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

List Rendering

In this lesson, we're going to look at the concept of list binding. If you're coding along with the repo, you can checkout the `L5-start` branch or view the starting code on Codepen.

Our Goal

Render HTML lists from an array in our data.

Looping through data arrays

In the starting code, we now have an array of `details`.

📄main.js

```
const app = Vue.createApp({
    data() {
        return {
            ...
            details: ['50% cotton', '30% wool', '20% polyester']
        }
    }
})
```

The question now is: how do we display this data as a list?

We'll start by creating an unordered list in our **index.html**. On the `li` inside of it, we'll add another Vue directive: `v-for`

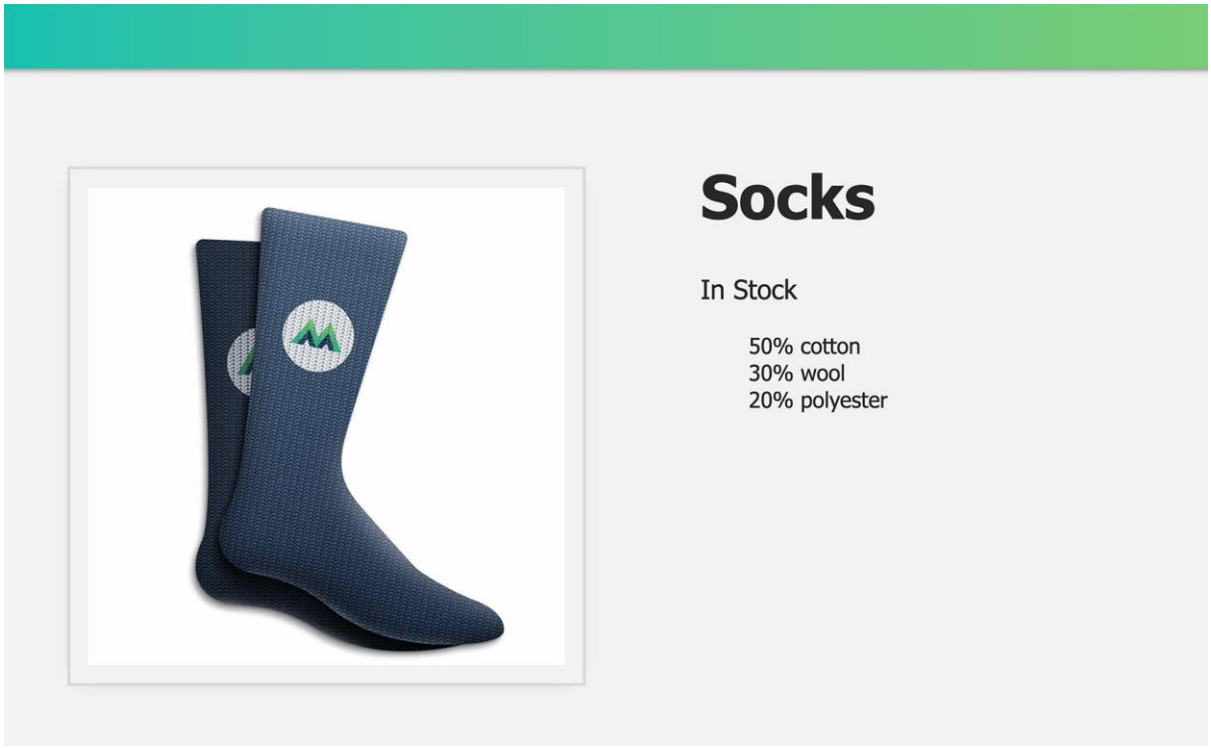
📄index.html

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

Inside the `v-for` expression, we wrote: `detail in details`. Here, `details` refers to the `details` array in our data, and `detail` is the alias for the current element from that array, as we're looping through it to print out a new `li`.

Each `li` will display that array element because in the inner HTML we've written an expression: `{{ detail }}` to print out each detail.

If we check the browser, we'll see a list of the `details` is displayed.



So far so good, but how is `v-for` actually working?

Product Variant Colors

To get more familiar with list rendering with `v-for`, we'll work on another example within our app. Let's add a new `variants` array to our data:

main.js

```
data() {
  return {
```

```
    ...
    variants: [
      { id: 2234, color: 'green' },
      { id: 2235, color: 'blue' }
    ]
  }
}
```

We now have an array that contains an object for each variant of our product. Each product variant has an `id`, and a `color`. So for our next task, we'll print out each variant color, and use the `id` to help Vue keep track of our list items.

index.html

```
<div v-for="variant in variants" :key="variant.id">{{ variant.color }}</div>
```

Notice how we're using dot notation to print out each `variant` as we loop through the `variants` array. But what is that `:key` attribute doing there?

Key Attribute: An essential for list items

By saying `:key="variant.id"`, we're using the shorthand for `v-bind` to bind the variant's `id` to the `key` attribute. This gives each DOM element a unique key so that Vue can grasp onto the element and not lose track of it as things update within the app.

This provides some performance improvements, and later down the line, if you're doing something like animating your elements, you'll find that the `key` attribute really helps Vue effectively manage your elements as they move around the DOM.

Coding Challenge

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

Add an array of `sizes` to the data object.

Use `v-for` to display the `sizes` in a list.

As a reminder, if you're coding along with our repo, you can check out `L5-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 Lesson Next Lesson

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

-
-
-
-