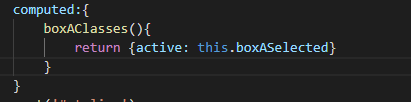
Acá con el click boxASeected pasa a ser true, entonces en la clase lo bindeamos y le ponemos un objeto con las clases css que van a recibir true o false, en este caso en el css tenemos demo y active, demo siempre se va a aplicar y active solo si boxASelected esta en true, ósea si se clickea ese div



Como demo no se modifica se puede poner aparte:



También se puede hacer retornando el objeto desde computed



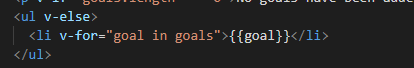


Para hacer como si fuera un else if se usa v-else-if

v-show=”goals.length > 0” con esta directiva si se cumple la condición se muestra esto en vez de modificar el DOM directamente como el v-if le pone un display none al contenedor

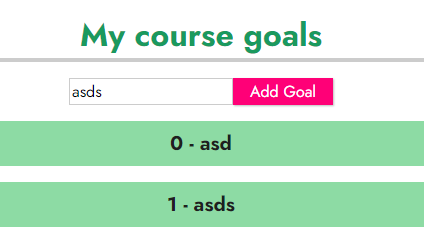
Cuando hacer una lista de agregar tareas x ej. Se recomienda usar en el for un in x ej: goal in goals

Para imrimir una lista

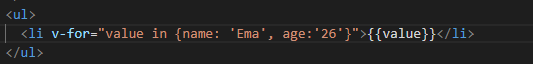


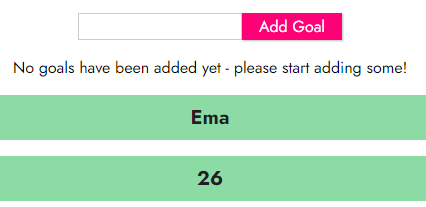
Si le queres agregar un índice:





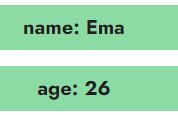
También podes recorrer un objeto con v-for y que te muestre en un li distinto cada propiedad del objeto:



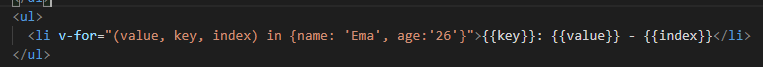


También podes mostrar el nombre de cada propiedad con key:



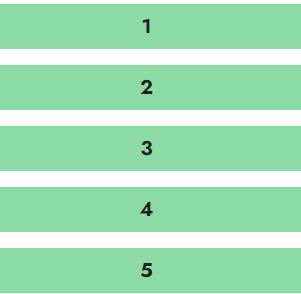


También a este le podes agregar el index si queres:



También podes recorrer números:

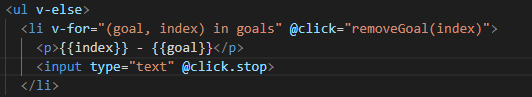




El index y goal se puede usar en todo el elemento, entonces podemos pasarlo como parámetro en la función del click



Si dentro del li tenemos un p y un input removiéndolo si se apreta el input también se elimina. Entonces para evitar eso le ponemos el @click.stop al input, no hace falta ponerle nada mas entre comillas:



En v-for y con Vue podes usar un atributo que es Key es recomendable usarlo cuando se usa v-for. Lo usamos para poder identificar a cada uno de los elementos de la lista,en este caso usamos goal, porque todos van a ser diferentes

Numero aleatorio entre 5 y 12



**Acceder a componente hijo o padre:**

Por ejemplo me paso con el input de search, quería ejecutar un método dentro de feedback pero llamarlo desde el search, tuve que hacer así:

En el search en la función donde se ejecuta el click para buscar resultados:

this.$eventBus.$emit('resetFeedback');

Entre paréntesis van a ir el nombre de la función que se va a llamar

Y en el feedback se hizo esto, dentro del created se puso esta función y dentro de el eventBus las funciones que queres que se ejecuten

  created: function(){

    this.$eventBus.$on('resetFeedback', () => {

        this.showFeedback();

        this.resetFeedback();

    });

Obtener un elemento del padre:

this.$parent.searchText

**Acceder a element que todavía no se muestra:**

          nextTick(()=>{

            this.$refs.firstElement.focus();

          })