EXERCISES QUE TRABAJAREMOS EN EL CUE:

- EXERCISE 1: INSTALACIÓN VUE.JS POR CDN.
- EXERCISE 2: DIRECTIVA V-BIND.
- EXERCISE 3: MÉTODOS, EVENTO REACTIVIDAD.

EXERCISE 1: INSTALACIÓN VUE.JS POR CDN.

Para partir aprendiendo Vue, necesitamos en primer lugar, poder instalarlo. Como se mencionó en la lectura de introducción, esto puede hacerse de distintas maneras; comenzaremos con la forma más sencilla y simple, a través de CDN, como si fuera una librería más de nuestro proyecto web.

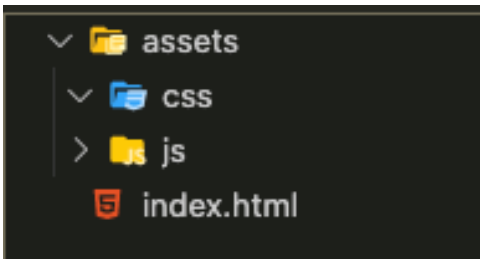
OBTENIENDO EL CDN.

El CDN que usaremos será desde la página: https://unpkg.com/.

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

Estructura de carpetas:

La estructura básica en la cual deberíamos tener ordenados nuestros archivos es:



En primer lugar, está la carpeta assets, donde podremos dejar los archivos necesarios para mantener ordenado nuestro proyecto. En el ejemplo, también se mencionan la carpeta css, para dejar las hojas de estilo; y la carpeta js, para dejar los archivos Javascript, tales como nuestros scripts o librerías.

Estructura del ejercicio de ejemplo.

Ahora, para comenzar a trabajar con Vue, solo necesitamos usar un archivo HTML, en el cual podemos incluir scripts, también es recomendable tener los archivos separados en carpetas, y luego llamarlas en dicho archivo HTML, donde desarrollaremos todo para que quede más claro. Posteriormente, podremos separar y ordenar el código generado en Javascript, en una carpeta js.

```
<!DOCTYPE html>
<html lang="es">
  <head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Intro Vue</title>
  </head>
  <body>
    <div id="app">
    </div>
    <script src="https://unpkg.com/vue/dist/vue.js"></script>
    <script>
      //acá debemos inicializar a Vue
    </script>
  </body>
</html>
```

Inicializar a Vue

Al tener la libreria copiada a través de CDN, lo que debemos hacer es inicializar Vue.

```
<script>
  const app = new Vue({
    el:"#app",
    data:{
      titulo:"Hola bienvenido a vue.js",
    }
  })
</script>
```

- new Vue(): es la forma de inicializar el framework, dentro de ella debemos crear un objeto.
- el: necesita un identificador único para poder insertar el contenido generado por Vue. En el ejemplo, lo hicimos con **id= “app”**, pero puede llamarse como prefieras.
- data: es donde estarán guardados todos los datos que necesitaremos mostrar dentro del ** **. Aquí podemos tener distintos tipos de datos, por ejemplo:
  - Números.
  - Cadenas de texto.
  - Booleanos.
  - Arrays.
  - Objetos.

RENDERIZADO DE DATOS DESDE VUE A HTML

Vue.js utiliza una sintaxis de template basada en HTML, que le permite vincular de forma declarativa, el DOM renderizado a los datos de la instancia de Vue subyacente. Todas las templates de Vue.js son HTML válidas, las cuales pueden analizarse mediante navegadores compatibles con sus especificaciones y analizadores.

Para mostrar los datos generados dentro del objeto data, debemos usar sintaxis de *mustaches* (llaves dobles), dentro el div id= “app”.

```
<div id="app">
    <h2>{{titulo}}</h2>
</div>
```

Ahora, la etiqueta dentro del *mustache,* se reemplazará con el valor de la propiedad título en el objeto de datos correspondiente (data), y ésta se actualizará cada vez que cambie la propiedad.

```
data:{
        titulo:"Hola bienvenido a vue.js",
      }
```

Resultado visto desde el navegador

Otra opción de renderizado

Una opción que nos puede servir para inyectar código HTML desde Javascript, es la directiva v-html , la cual permite agregar etiquetas html desde Vue.

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

```
data:{
  html:"<h1>Html agregado con v-html</h1>"
}
```

Ahora, el resultado visto desde el navegador:

EXERCISE 2: DIRECTIVA V-BIND.

DIRECTIVAS EN VUE

Nos permiten poder agregar ciertas propiedades a atributos HTML, y éstas siempre serán acompañadas por el prefijo **v-**. Algunas de las posibilidades que entrega:

- Enlazar datos.
- Enlazar eventos.
- Crear condicionales if.
- Ejecutar Ciclos for.

Directiva v-bind

Las llaves doble o *mustaches*, no se pueden utilizar dentro de los atributos HTML. En su lugar, use una directiva **v-bind**.

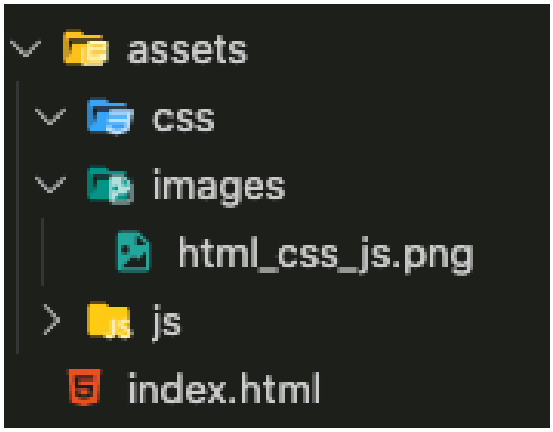
Ésta nos ayuda a poder enlazar datos desde el objeto data, hacia atributos HTML.

Para aprender a utilizar la directiva v-bind, utilizaremos la misma base creada en el ejercicio anterior (1).

Como recomendación, puedes copiar el index.html del ejercicio 1 y borrar la data y el contenido del `<div id="app">`, también puedes hacer lo mismo con el objeto data.

Paso 1: Agregando la carpeta images.

Vamos a crear una nueva carpeta dentro de images, y en esta pondremos una imagen que nos servirá de ejemplo.



Paso 2: Crear etiqueta HTML

Luego, utilizaremos una de las etiquetas más conocidas dentro de HTML, llamada img. Como sabemos, para poder mostrar una imagen, ésta debe hacerse en el atributo src, el cual le indicará a la etiqueta donde obtenerla. Es aquí que aplicaremos la directiva v-bind, para entregarle la ruta de la imagen desde los datos que están en la instancia de Vue.

```
1 <div id="app">
2     <img v-bind:src="imagen_src" alt="herramientas">
3  </div>
```

Paso 3: Creando dato.

Anteriormente, enlazamos el dato “imagen_src”, ahora debemos crearlo como un string, y adjuntaremos el camino donde dejamos la imagen.

```
<script>
    const app = new Vue({
      el:"#app",
      data:{
        imagen_src: "assets/images/html_css_js.png"
      }
    })
  </script>
```

Atajo directiva v-bind

Ha sido creado por los desarrolladores de Vue, éste consiste en solo anteponer dos puntos “:” al atributo HTML que deseemos ligarle algún dato.

```
<div id="app">
    <img :src="imagen_src" alt="herramientas">
  </div>
```

Con esto, obtenemos el mismo resultado, pero ahorrando líneas de código.

EXERCISE 3: MÉTODOS, EVENTO REACTIVIDAD.

Los métodos serán revisados a profundidad en capítulos posteriores, pero para empezar a aprender cómo son, de forma básica, es pertinente agregarlos para observar cómo pueden ser ligados a acciones del usuario, a través del evento “Clic”, y aprovecharemos de ver la reactividad de Vue.

Para comenzar, vamos a crear la estructura de carpetas y un archivo index.html

En dicho archivo index.html, vamos a agregar el siguiente código:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Intro Vue</title>
</head>
<body>
  <div id="app">
    <h1>Contador de likes {{likes}}</h1>
    <button v-on:click="addLike">Agregar like</button>
    <button v-on:click="removeLike">Quitar Like</button>
  </div>
  <script src="https://unpkg.com/vue/dist/vue.js"></script>
  <script src="assets/script.js"></script>
</body>
</html>
```

Representación visual de lo que hace este código

El código HTML que escribimos al comienzo debe generarnos el resultado de la imagen superior. Éste contará con un título H1, el cual aparte del mensaje tiene un dato {{likes}}, que será el encargado de mostrar cuantas veces el usuario hace clic en el botón “agregar like”. Ahora, dicho botón está ligado en sus atributos al evento “clic”, que contiene un método (función) para poder sumar. Por su parte, el botón “Quitar like”, lo que hace es descontar uno de éstos, y es por ello que creamos el método “removeLike”.

Evento v-on:click:

Es uno de los más utilizados en Vue, y éste nos permite enlazar un evento a un método. Ejemplo: el caso de **“addLike”** y **“removeLike”.**

Métodos:

Son funciones que nos permiten agregar lógica a nuestro sitio, de igual forma que utilizamos funciones en JavaScript.

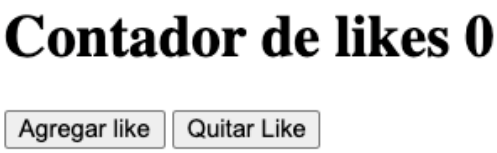
Para lograrlo, debemos crear un archivo JavaScript llamado scrpt.js, dentro de la carpeta js.

Ahora, comenzamos creando la instancia de **vue “new vue”**. En el objeto data, haremos un dato llamado likes, que es un número y será el que muestre cuantas veces se ha agregado o quitado un like.

Dentro del objeto **methods**, crearemos las funciones que indicamos en los botones, y ellos corresponden a **addLike** y **removeLike**. **addLike**, tomará el dato “likes” y le sumará 1 cada vez que el método sea invocado; y **removeLike**, tomará el mismo dato, y preguntará si este es mayor a 0, si es así procederá a restar 1 al dato “likes”.

```
const app = new Vue({
  el:"#app",
  data:{
    likes:0,
  },
  methods:{
    addLike: function(){
      this.likes++;
    },
    removeLike: function(){
      if(this.likes >0){
        this.likes--;
      }
    }
  }
})
```

En este ejemplo se está utilizando un numero **(int)**, para poder sumar o restar cuando el usuario haga click.



Atajo directiva v-on

Como revisamos en el código anterior, para poder enlazar un evento Javascript a Vue, lo debemos hacer con la directiva v-on; pero los desarrolladores de Vue han creado un atajo: @.

Para no tener que escribir

```
<button v-on:click="addLike">Agregar like</button>
```

Podemos escribir el atajo @, de la siguiente forma:

```
<button @click="addLike">Agregar like</button>
```