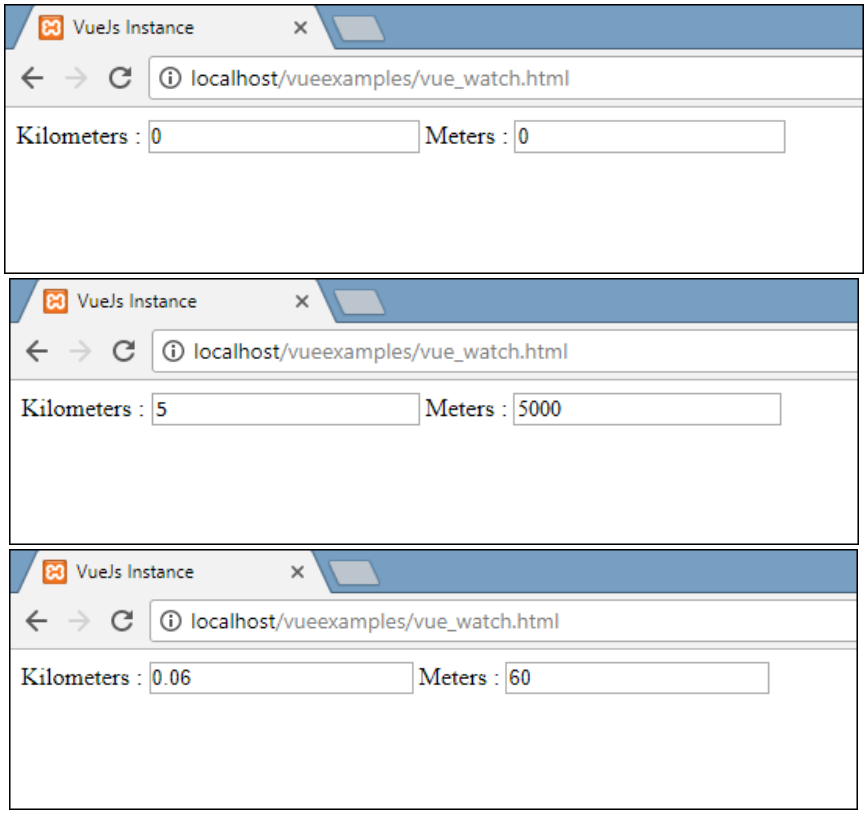
Fuente:tutorialspoint.com

Watchers

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "computed_props">
      Kilometers : <input type = "text" v-model = "kilometers">
      Meters : <input type = "text" v-model = "meters">
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#computed_props',
        data: {
          kilometers : 0,
          meters:0
        },
        methods: {
        },
        computed :{
        },
        watch : {
          kilometers:function(val) {
            this.kilometers = val;
            this.meters = val * 1000;
          },
          meters : function (val) {
            this.kilometers = val/ 1000;
            this.meters = val;
          }
        }
      });
    </script>
  </body>
</html>
```

Se crean dos textboxes, uno con kilómetros y otro con metros. En **data** ambas propiedades son inicializadas en cero. Existe dentro de la instancia Vue un objeto **watch** que se crea con dos funciones, cuyo objetivo es convertir de kilómetros a metros y viceversa.

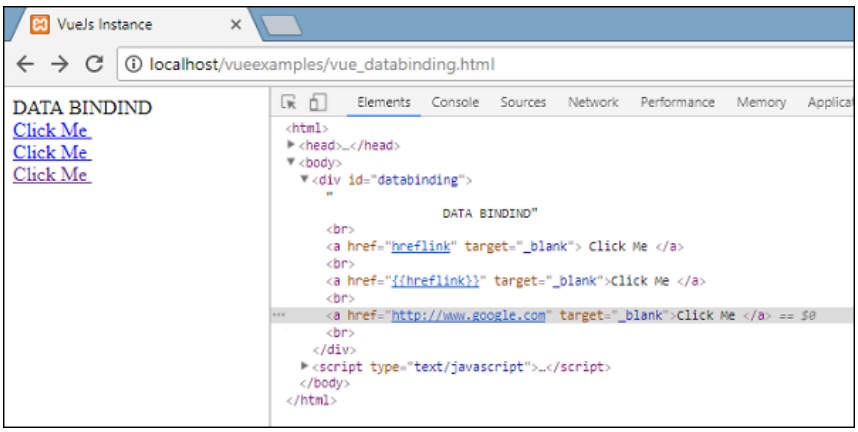
Cada vez que se ingrese algún valor en los textboxes, **watch** se encarga de actualizarlos calculando lo que está declarado en las funciones.



Más sobre manipulación de atributos

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      {{title}}<br/>
      <a href = "hreflink" target = "_blank"> Click Me </a> <br/>
      <a href = "{{hreflink}}" target = "_blank">Click Me </a>  <br/>
      <a v-bind:href = "hreflink" target = "_blank">Click Me </a>   <br/>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          title : "DATA BINDING",
          hreflink : "http://www.google.com"
        }
      });
    </script>
  </body>
</html>
```

En el ejemplo anterior se presentan tres supuestas formas de enlazar el atributo **href** a los enlaces. Pero sólo una es correcta al revisar el ejemplo en el navegador.



Como fue demostrado en anteriores ejemplos, para enlazar atributos desde una instancia Vue al DOM se utiliza la directiva **v-bind:atributo**. En este caso trabajaremos sobre **href.**

```
<a v-bind:href = "hreflink" target = "_blank">Click Me </a>
```

Vue.JS nos brinda un atajo a dicha directiva eliminando **v-bind.** Ninguna de las propiedades de Vue.JS será mostrada en el inspector del navegador.

```
<a :href = "hreflink" target = "_blank">Click Me </a>
```

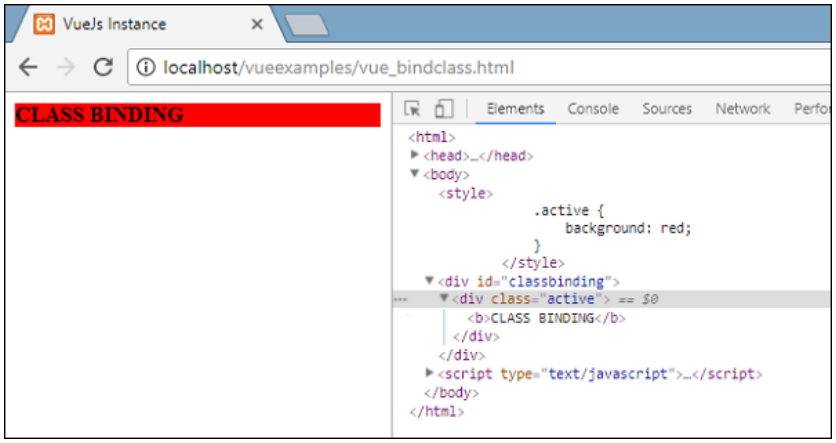
Enlazando clases HTML

Para enlazar clases se utiliza también la directiva **v-bind**, pero utilizando el atributo **class**.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <style>
      .active {
        background: red;
      }
    </style>
    <div id = "classbinding">
      <div v-bind:class = "{active:isactive}"><b>{{title}}</b></div>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#classbinding',
        data: {
          title : "CLASS BINDING",
          isactive : true
        }
      });
```

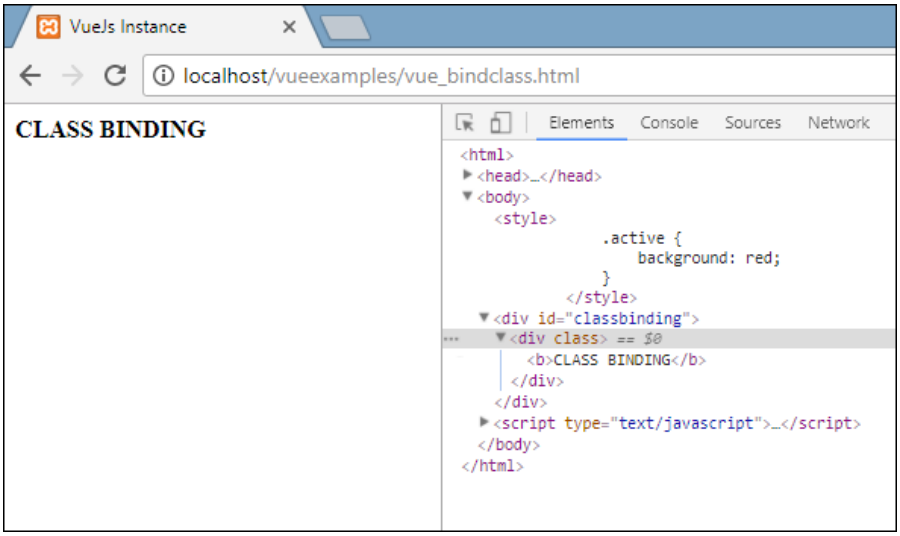
```
    </script>
  </body>
</html>
```

isactive es una variable booleana, la cual puede contener en su interior true o false lo cual hará que se aplique o no la clase **active** al div enlazado a la instancia Vue. La clase active cambia el fondo del elemento a un color rojo.



En el siguiente ejemplo a dicha variable se le asigna el valor false.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <style>
      .active {
        background: red;
      }
    </style>
    <div id = "classbinding">
      <div v-bind:class = "{active:isactive}"><b>{{title}}</b></div>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#classbinding',
        data: {
          title : "CLASS BINDING",
          isactive : false
        }
      });
    </script>
  </body>
</html>
```



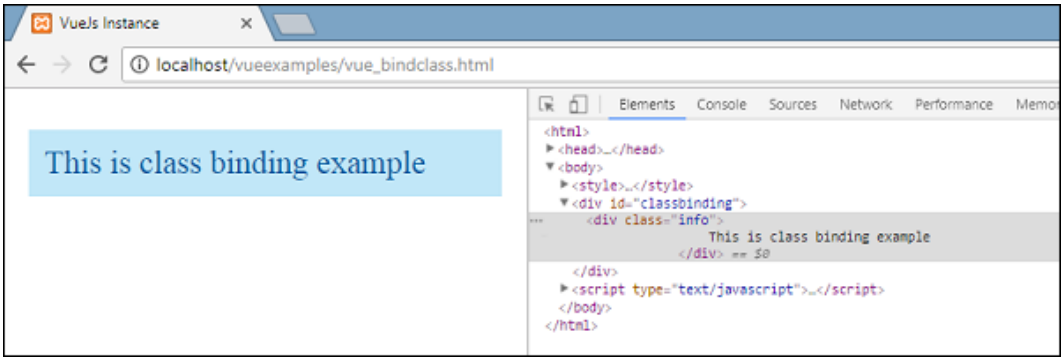
El siguiente ejemplo enlaza múltiples clases al elemento HTML.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <style>
      .info {
        color: #00529B;
        background-color: #BDE5F8;
      }
      div {
        margin: 10px 0;
        padding: 12px;
      }
      .active {
        color: #4F8A10;
        background-color: #DFF2BF;
      }
      .displayError{
        color: #D8000C;
        background-color: #FFBABA;
      }
    </style>
```

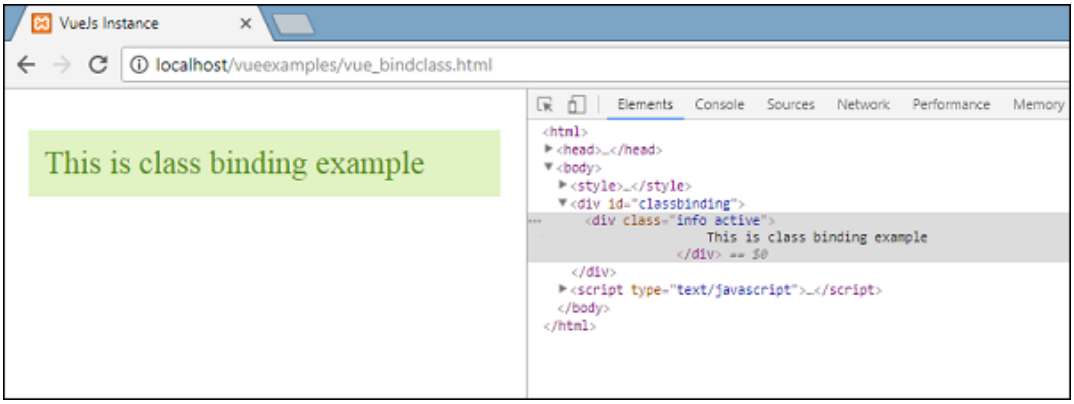
```
<div id = "classbinding">
```

```
        <div class = "info"  v-bind:class = "{ active: isActive, 'displayError': hasError }">
          {{title}}
        </div>
    </div>
    <script type = "text/javascript">
        const vm = new Vue({
          el: '#classbinding',
          data: {
            title : "This is class binding example",
            isActive : false,
            hasError : false
          }
        });
    </script>
  </body>
</html>
```

Con ambas variables con valores false el elemento HTML se ve de la siguiente manera:



Con la variable isactive en true:



Con ambas variables en true:

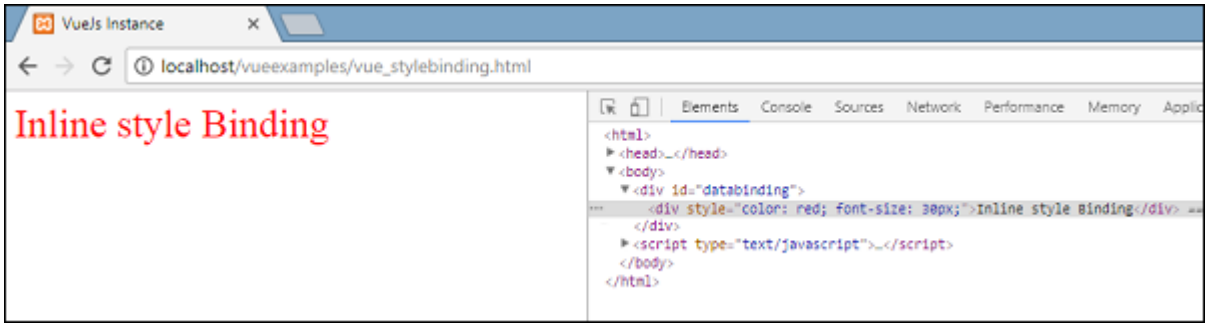
El siguiente ejemplo aplica **v-bind** para clases en un componente.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <style>
      .info {
        color: #00529B;
        background-color: #BDE5F8;
      }
      div {
        margin: 10px 0;
        padding: 12px;
        font-size : 25px;
      }
      .active {
        color: #4F8A10;
        background-color: #DFF2BF;
      }
      .displayError{
        color: #D8000C;
        background-color: #FFBABA;
      }
    </style>
    <div id = "classbinding">
      <new_component class = "active"></new_component>
    </div>
    <script type = "text/javascript">
      var vm = new Vue({
        el: '#classbinding',
        data: {
          title : "This is class binding example",
          infoclass : 'info',
          errorclass : 'displayError',
          isActive : false,
          haserror : true
```

```
        },
        components:{
          'new_component' : {
            template : '<div class = "info">Class Binding for component</div>'
          }
        }
      });
    </script>
  </body>
</html>
```

Enlazar estilos inline

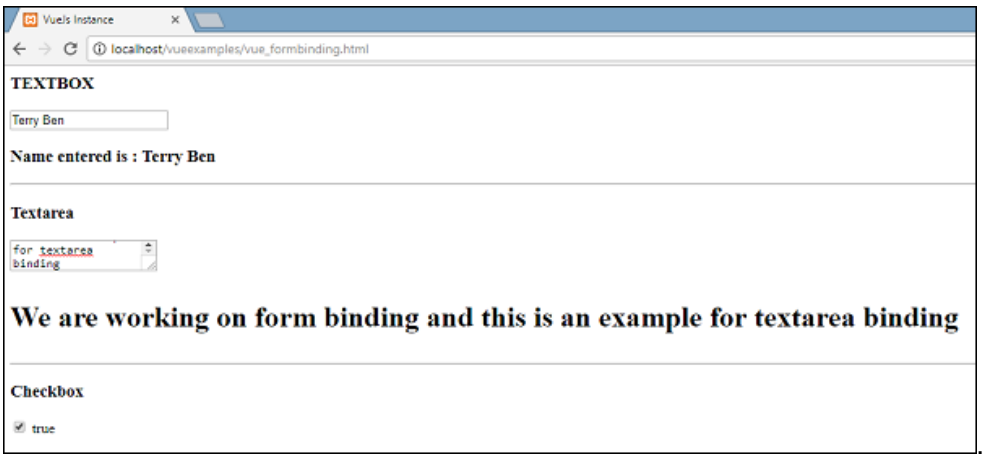
```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <div v-bind:style = "{ color: activeColor, fontSize: fontSize + 'px' }">{{title}}</div>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          title : "Inline style Binding",
          activeColor: 'red',
          fontSize :'30'
        }
      });
    </script>
  </body>
</html>
```



Enlazar elementos input de un formulario

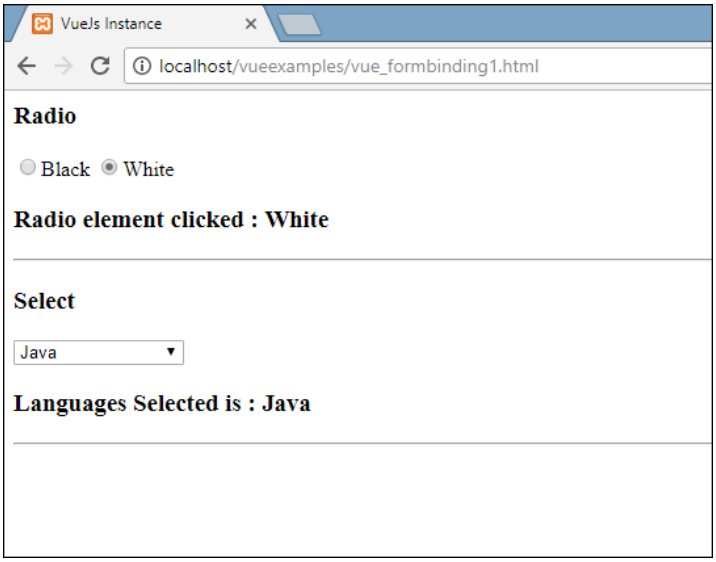
```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <h3>TEXTBOX</h3>
      <input  v-model = "name" placeholder = "Enter Name" />
      <h3>Name entered is : {{name}}</h3>
      <hr/>
      <h3>Textarea</h3>
      <textarea v-model = "textmessage" placeholder = "Add Details"></textarea>
      <h1><p>{{textmessage}}</p></h1>
      <hr/>
      <h3>Checkbox</h3>
      <input type = "checkbox" id = "checkbox" v-model = "checked"> {{checked}}
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          name:'',
          textmessage:'',
          checked : false
        }
      });
    </script>
  </body>
</html>
```

Lo que escribamos en el textbox se mostrará debajo. **v-model** tiene asignado el valor de la propiedad **name** y dicho nombre es mostrado en {{name}}, que muestra lo que tenga escrito en su interior el textbox, lo mismo sucede con el checkbox y el textarea.



Radio y Select

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <h3>Radio</h3>
      <input type = "radio" id = "black" value = "Black" v-model = "picked">Black
      <input type = "radio" id = "white" value = "White" v-model = "picked">White
      <h3>Radio element clicked : {{picked}} </h3>
      <hr/>
      <h3>Select</h3>
      <select v-model = "languages">
        <option disabled value = "">Please select one</option>
        <option>Java</option>
        <option>Javascript</option>
        <option>Php</option>
        <option>C</option>
        <option>C++</option>
      </select>
      <h3>Languages Selected is : {{ languages }}</h3>
      <hr/>
    </div>
    <script type = "text/javascript">
      var vm = new Vue({
        el: '#databinding',
        data: {
          picked : 'White',
          languages : "Java"
        }
      });
    </script>
  </body>
</html>
```



Modificadores

Veremos tres tipos: **trim**, **number**, y **lazy**.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <span style = "font-size:25px;">Enter Age:</span> <input v-model.number = "age"
type = "number">
      <br/>
      <span style = "font-size:25px;">Enter Message:</span> <input v-model.lazy = "msg">
      <h3>Display Message : {{msg}}</h3>
      <br/>
      <span style = "font-size:25px;">Enter Message : </span><input v-model.trim =
"message">
      <h3>Display Message : {{message}}</h3>
    </div>
    <script type = "text/javascript">
      var vm = new Vue({
        el: '#databinding',
        data: {
          age : 0,
          msg: '',
          message : ''
        }
      });
    </script>
  </body>
</html>
```

El modificador **number** sólo permite el ingreso de números, no tomará otra entrada salvo que sean números.

```
<span style = "font-size:25px;">Enter Age:</span> <input v-model.number = "age" type = "number">
```

El modificador **lazy** mostrará el contenido presente en el textbox cuando el usuario abandone el mismo.

```
<span style = "font-size:25px;">Enter Message:</span> <input v-model.lazy = "msg">
```

El modificador **trim** eliminará los espacios al principio y al final de lo ingresado en el textbox.

```
<span style = "font-size:25px;">Enter Message : </span><input v-model.trim = "message">
```

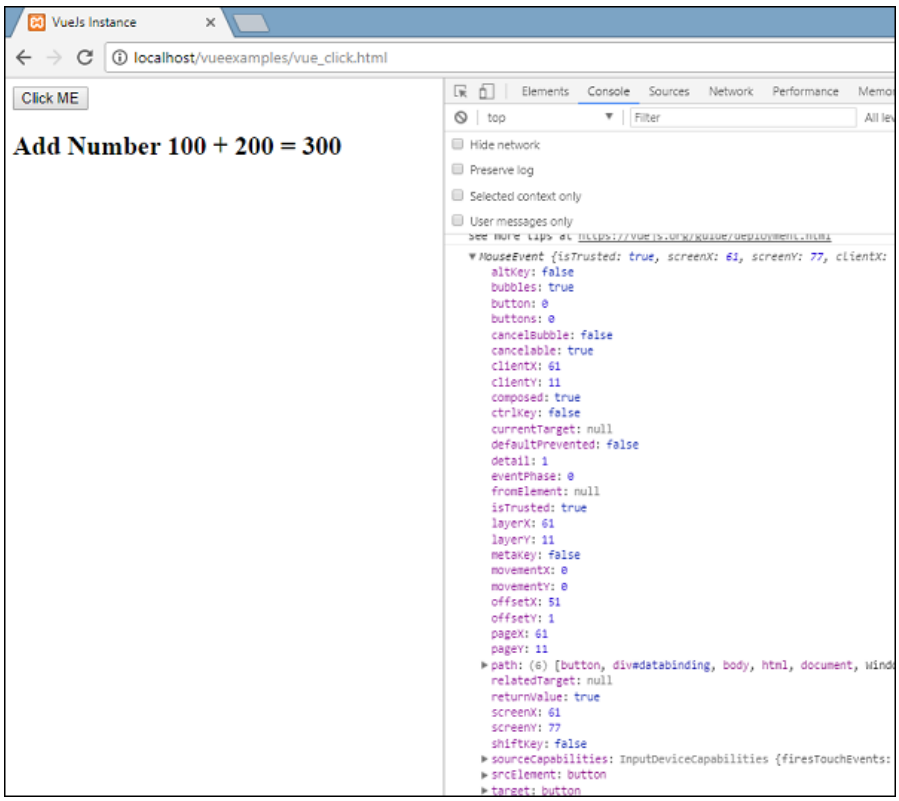
Eventos

Utilizaremos la directiva **v-on** en los elementos HTML del DOM para escuchar a los eventos.

Evento de Click

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <button v-on:click = "displaynumbers">Click ME</button>
      <h2> Add Number 100 + 200 = {{total}}</h2>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
         el: '#databinding',
         data: {
```

```
            num1: 100,
            num2 : 200,
            total : ''
        },
        methods : {
            displaynumbers : function(event) {
              console.log(event);
              return this.total =  this.num1+ this.num2;
            }
        },
    });
  </script>
 </body>
</html>
```



El siguiente código asignará un evento de click al elemento del DOM.

```
<button v-on:click = "displaynumbers">Click ME</button>
```

Vue.JS nos brinda una propiedad atajo para los eventos:

```
<button @click = "displaynumbers">Click ME</button>
```

Cuando se cliquee el botón, se llamará al metodo displaynumbers.

Eventos mouseover, mouseout

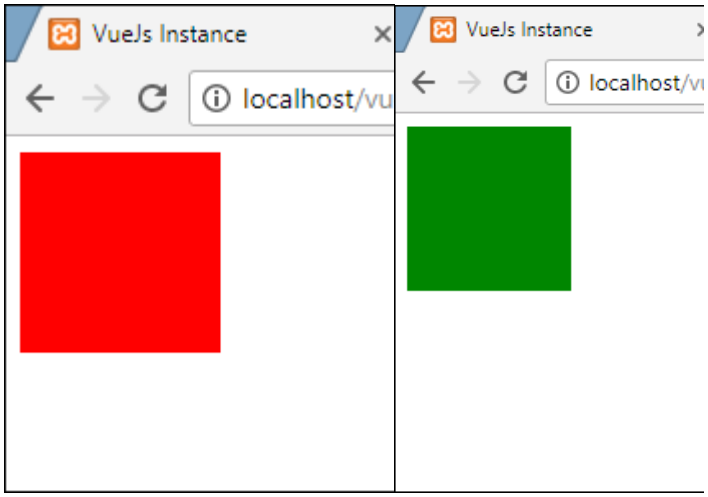
```
<html>
```

```
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <div v-bind:style = "styleobj" v-on:mouseover = "changebgcolor" v-on:mouseout =
"originalcolor"></div>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          styleobj : {
            width:"100px",
            height:"100px",
            backgroundColor:"red"
          }
        },
        methods : {
          changebgcolor : function() {
            this.styleobj.backgroundColor = "green";
          },
          originalcolor : function() {
            this.styleobj.backgroundColor = "red";
          }
        },
      });
    </script>
  </body>
</html>
```

Se crea un div con ancho y alto de 100 px. Se le asigna un background de color rojo. Cuando el mouse se posiciona sobre él se cambiará dicho fondo al color verde y cuando el mouse salga del elemento volverá al color rojo mediante los métodos changebgcolor y originalcolor.

```
<div v-bind:style = "styleobj" v-on:mouseover = "changebgcolor" v-on:mouseout =
"originalcolor"></div>

changebgcolor : function() {
  this.styleobj.backgroundColor = "green";
}
originalcolor : function() {
  this.styleobj.backgroundColor = "red";
}
```



Modificadores de eventos

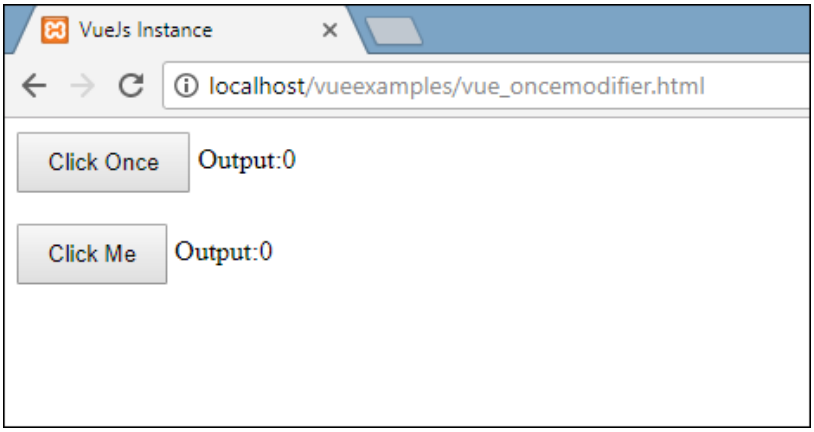
Vue.JS tiene disponibles modificadores de eventos en la directiva **v-on**.

.once

Permite que el evento se ejecute sólo una vez.

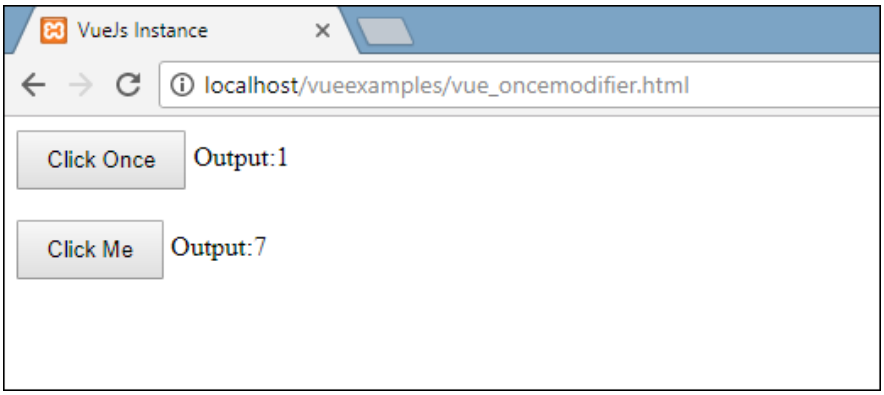
```
<button v-on:click.once = "buttonclicked">Click Once</button>
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <button v-on:click.once = "buttonclickedonce" v-bind:style = "styleobj">Click
Once</button>
      Output:{{clicknum}}
      <br/><br/>
      <button v-on:click = "buttonclicked"  v-bind:style = "styleobj">Click Me</button>
      Output:{{clicknum1}}
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          clicknum : 0,
          clicknum1 :0,
          styleobj: {
            backgroundColor: '#2196F3!important',
            cursor: 'pointer',
            padding: '8px 16px',
            verticalAlign: 'middle',
          }
        },
```

```
        methods : {
            buttonclickedonce : function() {
              this.clicknum++;
            },
            buttonclicked : function() {
              this.clicknum1++;
            }
          }
        });
    </script>
  </body>
</html>
```



En el anterior ejemplo se crean dos botones. Uno que posee el modificador **.once** y otro no. Uno podrá ejecutar tantas veces como el usuario desee el evento y el otro (con el modificador .once) no.

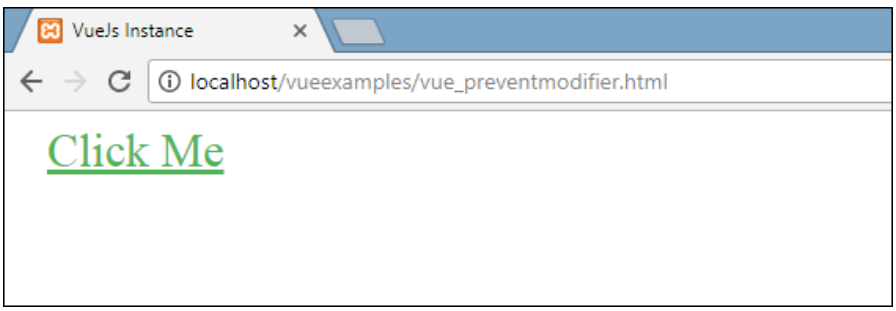
```
<button v-on:click.once = "buttonclickedonce" v-bind:style = "styleobj">Click Once</button>
<button v-on:click = "buttonclicked"  v-bind:style = "styleobj">Click Me</button>
buttonclickedonce : function() {
  this.clicknum++;
},
buttonclicked : function() {
  this.clicknum1++;
}
```



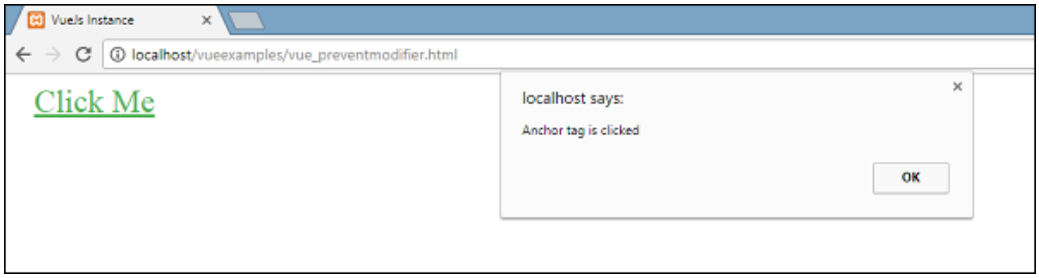
.prevent

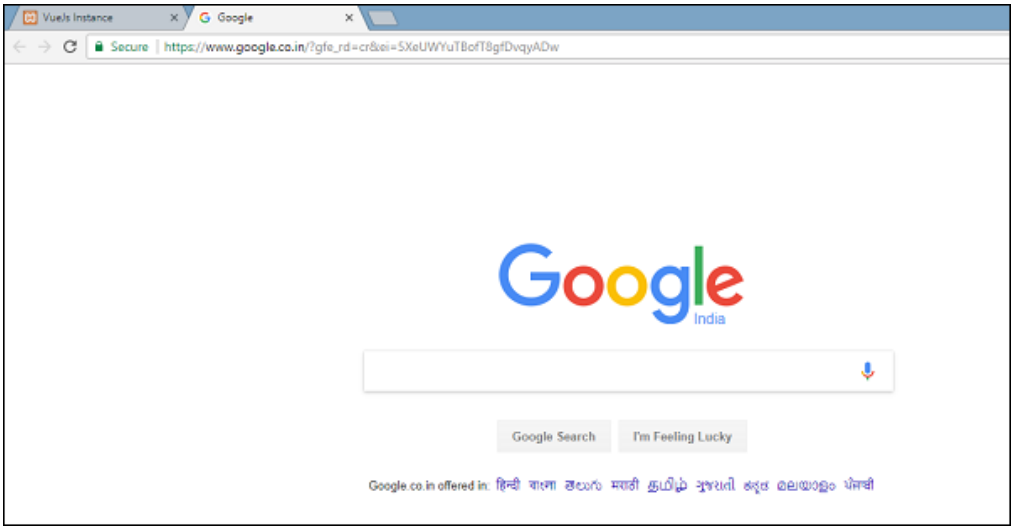
```
<a href = "http://www.google.com" v-on:click.prevent = "clickme">Click Me</a>
```

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <a href = "http://www.google.com" v-on:click = "clickme" target = "_blank" v-bind:style
= "styleobj">Click Me</a>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          clicknum : 0,
          clicknum1 :0,
          styleobj: {
            color: '#4CAF50',
            marginLeft: '20px',
            fontSize: '30px'
          }
        },
        methods : {
          clickme : function() {
            alert("Anchor tag is clicked");
          }
        }
      });
    </script>
  </body>
</html>
```



Si cliqueamos el link nos aparecerá un alert() que nos informa que el link fué cliqueado.





Si el link no posee el modificador **.prevent** se abrirá el enlace al que apunta el link. Dicho comportamiento es el que tienen por defecto los enlaces al ser clickeados. Con .prevent evitamos dicho comportamiento y ejecutamos la función especificada en las comillas.

```
<a href = "http://www.google.com" v-on:click.prevent = "clickme" target = "_blank" v-bind:style = "styleobj">Click Me</a>
```

Modificadores de teclas

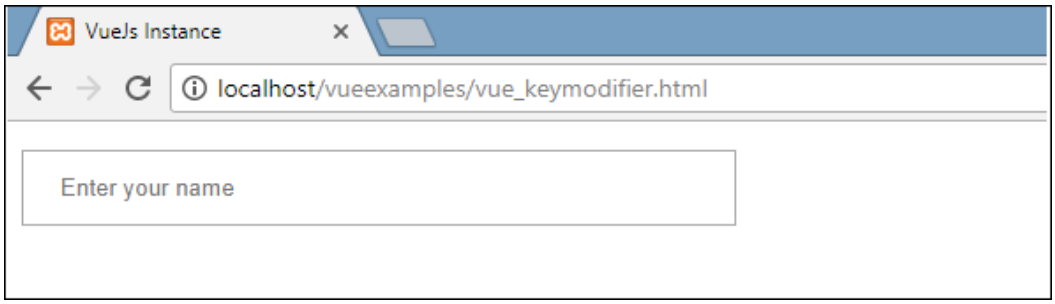
Vue.JS ofrece modificadores de teclas mediante los cuales se manejan los eventos. Si necesitaramos que un textbox llame a un método definido en una instancia Vue sólo cuando se pulsa la tecla enter deberíamos agregar un modificador.

```
<input type = "text"  v-on:keyup.enter = "showinputvalue"/>
```

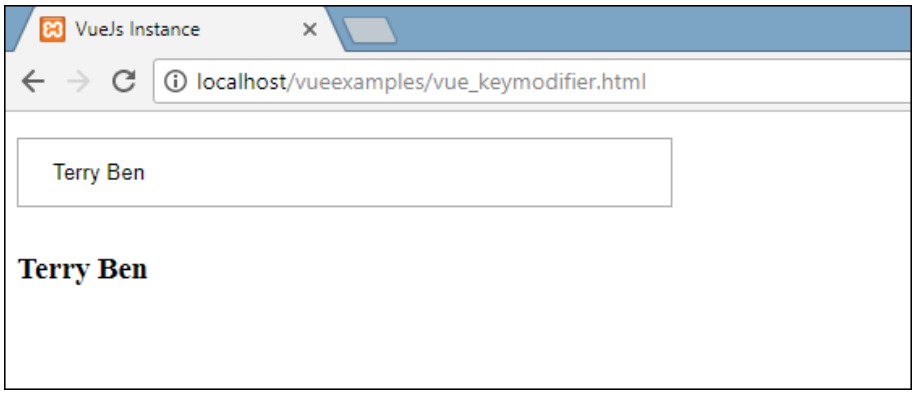
Podemos utilizar múltiples teclas. Por ejemplo: V-on.keyup.ctrl.enter

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <input type = "text" v-on:keyup.enter = "showinputvalue" v-bind:style = "styleobj" placeholder = "Enter your name"/>
```

```
        <h3> {{name}}</h3>
    </div>
    <script type = "text/javascript">
        const vm = new Vue({
            el: '#databinding',
            data: {
                name:'',
                styleobj: {
                    width: "30%",
                    padding: "12px 20px",
                    margin: "8px 0",
                    boxSizing: "border-box"
                }
            },
            methods : {
                showinputvalue : function(event) {
                    this.name=event.target.value;
                }
            }
        });
    </script>
  </body>
</html>
```



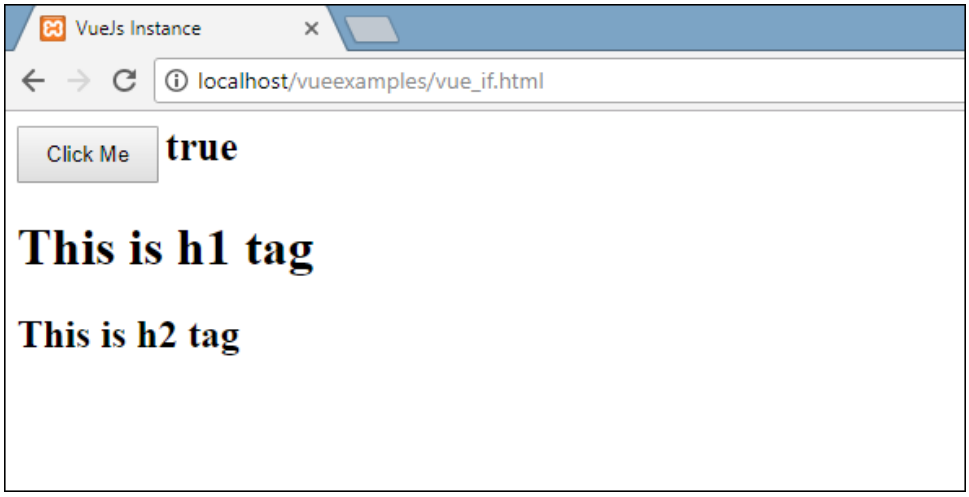
Al presionar Enter:



Renderizado condicional

v-if

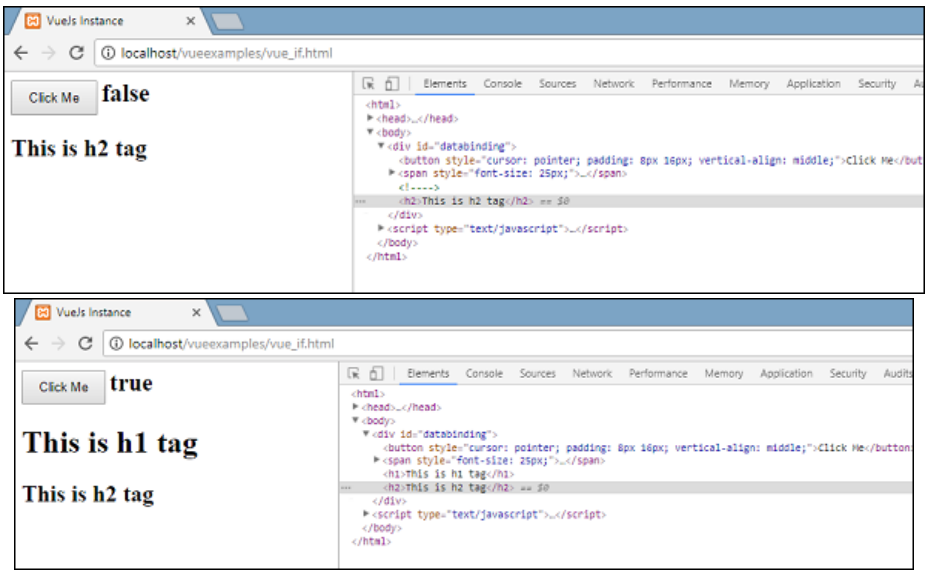
```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <button v-on:click = "showdata" v-bind:style = "styleobj">Click Me</button>
      <span style = "font-size:25px;"><b>{{show}}</b></span>
      <h1 v-if = "show">This is h1 tag</h1>
      <h2>This is h2 tag</h2>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          show: true,
          styleobj: {
            backgroundColor: '#2196F3!important',
            cursor: 'pointer',
            padding: '8px 16px',
            verticalAlign: 'middle',
          }
        },
        methods : {
          showdata : function() {
            this.show = !this.show;
          }
        },
      });
    </script>
  </body>
</html>
```



Se crea un botón con dos etiquetas de encabezados con un mensaje en el interior de ellas. Una variable llamada **show** es declarada e inicializada con un valor **true**. Su valor se muestra cerca del botón. Cada vez que se clickea el botón se llama al método **showdata** que cambia el estado de dicha variable de false a true y viceversa

```
<button v-on:click = "showdata" v-bind:style = "styleobj">Click Me</button>
<h1 v-if = "show">This is h1 tag</h1>
```

Si el valor de la variable show es falso el tag h1 no se mostrará.



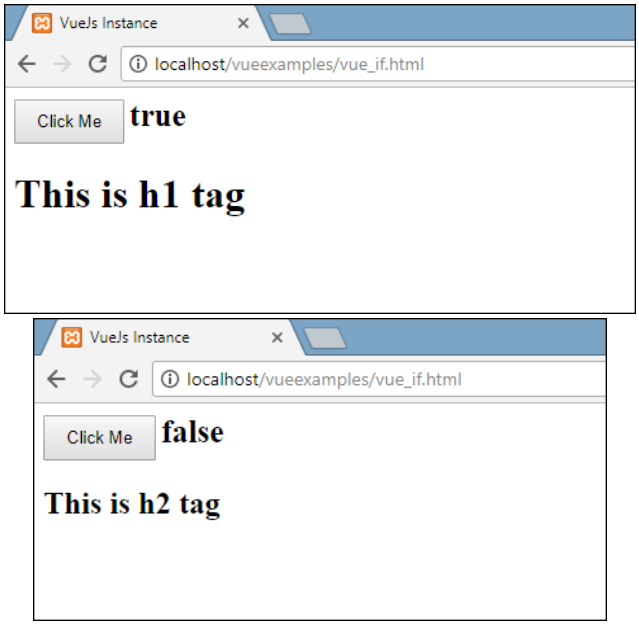
v-else

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <button v-on:click = "showdata" v-bind:style = "styleobj">Click Me</button>
      <span style = "font-size:25px;"><b>{{show}}</b></span>
      <h1 v-if = "show">This is h1 tag</h1>
      <h2 v-else>This is h2 tag</h2>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          show: true,
          styleobj: {
            backgroundColor: '#2196F3!important',
            cursor: 'pointer',
            padding: '8px 16px',
            verticalAlign: 'middle',
          }
        },
        methods : {
          showdata : function() {
            this.show = !this.show;
          }
        },
      });
    </script>
  </body>
</html>
```

v-else se añade utilizando la siguiente sintaxis:

```
<h1 v-if = "show">This is h1 tag</h1>
<h2 v-else>This is h2 tag</h2>
```

Ahora, si **show** es true, se mostrará “This is h1 tag”, si es false, se mostrará “This is h2 tag”



v-show

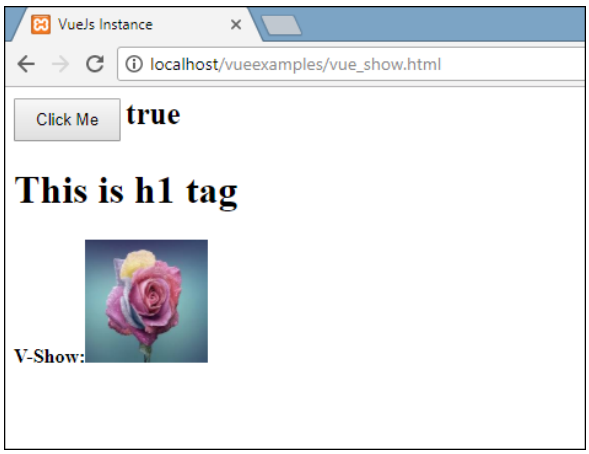
v-show se comporta igual que v-if. También muestra y oculta los elementos basado en una condición. La diferencia es que v-if los elimina del DOM si la condición es falsa y v-show sólo los oculta.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <button v-on:click = "showdata" v-bind:style = "styleobj">Click Me</button>
      <span style = "font-size:25px;"><b>{{show}}</b></span>
      <h1 v-if = "show">This is h1 tag</h1>
      <h2 v-else>This is h2 tag</h2>
      <div v-show = "show">
        <b>V-Show:</b>
        <img src = "images/img.jpg" width = "100" height = "100" />
      </div>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          show: true,
          styleobj: {
            backgroundColor: '#2196F3!important',
            cursor: 'pointer',
            padding: '8px 16px',
            verticalAlign: 'middle',
```

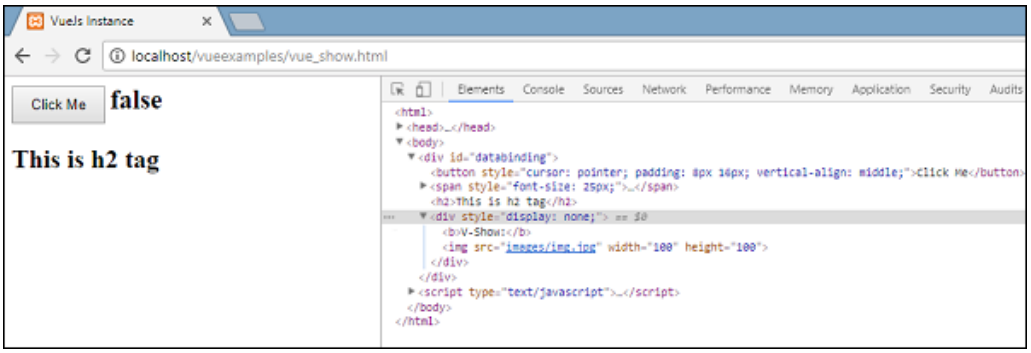
```
            }
        },
        methods : {
            showdata : function() {
                this.show = !this.show;
            }
        },
    });
    </script>
  </body>
</html>
```

Se asigna v-show a un elemento HTML de la siguiente manera:

```
<div v-show = "show"><b>V-Show:</b><img src = "images/img.jpg" width = "100" height = "100" /></div>
```



Si la variable **show** es true se muestra la imagen, si no, la misma se oculta. Si inspeccionamos el DOM vemos que sigue estando, nada más que está ocultada.



Renderizado de Listas

v-for

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <input type = "text" v-on:keyup.enter = "showinputvalue"
        v-bind:style = "styleobj" placeholder = "Enter Fruits Names"/>
      <h1 v-if = "items.length>0">Display Fruits Name</h1>
      <ul>
        <li v-for = "a in items">{{a}}</li>
      </ul>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          items:[],
          styleobj: {
            width: "30%",
            padding: "12px 20px",
            margin: "8px 0",
            boxSizing: "border-box"
          }
        },
        methods : {
          showinputvalue : function(event) {
            this.items.push(event.target.value);
          }
        },
      });
    </script>
  </body>
</html>
```

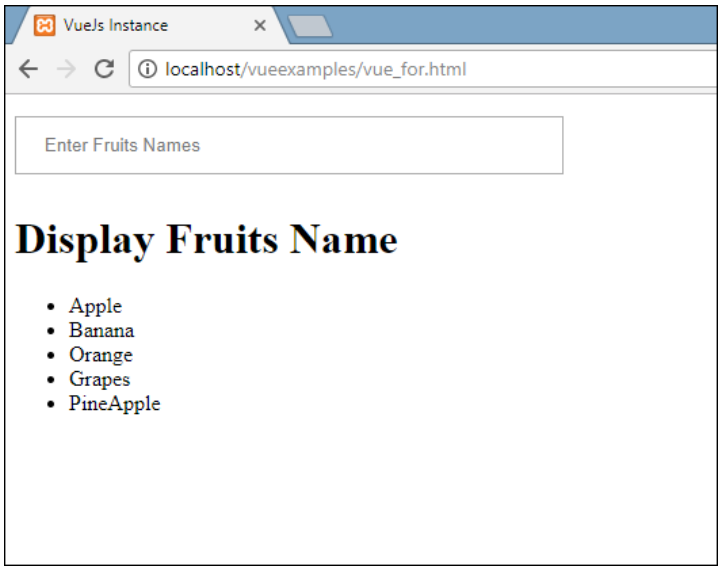
Se declara como array una variable llamada **items**. En la sección de métodos hay uno que se llama showinputvalue, el cual es asignado al textbox y recibe el nombre de las frutas. En el método dichos nombres serán agregados al array mediante el siguiente código.

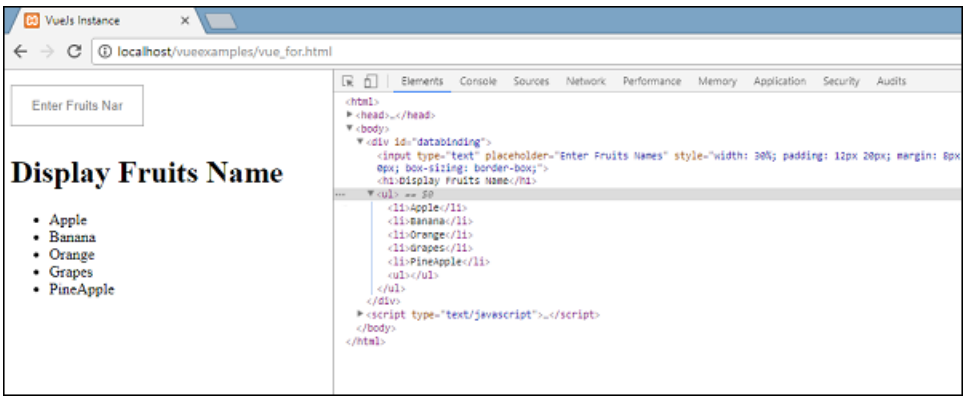
```
showinputvalue : function(event) {
  this.items.push(event.target.value);
}
```

Usaremos **v-for** para mostrar las frutas ingresadas al array mediante el siguiente código. v-for sirve para iterar sobre los elementos presentes en el array.

```
<ul>
  <li v-for = "a in items">{{a}}</li>
</ul>
```

Para iterar sobre el array en bucle utilizamos v-for = ”a in items” en donde **a** tendrá en su interior cada uno de los valores del array de forma secuencial y los mostrará mientras que queden ítems restantes que recorrer.





Si quisieramos mostrar el índice del array se debería utilizar el siguiente código.

```
<html>
  <head>
    <title>VueJs Instance</title>
    <script type = "text/javascript" src = "js/vue.js"></script>
  </head>
  <body>
    <div id = "databinding">
      <input type = "text" v-on:keyup.enter = "showinputvalue"
        v-bind:style = "styleobj" placeholder = "Enter Fruits Names"/>
      <h1 v-if = "items.length>0">Display Fruits Name</h1>
      <ul>
```

```
        <li v-for = "(a, index) in items">{{index}}--{{a}}</li>
      </ul>
    </div>
    <script type = "text/javascript">
      const vm = new Vue({
        el: '#databinding',
        data: {
          items:[],
          styleobj: {
            width: "30%",
            padding: "12px 20px",
            margin: "8px 0",
            boxSizing: "border-box"
          }
        },
        methods : {
          showinputvalue : function(event) {
            this.items.push(event.target.value);
          }
        },
      });
    </script>
  </body>
</html>
```

Para conseguir el índice se agrega entre paréntesis y separado por una coma de **a**, la palabra **index**, cuyo valor es posteriormente mostrado en {{index}}.

```
<li v-for = "(a, index) in items">{{index}}--{{a}}</li>
```

En **(a, index)**, **a** es el valor e **index** es la llave.