**Mix text and variables with template literals**

Selecting transcript lines in this section will navigate to timestamp in the video

- At this point we need to take a quick detour and talk about how to output content from JavaScript to an HTML document. I'm sure you're wondering why we've been spending so much time on the console, but nothing really happens in the browser. Well, we're about to bridge that gap. We need a way of effectively bridging the gap between what happens in the JavaScript world and what happens in the human world, and HTML functions as that bridge. A typical use of JavaScript is to either generate HTML, modify existing HTML, or remove HTML from a live document in the browser. So from this perspective, it's useful to think of JavaScript as an interactive layer on top of the content, the HTML, and its presentation, the CSS. And we can use Java scripts to manipulate the HTML and CSS to get the browser to do what we want. The most basic way of doing this is to use JavaScript to inject new HTML content into the document. To do that we first need to access an element within the document. We can then manipulate with JavaScript and now all that work we've been doing so far with objects comes into play because the entire document is an object in the browser. Let me show you. If I go to this new HTML document from the exercise files and just type in documents in the console and hit return, we get the document object and return. And this really is the full HTML output of the current document. You can see it here. We have an article with an H one, then an unordered list. And if I go over into index.HTML, these match exactly. So what happens is when the browser renders a document, it creates a document object model of that document. And we can then access that document object using JavaScript. I will talk about this in great detail in the next chapter. For now I want to show you how to directly manipulate this using something called a template literal. What I want to do is change the output here so that instead of just listing the properties we also get the property values. But the property values are sitting in our JavaScript object. So what I'm going to do is replace the content over here on the browser with content generated with JavaScript. So I'll show you first how we do it and then we'll use the template literal to output exactly the content we want. Here's the key: we can access anything within the document object model. So for example, I can say document DOT's body. This gives me the body elements. And then we can say give me the inner HTML of the body elements. So I'll say inner HTML. This is a property of the body elements. And it'll return to me all the inner HTML here. Now, instead of asking for it to be returned I'm going to reset it to something else. So I'll just type in some text here and then hit return. Now what happens is in the browser, I wiped out everything inside the body element and replaced it with just that string of texts, some texts and nothing else. So the main and the article, everything else has gone. Now, it's only gone in this current document object model in the browser. So if I reload the browser, everything comes back again. Okay, keep this in mind. Now we want to create a template where we use this HTML that you have here. And then we fill it in with data from our JavaScript object. So I'll copy the main, actually I'll cut out the main completely so the body is empty and save it. Now we have nothing in the browser. That's where we want to start. Then I'll go to my Java script and here I'll create a new constant, call it content. And then I'll set it equal to two back ticks. Now notes back ticks are different from quotation marks and they're also different from single quotation marks. It's a left-leaning, a weird tick that you don't normally use an English language on American keyboards. You'll find it directly under the escape key. On other language keyboards, it may be hiding somewhere else but you should be able to find it. The back ticks tell the browser that anything inside here is a template literal meaning we can mix HTML and strings with JavaScript expressions to literally create templates. And that's what we want to do here. So I'll paste in that HTML I copied out from index and then maybe clean up the layout a little bit here so that it's nice and clean. Then I'll use Java scripts to get the document and the body inside the document and the inner HTML inside the body and set it equal to the contents of this new constant content. Save that. And now JavaScript is injecting this main and everything else into the HTML on the fly. Cool, now we need to somehow pull the values from our everyday pack object into this template literal. And this is why it's called a template literal. Inside of template literal, anytime you want to call in something generated by Java script, we add a placeholder. It looks like this, take the title here and then set dollar sign and then curly brackets. And inside the curly bracket we can place any JavaScript expression. That JavaScript expression will be rendered and the output will be placed where this placeholder is. So in our case, we're looking for the everyday backpack object and I want the name. So name, save this and the name property is now sent over. It looks the same, but what if I changed the name property of our objects to Green frog? Save. And now it changes over here as well to green frog because we are now injecting the value from the everyday pack object into our template literal. Now we can do the same with all of the other properties to fully fill out our object. The template literal allows me to inject any JavaScript expression so we can call for properties. We can also run functions like this method here the backpack age that outputs the age and we can even run advanced functions or call in new functions if we want to. What's happening is we're just co-mingling strings, in this case HTML with generated JavaScript output. And this is what makes template literals so powerful. Save this and you'll see over in the browser we now have full output that uses the data from our object co-mingled with HTML to give us all the information we need in the browser. We're effectively bridging the JavaScript world and the human world through injecting HTML into the document. Thanks to this template literal, we can now create a string like this with formatting and everything else and co-mingle JavaScript into that string for generated output. If you ever worked with template tags in another language like PHP, you'll recognize this pattern and moving forward, I'll use template literals to output my content whenever possible. And I encourage you to do the same.