Intro to Vue 3

Lessons

Communicating Events

In this lesson, we're going to look at the concept of communicating events that happen within our components. If you’re coding along with the repo, you can checkout the `L10-start` branch or the starting code on Codepen.

Our Goal

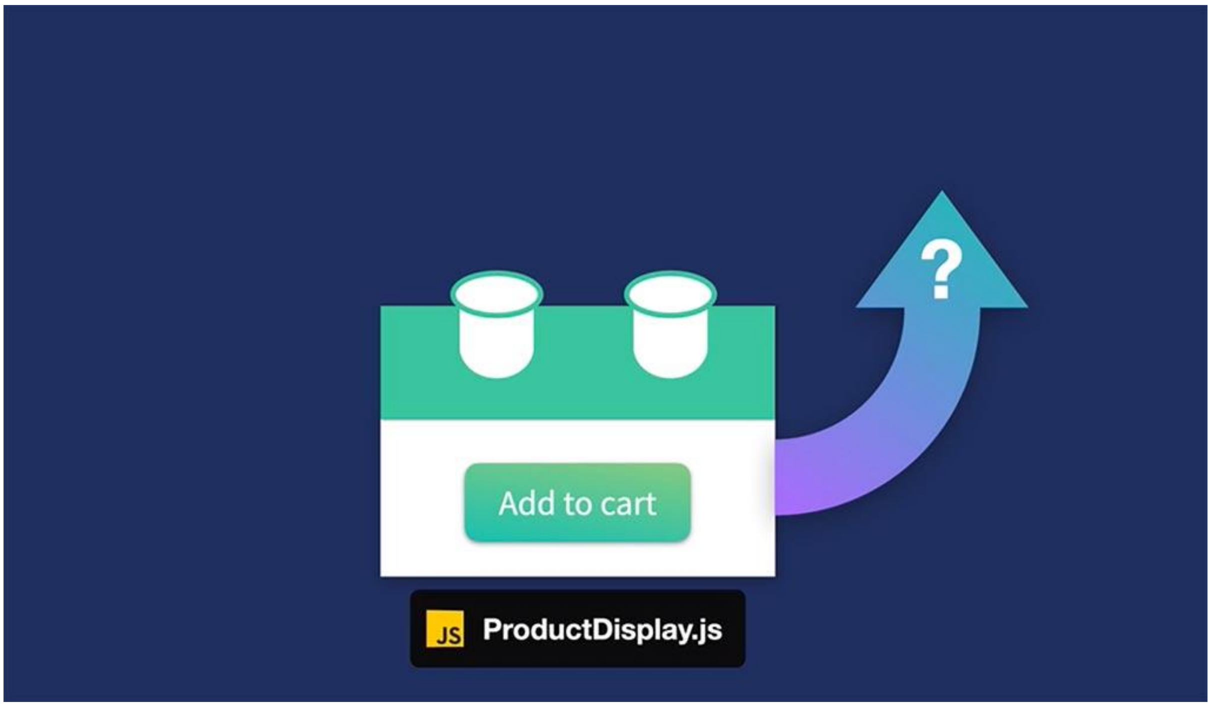
Give our component the ability to let its parent know an event happened within it.

Emitting the event

When we refactored things in our last lesson, moving product-related code into the new `product-display` component, we broke the ability to click the Add to Cart button and increment the value of `cart`. This is because the `cart` lives outside of the `product-display` component, inside the root Vue app in **main.js**.

We need to give the `product-display` component a way to *announce* that its button is clicked. How do we get this to happen?

We already know that props are a way to pass data down into a component, but what about when something happens within that component, like a button click? How do we let other parts of our app know that that event happened?



The answer is to emit that event, telling the parent that it happened. Let's add this ability within our `product-display` component, by amending the `addToCart()` method.

📄components/ProductDisplay.js

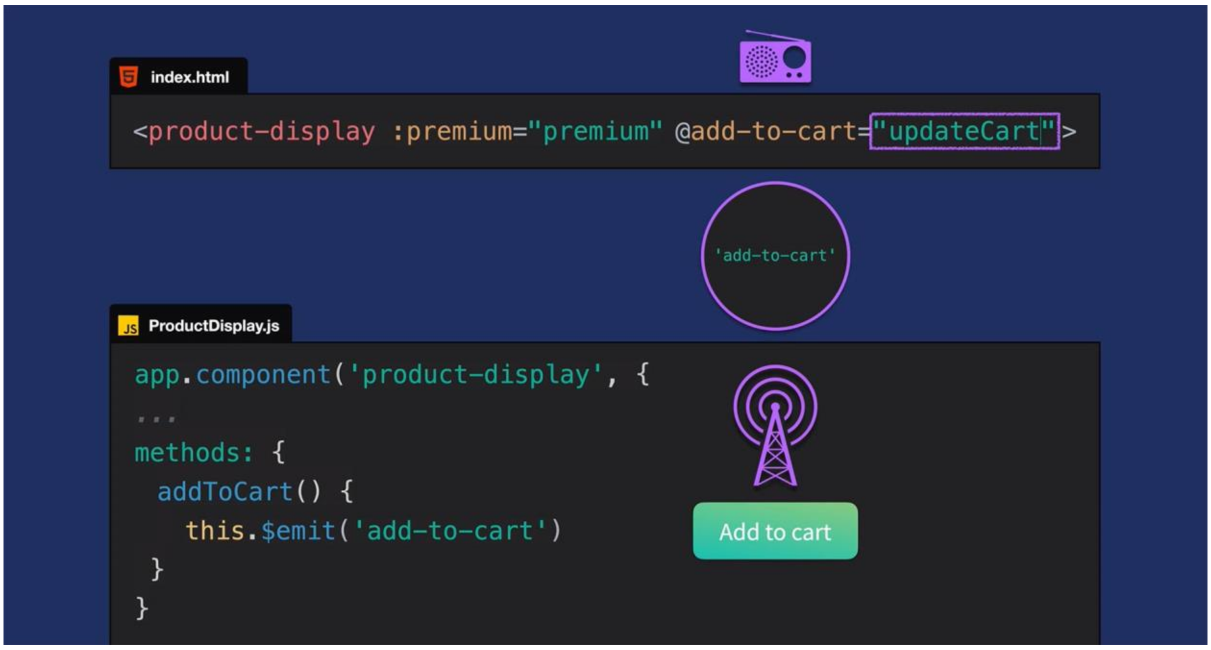
```
methods: {
  addToCart() {
    this.$emit('add-to-cart')
  }
  ...
 }
```

We'll write `this.$emit()` and emit an event called `'add-to-cart'`. So when the button is clicked, we're emitting, or bubbling up, that event. We can listen for that event from within the parent scope, where we're using `product-display`, by adding a listener: `@add-to-cart`

📄index.html

```
<product-display :premium="premium" @add-to-
cart="updateCart"></product-display>
```

When that event is “heard” by the parent, it will trigger a new method by the name of `updateCart`, which we’ll add within **main.js**.



index.html

```
<product-display :premium="premium" @add-to-cart="updateCart">
```

ProductDisplay.js

```
app.component('product-display', {
...
methods: {
  addToCart() {
    this.$emit('add-to-cart')
  }
}
```

📄main.js

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

If we check this out in the browser, we should now be able to click the Add To Cart button, which lets the parent know the `add-to-cart` event happened, triggering the `updateCart()` method.

Adding product id to the cart

To make our app more realistic, our `cart` shouldn't just be a number. It should be an array that contains the ids of the products that are added into it. So let's do a bit of refactoring.

📄**main.js**

```
const app = Vue.createApp({
  data() {
    return {
      cart: [],
      ...
    }
  },
  methods: {
    updateCart(id) {
      this.cart.push(id)
    }
  }
})
```

Now, `cart` is an array and `updateCart(id)` pushes the product `id` into it. We just need to add a payload to our `add-to-cart` event emission, so `updateCart` has access to that `id`.

📄**components/ProductDisplay.js**

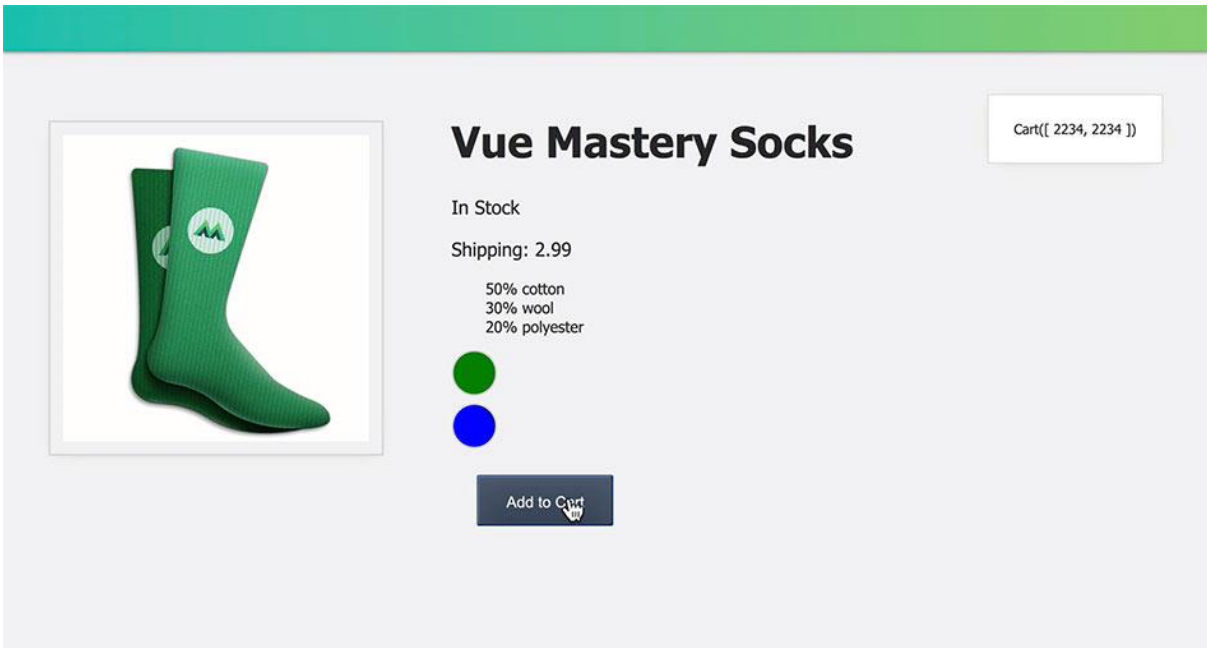
```
methods: {
  addToCart() {
    this.$emit('add-to-cart',
this.variants[this.selectedVariant].id)
  }
  ...
 }
```

Here, we've added a second parameter and grabbed the product's `id` much like we grabbed the `image` and `quantity` before:

components/ProductDisplay.js

```
computed: {
  image() {
    return this.variants[this.selectedVariant].image
  },
  inStock() {
  return this.variants[this.selectedVariant].quantity
  }
}
```

Now in the browser, you can see that we’re adding the product ids into the cart, which is now an array.



But we don’t need to actually display these ids. We just want to display how many items are in the cart. Fortunately, that’s a quick fix.

index.html

```
<div id="app">
  ...
  <div class="cart">Cart({{ cart.length }})</div>
  ...
 </div>
```

By adding `cart.length` we'll now only display the *amount* of items in the `cart`.

Coding Challenge

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

Add a new button to `product-display`, which removes the product from the `cart`.

As a reminder, if you're coding along with our repo, you can check out `L10-end` branch, and you can view the solution code on Codepen.

Download Video

Share Lesson

Lesson Resources

- Starting Code
- Ending Code

Discuss in our Facebook GroupSend us Feedback

Previous LessonNext Lesson

Vue mastery

VUE MASTERY

-
-
-
-
-
-
-
-

ABOUT US

-
-
-
-