* - [Instructor] A common task in JavaScript ***is to modify the classes of an element*** in some way .
* This is *a simple way of changing the appearance or behavior of an element without having to inject CSS into the HTML itself* .
* Instead***, we just add a class or remove a class*** to make something happen .
* Examples of this include hiding and showing a panel, or highlighting a button when something else has happened, like now that you've done step one, click here to continue, et cetera .
* Class manipulation is common enough, that we have two specialized properties for it .
* The first one and the oldest one is class name, and it's available for all elements in the dom .

Text

Description automatically generated

* Class name holds a string, listing out all the classes appended to an element .
* We can see it if we go to our exercise files and the console .
* So the H1 here has the class backpack name, and we can call up the H1 in JavaScript and get just that output .
* So document, query selector, H1 .
* Now we have the whole element .
* Then I can call for the class name property and I get just backpack name as a string in return .

Graphical user interface, text, application

Description automatically generated

* Now this is a property, meaning I can reset this property and reassign it any other value .
* So if I want to, I can say equals and then put in a different value, and watch what happens up here in the Dom when I hit return .

Graphical user interface

Description automatically generated with medium confidence

* The class Changes because I'm reassigning the class name property .
* This is really powerful .
* It's also a bit problematic .
* There **are two challenges** with using class name .
* **First off,** *if you have an element with several classes like we have here in the UL, we have several elements that have several classes, you can't just replace the classes with a new class, because that way you're wiping out all the classes, you're exchanging all the content within the property for some other content .*
* So if you have a property with several classes and you want to just change one of the classes you have to somehow go in and parse out which classes you want to keep, and which ones you want to change and it gets really clunky and requires a lot of code .
* **The other problem is even more serious .**
* In React and other JavaScript frameworks*, the term class name is used in place of class in markup to avoid collisions with the JavaScript class keyword which is used to set up classes* .
* This means if you're trying to target the class name property, but you're inside a JavaScript framework, you're actually doing something else which can cause unexpected behavior and will not result in what you want .
* *As a result we really shouldn't be using the class name property in frameworks, and we should be careful about using the class name property anywhere else as well .*
* Instead to get around both of these problems, we can use the **class list property .**
* Graphical user interface, text, application, email

  Description automatically generated
* The class list property gives us **a Dom token collection** of all the classes appended to an element .
* Let me show you what that means .
* If I go back to my code example here, let's go back to the browser and try to target an element that has multiple classes, like the first element in the list here .

Graphical user interface, text, application

Description automatically generated

* If I now call the **class list** property, you'll see I get a Dom token list, it's an array listing out each of the classes in turn .
* That means I can now work with each of these classes individually, and I'm not required to replace the entire string .
* What makes this property even more powerful is it comes with its own set of methods to manipulate the individual classes .
* So if I want to add a new class, I can use the add method .

Graphical user interface, text, application

Description automatically generated

* I say, class list dot, add, and inside I specify what class or classes I want to add .
* So I want to add new class .
* And you can see, the new class is added automatically .



* There's also a **remove** method .
* *So if I say remove, I remove that class, there's also a toggle method, so if I want to add a class, if it's not there, and remove it, if it is there, I can use the* **toggle** method .
* So watch up here when I hit it once, it's added if I hit it again, it's removed .
* And finally, there's a replace method where I can specify a class and then replace it with a different class .
* So I can say **replace** and then the first item will be the class I want to replace .
* So pack prop, comma and then the second item is the class I wanted to replace it with .
* So again, watch the item up here .
* Graphical user interface, text

  Description automatically generated with medium confidence
* I hit return, pack prop goes away and in place we have new class .
* Here's the bottom line .
* ***If you want to add, remove, toggle or replace classes you should always use the class list property, and use the methods that come with it .***
* If **on the other hand, you're looking for a string and you want just the output of all the classes to do something else with, you can use the class name property .**
* So in almost every case you want to use class list .