* - [Instructor] Accessing an entire object is useful in some cases, but in other cases you need access to individual properties and methods within that object .
* There are **two ways of accessing object properties**,
* ***dot notation and bracket notation .***
* Let's look at **dot notation** first .
* I want to output the value of just the pocket number property .
* To do that, I'll first console logout, the pocketNum value:, then I'll use dot notation by first accessing the backpack object, then adding a dot, and then adding the property I want, in this case pocket number .
* And you can see when I add the dot, VS Code automatically says, "Hey I see you're using dot notation .

Graphical user interface, text, application

Description automatically generated

* What is it you want?" And it suggests to me pocket number .
* Save, check it in the console, and you see now we have the full object, and then we have just a pocket number value .

Graphical user interface, text, application, email

Description automatically generated

* Dot notation is called dot notation because you literally use a dot to separate the different properties .
* So here we're digging one level in .
* So we're grabbing the backpack object, and then saying dot, just the property name .
* We can also use it to dig further into the object .
* If you look at the object here, you'll see the strapLength property has its own object inside it .
* with the values left and right .
* So if I wanted to output just the left strapLength value I can console logout, let's say, Strap length :L and then I say backpack, that's the object, then I want the strapLength property, and then inside the strapLength property, I want the left property .

Text

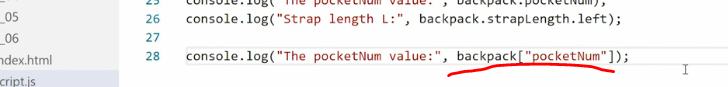
Description automatically generated

* Save .
* And now we have that value as well, Strap length L: 26 .

Graphical user interface, text, application

Description automatically generated

* Dot notation is the preferred way of accessing object properties because it's easy to read and understand .
* When I look at this, I immediately see, Oh, it's an object .
* And I'm looking for the strapLength property inside that object, and then the left property inside the strapLength property .
* So when you see a string like this with dots, you know you're accessing properties inside an object .
* *However, in some cases you need more control, either because you want to use a variable as the property name, or because the property name is non-standard for some reason .*
* For this, we have **bracket notation .**
* Let's look at how that works .
* So I'll copy this line here and then paste it in down here .
* Then to use bracket notation, I need to wrap the property name in quotation marks, because it's a string, and then square brackets .
* So if I highlight it, I can do quotation marks, and then I'll take the dot away, and put in the square bracket and whoops, put in a square bracket in its place like this .
* So as again see, the bracket notation is more clunky, but it also gives us more control .



* Save this output in the browser, and we have the pocketNum value again here .

Graphical user interface, text, application

Description automatically generated

* Now like I said, bracket notation allows us to do more advanced things .
* So let's say for example you want to pass the property value as a variable .
* You may have a function that outputs a specific value, and then you want to use that value .
* So for now, let's just say we want to set up a variable called query, and then we set that query to one of these values .
* We can set it to pocket number .

Graphical user interface, text, application

Description automatically generated

* Then, inside this bracket notation, I can use that query variable instead of a string .
* So what I'm doing now is saying, go and find the query variable, and then put the value of this variable inside here .
* So I need to put this in a string so that it works correctly .

Graphical user interface, application

Description automatically generated

* Save that, check it in the console, and it works the same way as before .
* What you see here can't be done using dot notation .
* If you place a variable inside dot notation, the script will simply break, because you're doing something incorrect .
* That's why we have this bracket notation .
* It gives us more control and allows us to do more things .
* ***There's also one additional situation where you might need to use bracket notation .***
* In JavaScript, the standard states that a property name can only contain letters, digits, dollar signs, and underscores .
* However, nothing actively prevents you or a piece of software from creating property names that break these conventions .
* So in theory, we could encounter a property name that starts with a digit, or it contains quotation marks or something else .
* When using dot notation, you can't access that property if it starts with a number, or uses a hyphen, or otherwise breaks the standard, because everything will break .
* In this circumstance, bracket notation comes to the rescue, because you are passing a quoted string that can be literally anything .
* So why would you ever come across properties with non-standard names? You'd never write them yourself, right? Well in JavaScript, we often work with generated data that has been transformed into JavaScripts and a JavaScript object from somewhere else .
* And these generators of that data do not always conform to JavaScript specifications, but thanks to bracket notation, JavaScript allows us to parse data even when it doesn't follow the rules .
* So for example, so if I have a property that breaks convention, let's say I have a property that has a hyphen in it, I can access that property using bracket notation, and nothing will break .



* So, in most cases, use dot notation because it's easy to understand .
* If you need to pass a variable into the property name, or you need to access a property that is somehow breaking convention, use bracket notation .