* - [Instructor] JavaScript can be **used to modify the attributes of any elements** .
* The attributes are stored in the **attributes property**, it can be accessed directly by calling it .
* One thing worth noting here is the *attributes property returns a different type of data from the class name and class list properties .*
* What we get is a *named node map* and it is not an array .
* So we can't use array methods on it .
* There's a simple reason for this .
* ***An attribute is a more complex piece of data, it has a key and a value and we need to be able to manipulate just the key or just the value or both at any time and a simple array wouldn't do that .***
* Let's look at the attributes property first .
* So I'll select this image here 'cause I know it has attributes and has a source attribute and an alt attribute .
* Graphical user interface, text, application, email, website

  Description automatically generated
* Then we can call it in using JavaScript and look at what that attributes property contains .
* What I get in return is a **named node map .**
* Inside here we have **the two attributes source and alt** and we can open each of them and get a full breakdown of all the information the browser has about this particular attribute .
* So in this case, we have a node value of assets, images, everyday, svg which matches what you see up here .

Graphical user interface, text

Description automatically generated

* Now, looking at this, you're probably also wondering how am I supposed to work with this pile of data? All the good news is, for attributes we have dedicated methods to do all the things you would normally want to do .
* So let me show you how that works .
* First***, let's test to see if the element has an attribute we're looking for .***
* So we'll say again, document, query selector and look for the image .
* And then we'll say, **has attribute** and pass in the value of the attribute we're testing for, so source .
* If the element in question has this attribute the return will be true .
* If it doesn't have the attribute, so title, that's an image there it shouldn't have a title and it doesn't .
* So we get false .

Graphical user interface, text, application

Description automatically generated

* So this is an easy way of testing does this attribute exist on this particular element .
* All right .
* So now that we know it has the attribute I want to see what that attribute contains, for that I can say, **get attribute** then we specify what attribute we're looking for and the browser will return the value assigned to that attribute .
* Graphical user interface, text, application, email

  Description automatically generated
* So in this case, it is the URL .
* And if I change this to the alt attribute, you'll see, even though it's blank up here, it actually contains something and contains the beginning and end quotation marks because this is an empty alt attribute .
* I can also set an attribute here .
* I can either set the value of an existing attribute or create a new attribute altogether .
* I do that using the set attribute method, here I first name the attribute I want to set then I add a comma and then put in whatever value I want to attribute to that attribute .
* So here I want to add text to the alt attributes .

Graphical user interface, text, application, email

Description automatically generated

* So I'll say, a drawing of a backpack, hit return and now that new value is appended to the existing attribute .
* I can use the set attributes to create new attributes as well .
* So if I go back here and say set attributes title .
* Now again, this is an image element, so it shouldn't have a title but I'm going to do this anyway .
* And then set the title to, this shouldn't be here and hit return .
* You'll see now the image gets a new attribute called title that has the value this shouldn't be here .
* All right .
* Now that I've added this attribute that shouldn't be here, I should really remove it as well .
* So I will say, **remove attribute** title and hit return .

Graphical user interface, text, application

Description automatically generated

* The title attribute is removed between has ***attribute get attributes, set, attribute, and remove attribute .***
* You have all the functions you need to be able to manipulate any attribute inside a tag .
* And note here that, ***everything inside the tag is considered an attribute .***
* That means if we target an element that has a class, the class will also show up as an attribute and we can actually use these attribute methods to manipulate classes if we want to .
* Now, keep in mind we have dedicated the properties and methods for classes but if you need to, you can also use all of these attributes to manipulate classes in any element .