# 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

5. List Rendering

3:31

6. Event Handling

4:31

7. Class & Style Binding

6:37

8. Computed Properties

6:23

9. Components & Props

6:38

10. Communicating Events

3:57

11. Forms & v-model

8:27

# Conditional Rendering

In this lesson, we're going to look at the concept of conditional rendering. If you're coding along with the repo, you can checkout the `L4-start` branch or use the starting code on Codepen.

---

# Our Goal

We want to display different HTML elements based upon a condition. We'll display a `p` tag that says "in stock" when our product is in stock, or one that says "out of stock" when it's not.

---

# To render or not to render

In our index.html file, we'll add two new `p` tags.

### index.html

```
<p>In Stock</p>
<p>Out of Stock</p>
```

We only want one of these to show up depending on if our product is in stock or not, so we'll head into our Vue app's data object and add a new `inStock` boolean.

### main.js

```
const app = Vue.createApp({
    data() {
        return {
            product: 'Socks',
            image: './assets/images/socks_blue.jpg',
            inStock: true // new data property //
        }
    }
})
```

Now that we've added the elements we want to conditionally render, and the condition (`inStock`) that we'll use to decide which one to render, we're ready to learn about another Vue directive.

---

# The v-if directive

We can add the `v-if` directive onto an element to render it based upon a condition, like so:

```
<p v-if="inStock">In Stock</p>
```

Now, this element will render only *if* `inStock` is truthy.

We can combine the `v-if` directive with its sister directive `v-else` to display another element as the fallback if the first condition turns out to be falsey.

**index.html**

```
<p v-if="inStock">In Stock</p>
<p v-else>Out of Stock</p>
```

Now, if `inStock` is `false`, we'll see "Out of Stock" gets rendered to the page.

---

# Show and Hide

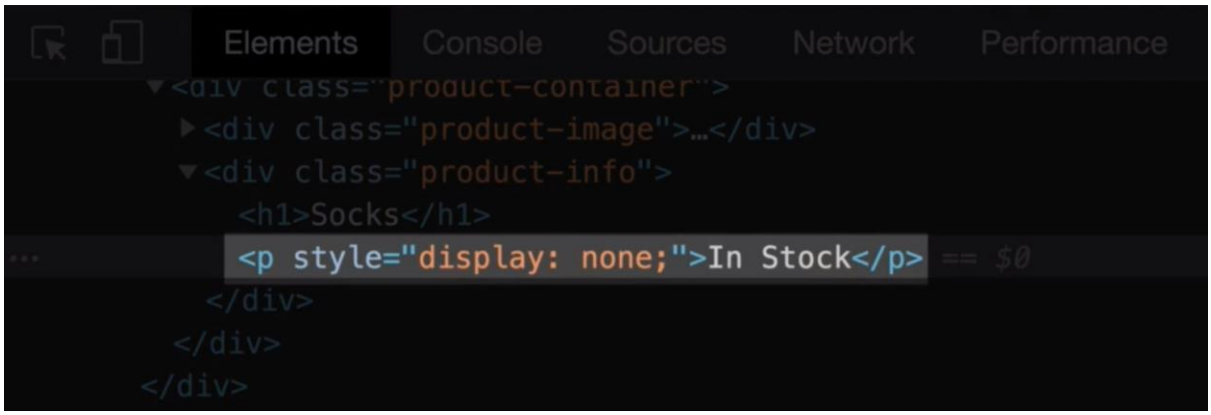
It's worth noting that you don't always need to pair `v-if` with `v-else`. There are plenty of use cases where you don't need a fallback element to render. However, in these cases, it is sometimes a better option to use the `v-show` directive.

### index.html

```
<p v-show="inStock">In Stock</p>
```

The `v-show` directive is used for toggling an element's **visibility** instead of adding and removing the element from the DOM entirely, like `v-if` does.

As you might imagine, this is a more performant option if you have something that's toggling off and on the screen often. We can verify this by setting `inStock` to `false` and viewing the element in the browser's Developer Tools. When `v-show` is used, we can see that the element is still present in the DOM, but it's now hidden with an inline style of `display: none;` added to it.




---

# Chained Conditional Logic

Earlier, we looked at `v-if` with `v-else`, now let's take a look at how we can add additional layers of conditional logic.

To do this, we'll replace `inStock` with `inventory`:

### main.js

```
const app = Vue.createApp({
    data() {
```

```
        return {
            ...
            inventory: 100
    }
```

Since our condition (`inventory`) is now an integer, we can use a bit more complex logic within our expression. For example:

### 📄index.html

```
<p v-if="inventory > 10">In Stock</p>
<p v-else>Out of Stock</p>
```

Now, we will only render the first `p` tag if `inventory` is greater than `10`.

Let's say we want display a new message when the product is almost sold out. In this situation, we could add another conditional level, where we're watching out for `inventory` to get below `10` but above `0`.

### 📄index.html

```
<p v-if="inventory > 10">In Stock</p>
<p v-else-if="inventory <= 10 && inventory > 0">Almost
sold out!</p>
<p v-else>Out of Stock</p>
```

The `v-else-if` directive gives us a middle layer of logic. So in this example, if `inventory` were `8`, this `p` tag would get rendered.

Of course, if `inventory` is zero, we'll default to the final level of `v-else` and display "Out of stock".

---

# Coding Challenge

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

Add an `onSale` Boolean to the data object.

Use `onSale` to conditionally render a `p` tag that says "On Sale," whenever `onSale` is `true`.

As a reminder, if you're coding along with our repo, you can check out `L4-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 Group Send us Feedback

Previous LessonNext Lesson

Vue mastery

As the ultimate resource for Vue.js developers, Vue Mastery produces weekly lessons so you can learn what you need to succeed as a Vue.js Developer.

## VUE MASTERY

- 
- 
- 
- 
- 
- 
- 
- 

## ABOUT US

- 
- 
- 
- 