**Class & Style Binding**

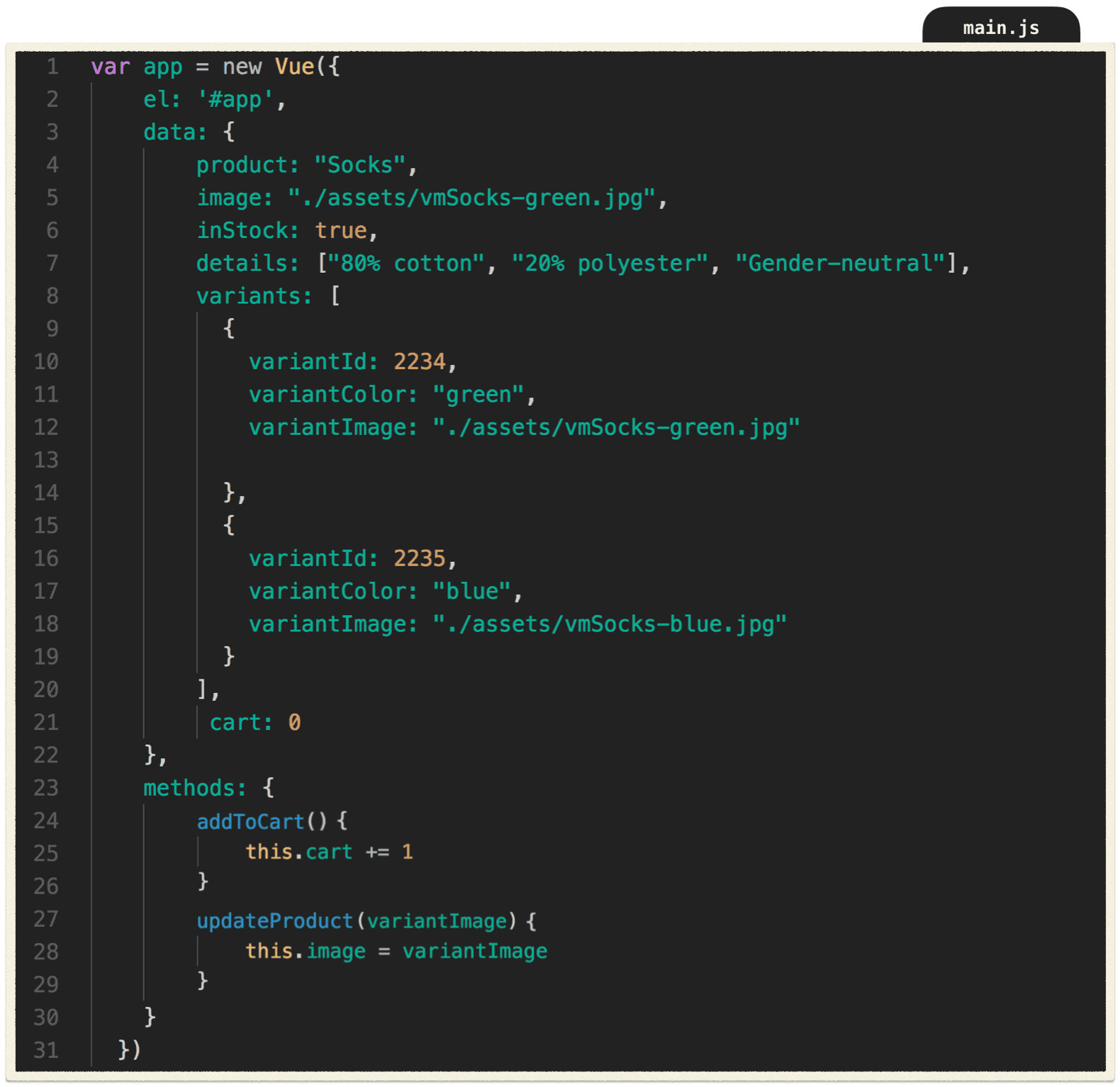
In this lesson we’ll be learning how to dynamically style our HTML by binding data to an element’s style attribute, as well as its class.

**Goal**

Our first goal in this lesson is to use our variant colors to style the background-color of divs. Since our variant colors are “green” and “blue”, we want a div with a green background-color and a div with a blue background-color.

**Starting Code**



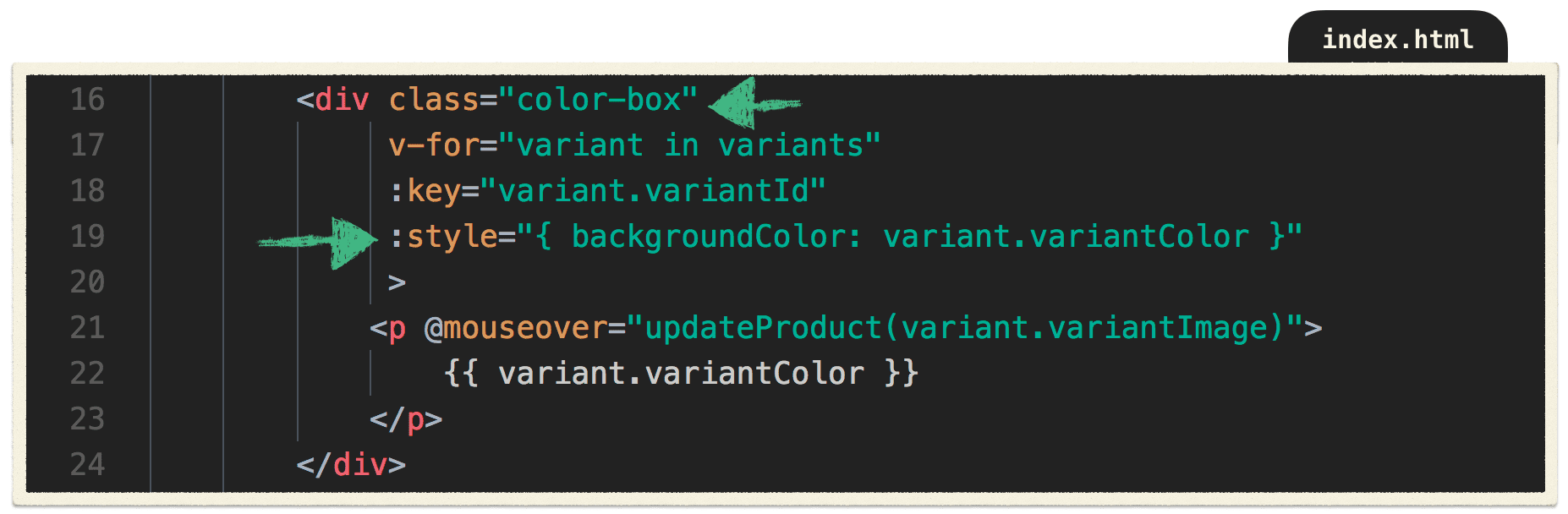


**Problem**

In the previous lesson, we created an event handler that updates the product’s image based on which p tag was hovered on. Instead of printing out the variant’s color into a p tag, we want to use that color to set the style of a div’s background-color. That way, instead of hovering over text in a p tag, we can hover over colored squares, which would update the product’s image to match the color that was hovered on.

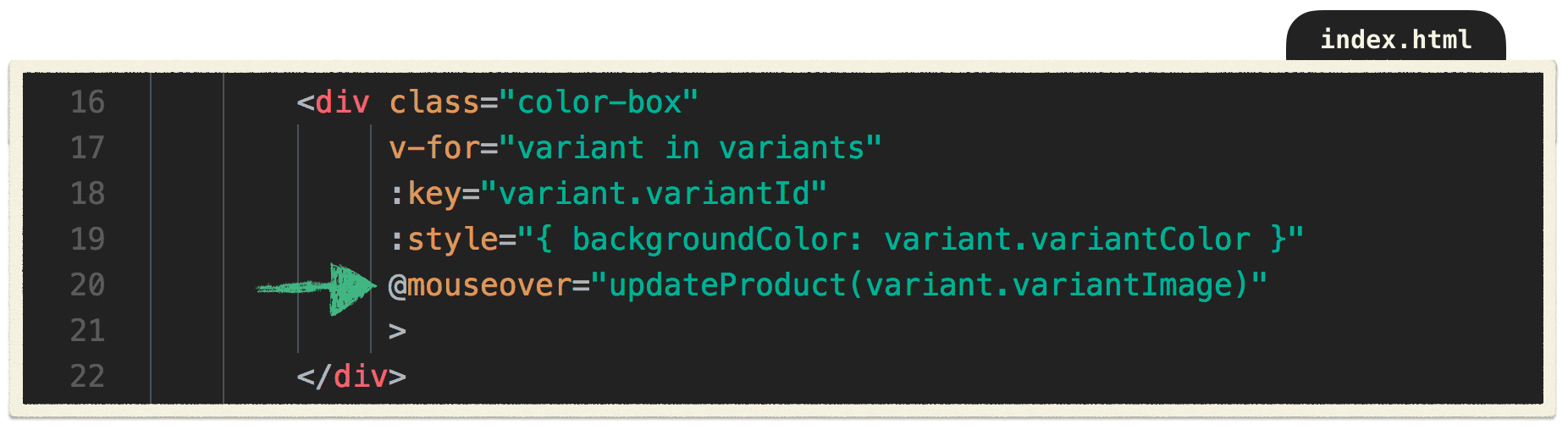
**Solution**

First, let’s add a class of color-box to our div, which gives it a width, height and margin. Since we’re still printing out “green” and “blue” onto the page, we can make use of those variant color strings and bind them to our style attribute, like so:



We are using an inline style to dynamically set the background-color of our divs, based on our variant colors ( variant.variantColor ).

Now that our divs are being styled by the variantColor, we no longer need to print them out. So we can delete the p tag and move its @mouseover into the div itself.



Now when we hover over the blue box and the blue socks appear, hover over the green box and the green socks appear. Pretty neat!

Now that we’ve learned how to do style binding, let’s explore class binding.

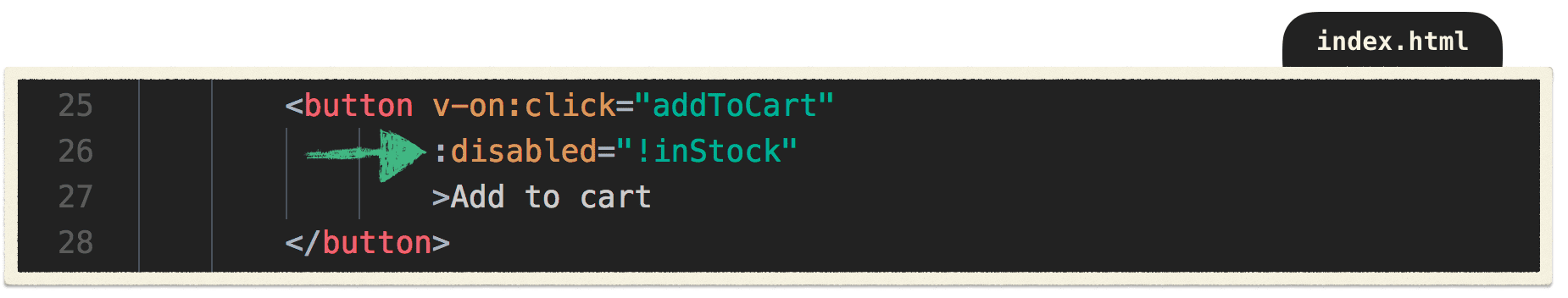
**Problem**

Currently, we have this in our data:



When this boolean is false, we shouldn’t allow users to click the “Add to Cart” button, since there is no product in stock to add to the cart. Fortunately, there’s a built-in HTML attribute, disabled, which will disable the button.

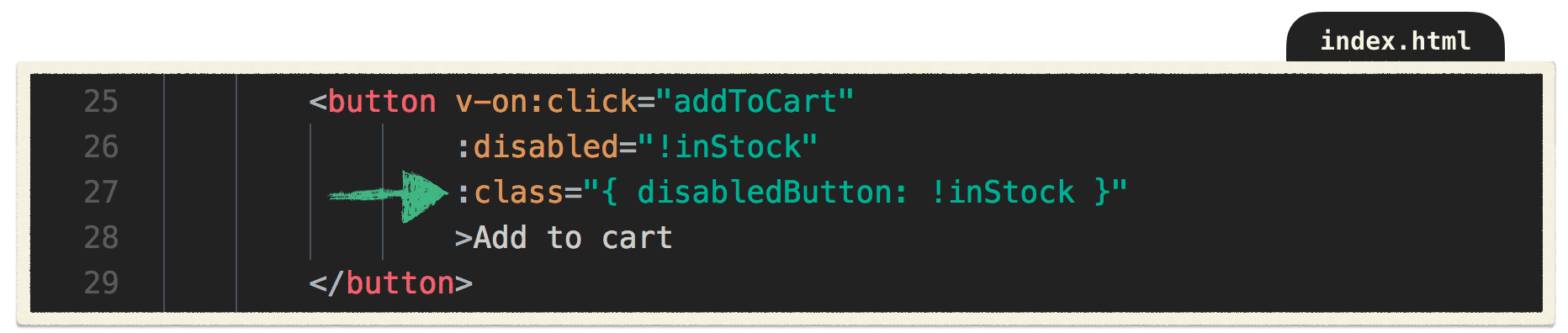
As we learned in our second lesson in this series, we can use attribute binding to add the disabled attribute whenever inStock is false, or rather *not true*: !inStock.



Now our button is disabled whenever inStock is false. But that doesn’t change the appearance of the button. In other words, the button still appears clickable, even though it’s not.

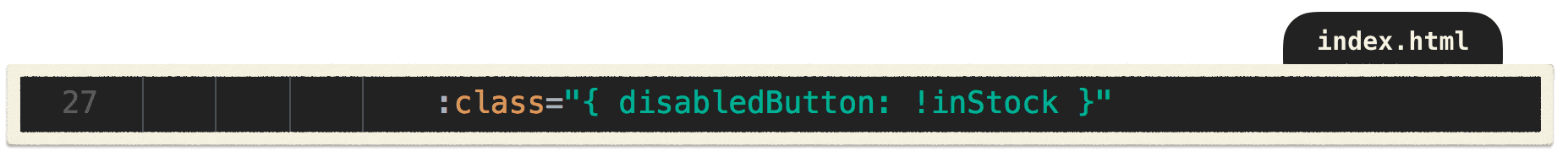
**Solution**

In a similar way to how we just bound inStock to the button’s disabled attribute, we can bind a disabledButton class to our button whenever inStock is false. That way, our button will also *appear* disabled.



It works! The button is now grayed out when inStock = false.

Let’s break this down.



We’re using the v-bind directive’s shorthand : to bind to our button’s class. Inside the brackets we’re determining the presence of the disabled-button class by the truthiness of the data property inStock.

In other words, when our product is not in stock ( !inStock ), the disabledButton class is added. Since the disabled-button class applies a gray background-color, the button turns gray.

Great! We’ve combined our new skill **class binding** with attribute binding to disable our button and turn it gray whenever our product inStock is false.

**What’d we learn**

* Data can be bound to an element’s style attribute
* Data can be bound to an element’s class
* We can use expressions inside an element’s class binding to evaluate whether a class should appear or not

**What else should we know?**

* You can bind an entire class object or array of classes to an element 

**Learn by doing**

**Challenge:**

When inStock is false, bind a class to the “Out of Stock” p tag that adds text-decoration: line-through to that element.