

Forms and Inputs with Vue.js



Outline and Learning Objectives

- **Vue.js Elements, a Quick Recap:**
 - to revise and understanding more in detail Vue.js Elements used for web-based mobile app development
- **Vue.js (2-Way) Data Binding with Form Elements:**
 - to understand how to bind a wide range of form elements with Vue.js properties by guaranteeing 2-Way real-time synchronisation between view and model
- **Vue.js Modifiers:**
 - to understand how to apply constraints and manipulate automatically data, in Form Elements, supported by Vue.js
- **Suggestions for Reading**

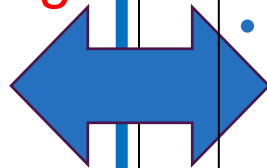
Vue.js Elements, a Quick Recap

Vue.js Elements: how and when to use them

- **Data:** it contains **properties**; some properties could come from **external sources**

- **Methods:** executed when called, **accept parameters**, **not used for automatic 2-way binding**

- **Computed Properties:** recomputed automatically when other inner (inside of its function) **data** variables are updated, **cannot accept parameters**, used for **developing robust user interactions** (automatic 2-way binding)



- **v-text:** used for populating the text content of the element where it is applied
 - example: **v-text="product.description"**
- **v-bind:** used for **binding HTML attributes**
 - example: **v-bind:src="product.image"**
- **v-model:** used for **binding Input Elements** (or Components)
 - example: **v-model="newTask"**
- **v-on:** used for managing DOM events
 - example: **v-on:click="toggleShowProduct"**
- Others: **v-show** (depending on a condition, can hide html, keeping HTML and hiding by using CSS) **v-if** (depending on a condition, can hide html, by completely removing it), **v-else** , **v-html** ...
- **Further Documentation:** [link](#)

- **Today** we will **revise/reuse** some of the above, and **introduce:** **v-for** , **v-bind:true-value**, **v-bind:false-value** , **v-model.trim** , **v-model.number** , ...

Where we stopped 2 weeks ago:

- We created a **product page** that shows **one product**
- It has a **'Add to cart' button** that will be **disabled once the stock level is zero**
- It has a **'Checkout' button** that **switch** the view between the **product and checkout page**
- So far, the **'checkout' page** is still **empty**



Cat Food, 25lb bag

A 25 pound bag of *irresistible*, organic goodness for your cat.

Price: 2000

Add to cart

This Week:

- We will create the **checkout page** similar to this
- On which the user can **enter her details**
- The **details will be checked** to ensure they are **correct**
 - For example, **only numbers are entered for phone number**

Pet Depot Checkout

Enter Your Information

First Name: Last Name:

Address:

City:

State: Zip / Postal Code:

☒ Ship As Gift? ☒ Home ☐ Business

First Name:
Last Name:
Address:
City:
Zip:
State:
Method: Home Address
Gift: Send As A Gift

Vue.js (2-Way) Data Binding (1/3)

`v-model` applied to Inputs and
Selects

v-model and 2-Way Data Binding

Pet Depot Checkout

Enter Your Information

First Name:

Last Name:

- We need to **get the user details from the html form**
- Previously we did this with `document.getElementById('#inputFieldID').value`
- Vue.js has a directive called `v-model` to make this easier
- `v-model` works **with all form inputs: text boxes, text areas, check boxes**, radio buttons, and dropdown menu.

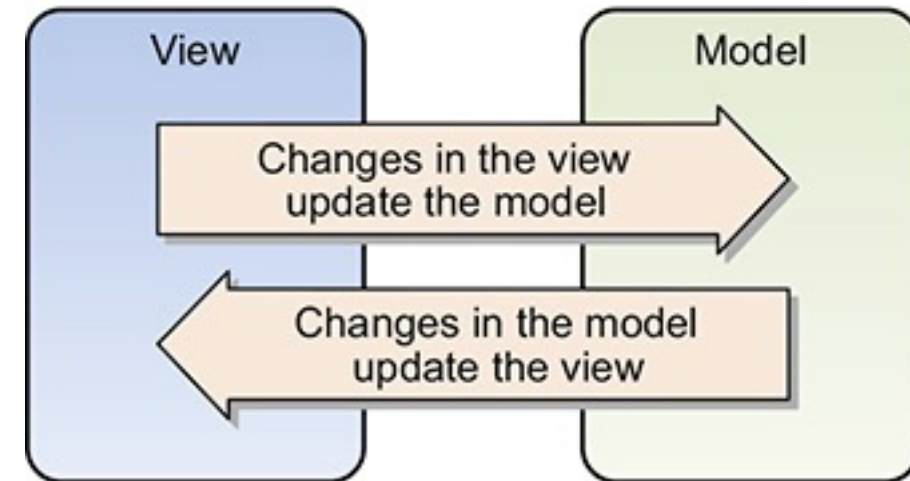
v-model directive

tag name

two-way data bound object

```
<input v-model="order.lastName"/>
```

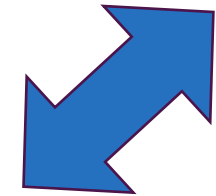
Two-way data binding



- When new value is entered in the form, the value of the bound vue property also changes
- When the value of the vue property changes, so does the value in the html input element
- `v-once` can be used for **one-way binding**

Binding Inputs with v-model: the Name

- As the user types in the name, it shows up in the pane below in **real-time**

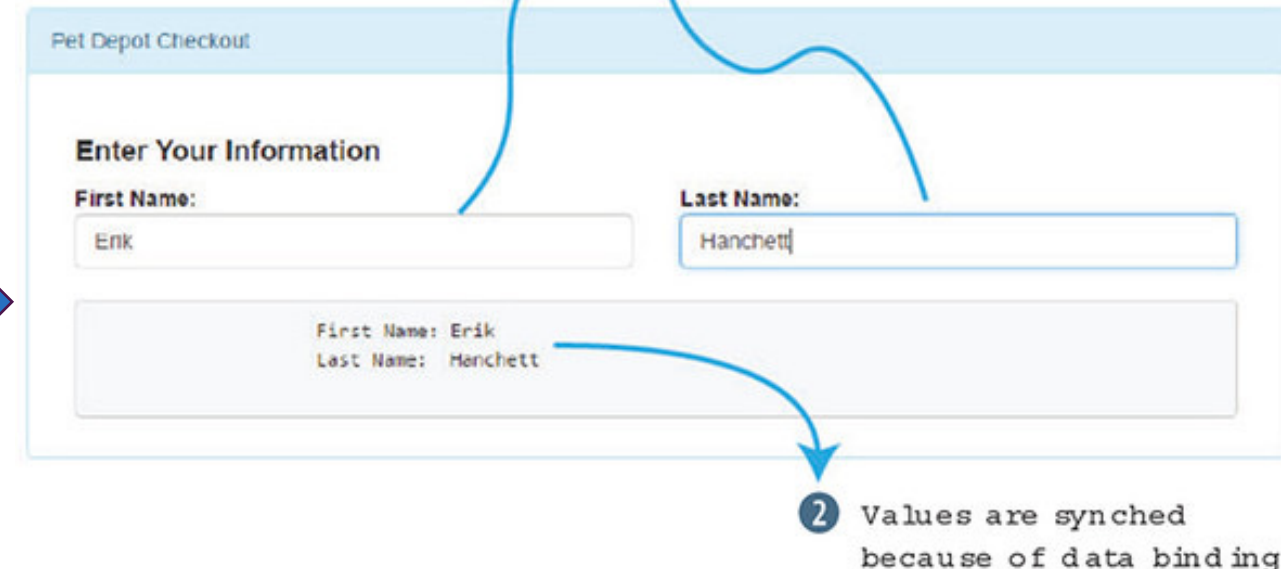
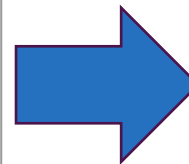


```
<h2>Checkout</h2>
<p>
  <strong>First Name:</strong>
  <!-- This input field is bound to 'firstName' in the 'order' object -->
  <input v-model="order.firstName"/>
</p>
<p>
  <strong>Last Name:</strong>
  <!-- This input field is bound to 'lastName' in the 'order' object -->
  <input v-model="order.lastName"/>
</p>
<h2>Order Information</h2>
<p>First Name: {{order.firstName}}</p>
<p>Last Name: {{order.lastName}}</p>
```

- Storing the name in data:



```
data: {
  sitename: 'Vue.js Pet Depot',
  showProduct: true,
  order: {
    firstName: '',
    lastName: ''
  },
  ...
}
```



Adding other Fields

- `v-model` applied also to other fields for having 2-Way Data Binding

```
<p>
  <strong>Address:</strong> <input v-model="order.address"/>
</p>
<p>
  <strong>City:</strong> <input v-model="order.city"/>
</p>
<p>
  <strong>State:</strong>
  <select v-model="order.state">
    <option disabled value="">State</option>
    <option>AL</option>
    <option>AR</option>
    <option>CA</option>
    <option>NV</option>
  </select>
</p>
<p>
  <strong>Zip/Postal Code:</strong> <input v-model="order.zip"/>
</p>
```

Vue.js Pet Depot

0 🛒 Checkout

Checkout

First Name:

Last Name:

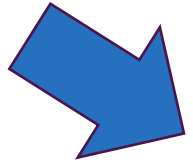
Address:

City:

State:

Zip / Postal Code:

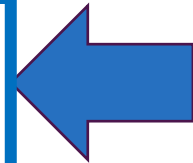
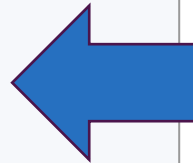
Displaying all the Fields



```
<h2>Order Information</h2>
<p>First Name: {{order.firstName}}</p>
<p>Last Name: {{order.lastName}}</p>
<p>Address: {{order.address}}</p>
<p>City: {{order.city}}</p>
<p>Zip: {{order.zip}}</p>
<p>State: {{order.state}}</p>
```

```
data: {
  sitename: "Vue.js Pet Depot",
  showProduct: true,
  order: {
    firstName: '',
    lastName: '',
    address: '',
    city: '',
    zip: '',
    state: ''
  },
},
```

- Note that we have not add the new fields to the data yet
- However, they can already be used, such as Zip: {{order.zip}}
- Because **Vue** can **implicitly add new properties**
- However, it is a **good practice to do this explicitly**



Vue.js Pet Depot

0 Checkout

Checkout

First Name:

Last Name:

Address:

City:

State:

Zip / Postal Code:

Order Information

First Name: Luca

Last Name: Piras

Address: Middlesex University

City: London

Zip: NW4 4BT

State: AR

Vue.js (2-Way) Data Binding (2/3)

`v-model` applied to Checkboxes and
Radio Buttons

Binding Checkboxes and Radio Buttons


- **Checkbox:** allow customer to **ship as a gift**
- **Radio button:** ship to 'home' or 'business' address; we must set the `v-model` in both check boxes to the same value

```
<p><input type="checkbox" id="gift" value="true" v-model="order.gift">
<label for="gift">Ship As Gift?</label></p>


<p><input type="radio" id="home" value="Home" v-model="order.method">
<label for="home">Home</label>
<input type="radio" id="business" value="Business" v-model="order.method">
<label for="business">Business</label></p>
```

Setting the Default Value

- You can set the default value for the input field using the property in `data`
- Below we set the default address to 'Home' and default gift option as 'false'



```
data: {
  sitename: "Vue.js Pet Depot",
  showProduct: true,
  order: {
    ...
    method: 'Home',
    gift: false
  },
},
```



☐ Ship As Gift?
☒ Home ☐ Business

Order Information

Gift? false

Method: Home

The “Place Order” Button

Add a 'Place order' button with `v-on`:

- `<button v-on:click="submitForm">Place Order</button>`
- Add the `submitForm` function to the Vue `methods`
 - `submitForm() {alert('Order submitted!')}`
- For now, this just displays an **alert**.
 - **Later we will add more functions such as user input validation.**

Vue.js

127.0.0.1:5500 says

Order submitted!

0 🛒 Check Out

OK

Checkout

First Name:

Last Name:

Address:

City:

State: State ▾

Zip / Postal Code:

☐ Ship As Gift?

☒ Home ☐ Business

Order Information

First Name:

Last Name:

Address:

City:

Zip:

State:

Gift? false

Method: Home

Place Order

Vue.js (2-Way) Data Binding (3/3)

```
v-bind (v-bind:true-value ,  
v-bind:false-value ,  
v-bind:value)
```

applied to

Checkboxes **and** Select Options

2 Problems

☐ Ship As Gift?



Gift? false


Method: Home

Place Order

[Problem 1: Checkbox Value] For the 'gift' box, we do not want the customers to see `true` or `false`

- but a message like **'Send as a gift'** or **'Not send as a gift'**

- **[Problem 2: Select Options Hard-Coded]** Currently the options in the select are **hard coded**



```
<strong>State:</strong>
<select v-model="order.state">
  <option disabled value="">State</option>
  <option>AL</option>
  <option>AR</option>
  <option>CA</option>
  <option>NV</option>
</select>
```


Binding Alternative Values

☐ Ship As Gift?



Gift? false

Method: Home

Place Order

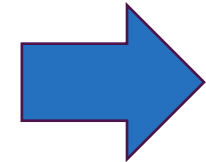
- For the **'gift' box**, we do not want the customers to see `true` or `false`
- but a message like **'Send as a gift'** or **'Not send as a gift'**

- This can be achieved by changing the **return value of the select options** with `v-bind:true-value` and `v-bind:false-value`.

```
<input type="checkbox" id="gift" value="true"
v-model="order.gift"
v-bind:true-value="order.sendGift"
v-bind:false-value="order.dontSendGift">
```

```
data: {
  order: {
    sendGift: 'Send as a gift',
    dontSendGift: 'Do not send as a gift'
  }
}
```

- The checkbox now binds with the intended text



☒ Ship As Gift?

☒ Home ☐ Business

Order Information

First Name:

Last Name:

Address:

City:

Zip:

State:

Gift? Send as a gift

Method: Home

Place Order

Binding Values to Select Options

State: CA

Order Information

First Name:

Last Name:

Address:

City:

Zip:

State: California

- Currently the options in the select are **hard coded**

```
<strong>State:</strong>
<select v-model="order.state">
  <option disabled value="">State</option>
  <option>AL</option>
  <option>AR</option>
  <option>CA</option>
  <option>NV</option>
</select>
```

- Now the select uses the “key”
as options (“AL”,
“AR”, etc,) and
returns the
“value”
 (“Alabama”,
“Arizona”, etc.)

- We need to read the list of states from data
and generate the options dynamically; we
add an array for the states in the data

```
<strong>State:</strong>
<select v-model="order.state" class="form-control">
  <option disabled value="">State</option>
  <option v-bind:value="states.AL">AL</option>
  <option v-bind:value="states.AR">AR</option>
  <option v-bind:value="states.CA">CA</option>
  <option v-bind:value="states.NV">NV</option>
</select>
```

```
data: {
  ...
  states: {
    AL: 'Alabama',
    AR: 'Arizona',
    CA: 'California',
    NV: 'Nevada'
  },
}
```

`v-for`

and Vue.js (2-Way) Data Binding

Using v-for

- With `v-for`, we can **iterate through the array** and **generate the options dynamically**, without having to list all the options in the select

```
<select v-model="order.state">
  <option disabled value="">State</option>
  <option v-for="(state, key) in states"
    v-bind:value="state">
    {{key}}
  </option>
</select>
```

```
data: {
  ...
  states: {
    AL: 'Alabama',
    AR: 'Arizona',
    CA: 'California',
    NV: 'Nevada'
  },
}
```

- In `v-for`, the `state` is a reference for element in the `states` array
- The `key` is an optional (but recommended) argument that specifies the **index of the current item**
- The `{{key}}` sets the key of `state` as the display for each select option

• The Generated HTML

```
<option value="Alabama">AL</option>
<option value="Arizona">AR</option>
<option value="California">CA</option>
<option value="Nevada">NV</option>
```

Using `v-for` without the “`key`”

The '`key`' in `v-for` is **optional**. The code below shows an **example without it**

```
<div id="app">
  <ol>
    <li v-for="state in states">{{state}}</li>
  </ol>
</div>

<script type="text/javascript">
  var webstore = new Vue({
    el: '#app',
    data: {
      states: ['Alabama', 'Alaska', 'Arizona', 'California', 'Nevada']
    }
  })
</script>
```

1. Alabama
2. Alaska
3. Arizona
4. California
5. Nevada

Vue.js Modifiers

Trimming the Input Values

- The `.trim` modifier can be used **to remove whitespaces before or after the text**
- We will use this for the **'firstName'** and **'lastName'** in our **form**. For instance:
 - `<input v-model.trim="order.firstName"/>`

First Name:

Order Information

First Name: Luca

Last Name:

Address:

City:

Zip:

State:

Method: Home Address

Gift: Send As A Gift

Not using `.trim` modifier:

```
<input v-model="order.firstName"/>
```

First Name:

Order Information

First Name: Luca

Last Name:

Address:

City:

Zip:

State:

Method: Home Address

Gift: Send As A Gift

Using `.trim` modifier:

```
<input v-model.trim="order.firstName"/>
```

The `.number` modifier

- The `.number` modifier changes the data type of `v-model` value to a number
 - the default type is 'string' even if you add `type='number'`
- This can be used to convert the type of ZIP input into number
 - `<input v-model.number="order.zip" type="number"/>`
- We can check this with the `typeof` operator in the console

Zip / Postal Code:

```
> webstore.order.zip
< ''
> typeof(webstore.order.zip)
< 'string'
> webstore.order.zip
< '12345'
> typeof(webstore.order.zip)
< 'string'
```

Not using `.number` modifier:

`<input v-model="order.zip" type="number"/>`

Zip / Postal Code:

```
> webstore.order.zip
< ''
> typeof(webstore.order.zip)
< 'string'
> webstore.order.zip
< 12345
> typeof(webstore.order.zip)
< 'number'
```

Using `.number` modifier:

`<input v-model.number="order.zip" type="number"/>`

Suggestions for Reading

Reading

Chapter 4 of the “**Vue.js in Action**” textbook

Questions?