VUE

# Introducción

Similar al uso de **Angular.js** y **React**, **Vue.js** provee una API para generar interfaces web cuyos datos **mantienen la correspondencia** (en ambos sentidos) entre el cliente y el servidor **a tiempo real**.

Estos elementos se generan mediante la **vista en HTML** y el **modelo en JS.**

**Vue.js** no soporta versiones inferiores a IE8 ya que utiliza **ECMAScript 5**.

# Vinculando datos

Para vincular los datos del modelo y de la vista

HTML (Vista)

|  |
| --- |
| <div id="app">  {{ message }} </div>  O también  <div id="app">  <input type="text" v-model="message" class="form-control"> </div>  JS (Modelo) |

|  |
| --- |
| new Vue({  el: '#app',  data: {  message: 'Hello Vue.js!'  } }) |

Resultado

Hello Vue.js!

# Listados

HTML

<div id="app">  
 <ul>  
 <li v-for="todo in todos">  
 <a href=”#”>{{ todo.text }}</span>  
 </li>  
 </ul>  
</div>

JS

new Vue({  
 el: '#app',  
 data: {  
 todos: [‘1’, ‘2’, ‘3’]  
 }

});

Resultado es una lista de elementos

* 1
* 2
* 3

# Eventos

Usaremos el atributo v-on:***event***=”***function***”para manejar eventos.

HTML

<div id="element" v-on:click="myfunction"></div>

JS

methods: {

        myfunction: function () {

            alert('you just clicked this');

        }

    }

# Filtros y orden

Los filtros en un array de objetos se usan con la clave **filterBy**. De la misma forma podemos ordenar con la clave **orderBy**.

HTML

<div id="myvueinstance" class="container">

    <div class="row">UI List Element</div>

    <input type="text" class="form-control" v-model="filterkey">

    <table class="table table-hover">

        <thead>

        <tr>

            <th><a href="#" v-on:click="sortvia('id')">id</a></th>

            <th><a href="#" v-on:click="sortvia('framework')">framework</a></th>

        </tr>

        </thead>

        <tbody>

        <tr v-for="framework in frameworks | filterBy filterkey | orderBy sortparam order">

            <td>{{ framework.id }}</td>

            <td>{{ framework.framework }}</td>

        </tr>

        </tbody>

    </table>

</div>

JS

var viewmodel = new Vue({

    el: '#myvueinstance',

    data: {

        sortparam: '',

        fitlerkey: '',

        order: 1,

        frameworks: [

            {id: '001', framework: 'angular'},

            {id: '002', framework: 'd3'},

            {id: '003', framework: 'node'},

            {id: '004', framework: 'jquery'},

            {id: '005', framework: 'reveal.js'},

            {id: '006', framework: 'impress.js'},

            {id: '007', framework: 'backbone.js'},

            {id: '008', framework: 'meteor.js'},

            {id: '009', framework: 'express'},

            {id: '010', framework: 'moment'},

            {id: '011', framework: 'underscore'},

            {id: '012', framework: 'gulp'},

            {id: '013', framework: 'react'},

            {id: '014', framework: 'ghost'},

            {id: '015', framework: 'sweetalert'},

            {id: '016', framework: 'select2'},

        ]

    },

    methods: {

        sortvia: function (sortparam, order) {

    this.order = this.order \* -1;

            this.sortparam = sortparam;

        }

    }

});

# Componentes

Podemos declarar componentes que se declaran mediante tags de html.

HTML

<div id="myvueinstance" class="container">

    <alert></alert>

</div>

JS

Vue.component('alert', {

    template: '<div class="alert alert-success" role="alert"><b>Smashing!</b> Nice work.</div>'

});

var viewmodel = new Vue({

    el: '#myvueinstance'

});

Además, podemos añadirles propiedades

HTML

<div id="myvueinstance" class="container">

    <alert type="info" bold="Greetings." msg="This is some information."></alert>

    <alert type="warning" bold="Slow down." msg="You might crash."></alert>

</div>

JS

Vue.component('alert', {

    props: ['type', 'bold', 'msg'],

    template: '<div class="alert alert-{{ type }}" role="alert"><b>{{ bold }}</b> {{ msg }}</div>'

});

var viewmodel = new Vue({

    el: '#myvueinstance'

});