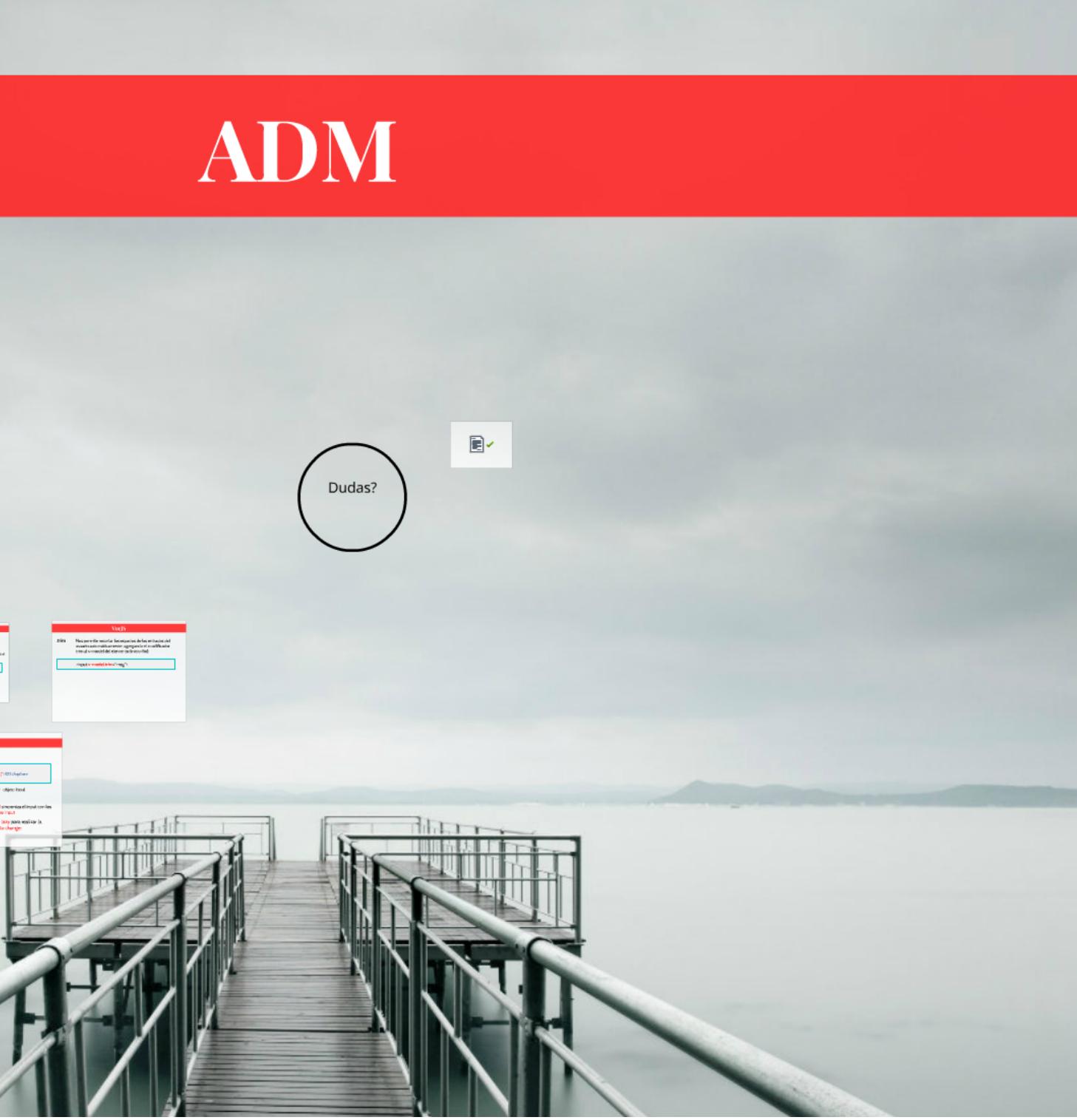


# ADM

Dudas?



Prezi

# ADM

Dudas?



Prezi

## Formularios

- Los formularios son el elemento de interacción elemental de todo sistema, web o aplicación.
- Hay ciertas cuestiones que tenemos que recordar dependiendo del elemento que queremos usar para el ingreso de datos.
- La directiva `v-model`, dentro del mundo vue, nos permite crear el data binding de 2 vías o el enlace de datos de 2 vías.
- Controles como `inputs`, `textarea` y `select` poseen esta directiva para poder actualizar de manera correcta el elemento según el tipo de entrada.

# VueJS

- En los elementos de tipo textarea la interpolacion no funciona

```
<textarea>{{text}}</textarea>
```



```
<textarea v-model="form_data.comentario"></textarea>
```



Checkbox simples con valor booleano:

```
<input type="checkbox" id="checkbox" v-model="checked">
<label for="checkbox">{{ checked }}</label>
```

- Los valores que retorna la elección pueden ser **true o false**

# VueJS

## Múltiples checkboxes vinculados al mismo Array:

```
<div>
  <input type="checkbox" id="pepe" value="Pepe" v-model="nombres">
    <label for="Pepe">Pepe</label>
  <input type="checkbox" id="thor" value="Thor" v-model="nombres">
    <label for="thor">Thor</label>
  <input type="checkbox" id="Randy" value="Randy" v-model="nombres">
    <label for="randy">Randy</label>

  <span>Nombres: {{ nombres }}</span>
</div>
```

```
new Vue({
  el: '#contenedor',
  data: {
    nombres: []
  }
})
```

Para guardar estos valores, cada checkbox debe tener su propio **value**. Caso contrario devolverá **null**

Recordar que *hay que definir estos valores como array dentro la logica de vuejs*

## Radio

```
<input type="radio" id="uno" value="Negro" v-model="color">
<label for="negro">Negro</label>

<input type="radio" id="Dos" value="Blanco" v-model="color">
<label for="blanco">Blanco</label>

<span>Eligió: {{ color }}</span>
```

## Selección simple

```
<select v-model="selected">
<option disabled value="">Seleccione un elemento</option>
<option>A</option>
<option>B</option>
<option>C</option>
</select>
<span>Seleccionado: {{ selected }}</span>
```

Desde la logica la variable selected debe inicializarse

```
data: {
  selected: ''
}
```

# VueJS

- Si el valor inicial del `v-model` no coincide con ninguna de las opciones, el elemento `<select>` se representará en un estado “**unselected**”
- En iOS, esto hará que el usuario **no pueda seleccionar el primer elemento porque iOS no dispara un evento de tipo change**.
- Se recomienda dar una **opción deshabilitada** con un **value vacío**,

## Selección de múltiple

```
<select v-model="selected" multiple>
  <option>A</option>
  <option>B</option>
  <option>C</option>
</select>
<span>Seleccionados: {{ selected }}</span>
```

## Select dinámico con v-for

```
<select v-model="elegido">
  <option v-for="item in options" v-bind:value="item.value">
    {{ item.texto }}
  </option>
</select>
<span>Seleccionado: {{ elegido }}</span>
```

Desde la lógica :

```
data: {
  elegido: 'A',
  options: [
    { texto: 'Uno', value: 'A' },
    { texto: 'Dos', value: 'B' },
    { texto: 'Tres', value: 'C' }
  ]
}
```

## Vincular valores

- En los casos de **radio**, **checkbox** y **option** de select, los **valores** de vinculación del v-model suelen ser strings o booleanos

```
<input type="radio" v-model="elegido" value="a">
```

(elegido es una string "a" cuando está chequeado)

```
<input type="checkbox" v-model="toggle">
```

(`toggle` es verdadero o falso )

# Vuejs

```
<select v-model="seleccion">  
    <option value="PC">Pc</option>  
</select>
```

(`seleccion` es un string "PC" cuando se selecciona la primera opción)

- Puede pasar que queramos **vincular el valor a una propiedad dinámica** en la instancia de Vue.
- Podemos usar **v-bind** para lograr eso. Además, el uso de **v-bind** nos permite vincular el valor de entrada a valores no string.

The slide has a red header bar with the word 'Vuejs'. Below it, there's a yellow box labeled 'En checkbox' containing the following code:

```
<input type="checkbox" v-model="sw" true-value="si" false-value="no"/>
```

Below the code, there's a note: "Los atributos **true-value** y de **false-value** no afectan la entrada del atributo value. Los browsers no envian casillas sin marcar".

There's also a yellow box labeled 'En Radio' containing the following code:

```
<input type="radio" v-model="elegido" v-bind:value="a">
```

Below the code, there's a note: "Cuando este elemento este marcado su valor será 'a'".

# Vuejs

## En checkbox

```
<input type="checkbox" v-model="sw" true-value="si"  
false-value="no"/>
```

- Los atributos **true-value** y de **false-value** no afectan la entrada del atributo value. Los browsers no envian casillas sin marcar

## En Radio

```
<input type="radio" v-model="elegido" v-bind:value="a">
```

Cuando este elemento este marcado su valor será "a"

## En select

```
<select v-model="selected">  
  <option v-bind:value="{ number: 123 }">123</option>  
</select>
```

objeto literal

## Modificadores

**.lazy** De forma predeterminada, **v-model** sincroniza el input con los datos **después de cada evento de tipo input**

Podemos agregar el **modificador lazy** para realizar la sincronización **después del evento change**:

# VueJS

```
<input v-model.lazy="msg" >
```

## .number

Podemos configurar que las entradas del usuario se escriban automáticamente **como un número**, agregando el modificador **number** al **v-model** del elemento:

```
<input v-model.number="edad" type="number">
```

- Esto suele ser útil ya que con **type="number"**, el valor returned por el elemento HTML siempre es una **cadena de texto**

# VueJS

.trim

Nos permite recortar los espacios de las entradas del usuario automáticamente, agregando el modificador trim al v-model del elemento (inicio-fin):

```
<input v-model.trim="msg">
```

# Dudas?



# ADM

Dudas?



Prezi