Introducción al desarrollo con Vue Js

Antes de iniciar de lleno a trabajar con Vue debemos de conocer un poco de su funcionamiento, directrices, tener clara la instancia de Vue, sus propiedades, los componentes y el mismo ecosistema, para esto se considera que es bueno tener clara la idea del ciclo de vida de una instancia o componente en Vue.js y nos ayudará a entender cuáles el comportamiento del framework.

Conceptos básicos para trabajar con vue:

Instancia de vue: Es la raíz, y se hace a través de una función constructora, generalmente esta función se instancia a partir de una clase Vue y se pasan objetos como parámetros, y estos pueden contener diferentes opciones para datos, plantillas, el elemento donde montar dichas plantillas, métodos y callbacks, por convención esta instancia lleva por nombre “vm” o “app”.

```
1   const app = new Vue();
```

Dentro de esta instancia creamos un objeto.

```
1   const app = new Vue({
2
3   });
```

Estructuras de control

Son simplemente atributos que dirigen y modifican el comportamiento de la aplicación dentro del html, algunas de las más comunes son:

v-bind: Generalmente se usa para bindear elementos en una sola dirección, generalmente se usa para enlazar atributos html.

v-model: Es una directiva que permite bindear de forma bidireccional, si el valor de entrada cambia los datos enlazados a este cambiarán en todos los elementos presentes.

v-on: Es una directiva usada para escuchar eventos de DOM, y ejecutar código JS cuando se activa.

v-for: Muchas veces usada para mapear listas de objetos y elementos, su sintaxis es especial y de forma “item in items” donde ítems es el origen de los datos.

v-if, v-else-if, v-else: Son similares y su función es agregar un elemento al DOM en función del valor de un campo.

Propiedades de una Instancia en Vue

- **el:** Esta propiedad define el tag html donde se ejecutará la instancia vue, es un selector css, generalmente este indica el principal de la aplicación.

- **data:** Es un objeto de propiedades que podemos usar en nuestra instancia, al momento de crear la instancia vue se encarga de leer todas estas propiedades y bajo su funcionalidad agregarles getters y setters para hacerlos accesibles.

- **template:** Esta propiedad indica con una cadena html generalmente escrita entre comillas inversas <` texto `> y reemplaza el contenido html que se ha fijado dentro de la propiedad “el”.

- **methods:** Es un objeto con funciones que pueden ser invocadas y están enlazadas con el objeto propio, estos pueden modificar las propiedades de nuestro objeto, o enviar eventos a componentes o instancias.

- **compute:** Es un objeto que contiene una serie de métodos que devuelven valores calculados a partir de las propiedades del objeto en cuestión.

- **hooks:** Es un objeto de funciones que se ejecutan en determinados momentos del ciclo de vida de la instancia, estos tienen un momento de ejecución específico, ejemplos de estos momentos son.
    - created, mounted, created, updated.

Hagamos nuestro primer componente en Vue

Vamos a crear un documento html y vamos a ir a vuejs.org y vamos a copiar los cdn de desarrollo y los vamos a colocar al final de nuestro documento html.

```
<!-- development version, includes helpful console warnings -->
<script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
```

Y en nuestra carpeta de trabajo creamos un documento JavaScript que estaremos enlazando justo debajo del cdn de Vue. Esto es debido a que los recursos que estaremos cargando en nuestro js necesitan de los métodos de Vue.

```
18  </html>
19  <script src="https://cdn.jsdelivr.net/npm/vue/dist/vue.js"></script>
20  <script src="01.js"></script>
21
```

Vamos a nuestro documento js y creamos una nueva instancia de Vue.

Y dentro del objeto de la instancia creamos la propiedad **el.** Y dentro de él vamos a capturar el div donde queremos pintar todos nuestros datos.

```
1  const app = new Vue({
2      el: '#app'
3  });
4
```

Dentro de nuestro html debemos crear este div.

```
15    <body>
16      <div id="app">
17
18      </div>
19    </body>
```

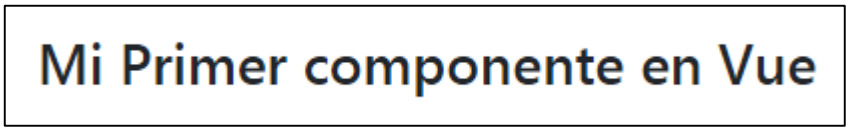
Ahora dentro de nuestra instancia de Vue vamos a inicializar el objeto **data** donde iremos almacenando todos los valores que queremos guardar en memoria. Y crearemos una propiedad llamada título con un valor, por ejemplo “mi primer componente en Vue”.

```
1  const app = new Vue({
2      el: '#app',
3      data: {
4          titulo: "Mi Primer componente en Vue"
5      }
6  });
7
```

Si ahora vamos a nuestro html y dentro de una tag abrimos dobles llaves podremos interpolar las propiedades que tengamos en **data**.

```html
<div id="app">
    <div class="container">
        <div class="row mt-3">
            <div class="col">
                <h3>{{titulo}}</h3>
            </div>
        </div>
    </div>
</div>
```

Nuestro documento se verá así.



Ahora vamos a crear una propiedad donde guardaremos un array. Para ellos dentro de data creamos una propiedad y a la hora de definir su valor abrimos corchetes. Además creamos otra propiedad más que le daremos el valor de un string vacío.

```js
const app = new Vue({
    el: '#app',
    data: {
        titulo: "Mi Primer componente en Vue",
        actividades: [],
        nuevaActividad: ''
    }
});
```

Ahora en nuestro html vamos a crear un input y un botón.

```html
<h4>Agregar Actividad</h4>
<input type="text" class="form-control"><br>
<button class="btn btn-block btn-primary">Agregar</button>
```

Dentro de nuestro input colocaremos una de las estructuras de control de Vue que será v-model y dentro de esta mandaremos a llamar a nuestra propiedad nuevaActuvidad, con esto le indicaremos a Vue que todo lo que escribamos en el input queremos que se relacione con dicha propiedad.

```html
<h4>Agregar Actividad</h4>
<input type="text" class="form-control" v-model="nuevaActividad"><br>
<button class="btn btn-block btn-primary">Agregar</button>
```

Luego en el objeto principal de nuestra instancia de Vue vamos a mandar a llamar a la propiedad **methods** que es la que permite definir las acciones de nuestro componente en su interior.

```
const app = new Vue({
    el: '#app',
    data: {
        titulo: "Mi Primer componente en Vue",
        actividades: [],
        nuevaActividad: ''
    },
    methods: {

    }
});
```

Y dentro de este crearemos la función agregar Actividad.

```
    methods: {
        agregarActividad: function(){

        }
    }
```

Ahora vamos en nuestro botón del html agregar otra estructura de control de Vue que detecte al presionar ejecute la función agregarActividad.

```
<button class="btn btn-block btn-primary" @click="agregarActividad">Agregar</button>
```

Ahora indicamos que la nuevaActvidad se introduzca dentro del array actividades y luego que limpie la propiedad nuevaActividad asignándole un string vacío como al principio.

```
agregarActividad: function(){
this.actividades.push(this.nuevaActividad);
   this.nuevaActividad='';
}
```

Utilizamos el operador this para referirnos a un elemento de un nivel diferente, en este caso al referirnos al nivel de data.

Vamos a ver si nuestra función está funcionando para ello vamos a en una tabla a mapear el array actividades.

Creamos la tabla y las tag que vamos a estar repitiendo en cada elemento serán las filas de la tabla (los tr y dentro de estos los td) entonces mandamos a llamar a la estructura **v-for** y mapeamos nuestra actividades como en un bucle foreach de cualquier lenguaje de programación. Con la forma actividad in actividades y en la celda interpolamos nuestra actividad mapeada.

```
<table class="table">
    <thead>
        <tr>
            <th>Actividades</th>
        </tr>
    </thead>
    <tbody>
        <tr v-for="actividad in actividades">
            <td>{{actividad}}</td>
        </tr>
    </tbody>
</table>
```

Ya falta poco para terminar nuestro primer componente en Vue, solamente faltaría crear un botón para eliminar las actividades que queramos de la lista.

```
<tr v-for="actividad in actividades">
    <td>{{actividad}}</td>
    <td><button class="btn btn-danger">Borrar</button></td>
</tr>
```

Actividades	Borrar?
algo	Borrar

Para saber la posición de nuestro array en la lista de actividades vamos a modificar nuestro **v-for** agregándole un parámetro adicional que será un index.

```
<tr v-for="(actividad, index) in actividades">
```

Ahora podremos manejar la posición de nuestro array en nuestro botón. En este agregaremos un @click que ejecute la acción borrar actividad y le pasamos por parámetro el index de nuestro array.

```
<tr v-for="(actividad, index) in actividades">
    <td>{{actividad}}</td>
    <td><button class="btn btn-danger" @click="eliminarActividad(index)">Borrar</button></td>
</tr>
```

Y ahora en nuestra propiedad **methods** de nuestra instancia de Vue vamos a crear nuestra función eliminarActividad.

```
eliminarActividad: function(index){

}
```

Y ahora solo llamaremos a un método de JavaScript que permite remover un elemento de un array pasando como parámetro la posición.

```
eliminarActividad: function(index){
    this.actividades.splice(index, 1);
}
```