Vue 3 is officially released. Learn it now with a 30% discount

Claim Offer

Vue mastery

Intro to Vue 3

Lessons

1. Intro to Vue 3

2:22

☐

2. Creating the Vue App

6:56

☐

3. Attribute Binding

3:53

☐

4. Conditional Rendering

5:11

Event Handling

In this lesson, we're going to look at the concept of Event Handling. If you're coding along with the repo, you can checkout the `L6-start` branch or the starting code on Codepen.

In the starting code, you'll see that we now have an Add to Cart `button`, along with a cart `div`, which includes an expression to print out the value of our new `cart` data.

index.html

```
<div class="cart">Cart({{ cart }})</div>
...
<button class="button">Add to Cart</button>
```

main.js

```
data() {
  return {
    cart: 0,
    ...
  }
}
```

Our Goal

We want to be able to click the `button` and increment the value of `cart`.

Listening for Events

In order to know when the button is clicked, we need to be listening for events on that element, specifically *click* events. We can achieve this by using another Vue directive: `v-on`.

📄**index.html**

```
<button class="button" v-on:click="logic to run">Add to
Cart</button>
```

Here, we are telling `v-on` what type of event to listen for: a `click`. Inside the quotes, we place the logic (or method name) we want to run when that event happens.

If we write `v-on:click="cart += 1"`, we'll increment the value of cart by `1`, when a click event happens.

Triggering a method

Because the logic `cart += 1` is very simple, we could keep it in-line on the `button` element, like we have it. But often, we need to trigger more complex logic. In those situations, we can add a method name to fire when the event happens. So let's do that now.

📄**index.html**

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

Now, when the button is clicked, the `addToCart` method will be run. Let's add that method to our Vue app's options object, like so:

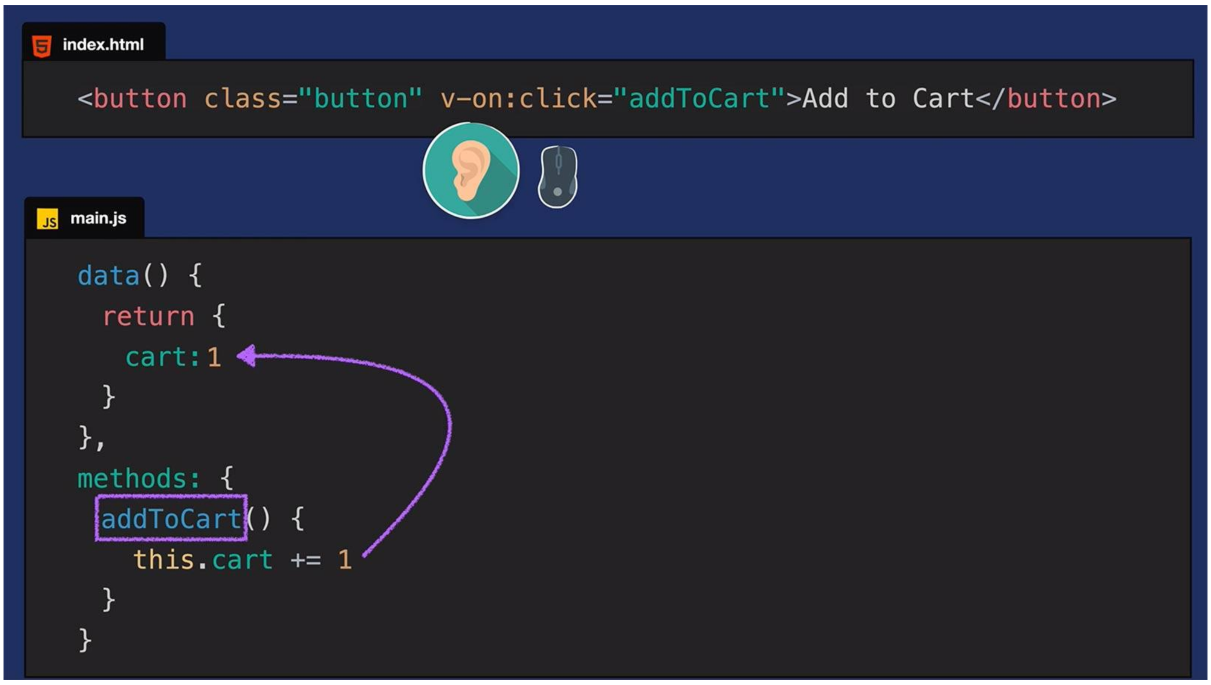
```
const app = Vue.createApp({
  data() {
    return {
      cart: 0,
      ...
    }
  },
  methods: {
    addToCart() {
      this.cart += 1
    }
  }
})
```

Notice how we added the `methods` option, and inside of that we added the new `addToCart` method, which contains the same logic we just had in-line. The difference here is now we're saying `this.cart` to refer to *this* `cart` in *this* Vue instance's data.

In the browser, we should now be able to click the **Add To Cart** `button` and see the value of cart climb up by 1.

Understanding v-on

Let's take a deeper look into how this event handling is working.



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

```
data() {
  return {
    cart:1
  }
},
methods: {
  addToCart() {
    this.cart += 1
  }
}
```

By adding `v-on` to an element, we're essentially giving it an ear that can listen for events. In this case, we've specified that we're listening for click events. When a click happens, the `addToCart` method runs, which as we just saw, takes the value of cart and increments it by one.

A shorthand for v-on

As you can imagine, listening for events on your elements is super common. Just like how `v-bind` had a shorthand (`:`), `v-on` has a shorthand: `@`

So our code could be simplified to:

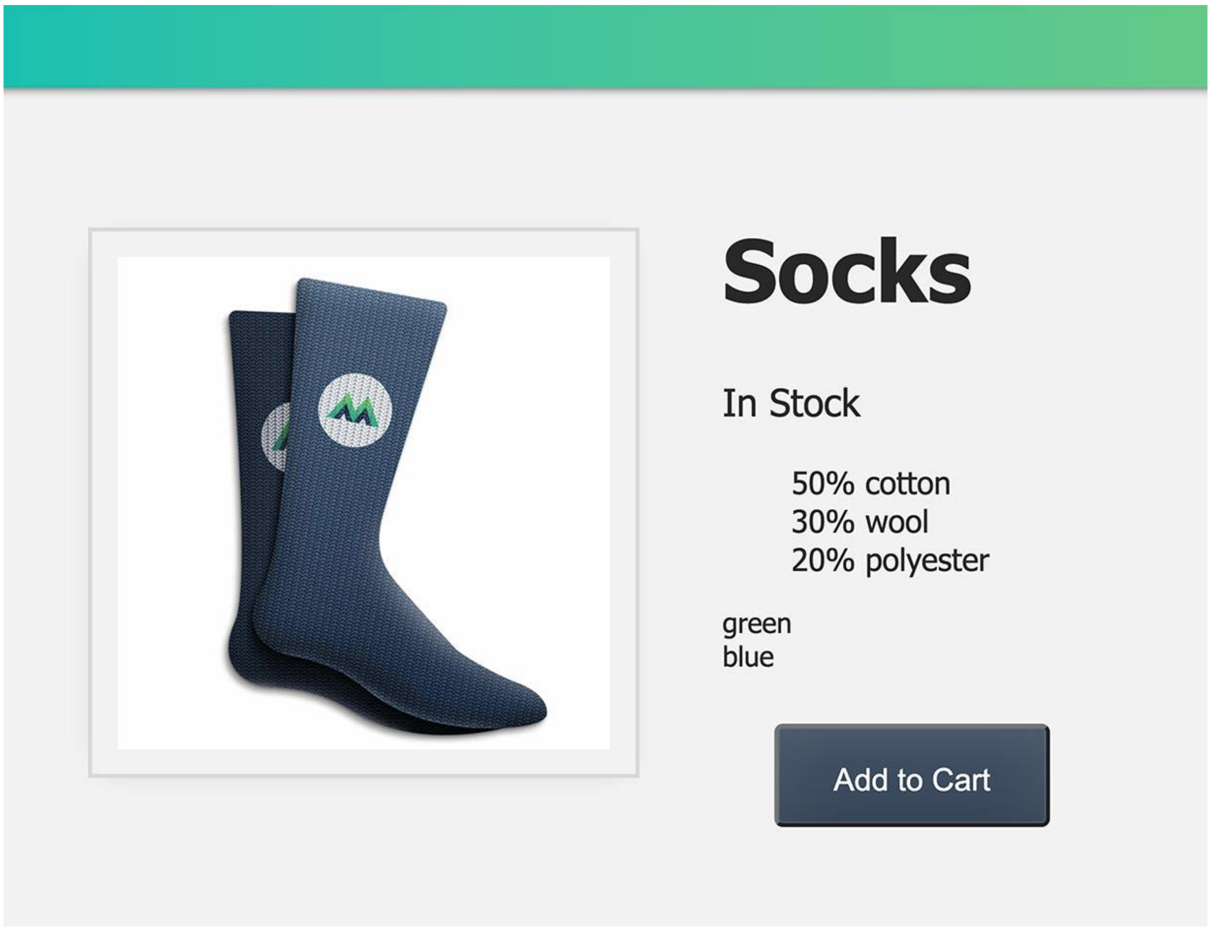
index.html

```html
<button class="button" @click="addToCart">Add to
Cart</button>
```

Another Example: Mouseover Events

Now that we understand the basics of event handling, let's listen for another kind of event within our Vue app.

Currently we're displaying the variant colors, "green" and "blue", just below the product details:


Socks

In Stock

50% cotton
30% wool
20% polyester

green
blue

Wouldn't it be nice if, when we hovered our mouse over "green" and "blue", we triggered an update of the image to the green and blue image, respectively? Let's add the ability to listen for `mouseover` events (Vue's term for "hover") on these color names.

Because we want to update the image that we're displaying when we mouse over the variant colors, I've added a new property to each variant object.

main.js

```
data() {
  return {
    ...
    variants: [
      { id: 2234, color: 'green', image:
'./assets/images/socks_green.jpg' },
      { id: 2235, color: 'blue', image:
'./assets/images/socks_blue.jpg' },
    ]
  }
}
```

Now each variant has an image path for the green and blue socks, respectively. We're ready to add a listener for `mouseover` events on the variant color `div`.

main.js

```
<div v-for="variant in variants" :key="variant.id"
@mouseover="updateImage(variant.image)">{{
variant.color }}</div>
```

When a `mouseover` event happens, we're triggering the `updateImage` method, passing in the image path of each variant. That method looks like this:

```
methods: {
  ...
  updateImage(variantImage) {
    this.image = variantImage
  }
}
```

It expects the `variantImage` as the parameter, and when it's run, it sets `this.image` (in *this* Vue instance's data) equal to the variant image that was passed in.

Now in the browser, when we hover our mouse over "green", we should see the green image. When we hover over "blue", we should see the blue image.

Coding Challenge

We've reached the end of the lesson and we're onto our challenge:

Create a new button that decrements the value of cart.

As a reminder, if you're coding along with our repo, you can check out `L6-end` branch, and you can view the solution code on Codepen.

Lesson Resources

- Starting Code
- Ending Code

VUE MASTERY

-
-
-
-
-
-
-
-

ABOUT US

-
-
-
-