Claro, aquí tienes el texto ordenado manteniendo el idioma original:

In this video, I’d like to talk about reactive values and statements. But to do that, I’d like to flesh out an example so it’s easier to understand. Right now, we just have these two values: name and belt color. But what I want to do is expand this so that we have a first name and a last name.

So, let’s create those. First, I’ll change the first name to “Jimmy” and then add a last name, setting it to “Hendrix”. Now I have a first name and a last name. In fact, I’m going to create an input for each one, so a user can choose what these different values are, much like we have the belt color input right here.

I’ll delete a lot of the unnecessary code, like the button, and remove the stuff at the top. We don’t need that anymore. I’ll keep this part, though, because that’s where we output the belt color. Now I’ll add a couple of inputs. First, I’ll bind the first name:

<input bind:value={firstName} />

Then, I’ll duplicate this and change it to last name:

<input bind:value={lastName} />

Now we have three inputs, so the user can choose the first name, last name, and belt color. We’re still outputting the belt color here, but instead of showing it directly, let me get rid of that line and save the file. Now, we’re only outputting the belt color without the text description.

If I go over here, I can see that if I change the belt color to “red”, it updates here. But these two, if we change them, nothing updates because we’re not outputting those values yet. So, if I wanted to output the full name, I’d have to concatenate the first name and last name somehow.

I could do something like this:

firstName + " " + lastName

That works. If I save it, it says “Jimmy Hendrix black belt”. If I change the first name to “Jimmy Foo”, it updates here too. But instead of doing this concatenation manually, it would be nice if we had a value called fullName that we could output, and it would automatically update whenever the first name or last name changes. This way, it would watch those values and update itself based on them.

Now, we can use what’s known as a **reactive value** to do that.

### Reactive Values

Reactive values in Svelte are just values that automatically update when the data they depend on changes. They basically watch certain values in your code and react to them by updating the corresponding value. They’re a little like **computed properties** in Vue.js, if you have any experience with that.

The way we do it is by using this syntax: $: followed by the name of the reactive value. I’m going to call it fullName. Now I set it equal to something. All I need to do is concatenate these two values, like this:

$: fullName = `${firstName} ${lastName}`;

This is our reactive value. Now, we can output it over here. Instead of concatenating the full name in multiple places, we can just output the fullName reactive value:

<h1>{fullName}</h1>

This is much easier, especially if we need to output the full name in several places. Now, if I save it, we see that we can change the first name, the last name, and the belt color, and everything updates automatically.

This is a simple example, but the main point is to show you how reactive values work. Later on, we’ll use more complex reactive values, but I just wanted to introduce the concept now so you’re not confused when we get to more advanced examples.

### Reactive Statements

Now, let’s talk about **reactive statements**. They’re similar to reactive values, but they’re not just simple values; they’re statements that run whenever the data inside them changes. In other words, instead of assigning a value, we’re running a block of code whenever the values inside the statement change.

The way we do this is almost the same as with reactive values. We use the $: syntax followed by the code we want to run. For example, I could say:

$: console.log(beltColor);

This reactive statement will log the beltColor to the console whenever it changes. So, if I save it and go over here, open the console, and change the belt color, you’ll see it automatically logs out the new color each time.

Now, if you want to run more complex code—let’s say you want to log both the belt color and the full name—you can use curly braces {} to create a code block:

$: {  
 console.log(beltColor);  
 console.log(fullName);  
}

This will run the entire block of code every time either of these values changes. You don’t need both to change; if just one of them changes, the whole block will run.

So, if I save this and go over here, change the belt color to red, it will output both the color and the full name to the console every time.

By the way, we can use other reactive values inside reactive statements as well. So, they’re very flexible.

### Summary

These are just simple examples, but I wanted to introduce you to the concepts of **reactive values** and **reactive statements**. They’re powerful features in Svelte that let you automatically react to changes in state and run code when needed. These examples are basic, but later on, we’ll use them in more complex ways, and I wanted to make sure you’re familiar with them when that happens.