Vue JS 2.0

# Vue Instance

Composed by:

1. el: String HTML selector
2. data: Object containing data values
3. methods: Functions to execute whenever render.
4. computed: Functions to execute when reactive dependency is updated.

el: '#app',

data: {

message: 'Hello Vue!'

}

methods: {

changeTitle: function(){

this.message = “hola”;

}

}

# Declarative rendering

Rendering data to the DOM using template syntax:

## HTML

<div id="app">

<p>{{ message }}</p>

<p>{{ getMessage() }}</p>

</div>

**new** Vue({

el: '#app',  
 data: {  
 message: 'Hello Vue!'  
 },  
 methods: {  
 getMessage: **function** () {  
 **return** 'Hello Vue!'  
 }  
 }

});

# Directives

Directives are used to indicate they are special attributes provided by Vue.

## [v-text](https://vuejs.org/v2/api/#v-text) (print text)

Updates the element’s textContent. If you need to update the part of textContent, you should use {{ Mustache }} interpolations.

<span v-text="msg"></span>

<!-- same as -->

<span>{{msg}}</span>

## [v-html](https://vuejs.org/v2/api/#v-html) (escape HTML)

Updates the element’s innerHTML

## [v-show](https://vuejs.org/v2/api/#v-show) (display none if false)

Toggle’s the element’s display CSS property based on the truthy-ness of the expression value.

## [v-if](https://vuejs.org/v2/api/#v-if) (render if true)

Conditionally render the element based on the truthy-ness of the expression value. The element and its contained directives / components are destroyed and re-constructed during toggles.

|  |
| --- |
| <div id="app-3">  <p v-if="seen">Now you see me</p>  </div> |

|  |
| --- |
| var app3 = new Vue({  el: '#app-3',  data: {  seen: true  }  }) |

## [v-else](https://vuejs.org/v2/api/#v-else) / v-else-if

enote the “else block” for v-if or a v-if/v-else-if chain.

## [v-for](https://vuejs.org/v2/api/#v-for) (Loop)

v-for directive can be used for displaying a list of items using the data from an Array:

|  |
| --- |
| <div id="app-4">  <ol>  <li v-for="todo in todos">  {{ todo.text }}  </li>  </ol>  </div> |

|  |
| --- |
| var app4 = new Vue({  el: '#app-4',  data: {  todos: [  { text: 'Learn JavaScript' },  { text: 'Learn Vue' },  { text: 'Build something awesome' }  ]  }  }) |

## [v-on](https://vuejs.org/v2/api/#v-on) (one way binding)

To let users interact with your app, we can use the v-on directive to attach event listeners that invoke methods on our Vue instances:

|  |
| --- |
| <div id="app-5">  <p>{{ message }}</p>  <button v-on:click="reverseMessage">Reverse Message</button>  </div> |

|  |
| --- |
| var app5 = new Vue({  el: '#app-5',  data: {  message: 'Hello Vue.js!'  },  methods: {  reverseMessage: function () {  this.message = this.message.split('').reverse().join('')  }  }  }) |

### Modifiers

* .stop - call event.stopPropagation().
* .prevent - call event.preventDefault().
* .capture - add event listener in capture mode.
* .self - only trigger handler if event was dispatched from this element.
* .{keyCode | keyAlias} - only trigger handler on certain keys.
* .native - listen for a native event on the root element of component.
* .once - trigger handler at most once.
* .left - (2.2.0+) only trigger handler for left button mouse events.
* .right - (2.2.0+) only trigger handler for right button mouse events.
* .middle - (2.2.0+) only trigger handler for middle button mouse events.
* .passive - (2.3.0+) attaches a DOM event with { passive: true }.

### Argument $event - receiving event as argument

For fetching the event as argument we must use protected name “$event”:

|  |
| --- |
| <div id="app-5">  <p v-on:mousemove="counter()">Reverse Message</p>{{count}} –  <p v-on:mousemove="stop($event)">Reverse Message</p>{{count}}  </div> |

|  |
| --- |
| var app5 = new Vue({  el: '#app-5',  data: {  count: 1  },  methods: {  counter: function () {  this.count += 1;  }  Stop: function(event){  Event.stopPropagation();  }  }  }) |

## [v-bind](https://vuejs.org/v2/api/#v-bind) (update value)

Keep and update value.

<div id="app-2">

<span v-bind:title="message">

Hover your mouse over me for a few seconds

to see my dynamically bound title!

</span>

</div>

var app2 = new Vue({

el: '#app-2',

data: {

message: 'You loaded this page on ' + new Date()

}})

Message is bind with Vue so by running app2.message = ‘Hola’ attribute title will change.

### binding an object of attributes

<div v-bind="{ id: someProp, 'other-attr': otherProp }"></div>

## [v-model](https://vuejs.org/v2/api/#v-model) (two ways binding)

v-model directive that makes two-way binding between form input and app state a breeze:

|  |
| --- |
| <div id="app-6">  <p>{{ message }}</p>  <input v-model="message">  </div> |

var app6 = new Vue({

el: '#app-6',

data: {

message: 'Hello Vue!'

}

})

## [v-pre](https://vuejs.org/v2/api/#v-pre) (skip compilation)

## [v-cloak](https://vuejs.org/v2/api/#v-cloak) (hide un-compiled bindings)

This directive will remain on the element until the associated Vue instance finishes compilation. Combined with CSS rules such as [v-cloak] { display: none }, this directive can be used to hide un-compiled mustache bindings until the Vue instance is ready.

## [v-once](https://vuejs.org/v2/api/#v-once) (compile once only )

|  |
| --- |
| Initialize (v-once) For rendering the first value and don’t changing any more, use v-once. In this example first paragraph shows Hello and last shows Adios.  <div id="app-6">  <p>{{ message }}</p>  <p>{{ setMessage() }}</p> </div> <script> **var** app6 = **new** Vue({  el: '#app-6',  data: {  message: 'Hello Vue!'  },  methods: {  setMessage: **function**(){  **this**.message = 'Adios!'  }  } }); |