VUE.JS: DECLARATIVE RENDERING

IGNACIO ANAYA
IANAYA89.COM

PROPIEDADES

```
const vm = {

  data() {
    return {
      message: 'Hello Vue!'
    }
  }

}

const app = new Vue(vm) // Instancia de Vue
```

EXPRESIONES

» Mustache syntax

» Templates Dinámicos

```
<div>
  {{ 'hello ' }} {{ 'world!'}}
  {{ 1 + 2 }}
  {{ true ? 'yes' : 'no' }}
  {{ str.replace(/-/g, '') }}
  {{ JSON.stringify(obj) }}
</div>
```

EXPRESIONES
EJERCICIOS

1. Llevar el ejemplo a CodePen.

2. Probar código JS valido para expresiones.

DIRECTIVAS

» Marcadores HTML para manipular el DOM

» Enlazar eventos

» Atributos Dinámicos

» Two Way Data Binding

DIRECTIVAS BÁSICAS

» v-if v-else-if v-else

» v-switch

» v-show

» v-text

» v-html

DIRECTIVAS BÁSICAS
EJEMPLO

```
<div id="app">
  <p v-if="show">
    {{ message }}
  </p>
  <p v-else>
    Texto Oculto
  </p>
</div>

<script>
  const vm = {
    el: '#app',
    data() {
      return {
        message: 'Hello Vue!',
        show: true
      }
    }
  }

  const app = new Vue(vm)
</script>
```

DIRECTIVAS BÁSICAS
EJERCICIOS

1. Llevar el ejemplo a CodePen.
2. Reemplazar la expresión {{ message }} por un directiva v-text.
3. Cambiar el valor de show a true
4. Reemplazar la directiva v-if por v-show y observar las diferencias en el DOM.

v-for

Permite definir un template HTML para un colección de elementos.

v-for
EJEMPLO

```
<div id="app">
  <ul>
    <li v-for="i in items">
      {{ i }}
    </li>
  </ul>
</div>

<script>
  const vm = {
    el: '#app',
    data() {
      return {
        items: [1, 2, 3, 4, 5]
      }
    }
  }

  const app = new Vue(vm)
</script>
```

v-for
EJERCICIO

1. Llevar el ejemplo a CodePen.
2. Reemplazar los elementos de items por un objetos de tipo Person (una propiedad name y otra age).
3. Cambiar la expresión {{ i }} por {{ JSON.parse(i) }}.
4. Cambiar de nuevo la expresión para que muestre explícitamente las propiedades name y age de cada Person.

v-bind

Permite hacer dinámico cualquier atributo tradicional de un elemento HTML.

» src, title, alt, href, id, class, style, etc.

» v-bind: es lo mismo que : (shortcut).

v-bind
EJEMPLO

```
<div id="app">
  <a v-bind:href="url">Link Dinámico</a>
</div>

<script>
  const vm = {
    el: '#app',
    data() {
      return {
        url: 'http://google.com'
      }
    }
  }

  const app = new Vue(vm)
</script>
```

v-bind
EJERCICIOS

1. Llevar el ejemplo a CodePen.
2. Replicar el comportamiento sobre un elemento <img> y el atributo src.
3. Agregar una clase de CSS, una propiedad en data con un valor boolean y usar v-bind en un elemento para aplicar la clase de CSS en base a la propiedad.

v-model

Two-Way Data Binding: vincula un input, select o textarea con una propiedad.

v-model
EJEMPLO

```
<div id="app">
  <input type="text" v-model="name">
  <p>
    {{ name }}
  </p>
</div>

<script>
  const vm = {
    el: '#app',
    data() {
      return {
        name: ''
      }
    }
  }

  const app = new Vue(vm)
</script>
```

v-model
EJERCICIO

1. Llevar el ejemplo a CodePen.
2. Agregar 3 elementos: checkbox, select y textarea.
3. Agregar una propiedad en data para que cada elemento.
4. Enlazar elementos con propiedades usando v-model.

v-on

Permite enlazar cualquier tipo de evento a un método de nuestro ViewModel.

» v-on es lo mismo que @ (shortcut).

» Objeto methods.

v-on
EJEMPLO

```
<div id="app">
  <input type="text" v-model="name">
  <button v-on:click="formatName">Format</button>
  <p>
    {{ name }}
  </p>
</div>

<script>
  const vm = {
    el: '#app',
    data() {
      return {
        name: 'Ignacio Anaya'
      }
    },

    methods: {
      formatName () {
        this.name = this.name.split(' ').join('-').toLowerCase()
      }
    }
  }

  const app = new Vue(vm)
</script>
```

v-on
EJERCICIOS

1. Llevar el ejemplo a CodePen.
2. Agregar un método que imprima el valor de text por consola.
3. Agregar un evento de tipo keyup al evento y enlazarlo con el método creado.
4. Ver ejemplos de modifiers.

PROPIEDADES COMPUTADAS

Propiedades generadas en base a otros propiedades

» Funciones que retornan un nuevo valor

» Objeto computed

PROPIEDADES COMPUTADAS
EJEMPLO

```
<div id="app">
  {{ full }}
</div>

<script>
const app = new Vue({
  el: '#app',

  data: {
    first: 'John',
    last: 'Doe'
  },

  computed: {
    full () {
      return `${this.first} ${this.last}`
    }
  }
})
</script>
```

PROPIEDADES COMPUTADAS
EJERCICIO

1. Llevar el ejemplo a CodePen.

WATCHERS

Funciones para ejecutar acciones en base a cambios de propiedades

» Nombre del watcher = nombre de propiedad

» Objeto watch.

WATCHERS
EJEMPLO

```
<div id="app">
</div>

<script>
const app = new Vue({
  el: '#app',

  data: {
    first: 'John',
    last: 'Doe'
  },

  watch: {
    first(newVal, oldVal) {
      console.log(newVal, oldVal)
    }
  }
})
</script>
```

WATCHERS
EJERCICIO

1. Llevar el ejemplo a CodePen.