

Interaction

Outline

- The '*Add to cart*' button
- Check the inventory
- Show the 'checkout' page

Interaction

- We did this earlier in the ToDo example:
 - When click the 'Add' button,
 - A new item is added to the todo list

My Todo List

- task 1
- task 2
- task 3

- This can be done either plain JavaScript

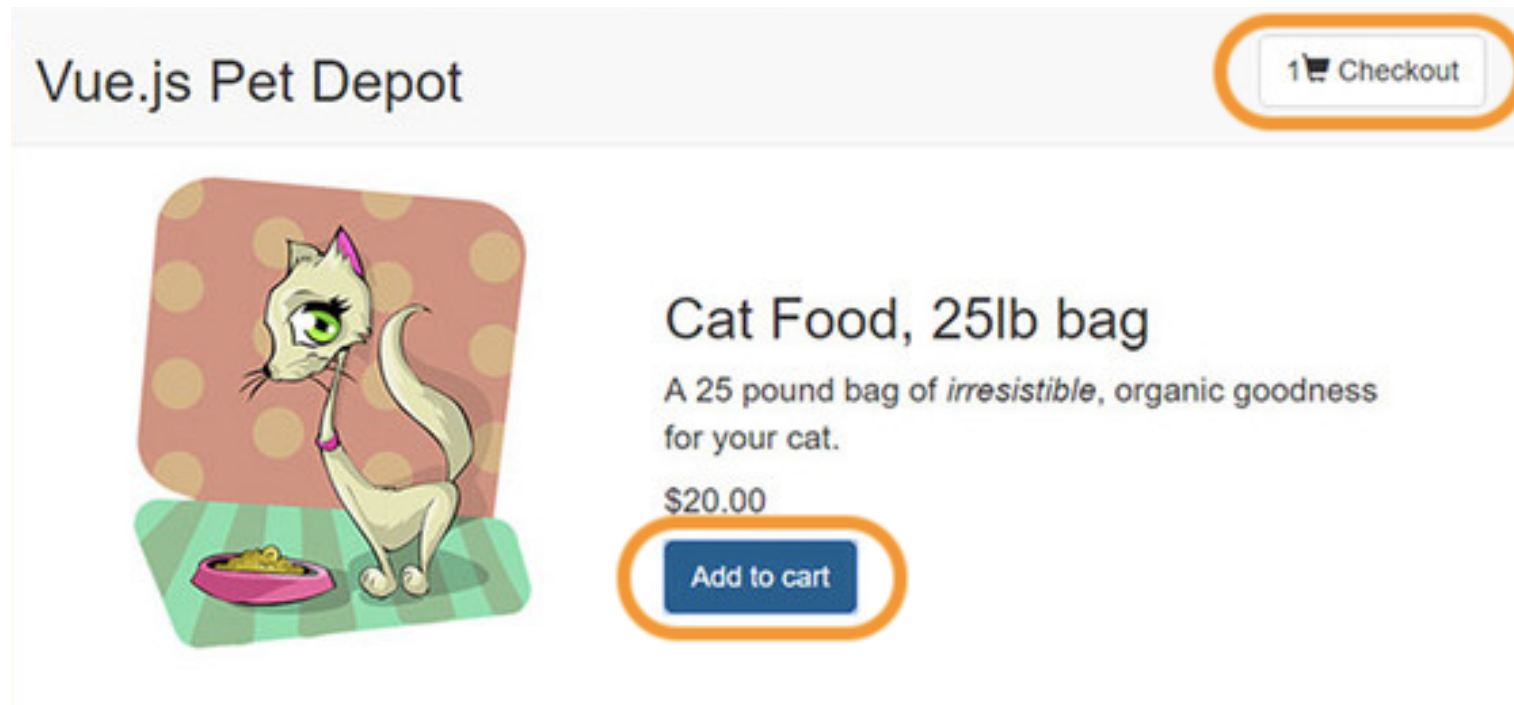
```
let addButton = document.getElementById('addButton');
let taskList = document.getElementById('taskList');
addButton.onclick = () => {
    let taskField = document.getElementById('newTask');
    let taskItem = document.createElement('li');
    taskItem.innerText = taskField.value;
    taskList.appendChild(taskItem);
    taskField.value = '';
}
```

Or use Vue.js

```
<form>
  <input v-model="newTask">
  <input type="button" value="add" v-on:click='addItem'>
</form>
```

```
let app = new Vue({
  el: '#app',
  data: { newTask: '' },
  methods: {
    addItem: () => {
      let newItem = document.createElement('li');
      newItem.innerText = app.newTask;
      document.getElementById('taskList').appendChild(newItem);
      app.newTask = '';
    }
  }
})
```

- Today we will try to add a 'Add to cart button' to the product page.
- And the 'Checkout' button.

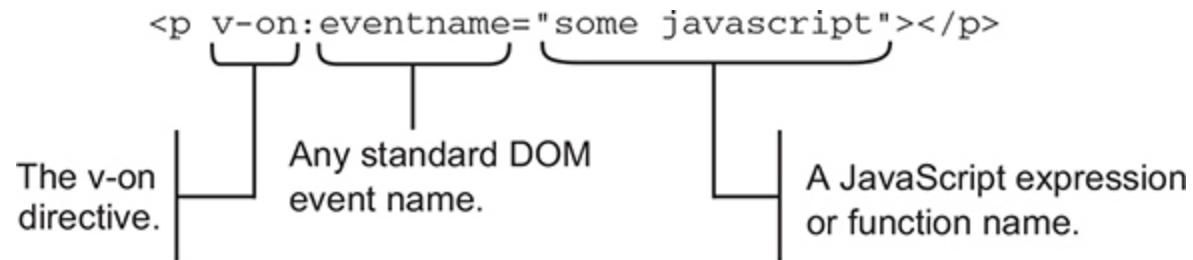


Product Information and Shopping Cart Array

```
data: {  
  sitename: "Vue.js Pet Depot",  
  product: { // product information similar to the last time  
    id: 1001,  
    title: "Cat Food, 25lb bag",  
    description: "A 25 pound bag of <em>irresistible</em>,  
      organic goodness for your cat.",  
    price: 2000,  
    image: "assets/images/product-fullsize.png",  
  },  
  cart: [] // array to store items in shopping cart  
},
```

Binding To Dom Events

- Event bindings use the `v-on` directive to bind a snippet of JavaScript, or a function, to a DOM element.



```
<input type="button" value="add" v-on:click='addItem'>
```

```
methods: {  
  addItem: () => {  
    ...  
  }  
}
```


v-on shorthand @

- Instead of using `v-on` , you can replace it with the `@` symbol

```
<input type="button" value="add" v-on:click='addItem'>
```

- is the same as

```
<input type="button" value="add" @click='addItem'>
```

Bind an Event to the Add to Cart Button

- First, we create the event function

```
methods: {  
  addToCart: function() {  
    this.cart.push( this.product.id );  
  }  
}
```

- Adding a product to the cart means pushing the product's `id` property from the product data onto the `cart` array
- Pushing the **product** onto the cart array would push a **reference** to the product object defined in our data, not a copy.

The 'Add to cart' button

```
<button v-on:click="addToCart">  
  Add to cart  
</button>
```

- Now put everything together

```
<div id="app">
  <header>
    <h1 v-text="sitename"></h1>
  </header>
  <main>
    <figure>
      
    </figure>
    <h2 v-text="product.title"></h2>
    <p v-html="product.description"></p>
    <p>Price: {{product.price}}</p>
    <button v-on:click="addToCart">
      Add to cart
    </button>
  </main>
</div>
```

```
var webstore = new Vue({
  el: '#app',
  data: {
    sitename: 'Vue.js Pet Depot',
    product: {
      id: 1001,
      title: "Cat Food, 25lb bag",
      description: "A 25 pound bag of <em>irresistible</em>,"
        + "organic goodness for your cat.",
      price: 2000,
      image: "images/product-fullsize.png"
    },
    cart: []
  },
  methods: {
    addToCart: function () {
      this.cart.push(this.product.id);
    }
  }
});
```

- You need to click on the `<Root>` to update the cart array

The screenshot shows a web browser window with the address bar displaying `127.0.0.1:5500/week04-interaction-git/pet-depot.html`. The page title is "Vue.js Pet Depot". The main content area features a cartoon illustration of a white cat with green eyes sitting on a green and white striped mat, eating from a pink bowl. Below the illustration, the text "Cat Food, 25lb bag" is displayed, followed by a description: "A 25 pound bag of *irresistible*,organic goodness for your cat." and the price "Price: 2000". An "Add to cart" button is located at the bottom left of the product information.

The right side of the image shows the Vue.js developer tools interface. The "Vue" tab is selected, showing the component tree with the root component `<Root> = $vm0`. Below this, the "data" section is expanded, showing the following structure:

```
data: {
  cart: Array[3],
  product: Object
}
```

The `product` object contains the following properties:

- `description`: "A 25 pound bag of *irresistible*,organic goodness for your cat."
- `id`: 1001
- `image`: "images/product-fullsize.png"
- `price`: 2000
- `title`: "Cat Food, 25lb bag"
- `siteName`: "Vue.js Pet Depot"

Outline

- The '*Add to cart*' button
- **Check the inventory**
- Show the 'checkout' page

Computed Property

- The checkout button will show the number of items in the cart;
- We will use *computed property* to achieve this;
- Computed properties can be bound to the DOM like any other property defined in the `data` ;
- Its value is usually derived from the current state of an application.

Item Count

```
computed: { // the Computed Property object
  cartItemCount: function() { // the property name
    // its value is calculated when it is called
    return this.cart.length || '';
  }
},
```

- The `cartItemCount` should not be in the `data` object
 - Because its value changes as the result of user interaction.

Checkout Button

```
<head>
  // We are using Font Awesome to create the cart icon
  <link rel="stylesheet"
    href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css">
</head>
<body>
  <header>
    <h1>{{ sitename }}</h1>
    <button>
      <!-- 'cartItemCount' is used the same way as a data property. -->
      {{ cartItemCount }}
      <!-- add the cart icon -->
      <span class="fas fa-cart-plus"></span> Checkout
    </button>
  </header>
  ...
</body>
```

Vue.js Pet Depot

0  Checkout



Cat Food, 25lb bag

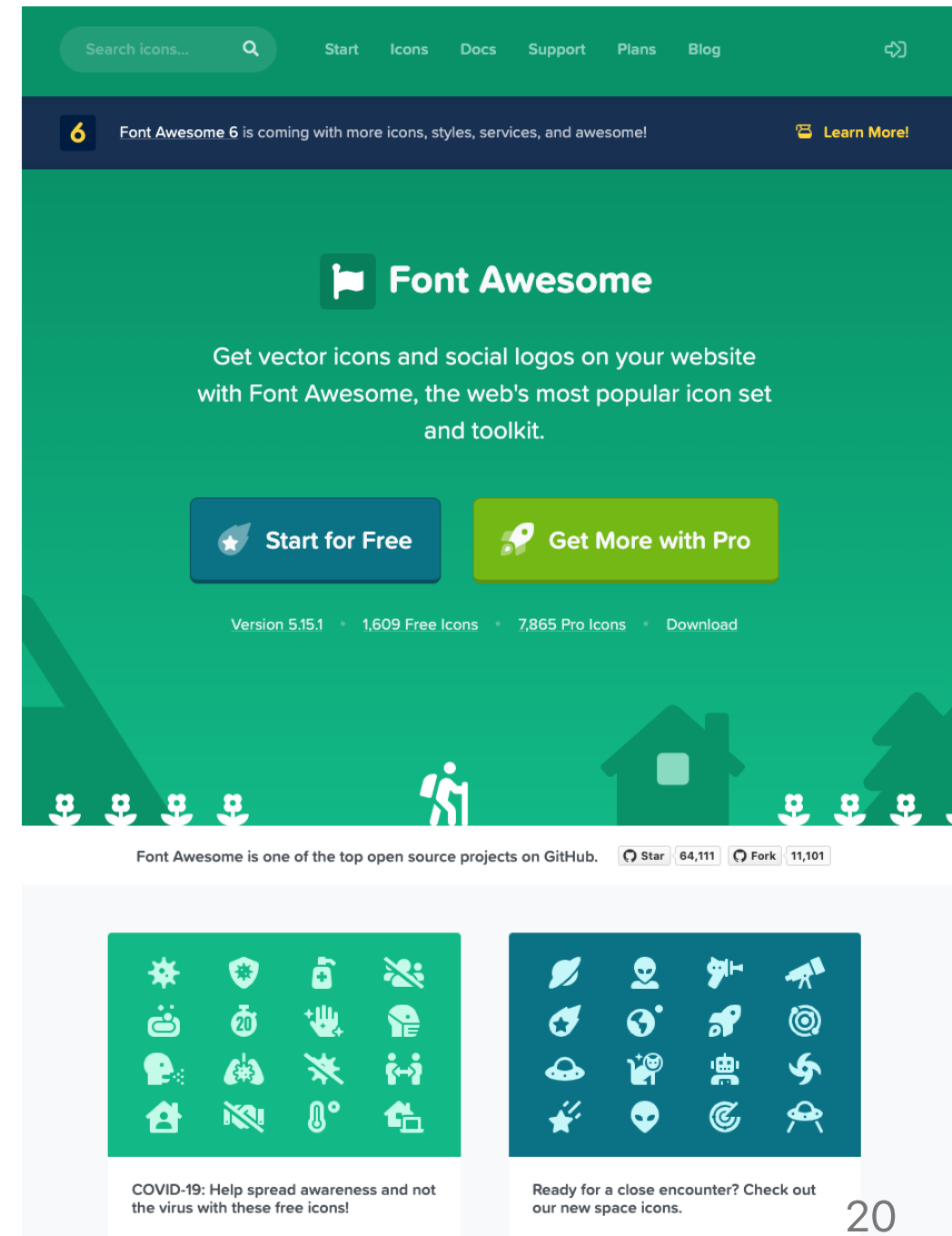
A 25 pound bag of *irresistible*, organic goodness for your cat.

Price: 2000

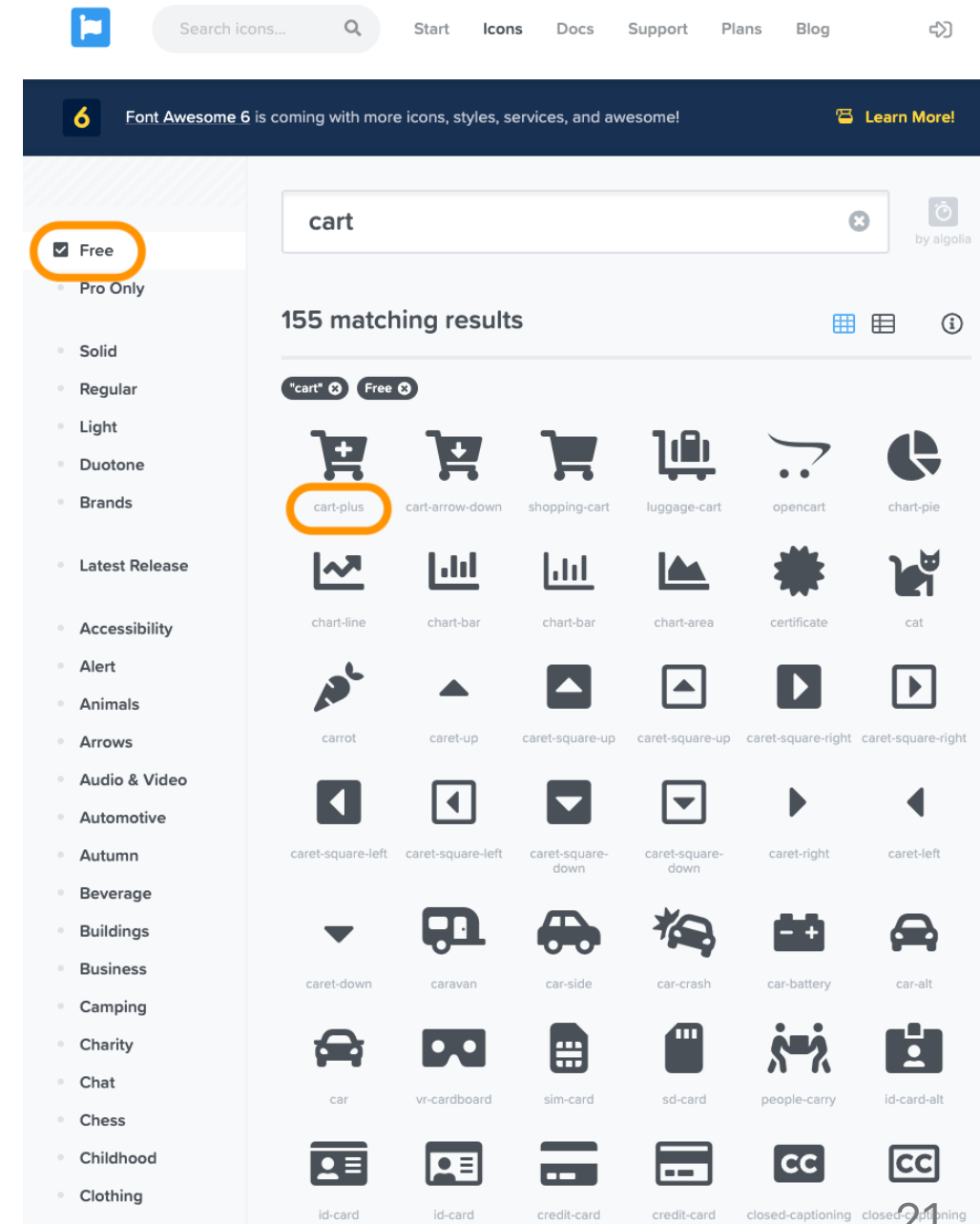
Add to cart

Font Awesome

- It allows you to add icon to your page as text instead of image.
- You need to first load it as an external css file
- ```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/5.15.1/css/all.min.css">
```



- Then search for the icon you want, such as 'cart' at <https://fontawesome.com/icons>
  - Make sure to choose the 'free' icons
- Note the code for your icon,
  - In this case 'cart-plus'
- Add the icon to your page like
  - `<span class="fas fa-cart-plus">`  
`</span>`
  - The class `fas` means it is a Font Awesome icon
  - The other class selects the icon: `fa-`  
`icon-code`



# Inventory

- We need to make sure there is still stock available if a customer wants more
- We will use a new property `availableInventory` to record the stock level

```
data: {
 sitename: "Vue.js Pet Depot",
 product: {
 id: ... ,
 title: ... ,
 description: ... ,
 price: ... ,
 image: ... ,
 availableInventory: 5
 }
 cart: []
}
```

# Check Stock Level

- We don't want to change the `availableInventory`
  - because that should only happen after user finished checkout.
- But we do want to restrict the amount of product a customer can add to their cart
  - It should not be more than the `availableInventory`.

```
computed: {
 cartItemCount: function() {
 ...
 },
 canAddToCart: function() {
 return this.product.availableInventory > this.cartItemCount;
 }
}
```

- We use a computed property `canAddToCart` to check this
- Note that we use another computed property `cartItemCount` just as a 'data' property



## Hide the Button with `v-show`

- We will stop a customer from adding more product if the number in the cart is more than the stock level
- We do this by hiding the 'Add to cart' button
- We can use the `v-show` for this.
  - It only shows a HTML element if the condition is true

```
<button
 v-on:click="addToCart"
 v-show="canAddToCart">
 <!-- only show the button when 'canAddToCart' is true -->
 Add to cart
</button>
```

## No more 'Add to cart' button

### Vue.js Pet Depot

5  Checkout



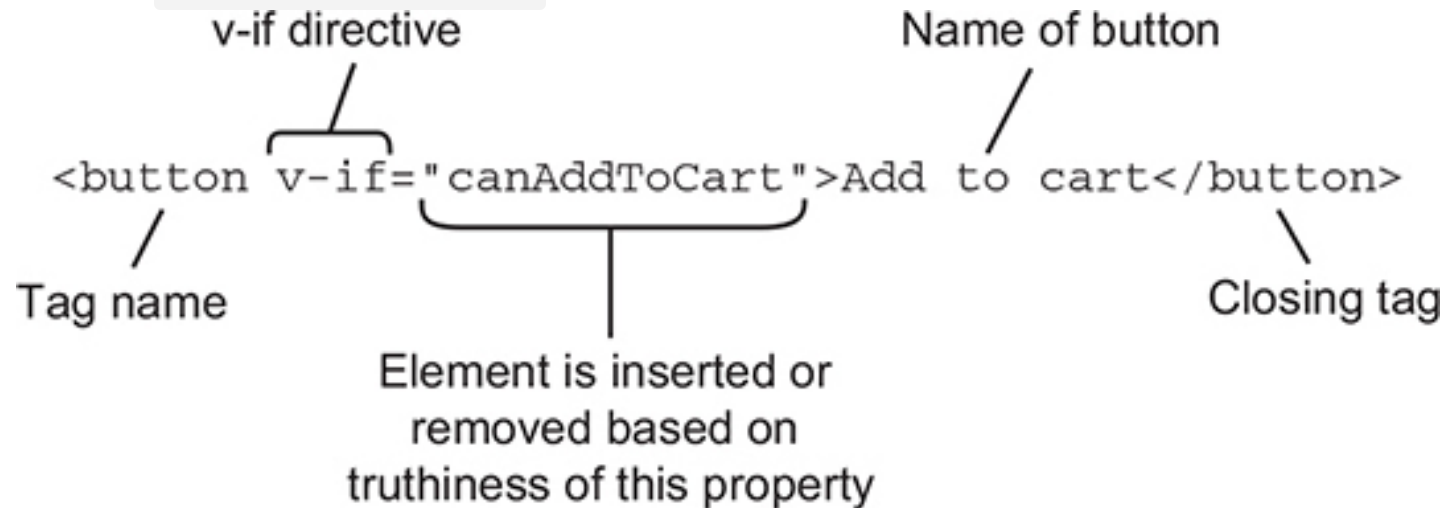
### Cat Food, 25lb bag

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

Price: 2000

# A Disabled Button

- It is rare for a e-commerce site to actually make a button disappear
  - Which is not really a good user experience
- Instead, more likely a button is **disabled** to achieve a similar effect
- This can be done with `v-if` and `v-else`
- If the `canAddToCart` is true the button appears, if not, the button doesn't appear.




- The idea: we will have two similar buttons: one is enabled, and the other is disabled
- We show the enabled button when user can add more, and the other button when user can't
- We use `v-if` and `v-else` to select which button is show

```
<!-- This button will be displayed when 'canAddToCart' is True -->
<button v-on:click="addToCart"
 v-if="canAddToCart">
 Add to cart
</button>

<!-- This button will be displayed otherwise -->
<button disabled='disabled' v-else>
 Add to cart
</button>
```

Now the button is disabled,  
instead of disappears.

## Vue.js Pet Depot

5  Checkout



### Cat Food, 25lb bag

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

Price: 2000

Available stock: 5

Add to cart

# Outline

- The '*Add to cart*' button
- Check the inventory
- **Show the 'checkout' page**

# Toggling the Checkout Page

- So far there is no 'checkout' page yet
- We will add a checkout page that becomes visible when the 'checkout' button is clicked.
  - A first click of the 'checkout' button will show the checkout page
  - A second click of the 'checkout' button will hide the check out page, i.e., show the product page again

## Vue.js Pet Depot

0  Checkout



### Cat Food, 25lb bag

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

Price: 2000

Add to cart

# New Property and Method

```
data: {
 showProduct: true,
 ...
},

methods: {
 ...
 showCheckout() {
 this.showProduct = this.showProduct ? false : true;
 },
}
```



```
this.showProduct = this.showProduct ? false : true;
```

- The `showCheckout` method toggles the `showProduct` property by using something called a **ternary operation** in JavaScript.
- The ternary operator is a shortcut for the `if` and `else` statement
- It takes three parameters.
  - i. The first parameter is the condition, in this case, `this.showProduct` .
  - ii. If it resolves to `true` , it then returns the first expression, `false` .
  - iii. Otherwise it returns the last expression, `true` .
- This is equivalent to:

```
if (this.showProduct) this.showProduct = false;
else this.showProduct = true;
```

```
methods: {
 ...
 showCheckout() {
 this.showProduct = this.showProduct ? false : true;
 },
}
```

- The `showCheckout` method definition was missing the `function()` declaration.
- This is another shorter syntax for method definitions introduced in ES6, also known as ES2015.
- This is equivalent to:

```
methods: {
 ...
 showCheckout: function() {
 this.showProduct = this.showProduct ? false : true;
 },
}
```

## Change the value of `showProduct`

- Now we will use the 'checkout' button to change the value of `showProduct`

```
<button v-on:click='showCheckout'>
 {{cartItemCount}}

 Checkout
</button>
```

# Display the Checkout Page

- We will use `v-if` to toggle the checkout page based on the value of `showProduct`

```
<main>
 <div v-if='showProduct'>
 <!-- the code for the product page -->
 ...
 </div>
 <div v-else>
 <!-- the code for the checkout page -->
 ...
 </div>
</main>
```

- Clicking the 'Checkout' button will toggle between the product page and the empty checkout page.

## **Reading: Chapter 3: Adding Interactivity**