Los componentes son bloques de código JavaScript que nos permite interpolar bloques de código html desde un archivo js, a la hora de crear webs con Vue lo que se acostumbra a hacer es dividir las partes de nuestra web en componentes individuales (también llamados módulos) y colocarlos uno a uno dentro de un componente mayor (componente padre) y la comunicación de estos componentes se da a través de las propiedades (Props) que permite mandar datos de un componente a otro (con ciertas restricciones)

1. **Creando nuestro html**

Para crear componentes con Vue Js desde el lado de nuestro html vamos a crear una estructura básica y a enlazar con los CDN´s de desarrollo de Vue y de Bootstrap.

Enlazamos con nuestro archivo JavaScript desde el cual estaremos creando nuestro primer componente. Y dentro del contenedor app (que es donde generamos nuestra instancia de Vue, procedemos a crear una etiqueta html con un nombre que queramos.

```
18          <div class="row">
19            <div class="col-6">
20              <componente-padre>
21
22              </componente-padre>
23            </div>
24          </div>
25        </div>
```

2. **Creando el componente.**

Para ello antes de crear nuestra instancia de Vue vamos a crear un componente de Vue dentro de nuestro archivo JavaScript. Que como parámetro principal recibirá el nombre de la etiqueta que creamos en nuestro html.

```
1  Vue.component('componente-padre', {
2
3  });
4
```

Luego abajo del componente creamos la instancia de Vue.

```
1  Vue.component('componente-padre', {
2
3  });
4
5  const app = new Vue({
6      el: 'app'
7  })
8
```

3. **Partes principales de un componente**

El template: es la primera propiedad de los componentes y su función es renderizar un bloque de código html. Para poder hacer esto todo aquello que vamos a renderizar debe de estar dentro de un contenedor y dentro de los template iteradores (tildes invertidas).

```
Vue.component('componente-padre', {
    template:
    `
    <div>
        <h5 class="text-primary">Ejemplo</h5>
    </div>
    `
});

```

La data: los datos de nuestra instancia de Vue ya no los manejaremos desde app sino que los manejaremos desde nuestro componente de Vue. Solo que esta data será una función que debe de retornar un objeto.

```
Vue.component('componente-padre', {
    template:
    `
    <div>
        <h5 class="text-primary">{{titulo}}</h5>
    </div>
    `,
    data(){
        return {
            titulo: 'Ejemplo'
        }
    }
});

```

Ahora ya podemos manipular datos dentro de nuestro componente de Vue.

Methods: esta propiedad también la manipularemos dentro de nuestro componente de Vue. Colocándolo justo debajo de la data.

```
    },
    methos: {

    }
});

```

EJEMPLO:

Crea un documento html y manda a llamar un componente crea ese componente y renderiza este template:

```
Vue.component('componente-padre', {
    template:
    `
    <div>
        <h5 class="text-primary">{{titulo}}</h5>
        <input type="text" v-model="fruta.nombre"
        class="form-control" v-on:keyup.enter="agregarFruta">
        <button class="btn btn-block btn-success"
        @click="agregarFruta">Agregar Fruta</button><br>
        <div class="aler alert-success" v-for="frut in frutas">
            {{ frut.nombre }}
        </div>
    </div>
    `,
```

Ahora vamos a crear nuestra data donde manejaremos nada más dos datos por el momento: nuestro título, fruta con la diferencia que fruta será un objeto que contendrá dos atributos el nombre y la cantidad. Y un array frutas.

```
    data(){
        return {
            titulo: 'Ejemplo de componentes',
            frutas: [],
            fruta: {
                nombre: '',
                cantidad: 0
            }
        }
```

Y por último nuestros métodos, donde por el momento solo vamos a tener a la función agregarFruta.

```
    methods: {
        agregarFruta: function() {
            this.frutas.push({
                nombre: this.fruta.nombre,
                cantidad: this.fruta.cantidad
            });
            this.fruta.nombre='';
        }
    }
```

Con este sistema bastante corto ya tenemos como agregar frutas de manera dinámica, ahora solo falta que podamos modificar la cantidad de cada fruta de una a una.

Para ello vamos a hacer unos cambios dentro de nuestro template.

```
10          <div class="aler alert-success" v-for="(frut, index) in frutas">
11              {{ frut.nombre }} - {{ frut.cantidad }}
12              <button class="btn btn-info" @click="frut.cantidad++">
13                  +
14              </button>
15          </div>
```

EJERCICO: ahora agrega las funcionalidades restantes al componente (eliminar y actualizar) y configúralo para guardar estos datos en el LocalStorage del navegador.