

ADM

Dudas?

The background of the slide features a long, narrow wooden pier with black metal railings, stretching from the bottom center towards the horizon. The pier is set against a vast, light-colored sky filled with soft, grey clouds. In the far distance, faint outlines of hills or mountains can be seen across the water.

The slide contains a grid of 24 small, rectangular cards, each representing a different topic or concept related to ADM. The cards are arranged in four rows and six columns. Each card has a red header bar with white text and a white body area containing various text and diagrams. Some cards have blue callout boxes with arrows pointing to specific parts of the text or diagrams.

- Card 1:** Descripción de la función.
- Card 2:** Función de la administración.
- Card 3:** Función de la administración.
- Card 4:** Función de la administración.
- Card 5:** Función de la administración.
- Card 6:** Función de la administración.
- Card 7:** Función de la administración.
- Card 8:** Función de la administración.
- Card 9:** Función de la administración.
- Card 10:** Función de la administración.
- Card 11:** Función de la administración.
- Card 12:** Función de la administración.
- Card 13:** Función de la administración.
- Card 14:** Función de la administración.
- Card 15:** Función de la administración.
- Card 16:** Función de la administración.
- Card 17:** Función de la administración.
- Card 18:** Función de la administración.
- Card 19:** Función de la administración.
- Card 20:** Función de la administración.
- Card 21:** Función de la administración.
- Card 22:** Función de la administración.
- Card 23:** Función de la administración.
- Card 24:** Función de la administración.



Prezi

ADM

Dudas?

The slide displays a collection of 20 small, rectangular cards arranged in a grid-like pattern, each containing text and a small icon. The cards are organized into four main columns:

- Column 1:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 2:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 3:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 4:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".



Prezi

Directiva v-on

- **V-on** : Esta directiva se utiliza para vincular eventos a un botón, formulario o input o en cualquier cosa.

v-on:nombreEvento="metodo"

- **nombreEvento=** puede ser cualquier cosa. Puede ser evento de **click**, evento de tipo **mouseover**, evento de **keypress**, **focus**, etc.
- Esto **une los eventos** a un formulario, botón o **cualquier etiqueta** (párrafo, encabezado, div, etc.)

```
<div v-on:click="mostrar">  
  <p>Mostrar</p>  
</div>
```

VueJS

- Es muy importante tener presente que cualquier variable, o método que pongamos en el HTML, debe estar declarado en el código de nuestra app, caso contrario la consola arrojará error.
- Desde la lógica tendremos dentro de la instancia de vue una propiedad llamada **methods**
- Dentro, crearemos las funciones, es decir, tantos métodos como queramos separados por comas.

```
methods : {  
    guardar : function(){  
    },  
    editar: function(){  
    }  
}
```

```
<form v-on:submit="guardar">  
  
    <!--Elementos del form-->  
  
    <button>Guardar</button>  
</form>
```

VueJS

- En el caso de ser un **form**, el comportamiento al enviar el mismo es **recargar la página**.
- Vue posee **modificadores** para los eventos.
- Después del **evento** se coloca **punto** y el **modificador** deseado.

```
<form v-on:submit.prevent="guardar">
```

```
<!--controles del form-->
```

```
</form>
```

No se recarga la página

VueJS

- Cuando usamos la directiva **v-bind** para pasar por argumento atributos de etiquetas podíamos usar el atajo :
- Si queremos usar la directiva **v-on** podemos usar el atajo **@**

```
<p @click="mostrar">Ver</p>
```

Modificadores de teclas

- Cuando queremos escuchar eventos del teclado, podemos mediante el **keycode** asignar un evento.
- Podríamos asignar **keyup** (se dispara cuando el usuario deja de presionar una tecla) si el usuario presiona determinada tecla.

VueJS

- La función enviar se ejecutará cuando el usuario suelta la tecla enter, en este caso pasando su keycode.

```
<input v-on:keyup.13="enviar" type="text" v-model="dato"/>
```

- <https://keycode.info/> --> presionan la tecla del teclado que quieren averiguar cual es su keycode y te lo devuelve

32

event.key

(Space character)

event.location

(?)

0
(General keys)

event.which

(deprecated)

32

event.code

Space

VueJS

- Vue ofrece poder trabajar con alias para las teclas más utilizadas:

.enter

.tab

.delete (captura ambas teclas “Delete” y “Backspace”)

.esc

.space

.up

.down

.left

.right

Enlace de datos de 2 vías

- Cuando trabajamos con formularios tenemos que tener presente como vamos a tratar los datos.
- Hay un concepto fundamental en vuejs que tiene que ver con el **enlace de datos**.
- Hasta ahora vimos como **los datos inicializados desde el modelo, se mostraban en la vista. (Enlace de una vía.)**
- Para que el **enlace de datos de 2 vías** ocurra necesitamos de la **participación del usuario**.

Vuejs

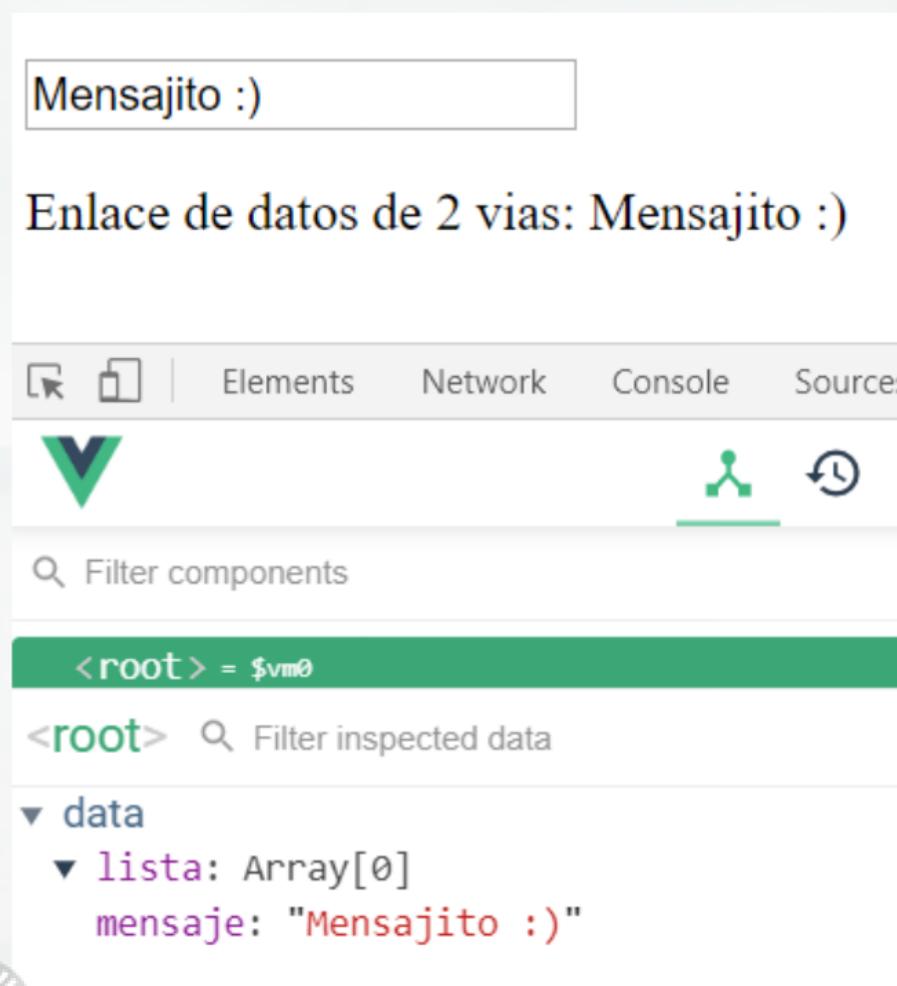
- Con la directiva **v-model** podemos crear enlace de datos bidireccionales.
- Se usa en elementos de tipo **input**, **textarea** y **select** en formularios.
- La directiva busca automáticamente la manera correcta de **actualizar el elemento** según el tipo de entrada.
- Al usar **v-model**, le decimos a Vue **qué variable** debe unirse con esa **entrada**, en este caso el **mensaje**

```
<input type ="text" v-model="mensaje" />
<p>{{mensaje}}</p>
```

```
data: {
  mensaje:" Mensajito :)",
}
```

VueJS

- El enlace de datos bidireccional significa que *si cambia el valor de un modelo en su vista, todo se mantendrá actualizado*



Mensajito :)

Enlace de datos de 2 vias: Mensajito :)

Elements Network Console Source

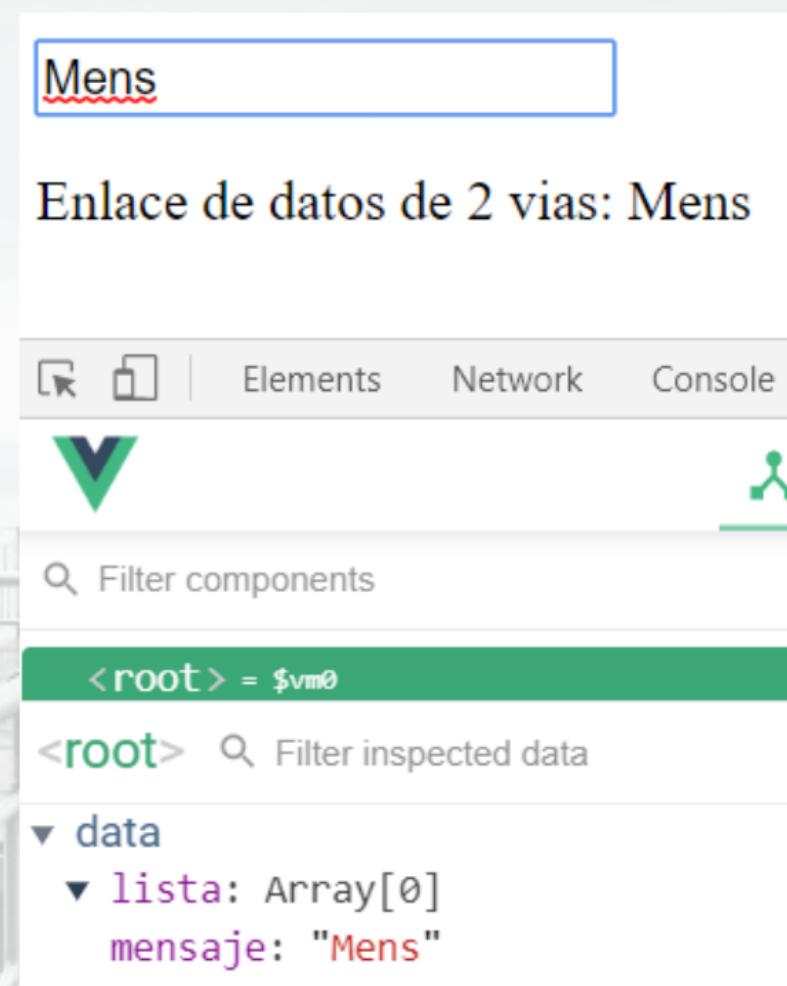
<root> = \$vm0

<root> Filter inspected data

▼ data

 ▼ lista: Array[0]

 mensaje: "Mensajito :)"



Mens

Enlace de datos de 2 vias: Mens

Elements Network Console

<root> = \$vm0

<root> Filter inspected data

▼ data

 ▼ lista: Array[0]

 mensaje: "Mens"

Manipular datos

- Los formularios son un medio efectivo para lograr interacción con el usuario
- Si queremos que el usuario ingrese datos e ir mostrándoselos a medida que lo hace tenemos que resolver la estructura que poseerá esa info.
- Por empezar, tener presente que las variables deben inicializarse dentro de data.
- Y las funciones que creermos, dentro de la propiedad methods.



VueJS

```
var app= new Vue({  
    el: ".contenedor",  
    data: {  
        mensaje:"Mensajito :)",  
        lista:[],  
    },  
    methods:{  
        agregar : function(){  
            this.lista.push(this.mensaje);  
            this.mensaje="";  
        }  
    }  
})
```

- Partimos de un dato inicializado desde el modelo, vamos agregar cada mensaje nuevo que el usuario escriba.
- Para poder acceder a las propiedades del objeto data desde las funciones, tenemos que usar **this**
- Por medio del método **push** vamos a ir agregándolos al array lista.

VueJS

```
<form v-on:submit.prevent>
  <input type ="text" v-model="mensaje" @keyup.13="agregar"/>
</form>
<p> Enlace de datos de 2 vias: {{mensaje}}</p>

<ul>
  <li v-for="(item, index) in lista">{{item}}</li>
</ul>
```

- Con el modificador `.prevent` anulamos el comportamiento por defecto del `form` en el `submit`.
- Ejecutando la función `agregar`, cada vez que el usuario presione `enter` el mensaje lo vamos agregando por medio del `push` al array `lista`
- Por medio de la directiva `v-for`, recorremos cada `item` que posea el array

VueJS

- Si queremos lograr más interacción y que el usuario pueda borrar los datos que fue ingresando, necesitamos crear otra función y desde la vista un botón asociado a la misma función.

```
<li v-for="(item, index) in lista">{{item}}  
    <button @click="borrar(index)">Borrar</button>  
</li>
```

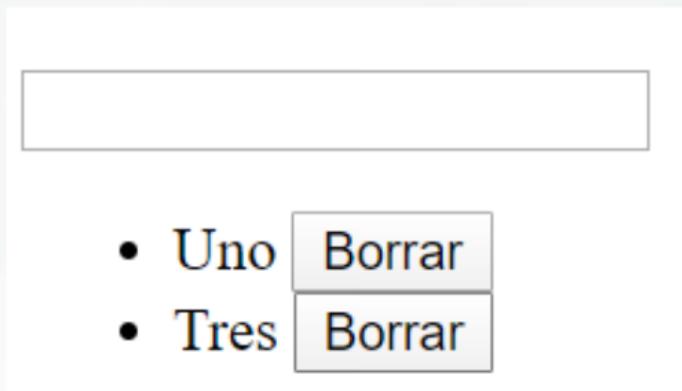
Cada vez que escribamos una función, dentro de la propiedad **methods** cada función debe estar separada por una coma.

```
methods:{  
    agregar : function(){  
    },  
    borrar : function(){  
    }  
}
```

VueJS

- El **método splice** nos permite **elegir el elemento del array que queramos eliminar, (index)** como segundo argumento le indicamos cuantos a partir del número del index. (En este caso, ese elemento solo.)

```
borrar : function(index){  
    this.lista.splice(index,1)  
}
```



Una vez eliminado el elemento Dos, los elementos que quedan aún en el array se seguirán mostrando.

VueJS

- Si quisieramos verificar que el usuario ingresa un dato y no guarda un elemento vacío:

```
data: {  
  mensaje:"",  
  verificar:true  
}
```

- Si la variable está vacía, cambiamos el valor de la variable **verificar** a **false**
- Caso contrario **verificar** es **true**, y podemos hacer push

```
methods:{  
  agregar: function(){  
    if (this.mensaje == "") {  
      this.verificar = false  
    }else{  
      this.verificar = true  
      this.lista.push(this.mensaje);  
      this.mensaje="";  
    }  
  }  
}
```

VueJS

```
<div class="error" :class="verificar ? 'classBien' : 'classError'">  
  <p>No puede quedar vacío</p>  
</div>
```

- El mensaje solo se le mostrará al usuario en caso de soltar el enter y el input no tenga contenido.

Cuántos elementos fue ingresando el usuario?

- Desde el objeto data podemos inicializar una variable total en 0
- A medida que se haga el push establecer que el valor de total va a ser el .length de lo que tenga el array lista

```
this.total= this.lista.length;
```

VueJS

- Cuando el usuario borre algún elemento deberíamos ir restando del valor de total

```
borrar: function(index){  
    this.lista.splice(index,1)  
    this.total --  
},
```

Lista de elementos seleccionable

- Por defecto el usuario ingresa tareas pendientes, asociadas a una clase "pendientes"
- Y si selecciona alguna cambiamos la clase a "terminadas"

<p> Datos Ingresados: {{total}}</p>

Lista Pendientes:

Enlace de datos de 2 vias:

Concirnar tacos, indice: 0
Ordenar escritorio, indice: 1
Regar plantas, indice: 2

Borrar
Borrar
Borrar

Datos Ingresados: 3

Terminadas : 1

VueJS

- En este caso vamos a crear un objeto que contenga 2 propiedades, **mensaje** y **css**
- Cuando el usuario vaya cargando las nuevas tareas iremos haciendo push de estos objetos al array lista

```
<li v-for="(item, index) in lista" :key="index" :class="item.css" >
  <span @click="toggle(index)">{{item.mensaje}}, indice: {{index}}</span>
  <button @click="borrar(index)">Borrar</button>
</li>
```

:class="item.css --> lo definiremos con la clase **pendientes** cada vez que el usuario agregue un elemento

VueJS

- Desde la lógica:

```
toggle: function(index){  
  if (this.lista[index].css == "pendientes"){  
    this.sumarTerminadas ++; //sumará y su css pasará a terminadas  
    this.lista[index].css = 'terminadas'  
  }else{  
    this.sumarTerminadas --; //restará y el su css pasará a pendientes  
    this.lista[index].css = 'pendientes'  
  }  
}
```

this.lista[index] --> Nos va a permitir afectar a la posición de ese elemento que estamos haciendo click

Ir a la ferreteria, indice: 0
Limpiar estantes, indice: 1

Borrar

Borrar

Filtros

- En sus primeras versiones el core de vuejs tenía incorporado un recurso muy útil para la muestra de información.
- Actualmente la versión 2 no la trae dentro de sus funcionalidades básicas.
- Vuejs nos permite crear nuestros propios filtros:

```
filters: {  
    mayuscula: function (value) {  
        if (!value) return ''  
        value = value.toString()  
        return value.charAt(0).toUpperCase() + value.slice(1)  
    }  
}
```

Filtros

`charAt(0)` selecciona el primer carácter de la cadena.

`toUpperCase()` la pasamos a mayúsculas.

`slice()` obtenemos el resto de la cadena (desde el segundo carácter, índice 1 , hasta el final de la cadena).

- Hay varios recursos desarrollados para incorporar dentro de nuestros proyectos.
- <https://github.com/freearhey/vue2-filters>, ofrece una colección de filtros.

VueJS

- Para poder incorporarlo a nuestro proyecto necesitamos agregar un archivo

```
<script src="vue.js"></script>
<script src="vue2-filters.min.js"></script>
```

- Cualquier recurso que dependa de Vuejs, **siempre va debajo de la etiqueta script que lo incorpora**, caso contrario arrojará error.
- Los **filtros solo cambian la representacion del dato**, no cambia el dato original.
- Los **filtros** pueden usarse **dentro de las interpolaciones y en la directiva v-bind**

VueJS

- Dentro de la interpolación **se usa pipe** y luego **el nombre del filtro** que queremos aplicar.

```
<p> {{mensaje | nombreFiltro}} </p>
```

- Funcionalidades básicas que podemos obtener :
- mayúsculas
- Primer letra en mayúscula
- minúscula
- colocar texto por defecto si la variable no tiene contenido.
- mostrar un número de caracteres
- números
- moneda
- etc.

VueJS

- El valor que contenga la interpolación mensaje será afectado por el filtro que se aplique :

`{{ mensaje | uppercase }}` → en mayúsculas

`{{ mensaje | lowercase }}` → en minúsculas

`{{ mensaje | capitalize }}` → primer letra de cada palabra en mayúsc..

`{{ mensaje | capitalize({ onlyFirstLetter: true }) }}`

→ de toda la frase, solo la primer letra de la primer palabra.

`{{ mensaje | placeholder('Texto por defecto') }}`

→ si la variable no tiene contenido, le asignamos un texto por defecto

VueJS

`{{ mensaje | truncate(5) }}` → muestra hasta 5 caracteres

`{4000.67 | number('0.0000')}` → agregar más decimales

`{ 4000 | number('-0') }` → agregar el signo de suma o resta

`{ 400200 | number('0a') }` → muestra la k de miles (K)

`{ 4000300 | number('0a') }` → muestra la m de millones (M)

`{ 4000000 | number('0,0', { thousandsSeparator: ' ' }) }`

→ un separador de miles diferente, con espacio o punto

`{ 4000000 | number('0.00', { decimalSeparator: ',', ' }) }`

→ un separador decimal diferente, con coma, barra, etc



Dudas?

ADM

Dudas?

The slide displays a collection of 20 small, rectangular cards arranged in a grid-like pattern, each containing text and a small icon. The cards are organized into four main columns:

- Column 1:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 2:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 3:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".
- Column 4:** Contains 5 cards. The first card is titled "Punto de partida". The second card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The third card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fourth card is titled "Punto de partida" and includes a red box with the text "Punto de partida". The fifth card is titled "Punto de partida" and includes a red box with the text "Punto de partida".



Prezi