* - [Instructor] Let's take a look at how to access elements in the DOM using JavaScript .
* In the exercise files for this movie, I've updated our examples so we have some actual content to work with .
* I've added some basic HTML and a style sheet in index .
* html and I've also expanded the HTML output in the template literal a little bit to make it possible to style it .
* I've added some classes .
* I've wrapped spans around the values, and I've also added an image at the top here .
* Looking at the project site in the browser you'll see, it looks quite different but there's nothing new here in terms of JavaScript .
* I've just expanded on what we've covered so far in the course .
* All right, anytime you want to do something in the DOM ***you start by finding the element or elements you want to work with .***
* For this, we have two methods: **query selector and query selector all .**
* And *they both apply to the document object .*
* These methods use standard CSS queries to climb through the DOM tree and find the specified elements .
* So if you know how to target things with CSS you also know how to target things using JavaScript via these methods .
* That's what makes them so powerful .
* Let me show you how this works .
* Let's say I want to target the main that wraps around the main content in this document .
* I can go to my console, grab the document object, and then use the query selector method The query selector and query selector all methods, both take a CSS query as their parameter and it's put inside quotation marks so we can put in a proper string .
* So here, because I'm targeting an element I'll just type in main .
* And this immediately gives me the main element .

Graphical user interface, application, Word

Description automatically generated

* Hit return and you can see when I hover over this element it gets highlighted in the browser because this is the actual element the browser is showing me .
* And I can then dig into this element and see to the article, to figure, just to each one and so on .
* The query selector method has returned a DOM object to me, and it is a pure JavaScript object I can then work with just like any other JavaScript object .
* Now, in this example, I targeted an element .
* I can also use the same method to target a class .
* So again document, query selector, only this time, I'll say .
* maincontent .

Graphical user interface, text, application

Description automatically generated

* It'll give me the same element, only this time I got it using the class name instead .
* What's cool about this is I can use any CSS query here .
* So if we look at our example, you'll see we have a list here, an unordered list .
* Let's say I want just the last list item .
* In that case I'll say document, query selector, and then say I know that it's inside the main, so main, and then I want L I, last child .

Graphical user interface

Description automatically generated with medium confidence

* Hit return and it gives me just that last child of the list .
* Inquiry selector, all CSS selectors and all CSS selector combinations work just like they do in CSS .
* This is basically using CSS in JavaScript .

Graphical user interface, text, application, email

Description automatically generated

* The query selector method I've used so far returns the first elements that matches the query we're passing, so we get one element and one DOM object in return .
* If we want to get a list of objects let's say all the list items or some other array of objects, we use the query selector all method instead .

Graphical user interface, application, Teams

Description automatically generated

* This one returns to me a node list which looks like an array that has all of those objects .
* So let's say we want all the list items here .
* In that case, let me just clear this .
* In that case I say document, query selector, all and then we'll say main L I .

Graphical user interface, application

Description automatically generated

* Now I get a node list and inside the node list, I have each of these items .
* Now I can go through them one at a time to see what's going on .
* Now that I have a node list, I can use a different method like for each to go through that list and do something to each of the items .
* So again, we're skipping ahead a little bit .
* We'll cover this function in more detail later on in the course, but I want to give you a preview .
* So I'll say document, query selector, all, just like I did before and target main L I .
* Then I'll append the for each array method on this .
* And inside for each, I'll say for each of the items, I want to set the item style, background color equal to red .

Graphical user interface, text, application

Description automatically generated

* Hit return .
* And now you see that each of the items have gotten the red background color .
* Now, just to show you, I'm not cheating, I can now go in here and change this to say, just last child and then give the last child the color blue maybe .
* And here you see, I can target just that last item .
* So even though I have a list of items I can still target either each of them or go through all of them and do something to them using JavaScript .
* And this is what makes these two methods so powerful .
* I now have access to any element in the document object model and I can do whatever I want to any of those elements at any time using JavaScript .
* Before you continue, I urge you to play around with these two methods in the console right now .
* You've seen how they work and now you can use them to target all those elements .
* Remember, I've expanded the HTML in this example specifically so you can target different things .
* So try to find the element that has to class background underscore underscore color or all the spans in the project or something else and see if you can get to target those individual items .
* Like I said, query selector uses CSS queries .
* So if you know how to target things with CSS queries you know how to target them using JavaScript .