

Proyecto de Software

Directivas Vue

Directivas Vue

- Son atributos específicos de Vue que comienzan con **v-**.
 - **v-text**, **v-once**, **v-html**.
 - **v-bind**, **v-model**.
 - Condicionales: **v-if**, **v-else**, **v-else-if**.
 - Bucles: **v-for**.
 - Eventos: **v-on**.
 - **v-show**.

Directiva **v-bind**:

- La **intepolación** `{{}}` no funciona para atributos, se utiliza **v-bind** para relacionar atributos con datos de Vue. Veamos ejemplo **v-bind**.

```
<div id="app">
  <span v-bind:title="message">
    Hover your mouse over me !!
  </span>
</div>
<script>
  const App = {
    data() {
      return {
        message: 'Fecha ' + new Date().toLocaleString()
      }
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Directiva condicional v-if:

- Veamos v-if.

```
<div id="app">
  <p>
    <span v-if="seen">Now you see me</span>
    <span v-else>Oh no 😞</span>
  </p>
</div>
<script>
  const App = {
    data() {
      return {
        seen: true
      }
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Bucles v-for:

- Veamos **v-for**. Modifiquemos **vue_app.products**, agregando: **vue_app.products.push("Manteca")** y eliminando: **vue_app.products.pop()**.

```
<div id="app">
  <ul>
    <li v-for="product in products">
      {{ product }}
    </li>
  </ul>
</div>
<script>
  const App = {
    data() {
      return {
        products: ['Harina', 'Arroz', 'Yerba']
      }
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Métodos y eventos **v-on**:

- La directiva **v-on** nos permite actuar cuando se produzca algún **evento DOM**.
- Dentro de la sección **methods** ponemos el método que se va a disparar cuando el evento se produzca. Veamos **v-on**.

```
<div id="app">
  <p>{{ message }}</p>
  <button v-on:click="reverseMessage">Reverse Message</button>
</div>
<script>
  const App = {
    data() {
      return { message: 'Bienvenidos a Proyecto de Desarrollo!!' }
    },
    methods: {
      reverseMessage: function () {
        this.message = this.message.split('').reverse().join('')
      }
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Eventos v-on:

- Incluso es posible "colgarse" de múltiples eventos:

```
<div v-on="  
  click      : onClick,  
  keyup      : onKeyUp,  
  keydown    : onKeyDown  
>  
</div>
```

- Notar que se modifica el estado de nuestra aplicación sin tocar el DOM, todo eso lo hace Vue.
- El código queda simplificado y enfocado en la lógica de lo que hay que resolver.

Directiva **v-model**:

- Hace la relación bidireccional entre un input y los datos de la aplicación Vue.
- Veamos **v-model** y **v-models**.

```
<div id="two-way-binding">
  <p>{{ message }}</p>
  <input v-model="message">
</div>

<script>
  const App = {
    data() {
      return {
        message: 'Bienvenidos a Proyecto!!'
      }
    }
  }
  vue_app = Vue.createApp(App).mount('#two-way-binding')
</script>
```


Propiedades computadas:

- Veamos [propiedades-computadas](#).
- Nos evita poner demasiada lógica en la visualización.

```
<div id="app">
  <ul>
    <li v-for="product in products">
      {{ product }}
    </li>
  </ul>
  Cantidad de elementos: {{ countProducts }}
</div>
<script>
const App = {
  data() {
    return { products: ['Harina', 'Arroz', 'Yerba'] }
  },
  computed: {
    // a computed getter
    countProducts: function () {
      return this.products.length
    }
  }
}
vue_app = Vue.createApp(App).mount('#app')
</script>
```

Watchers:

- Veamos [watcher](#). Para reaccionar cuando un dato cambia.

```
<div id="app">
  <ul>
    <li v-for="product in products">
      {{ product }}
    </li>
  </ul>
  Cantidad de elementos: {{ countProducts }}
</div>
<script>
const App = {
  data() {
    return {
      products: [ 'Harina', 'Arroz', 'Yerba'],
      countProducts: 3
    }
  },
  watch: {
    products: {
      handler () {
        this.countProducts=this.products.length
      },
      deep: true
    }
  }
}
vue_app = Vue.createApp(App).mount('#app')
</script>
```

Consumiendo una API con Vuejs

Ejemplo básico consultando una API con fetch

- Veamos [lista-api](#). En este caso utilizamos el hook **created** dentro del ciclo de vida de la instancia Vue.

```
<div id="app">
  Datos de: https://jsonplaceholder.typicode.com/
  <ul><li v-for="post in posts">
    {{ post.id }} - <b>{{ post.title }}:</b> {{ post.body }}
  </li></ul>
</div>
<script>
  const App = {
    data() { return { posts: [] } },
    created() {
      fetch('https://jsonplaceholder.typicode.com/posts')
        .then(response => response.json())
        .then(json => { this.posts = json })
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Utilizando el cliente HTTP **axios**

- Realiza los **XMLHttpRequests** del cliente (browser).
- Realiza las peticiones HTTP en node.js (servidor).
- Soporta API de **Promesas de JS**.
- Intercepta y transforma los datos de requerimientos y respuestas.
- Transforma automáticamente a JSON.
- Soporte del lado del cliente para protección contra **CSRF**.

Ejemplo básico consultando una API con axios

- Veamos [lista-api-axios](#). Ahora, abramos directamente el archivo local en el navegador.

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
<script src="https://unpkg.com/axios/dist/axios.min.js"></script>
</head>
<body>
  <div id="app">
    <div v-if="provincias && provincias.length">
      <h1>Provincias Argentinas:</h1>
      <ul>
        <li v-for="p in provincias">
          {{ p.id }} - {{ p.nombre }}
        </li>
      </ul>
    </div>
    <div v-if="errors && errors.length">
      <h1>Errores:</h1>
      <ul>
        <li v-for="error of errors">
          {{error.message}}
        </li>
      </ul>
    </div>
    Fuente: <a href="https://datos.gob.ar/" target="blank">https://datos.gob.ar/</a>
  </div>
</body>
<script>
  const App = {
    data() {
      return {
        provincias: [],
        errors: []
      }
    },
    created() {
      axios.get('https://apis.datos.gob.ar/georef/api/provincias')
        .then(response => {
          // JSON responses are automatically parsed.
          this.provincias = response.data.provincias
        })
        .catch(e => {
          this.errors.push(e)
        })
    }
  }
  vue_app = Vue.createApp(App).mount('#app')
</script>
```

Ciclo de vida de una instancia Vue:



Referencias Vue

- Guia Oficial Vue: <https://vuejs.org/guide/introduction.html>
- Vue Mastery's Intro to Vue 3: <https://www.vuemastery.com/courses/intro-to-vue-3/intro-to-vue3/>
- Ejemplos Vue: <https://vuejs.org/examples/>
- Online [Playground](#)

¿Dudas?

Seguimos con vuejs la próxima semana...