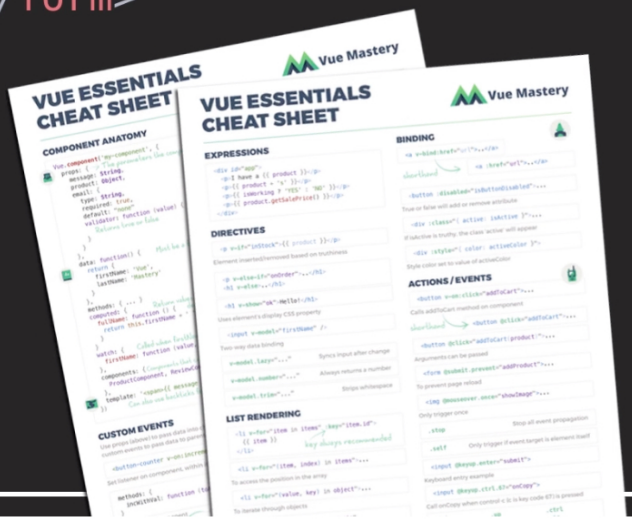
Intro to Vue.js

```
<button @click="addToCart">Add to cart</button>

<div @mouseover="updateProduct">Color</div>

<form @submit="addToCart">...</form>

<input @keyup.enter="send">
```

Modifier

.enter



Lessons

1. The Vue Instance
5:44

2. Attribute Binding
2:46

3. Conditional Rendering
3:44

4. List Rendering

Event Handling

In this lesson we'll be learning how to listen for DOM events that we can use to trigger methods.

Goal

We want to have a button that increments the number of items in our cart.

Starting Code

index.html

```html
<div id="app">
  <div class="product">
    <div class="product-image">
        <img :src="image" />
    </div>

    <div class="product-info">
        <h1>{{ product }}</h1>
        <p v-if="inStock">In Stock</p>
        <p v-else>Out of Stock</p>

        <ul>
            <li v-for="detail in details">{{ detail }}</li>
        </ul>

        <div v-for="variant in variants" :key="variant.variantId">
            <p>{{ variant.variantColor }}</p>
        </div>

    </div>
  </div>
</div>
```

main.js

```js
var app = new Vue({
    el: '#app',
    data: {
        product: "Socks",
        image: "./assets/vmSocks-green.jpg",
        inStock: true,
        details: ["80% cotton", "20% polyester", "Gender-neutral"],
        variants: [
          {
            variantId: 2234,
            variantColor: "green"
          },
          {
            variantId: 2235,
            variantColor: "blue"
          }
        ]
    }
  })
```

Problem

We need a button to listen for click events on it, then trigger a method when that click happens, in order to increment our cart total.

First, we'll add a new data property for our `cart`.

main.js

```
18          cart: 0
```

In our HTML, we'll create a `div` for our cart. We'll add a `p` inside it to display our `cart` data's value.

index.html

```
24          <div class="cart">
25              <p>Cart({{ cart }})</p>
26          </div>
```

We'll also make a `button` to add items to our `cart`.

index.html

```
22          <button v-on:click="cart += 1">Add to cart</button>
```

As you can see, we're using Vue's `v-on` directive to increment the value of `cart`

This works. But how is it working?

Let's dissect this syntax. We say `v-on`, which let's Vue know we're listening for events on this button, and after the `:` we specify the kind of event we are listening for, in this case: a click. Inside the quotes, we're using an expression that adds 1 to the value of `cart` every time the button is clicked.

This is simple, but not entirely realistic. Rather than using the expression `cart += 1`, let's make the click trigger a method that increments the value of `cart` instead, like this:

index.html

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

As you can see, `addToCart` is the name of a method that will fire when that click event happens. We haven't yet defined that method, so let's do that now, right on our instance.

Just like it does for its data, the Vue instance has an optional property for methods. So we'll write out our `addToCart` method within that option.

main.js

```
20    methods: {
21        addToCart() {
22            this.cart += 1
23        }
24    }
```

Now, when we click our `button`, our `addToCart` method is triggered, which increments the value of `cart`, which is being displayed in our `p` tag.

Let's break this down further.

Our `button` is listening for click events with the `v-on` directive, which triggers the `addToCart` method. That method lives within the methods property of the Vue instance as an anonymous function. The body of that function adds 1 to the value of `this.cart`. Because `this` refers to the data of the instance we're currently in, our function is adding 1 to the value of `cart`, because `this.cart` **is** the `cart` inside our data property.

If we just said `cart += 1` here, we'd get an error letting us know that "cart is not defined", so we use `this.cart` to refer to the `cart` from `this` instance's data.

You might be thinking, "But wait. We're only incrementing the number of items in the cart, we're not actually adding a product to the cart." And you're right. We'll build that out in a future lesson.

Now that we've learned the basics of event handling in Vue, let's look at a more complex example.

First, let's add a `variantImage` to each of our variants.

main.js

```
8       variants: [
9         {
10          variantId: 2234,
11          variantColor: "green",
12          variantImage: "./assets/vmSocks-green.jpg"
13
14        },
15        {
16          variantId: 2235,
17          variantColor: "blue",
18          variantImage: "./assets/vmSocks-blue.jpg"
19        }
20      ],
```

Now each variant has an image with green and blue socks, respectively.

Problem

We want to be able to hover our mouse over a variant's color and have its `variantImage` show up where our product image currently is.

Solution

We'll use the `v-on` directive again, but this time we'll use its shorthand `@` and listen for a `mouseover` event.

index.html

```
16      <div v-for="variant in variants" :key="variant.variantId">
17          <p @mouseover="updateProduct(variant.variantImage)">
18              {{ variant.variantColor }}
19          </p>
20      </div>
```

Notice that we're passing `variant.variantImage` in as an argument to our `updateProduct` method.

Let's build out that method.

main.js

```
27        updateProduct(variantImage) {
28            this.image = variantImage
29        }
```

This is very similar to what we did to increment the value of `cart` earlier.

But here, we are updating the value of `image`, and its updated value is now the `variantImage` from the variant that was just hovered on. We passed that variant's image into the `updateProduct` function from the event handler itself:

index.html

```
17            <p @mouseover="updateProduct(variant.variantImage)">
```

In other words, the `updateProduct` method is ready to fire, with a parameter of `variantImage`.

When it's called, `variant.variantImage` is passed in as `variantImage` and is used to update the value of `this.image`. As we just learned, `this.image` **is** `image`. So the value of `image` is now dynamically updating based on the variant that was hovered on.

ES6 Syntax

Instead of writing out an anonymous function like `updateProduct: function(variantImage)`, we can use the ES6 shorthand and just say `updateProduct(variantImage)`. These are equivalent ways of saying the same thing.

What'd we Learn

What'd We Learn

- The `v-on` directive is used to allow elements to listen for events
- The shorthand for `v-on` is `@`
- You can specify the type of event to listen for:
  - click
  - mouseover
  - any other DOM event
- The `v-on` directive can trigger a method
- Triggered methods can take in arguments
- `this` refers to the current Vue instance's data as well as other methods declared inside the instance

Learn by doing

Challenge:

Create a new button and method to decrement the value of `cart`.

Download Video

Share Lesson

Lesson Resource

- Get the blue socks image

Coding Challenge

Start the Challenge

View the Solution

Discuss in our Facebook Group

Send us Feedback

Previous Lesson

Next Lesson