HINTS

CONSEJOS CONCEPTUALES

- **BeforeCreate:** es la única función, de todos los lifeCycles, que no puede enlazarse a algún methods, para encapsular lógica, ya que en ese estado de vida del componente no podemos acceder a data, evento o métodos.
- **Instalar axios:**

  `npm install axios`
- **Instalar chartist:**

  `npm install --save chartist`

RECOMENDACIONES

Encapsular lógica en métodos:

Cuando usamos los lifeCycle, a veces es necesario incluir diferentes tareas, por ejemplo: la función created(). Luego, para hacer una llamada al api, es recomendable encapsular esta lógica, ya que en ese mismo estado se pudiese generar otra lógica para alguna acción del componente, lo que haría que el código se haga difícil de entender y mantener.

```
created(){

    axios.get("https://rickandmortyapi.com/api/character")
      .then(resp=>{
        console.log(resp)
      })

      .catch(error=>{
        console.log(error)
      })
  }
```

Como recomendación, es preferible que la lógica sea encapsulada en métodos, logrando así poder llamar distintas funciones, que hacen diferentes lógicas. En el siguiente ejemplo, se ha creado el método fetchCharacters(), para almacenar la lógica que antes estaba en la función created().

```
methods:{
    fetchCharacters(){
      axios.get("https://rickandmortyapi.com/api/character")
      .then(resp=>{
        console.log(resp)
      })
      .catch(error=>{
        console.log(error)
      })
    }
  },
  //lifecycles
  created(){
    this.fetchCharacters();
    //puedo llamar a otro method
  }
```

CLASS BINDING

En Vue.js, existe una manera muy sencilla de agregar clases dinámicamente a un elemento HTML. La forma de hacerlo está ligada a cómo funciona la directiva v-bind en Vue.

Para este ejemplo, utilizaremos el siguiente código CSS.

```
.red {
    color: red;
}
.green {
    color: green;
}
.price {
    letter-spacing: 4px;
}
```

Y definiremos el elemento ventas, con los siguientes valores de facturación por día.

```
const app = new Vue({
    el: "#app",
    data: {
        ventas: {
            '2020-01-01': 100,
            '2020-01-02': 200,
            '2020-01-03': 300
        }
    },
})
```

Ahora, supongamos que queremos mostrar en rojo la facturación menor o igual a 100 USD, y en verde la facturación por encima de 100 USD. Para esto, podemos utilizar en shortcut, la directiva v-bind, para colocar un objeto en el atributo Array, y decidir si se aplica cada clase:

```
<div id="app">
    <ul id="app">
        <li class="price" :class="{ red: price <= 100, green: price > 100 }"
        v-for="(price, date) in ventas">
            {{ date }} ____ $ {{ price }}
        </li>
    </ul>
</div>
```

También se puede observar, que hemos colocado a propósito, la clase price en el elemento li. Con esto, queremos mostrar que la directiva v-bind, adjunta la clase seleccionada a las que ya existen. Veamos el ejemplo anterior, en funcionamiento en el navegador.

- 2020-01-01 ____ $ 100
- 2020-01-02 ____ $ 200
- 2020-01-03 ____ $ 300

Sintaxis de Array

Si bien es posible pasar un objeto en el atributo class, también se puede un Array de clases a agregar. A continuación, veamos un ejemplo en donde adjuntamos un par de clases a un botón.

```
<button :class="['btn', 'btn-success']">Submit</button>
```

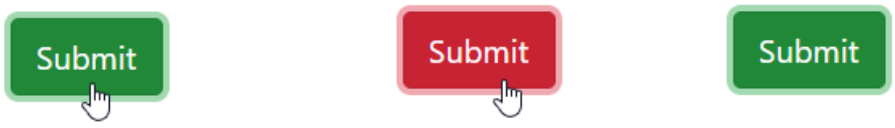
Esto es exactamente igual a hacer lo siguiente:

```
<button class="btn btn-primary">Submit</button>
```

A simple vista, pareciera no tener mucha utilidad, pero el verdadero poder de esta sintaxis, es que puedes combinarlo con la sintaxis de objeto. Entonces puede ser útil que revises si realmente necesitas la clase btn-success en un botón, o si necesitas btn-danger. Dicho esto, podríamos tener algo como lo siguiente:

```
<button :class="['btn', isError ? 'btn-danger' : 'btn-success']"
@click="isError = !isError">Submit</button>
```

Cada vez que se haga clic en el botón, cambiará de clase según el valor de isError (para darle estilo al botón, se utilizó el CDN de Bootstrap).



STYLE BINDING

Los estilos dinámicos sirven para vincular estilos en línea a elementos. El método es similar a :class

Ahora, en nuestro HTML tenemos lo siguiente:

```
<div id="app">

    <div class="container"
    :style = "{'color': color, 'fontSize': fontSize + 'px',  'padding': padding + 'px'}">

        Lorem, ipsum dolor sit amet consectetur adipisicing elit. Exercitationem laborum,
        doloribus eum provident similique natus atque praesentium temporibus tenetur quos,
        incidunt quibusdam ullam consectetur dolore labore at delectus facere quam. </div>

</div>
```

Y en nuestra data tendremos:

```
const app = new Vue({
    el: "#app",
    data: {
        color: 'red',
        fontSize: 20,
        padding: 40
    },
})
```

Como resultado, en nuestro navegador se verá algo similar:

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Exercitationem laborum, doloribus eum provident similique natus atque praesentium temporibus tenetur quos, incidunt quibusdam ullam consectetur dolore labore at delectus facere quam.

En la mayoría de los casos, escribir una lista larga de estilos directamente en la vista, no es fácil de leer y mantener, entonces podemos incluir nuestros estilos en un objeto de la siguiente manera:

```
const app = new Vue({
    el: "#app",
    data: {
        styleObject: {
        color: 'green',
        fontSize: 20 + 'px',
        padding: 40 +'px'
        }
    },
})
```

En nuestra vista usaremos :style = “styleObjetc” (que es el nombre del objeto definido en data), y cambiaremos el color de la letra a verde para notar la diferencia.

```
<div id="app">
    <div class="container"
    :style = "styleObject">

        Lorem, ipsum dolor sit amet consectetur adipisicing elit. Exercitationem laborum,
        doloribus eum provident similique natus atque praesentium temporibus tenetur quos,
        incidunt quibusdam ullam consectetur dolore labore at delectus facere quam.
    </div>
</div>
```

Finalmente, en nuestro navegador se verá así:

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Exercitationem laborum, doloribus eum provident similique natus atque praesentium temporibus tenetur quos, incidunt quibusdam ullam consectetur dolore labore at delectus facere quam.

Al usar varios objetos de estilos, se puede emplear la sintaxis de matriz (no se utiliza comúnmente).

```
<div :style = "[styleObject1, styleObject2]"> TEXTO </div>
```

Si deseas ampliar la información sobre este tema, puedes revisar en el siguiente enlace:

https://es.vuejs.org/v2/guide/class-and-style.html