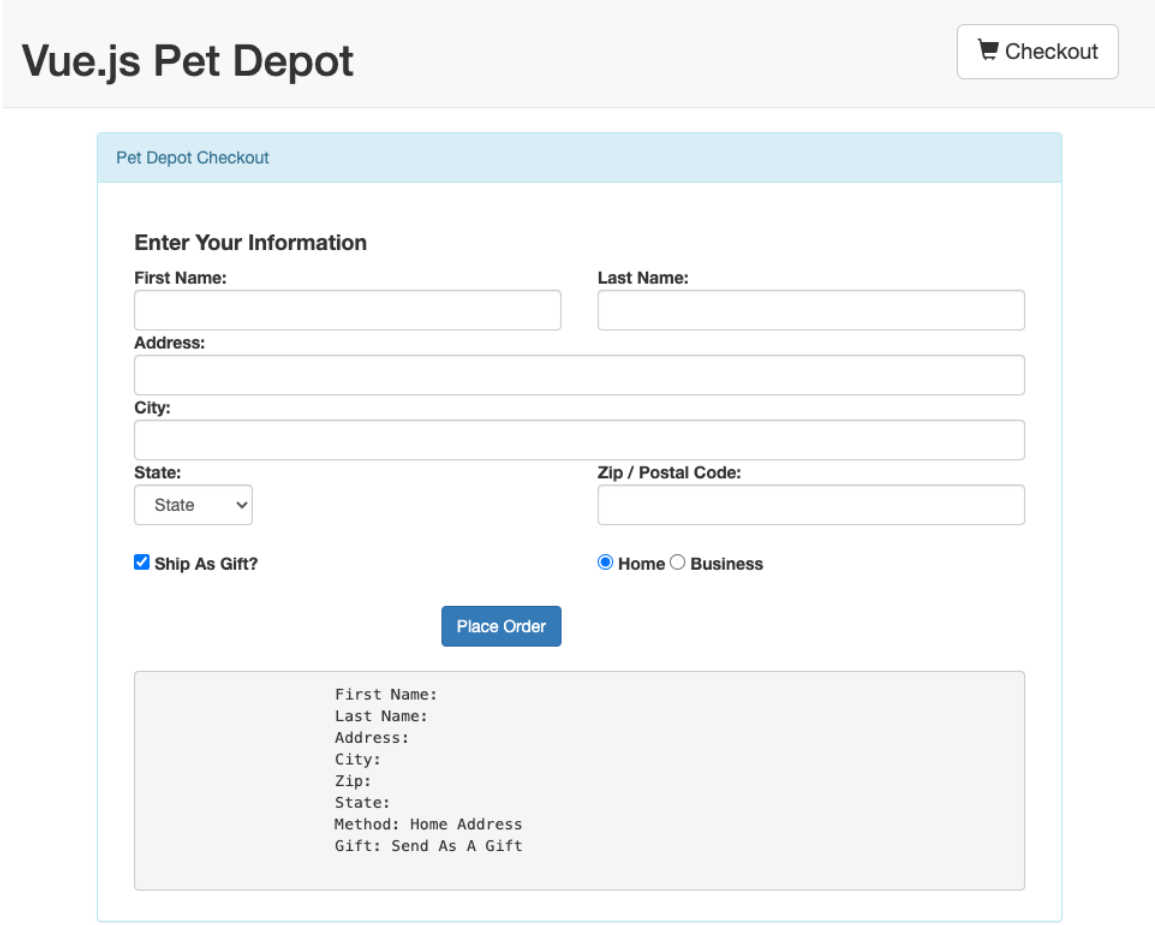
Forms and Inputs

Where we stopped two 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

This week

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



Outline

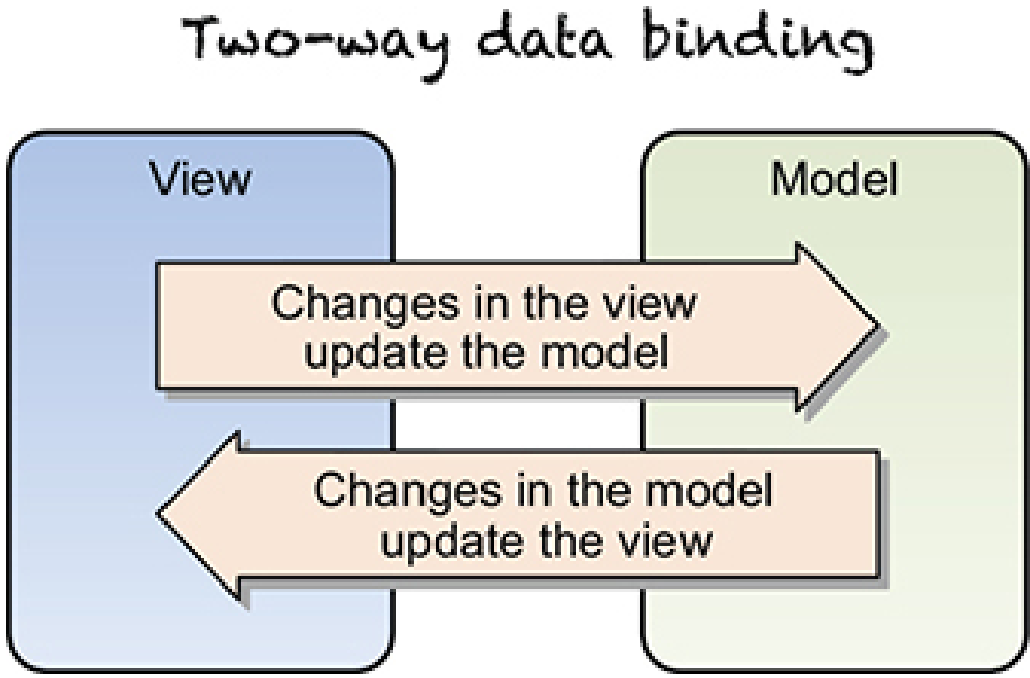
- Binding values with `v-model`
- Binding values with `v-bind` and `v-for`
- Modifiers

`v-model`

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

`v-model` binding is two way

- 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

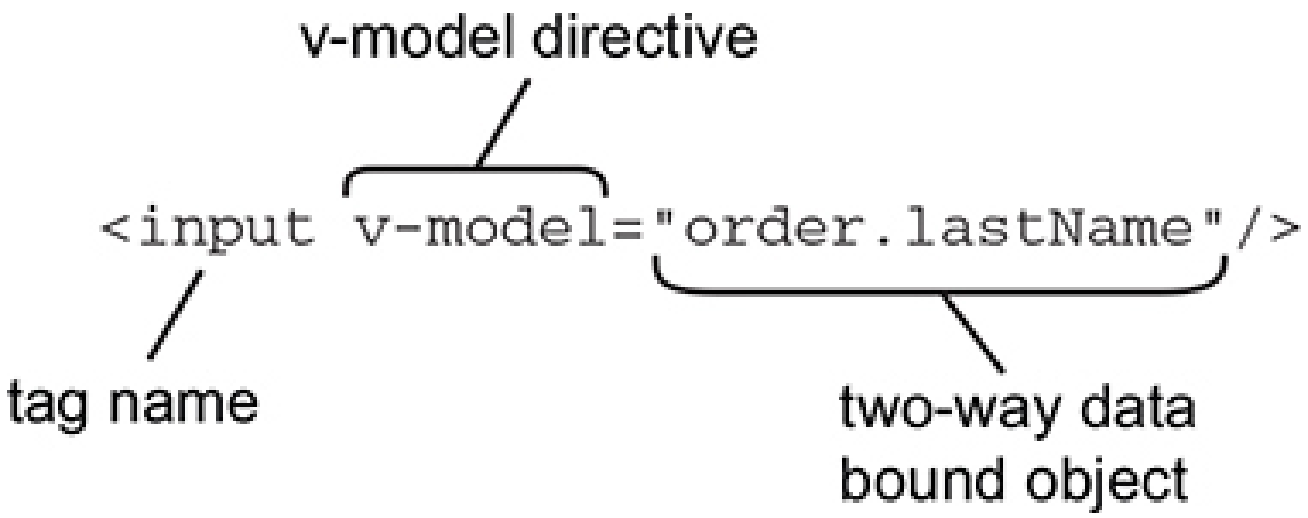


`v-model` vs. `v-bind`

- The `v-model` directive is used mainly for input and form binding.
  - We'll use the `v-model` here to bind text inputs on the checkout page.
- The `v-bind` directive is mostly used to bind HTML attributes.
  - We can use `v-bind` on an `src` attribute on an <img> tag, or
  - bind it to the class attribute on a tag.
- Both are useful, yet they're used in different situations.

An example

- The `input` is the form field that will be linked
- The `order.lastName' is the Vue` data` property that the form field will link to



Binding the name

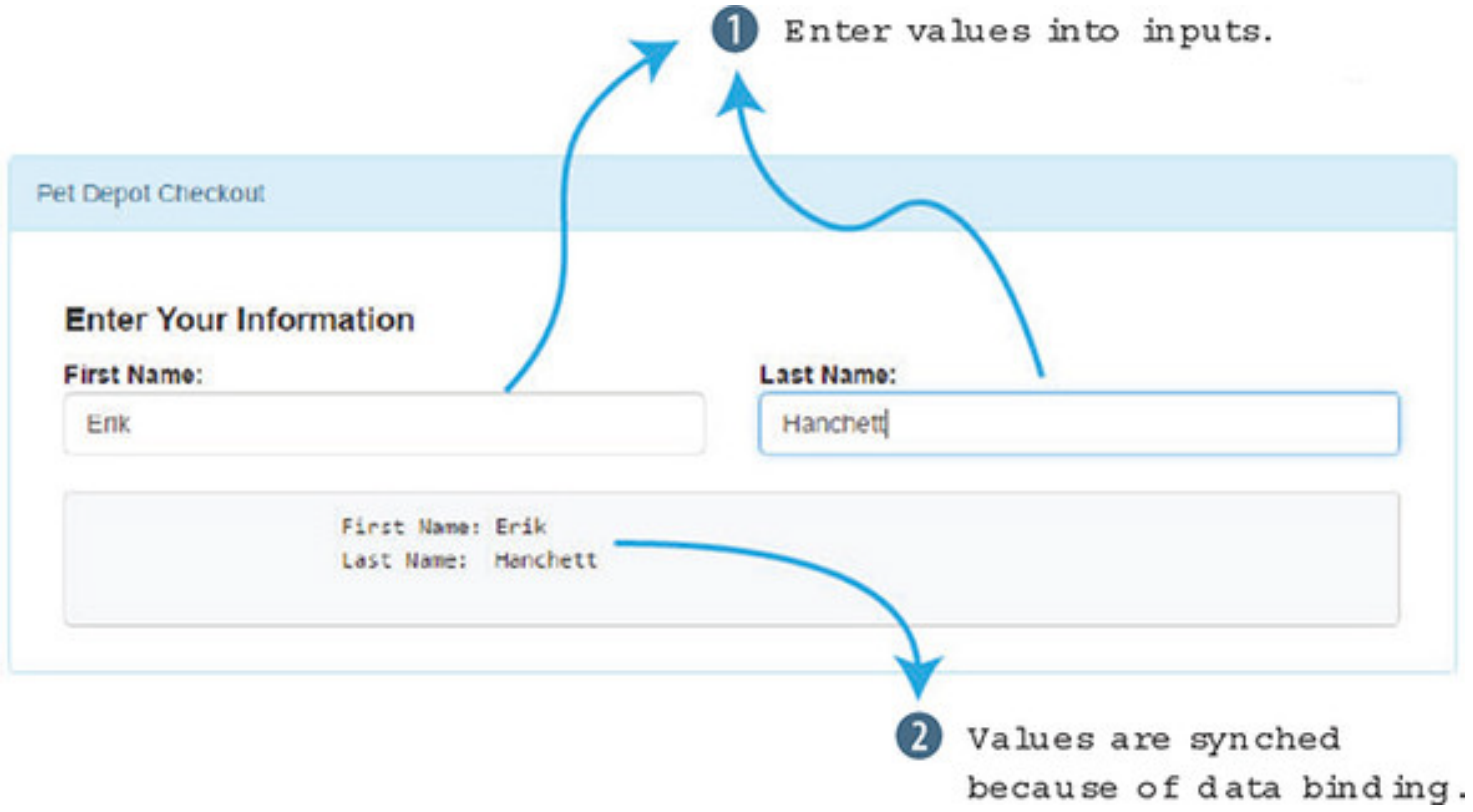
```
<h2>Checkout</h2>
<p>
  <strong>First Name:</strong>
  <!-- This input field is bound to 'firstName' in the 'order' object -->
  <input v-model="order.firstName"/>
</p>
<p>
  <strong>Last Name:</strong>
  <!-- This input field is bound to 'lastName' in the 'order' object -->
  <input v-model="order.lastName"/>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

Realtime update

As the user types in the name, it shows up in the pane below in realtime



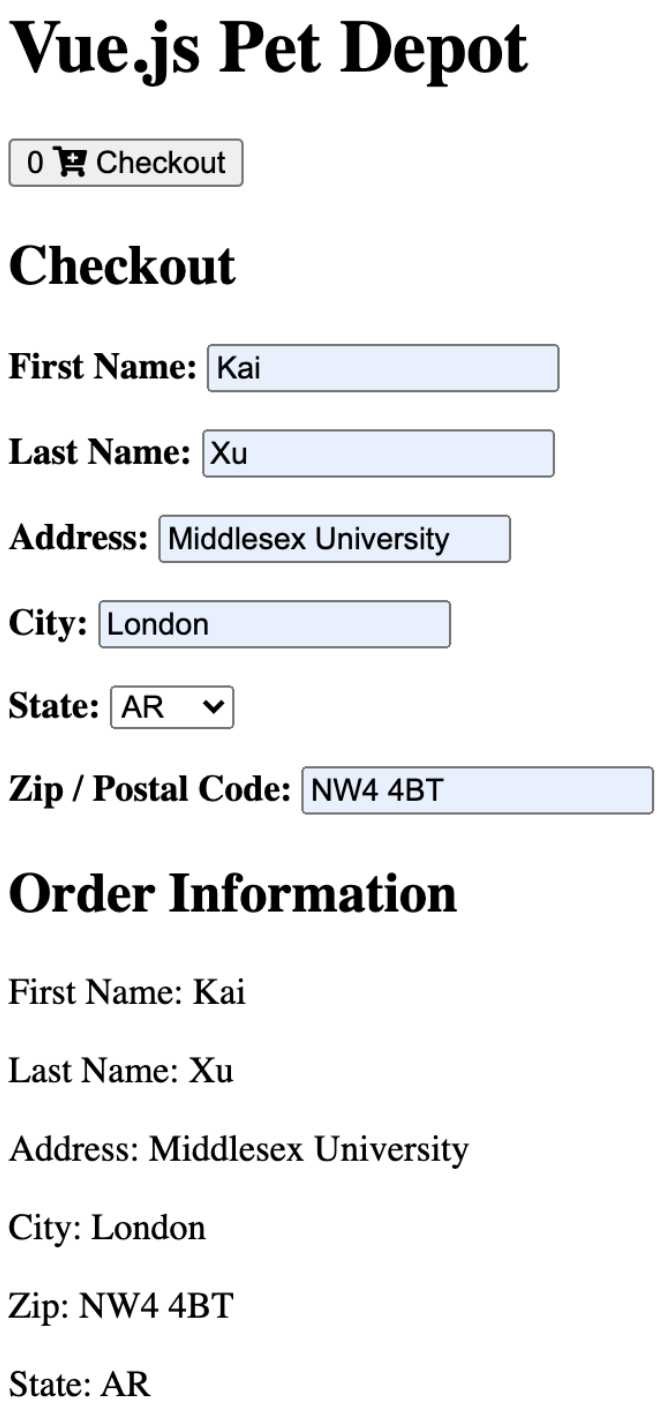
Adding other fields

```
<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>
```

Displaying all the fields

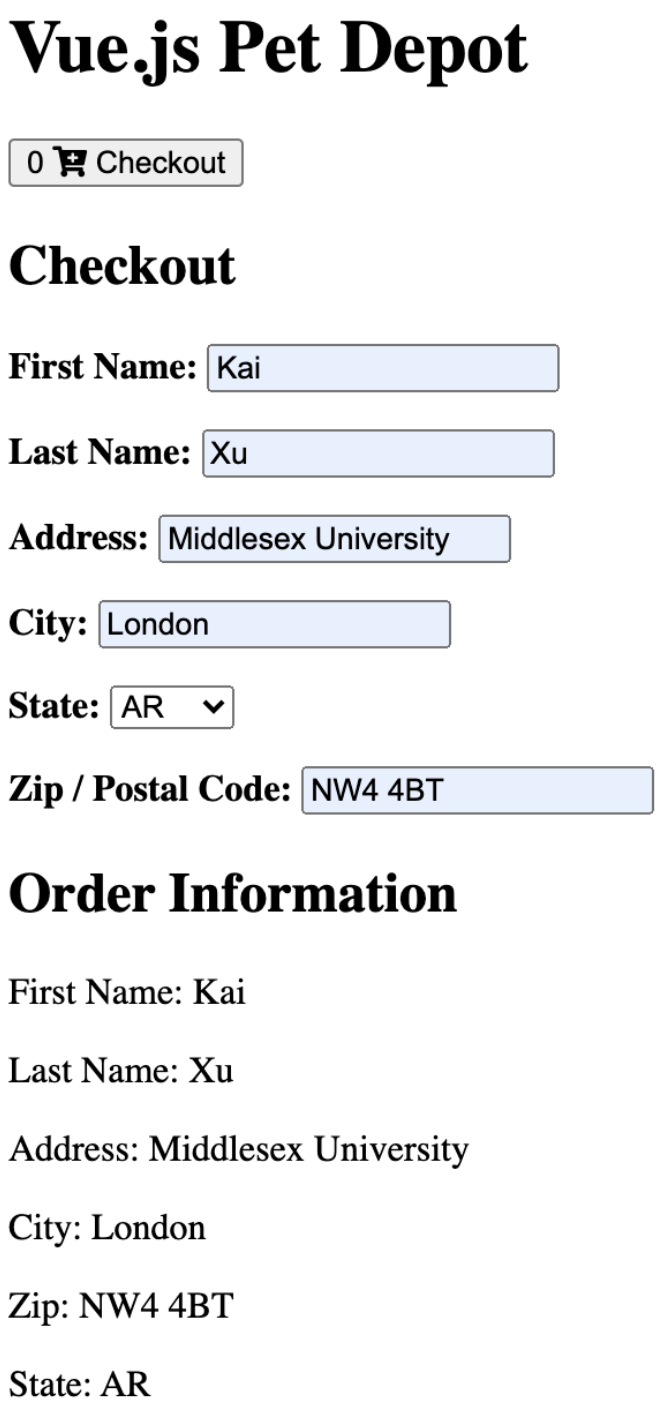
```
<h2>Order Information</h2>
<p>First Name: {{order.firstName}}</p>
<p>Last Name: {{order.lastName}}</p>
<p>Address: {{order.address}}</p>
<p>City: {{order.city}}</p>
<p>Zip: {{order.zip}}</p>
<p>State: {{order.state}}</p>
```

- Note that we haven't 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



Adding the fields to `order` object

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



Checkbox and radio button

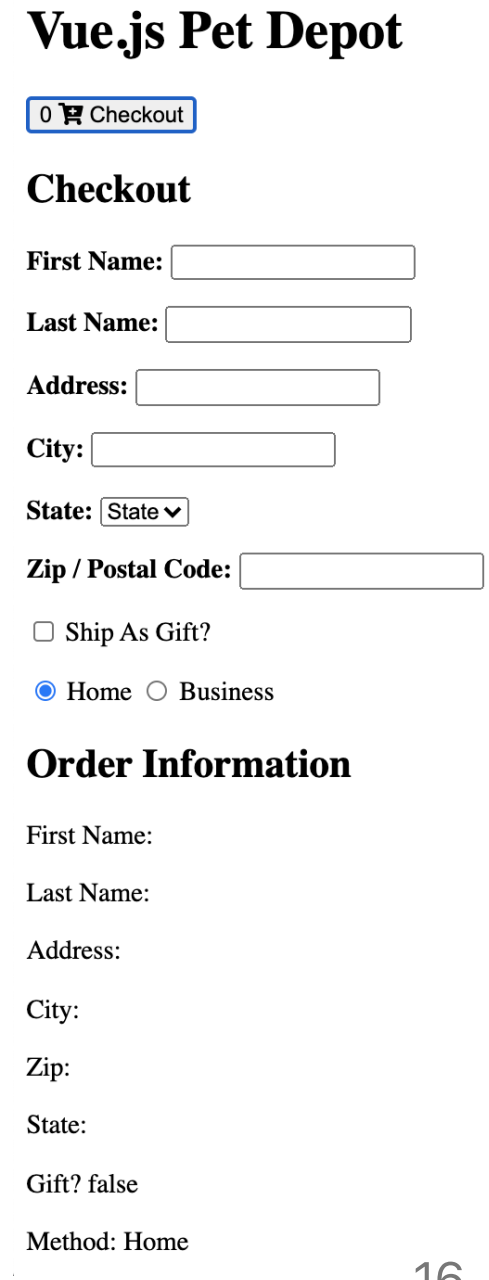
- Checkbox: allow customer to ship as a gift
- Radio button: ship to 'home' or 'business' address
    - We must set the `v-model` in both check boxes to the same value

```
<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

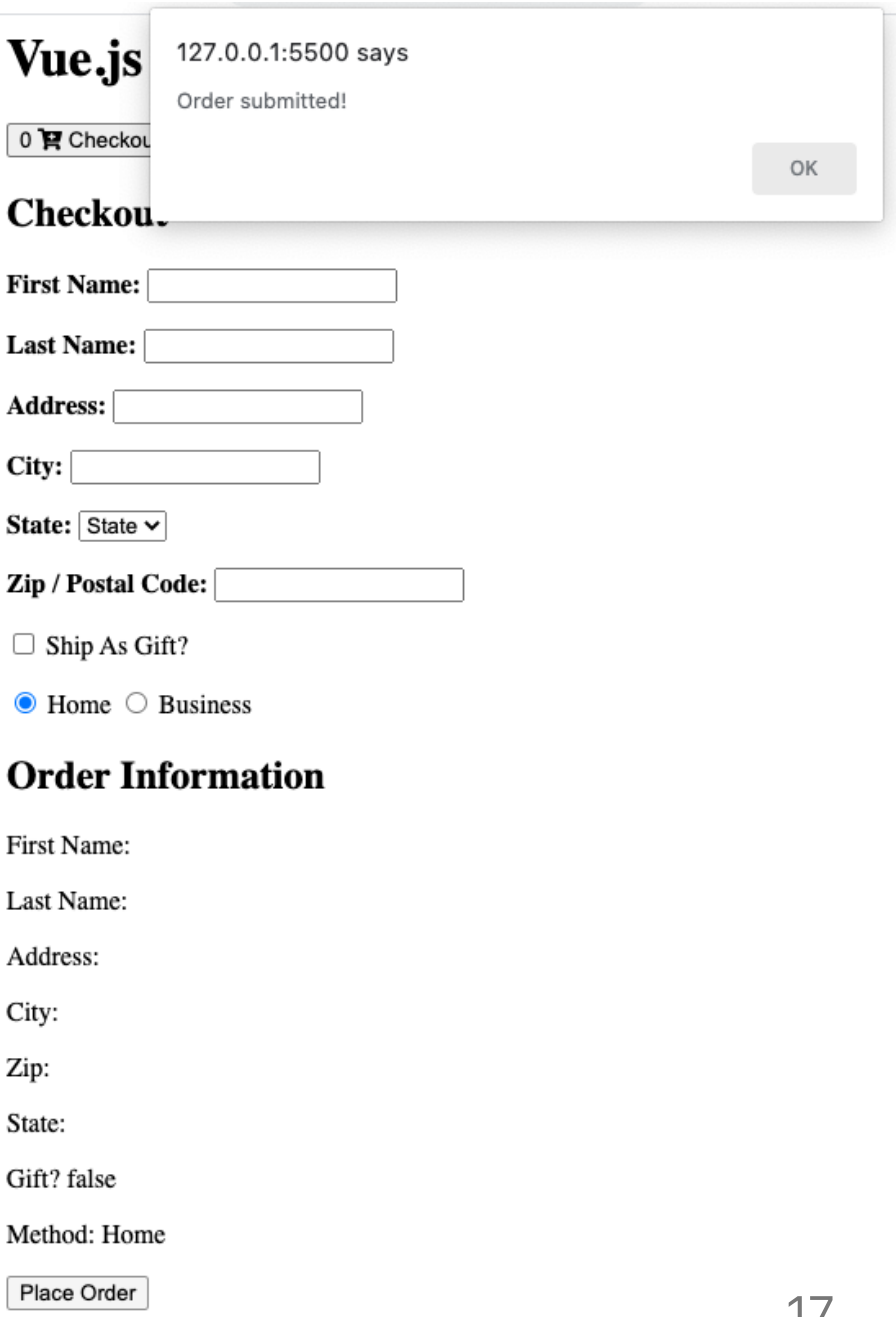
- You can set the default value for the input field using the property in `data`
- Below we set the default address to 'Home' and default gift option as 'false'

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



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 `method`
  - `submitForm() {alert('Order submitted!')}`
  - For now this just displays an alert. Later we will add more functions such as user input validation.



Outline

- Binding values with `v-model`
- **Binding values with `v-bind` and `v-for`**
- Modifiers

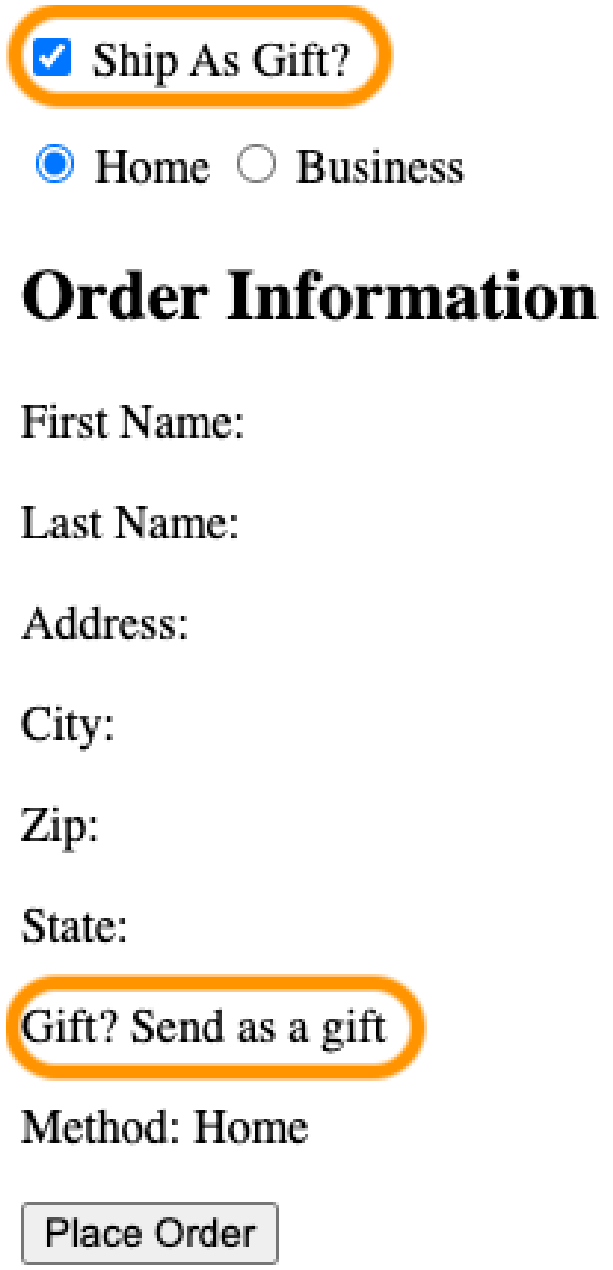
Changing the option value

- For the 'gift' box, we don't 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 returns the intended text.



Value binding for the dropdown menu options

- Currently the options in the dropdown menu is hard coded.
- It will be better if we can read the list of states from `data` and generate the options dynamically.

```
<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>
```

Change the options with `v-bind`

- Just as the last example, we can change the options with `v-bind`
- Instead of `true` or `false`, the values are 'send as a gift' and 'do not send as a gift'
- Here we use `v-bind:value` instead of `v-bind:true-value` or `false-value`

```
<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>
```

The `states` array

- The `states` is an array in `data`
- Now the dropdown menu use the 'key's as options: 'AL', 'AR', etc.
- And returns the 'value': 'Alabama', 'Arizona', etc.

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

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 dropdown menu

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

- In `v-for`, the `state` is an alias 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 dropdown option.

The generated HTML

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

- Now we can add all the states to the `states` array in `data`, and the dropdown selection options will be generated automatically.

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

Outline

- Binding values with `v-model`
- Binding values with `v-bind` and `v-for`
- **Modifiers**

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"/>`
- This can be checked with the `typeof` operator in the console

```
> 'before'
<· "before"
> webstore.order.zip
<· "1234"
> typeof(webstore.order.zip)
<· "string"
> 'after'
<· "after"
> webstore.order.zip
<· 1234
> typeof(webstore.order.zip)
<· "number"
>
```

Trimming the input values

- The `.trim` modifier can be used to remove white spaces before or after the text
- We will use this for the 'firstName' and 'lastName' in our form.
- `<input v-model.trim="order.firstName"/>`
- `<input v-model.trim="order.lastName"/>`
- Again we can check this in the browser console

```
> 'Before'
< "Before"
> webstore.order.firstName
< "    kai.  "
> webstore.order.lastName
< "   xu.  "
> 'after'
< "after"
> webstore.order.lastName
< "xu."
> webstore.order.firstName
< "kai."
>
```

Reading: Chapter 4 of 'Vue.js in Action' - Forms and Inputs