<codoacodo/>

Agencia de Aprendizaje a lo largo de la vida

FULL STACK PYTHON
Clase 19

Vue 1

BA

Introducción a Vue

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 18

DOM y Eventos

- Manipulación del DOM.
- Definición, alcance y su importancia..
- Eventos en JS.
- Eventos. ¿Qué son, para qué sirven y cuáles son los más comunes?
- Escuchar un evento sobre el DOM.

Clase 19

Introducción a Vue

- Introducción a Vue.js. ¿Qué es?
- Renderizado.
- Modificación del DOM.
- Instalación. CDN.
- Directivas condicionales, estructurales y de atributo.
- Métodos y eventos.
- Conceptos claves.

Clase 20

DOM y Eventos

- Aplicaciones Reactivas. Reactividad en 2 sentidos.
- Propiedades computadas.
- Componentes.
- Watchers.
- Acceder a los elementos del DOM utilizando $refs
- Concepto MVC y MVVM.

¿Qué es Vue.js?

Vue (pronunciado /vjuː/, como view) es un **framework** de Javascript progresivo de código abierto. Se utiliza para desarrollar interfaces web interactivas y está diseñado para simplificar el desarrollo web.
La librería central está enfocada solo en la capa de visualización, y es fácil de utilizar e integrar con otras librerías o proyectos web existentes en desarrollos front-end.
La instalación de Vue.js es muy fácil. Cualquier desarrollador puede crear interfaces web interactivas de forma sencilla. Vue fue creado por Evan You, un ex empleado de Google. La primera versión de Vue se lanzó en febrero de 2014.

DOM Virtual

Vue utiliza **DOM virtual**, también utilizado por React. Esto permite que los cambios no se realicen en el DOM, sino que se cree una **réplica del DOM** que está presente en forma de estructuras de datos JavaScript. Siempre que se deban realizar cambios, se realizan en las estructuras de datos de JS y esta última se compara con la estructura de datos original. Luego, los cambios finales se actualizan al DOM real, que el usuario verá cambiar. Esto es bueno en términos de optimización, es menos costoso y los cambios se pueden realizar a un ritmo más rápido.

DOM Virtual

Desde JavaScript haremos cambios en las estructuras de datos propias de JS, esa copia se compara con el DOM y actualiza los cambios. Esta es la gran ventaja de trabajar con Vue, ya que no se actualiza todo, sólo lo que cambia.

El **sistema de renderizado** de Vue se basa en este **DOM virtual**. El concepto de DOM Virtual fue iniciado por React y se ha adaptado en muchos otros marcos con diferentes implementaciones, incluido **Vue**.

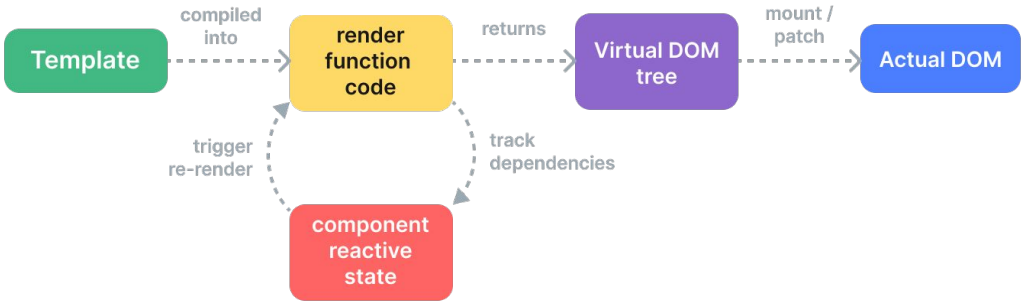
¿Qué es el renderizado?

El término **renderización** (del inglés *rendering*) es un anglicismo para representación gráfica, usado en la jerga informática para referirse al proceso de generar o representar algo en la pantalla. Los resultados de mostrar dicho modelo pueden llamarse **render**.

En una web el renderizado ocurre cuando se visita la página y su contenido se representa en la pantalla.

DOM Virtual y Render

El principal beneficio del **DOM virtual** es que le brinda al desarrollador la capacidad de crear, inspeccionar y componer estructuras de interfaz de usuario que desee de manera declarativa, mientras deja la manipulación del DOM al **renderizador**.



Reactividad en Vue

Una de las mejoras que trae la versión 3 de Vue es en el sistema reactivo.

¿Qué es reactividad? Reactividad es cuando la vista de la aplicación cambia cuando los valores reactivos dentro de estas cambian su valor.

Modelo de enlazado de datos: es la forma a través de la cual JavaScript se conecta (enlaza, comunica) con Vue, permitiendo comunicar el documento HTML con JS.

Componentes en Vue

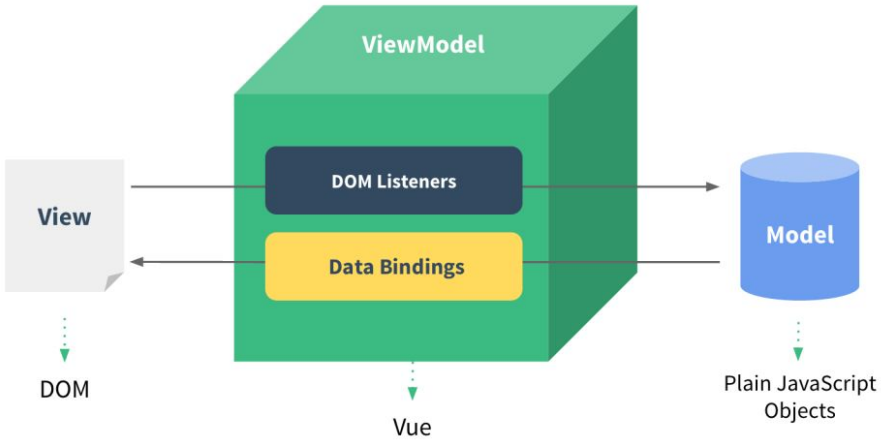
Los **componentes** son una de las características importantes de Vue.js que ayudan a crear elementos personalizados, que se pueden reutilizar en HTML.

Vue permite tomar una página web y dividirla en componentes cada uno con su HTML, CSS y JS necesario para generar esa parte de la página.

De esta manera, hacemos una "intervención por partes", por ejemplo para intervenir sobre el header y footer, que es siempre el mismo.

Instalación

Hay muchas formas de instalar Vue.js. Algunas de las formas de cómo realizar la instalación se comentan a continuación.



Comenzando con VUE.js

La forma más fácil de comenzar a usar Vue.js es crear un archivo index.html e incluir Vue desde el CDN de alguna de las siguientes formas:

```
<script src="https://unpkg.com/vue@3/dist/vue.global.js"></script>
```

```
<script src="https://unpkg.com/vue@3"></script>
```

La referencia al CDN **debe ir al final del <body>** y antes de la referencia a nuestro archivo .js.

```
const { createApp } = Vue
createApp({

}).mount('#app')
```

Esta constante conecta VUE con HTML y tiene un objeto de tipo VUE. Dentro de las llaves habrá propiedades y valores.

Hola mundo con VUE.js

En nuestro primer caso, tendremos en el documento HTML un elemento div con un ID que va a conectar con mi archivo JS:

```
<body>
    <div id="app">
        <p> {{mensaje}} </p>
        <p>Aprendiendo VUE en {{curso}}</p>
    </div>
    <script
src="https://unpkg.com/vue@3/dist/vue.global.js"></sc
ript>
    <script src="intro-vue.js"></script>
</body>
```

```
const { createApp } = Vue
createApp({
    data() {
        return {
            mensaje: 'Hola Mundo con Vue!',
            curso: 'Codo a Codo'
        }
    }
}).mount('#app')
```

La conexión desde JS con mi documento HTML a través de VUE se llama ***renderización declarativa*** (enlazamos el contenido de HTML a través de **Vue**.

Hola Mundo con Vue!

Aprendiendo VUE en Codo a Codo

Hola mundo con VUE.js

{ createApp } = Vue crea un objeto de tipo VUE que en su interior tiene una función **data()** que retorna datos (pares, propiedad: valor). A este objeto lo "montamos" (mount) dentro del #app creado en HTML. De esta manera accedemos a los datos creados dentro del objeto VUE.

```
const { createApp } = Vue
createApp({
    data() {
        return {
            mensaje: 'Hola Mundo con Vue!',
            curso: 'Codo a Codo'
        }
    }
}).mount('#app')
```

Renderización declarativa (interpolación)

Permite insertar texto en el documento HTML, valores, propiedades o atributos. VUE utiliza las **llaves dobles** para encerrar el dato que se quiere mostrar **{{ }}**, es similar a Template String de JS, que lo hace con **${ }:**

{{ mensaje }}

Con el uso de la doble llave **se vinculan los datos con el DOM**, reaccionando a esos nuevos valores. Al cambiar esa réplica del DOM (DOM virtual) lo vemos reflejados en el DOM, ya que el framework al detectar un cambio lo actualiza.

De esta manera, **los datos y el DOM ahora están vinculados**, y **todo es reactivo** (sólo se modifica ante los cambios). Si cambia el valor de la propiedad en el objeto VUE debería ver que se ha **renderizado** con el nuevo valor que se acaba de ingresar.

Directivas

Una **directiva** es el término usado para referirse a algunos atributos especiales que le indican a Vue.js que debe realizar ciertos cambios en un elemento del DOM, cada vez que la expresión asociada con dicha directiva cambie.

Vue utiliza **directivas** para aplicar un comportamiento especial al DOM. Las directivas permiten enlazar VUE con el documento HTML pero con los **atributos** de las etiquetas, no solo con el contenido.

Tienen el prefijo **v-** para indicar que son atributos especiales proporcionados por **Vue**.

Directivas | v-text

La directiva **v-text** es una directiva Vue.js que se usa para actualizar el **textContent** de un elemento con nuestros datos. Es una buena alternativa a la sintaxis {{ }}. Se aplica a un elemento HTML colocando **v-text** como atributo.
+info

```
const { createApp } = Vue
createApp({
    data() {
        return {
            desarrollador: 'Evan You',
            anio: 2014,
        }
    }
}).mount('#app')
```

```
<div id="app">
    VUE fue desarrollado por: <span v-text="desarrollador"></span>
en el año <span v-text="anio"></span>
</div>
```

VUE fue desarrollado por: Evan You en el año 2014

Directivas | v-html

La directiva **v-html** es una directiva Vue.js que se usa para actualizar el **innerHTML** de un elemento con nuestros datos. Esto es lo que lo separa de v-text, lo que significa que mientras v-text acepta una string y la trata como tal, v-html aceptará una string y la convertirá en HTML. +info

```
const { createApp } = Vue
createApp({
    data() {
        return {
            titulo: '<h1>Hola <span
class="clase"> mundo</span></h1>',
        }
    }
}).mount('#app')
```

```
.clase{
    border: 1px blue solid;
    border-radius: 5px;
    background-color: lightblue;
}
```

```
<div id="app">
    <div v-html="titulo"></div>
</div>
```

Directivas | v-bind

La directiva **v-bind** es una directiva Vue.js que permite enlazar (bindear) una variable de Vue con un atributo específico de una etiqueta HTML. Podemos vincular atributos o propiedades de componentes a un elemento. Si ese atributo está vinculado a nuestros datos definidos en la instancia de Vue, se pueden observar cambios dinámicos a medida que cambian los datos. +info

```
const { createApp } = Vue
createApp({
    data() {
        return {
            claseCSS: 'nuevaClase',
            url: 'https://www.google.com.ar/',
        }
    }
}).mount('#app')
```

```
<div id="app">
    <a v-bind:href="url" v-bind:class="claseCSS">Ir a
Google</a>
</div>
```

```
.nuevaClase{text-decoration: none;
    color: green;
    background-color: lightblue;
    padding: 5px;}
```

Ir a Google

Directivas | v-show

La directiva **v-show** es una directiva Vue.js que muestra u oculta algo, según del valor booleano que tenga: si es *true* lo muestra, si es *false* lo oculta con CSS, mediante un *display: none*.

```
const { createApp } = Vue
createApp({
    data() {
        return {
            login: true
        }
    }
}).mount('#app')
```

```
<div id="app">
    Usuario: codoacodo
    <span v-show="login">(Identificado...)</span>
</div>
```

Usuario: codoacodo (Identificado...)

Si el valor es *false* la etiqueta existe en el código fuente resultante del navegador, pero tiene aplicado un atributo style a display: none que hace que no se muestre visualmente. Esta directiva mantiene el código en el HTML pero lo mantiene oculto

Directivas | v-once

Todo elemento que posea la directiva **v-once** será renderizado sólo una vez.

```
<span v-once>Mensaje: {{ message }}</span>
```

Directivas | v-for (renderización de una lista)

Usamos la directiva **v-for** para representar una lista de elementos basada en un Array. La directiva **v-for** requiere una sintaxis especial en forma de ***item in items***, donde los ***items*** son el array de datos de origen y el ***item*** es un alias para el elemento del Array que se está iterando:

```
<ul id="app">
    <li v-for="fruta in frutas">
        {{ fruta }}
    </li>
</ul>
```

```
const { createApp } = Vue
createApp({
    data() {
        return {
            frutas: ['naranja','banana','durazno'],
        }
    }
}).mount('#app')
```

- naranja
- banana
- durazno

Cargamos ítems de en un array mediante la directiva **v-for**. El contenido puede ser dinámico.

fruta es cada elemento de la lista, mientras qué **frutas** es el array en sí.

Directivas | v-if, v-elseif y v-else

Dentro de la vista también podemos tener cierta lógica, por ejemplo con el **v-if** puedes hacer que ciertos elementos se muestren o no dependiendo del valor de una variable.

El **v-if** se puede aplicar a cualquier elemento html o componente. Además de **v-if** disponemos de **v-else** y **v-else-if**, útiles para resolver situaciones condicionales más complejas. Funcionan igual que sus contrapartes de JavaScript.

La directiva **v-if**, a diferencia de **v-show**, si corresponde no mostrar un elemento directamente lo elimina del DOM. Es decir, no se limita a ocultar el elemento si no que lo elimina de la estructura del documento.

Directivas | v-if, v-elseif y v-else

El contenido puede mostrarse dependiendo de una condición. En un listado de productos podemos destacar aquellos que tengan un stock igual a 0. Usamos **v-if** para determinar qué elementos no tienen stock (stock=== 0):

```
<div id="app">
    <ul>
        <li v-for="fruta in frutas">
            {{ fruta.nombre }} - {{ fruta.cantidad }}
            <span v-if="fruta.cantidad===0"> (Sin
Stock)</span>
        </li>
    </ul>
</div>
```

```
const { createApp } = Vue
createApp({
    data() {
        return {
            frutas: [
                {nombre:"naranja", cantidad: 10},
                {nombre:"banana", cantidad: 0},
                {nombre:"durazno", cantidad: 3},
                {nombre:"pera", cantidad: 0}
            ]
        }
    }
}).mount('#app')
```

- naranja - 10
- banana - 0 (Sin Stock)
- durazno - 3
- pera - 0 (Sin Stock)

*Iteramos sobre el array de frutas, mostrando de ese objeto dos propiedades: **nombre** y **cantidad**. La etiqueta **** aparece al cumplirse la condición.*

Directivas | v-if, v-elseif y v-else

Ampliaremos el ejemplo anterior incorporando más elementos al array y estableciendo otras condiciones: a) Stock = 0: Sin stock; b) Stock < 5: Stock bajo y c) Stock >=5 Stock alto. Para esto emplearemos **v-else-if** y **v-else**:

```
<div id="app">
    <ul>
        <li v-for="fruta in frutas">
            {{ fruta.nombre }} - {{ fruta.cantidad }}
            <span v-if="fruta.cantidad===0"> (Sin Stock)</span>
            <span v-else-if="fruta.cantidad<5"> (Stock Bajo)</span>
            <span v-else="fruta.cantidad>=5"> (Stock Alto)</span>
        </li>
    </ul>
</div>
```

- naranja - 10 (Stock Alto)
- banana - 0 (Sin Stock)
- durazno - 3 (Stock Bajo)
- pera - 0 (Sin Stock)

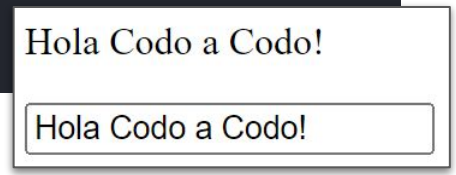
*El **v-for** itera sobre el arreglo y determina con los condicionales cuál es la situación de cada elemento (sin Stock Stock Bajo o Stock Alto)*

Directivas | v-model

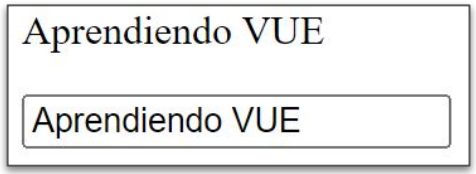
La directiva **v-model** establece un enlace bidireccional, es decir, vincula el valor de los elementos HTML a los datos de la aplicación.

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

```
const { createApp } = Vue
createApp({
    data() {
        return {
            mensaje: 'Hola Codo a Codo!'
        }
    }
}).mount('#app')
```



Lo que ingrese el usuario en el input modifica el mensaje, y el mensaje modifica el 1er párrafo. A medida que se escribe en el input, se ve reflejado el cambio en el párrafo.



Directivas | v-on

La directiva **v-on** permite escuchar eventos DOM y ejecutar instrucciones de JavaScript cuando se producen.

```
<div id="app">
        <button v-on:click="counter += 1">Agregar 1</button>
        <span> Clics: {{ counter }}</span>
 </div>
```

```
const { createApp } = Vue
createApp({
    data() {
        return {
            counter: 0
        }
    }
}).mount('#app')
```

*Se asocia la directiva **v-on** al evento **click** y cada vez que hacemos un click sobre el botón "Agregar 1" se incrementa en uno el valor de la propiedad **counter**.*

Conceptos claves

- **Data Binding:** la función de data binding ayuda a manipular o asignar valores a atributos HTML, cambiar el estilo, asignar clases con la ayuda de la directiva de enlace llamada v-bind disponible en Vue.
- **Event Handling:** v-on es el atributo agregado a los elementos DOM para escuchar los eventos en Vue.JS.
- **Computed Properties:** esta es una de las características más importantes de Vue. Ayuda a escuchar los cambios realizados en los elementos de la interfaz de usuario y realiza los cálculos necesarios. No hay necesidad de codificación adicional para este fin.
- **Templates:** Vue.JS proporciona plantillas basadas en HTML que unen el DOM con los datos de la instancia de Vue. Vue compila las plantillas en funciones virtuales DOM Render. Podemos hacer uso de la plantilla de las funciones de render y para hacerlo tenemos que reemplazar la plantilla con la función de render.
- **Directives:** Vue.JS tiene directivas integradas como v-if, v-else, v-show, v-on, v-bind y v-model, que se utilizan para realizar varias acciones en la interfaz.
- **Watchers:** los Watchers se aplican a los datos que cambian. Por ejemplo, elementos de input en formularios. Aquí, no tenemos que agregar ningún evento adicional. Los Watchers se encargan de manejar cualquier cambio de datos haciendo que el código sea simple y rápido.

Material extra

Artículos de interés

Material de lectura:

- Qué es Vue.js: https://vuejs.org/guide/introduction.html#what-is-vue
- Qué es Vue.js: https://lenguajejs.com/vuejs/introduccion/que-es-vue/
- Tutorial Vue 3
- Directivas Vue.js
- Ejemplos Vue.js
- Ejemplos de App en Vue

Videos:

- Vue 3 desde cero: https://escuelavue.es/cursos/curso-vue-3-desde-cero/
- Curso de Vue.js
- Novedades de Vue 3 en 10 Minutos. ¿Qué cambia?
- Renderizado Condicional, bucles y Listas en Vue 3

No te olvides de dar el presente

Recordá:

- Revisar la Cartelera de Novedades.
- Hacer tus consultas en el Foro.
- Realizar los Ejercicios obligatorios.

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto