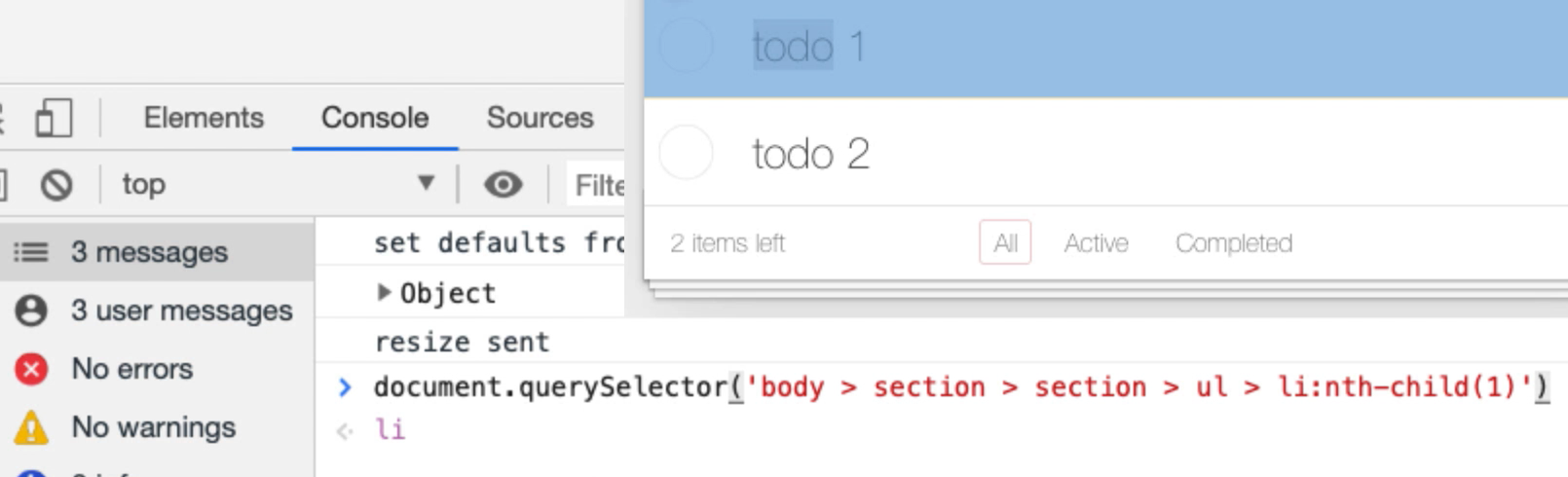
Now, one of the things we are going to do is we are going to use JavaScript to help us. When we do automated execution, we're going to use JavaScript within the console to do that.

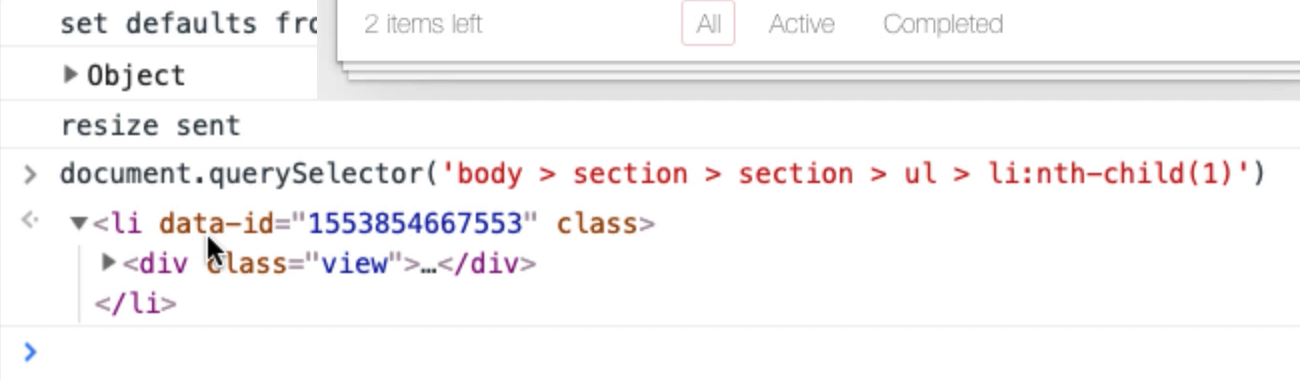
So, what happens if I do *Copy JS path*? I'm going to try and paste this in here and I find this isn't going to work because this is actually a JavaScript String. That's not going to find that on the page.

If I go into the console, which is the JavaScript console, and type this in,



Then you can see that by executing the JavaScript it's showing me what it's going to find.

If I click *Enter* here to run that, it has returned the web element on screen.



I just used JavaScript this document.querySelector with the CSS selector to locate the item.

**Now if I knew things about JavaScript, I could probably start manipulating this, so we need to learn some JavaScript commands for finding elements.**

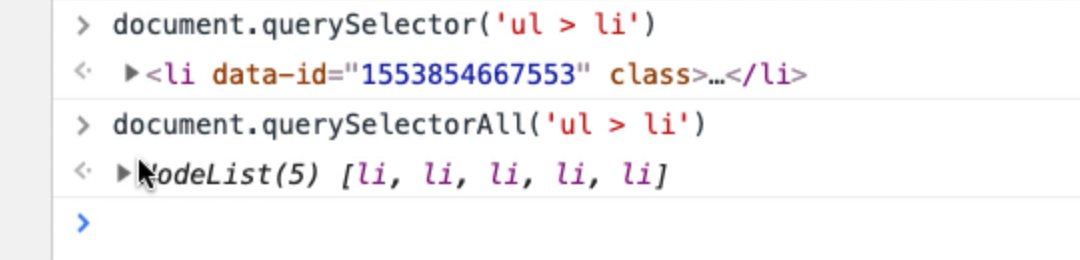
[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-queryselector)**Finding Elements - document.querySelector**

Now you see there that it's used document.querySelector — document.querySelector will return 1 item.

If I change this selection criteria here — document.querySelector(‘ul > li’) — and say just find me an unordered list and the list item beneath it, querySelector will return 1.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-queryselectorall)**Finding Elements - document.querySelectorAll**

There's another command — document.querySelectorAll — if I run that, it will return a list of all the query selectors that match.



And you can see now we've got 1, 2, 3, 4, 5. I thought there was only 2. So, what have we got here? These filters are also represented as list items in unordered list.

So, really, I do want to filter on the class todo-list.

Let me go back and do that. If I *press up*, it'll bring my command back. So, I want to say — document.querySelector(‘ul.todo-list > li’) — and there we go.



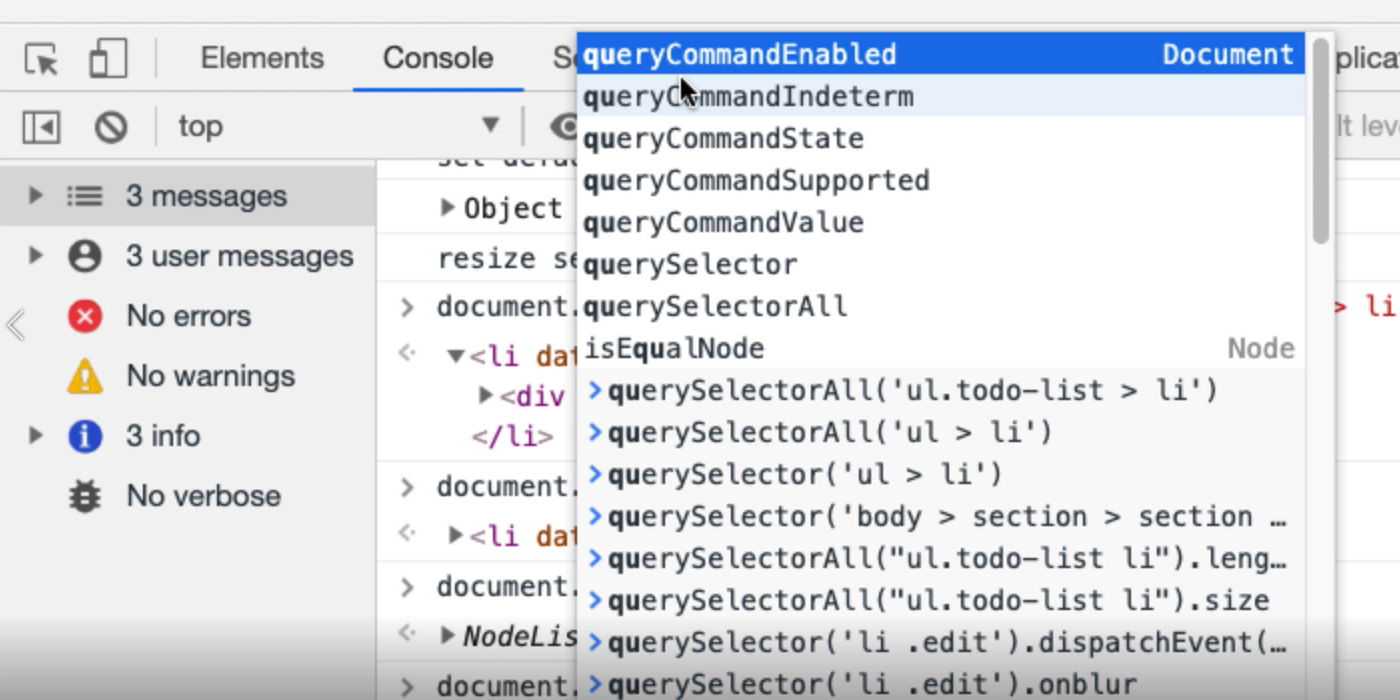
Now I know that I need my CSS selectors to be very specific because I have 2 unordered lists in here: the todo items and the filters.

I want to be able to make sure that when I click on something and it's an nth-child I'm clicking on the todo item and not the filter. So, I need to be very specific. This says find the ul with the class todo-list which has a li and it's selecting the list items.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#code-command-completion)**Code Command Completion**

Now the console has command completion.

If I do “document.querySelector”, you can see all the things I've written here in the past and it's giving me the commands that match that start with query.

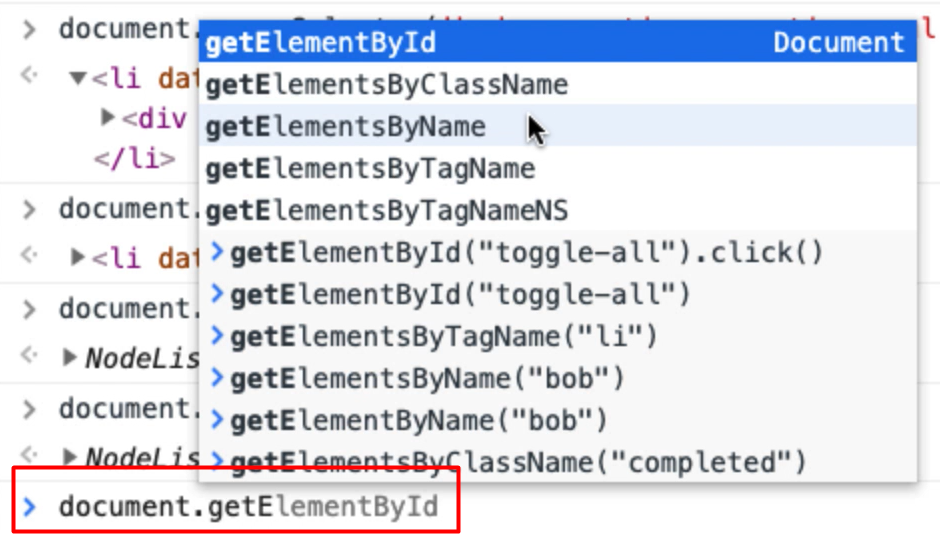


So, I've got code completion in the console to help me. If I can't quite remember a command, I can use code completion in the console to help.

So, querySelector will run a CSS selector and find the first item. querySelectorAll will run a CSS selector and find all the items that match.

**We have other ways of accessing elements if we want.**

There's a getElement set of commands — document.getElement



And here we can see we have getElementById, ByClassName, ByName and ByTag.

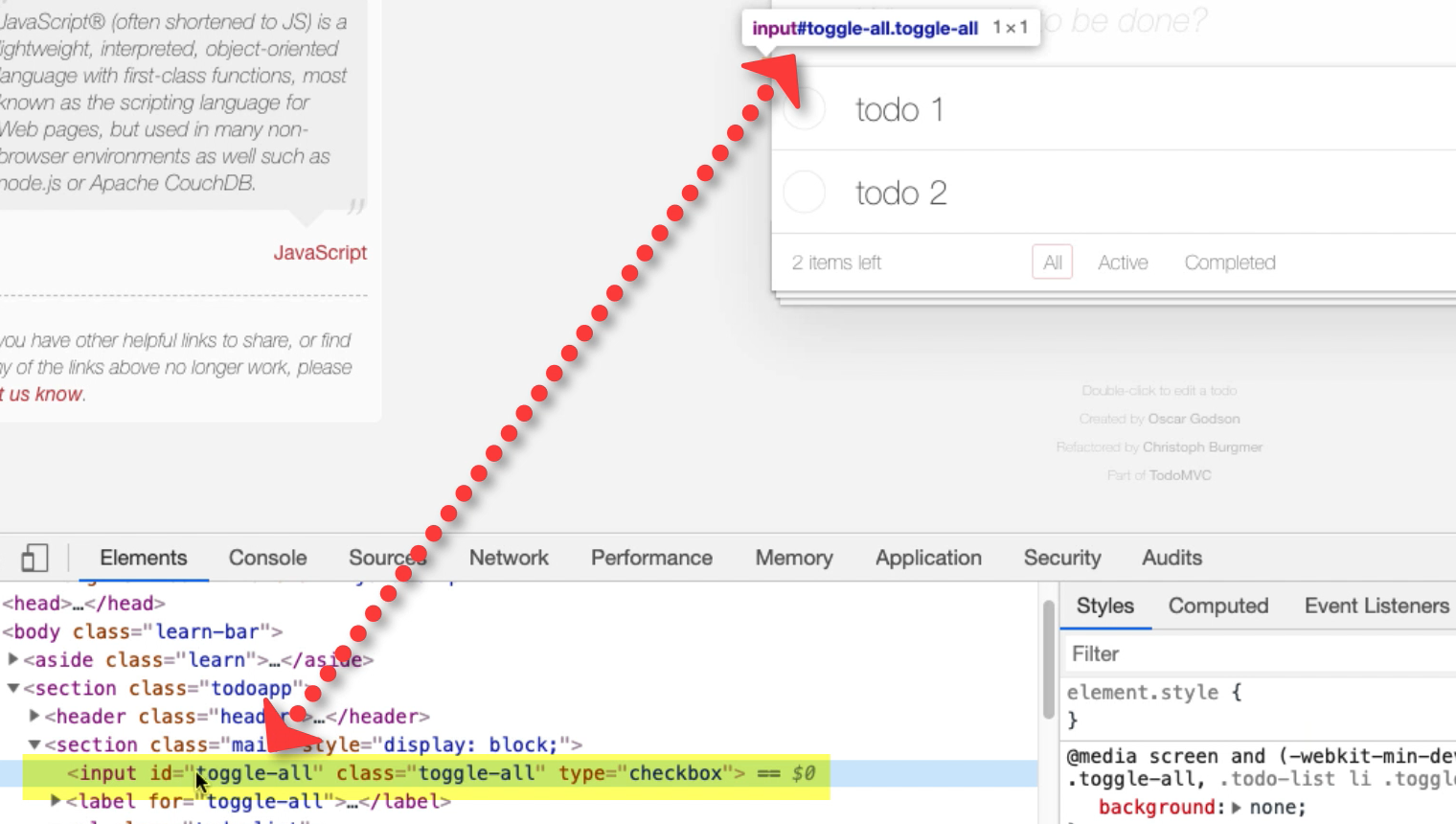
I can't remember what ByTagNameNS does . I never use that one. Let's look it up. So getElementByTagNameNS says “get me all the elements restricted to the senders of the specific element”. I never use that. I don't need to know everything, right?

One of the things I really demonstrate in here is I do quite a lot of automating through the console. I am still learning how to do this. We do not need to know and understand everything, right? So, don't feel that you do.

**All of these getElementsBy we can actually do exactly the same thing with CSS Selectors.**

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-getelementbyid)**Finding Elements - document.getElementById**

So, let's find something that has an Id, I think the “toggle all” has an Id here.



Toggle all, input with an Id, toggle-all.

So, if I take this, I go in here, I say: document.getElementById("toggle-all")



And it's found the toggle-all, right? That's that input up there.

**If in theory, if I knew how to click on something, I could find the element and click on it.**

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-getelementsbyclassname)**Finding Elements - document.getElementsByClassName**

So, class name — this toggle-all also has the class name. The input buttons have a class calledtoggle.

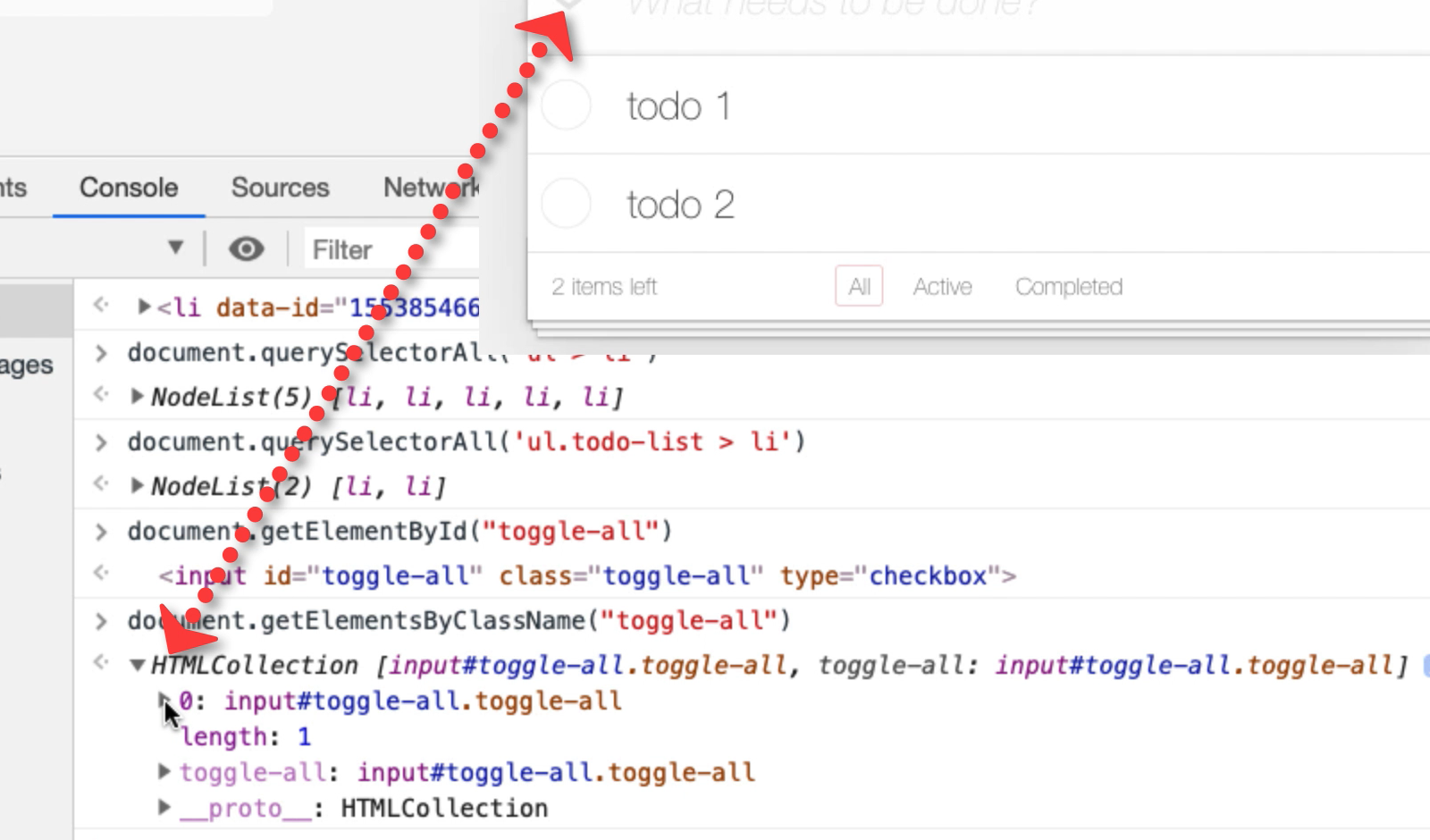
Let's experiment with that.

**Useful Tip about Code Completion with Typos**

So rather than getElementById, I'm going to say “getElementByClassName”… Now you can see I haven't gotten any code completion at this point because the code completion only clicks in when we've got the right command and that's a hint to me that I was typing that in incorrectly. Because I'd missed out the “Elements”, which is what the ByClassName is cause ByClassName can match multiple things. So, get in the habit of looking for cues when you're typing this in so that you don't make simple mistakes.

I want: document.getElementsByClassName("toggle")

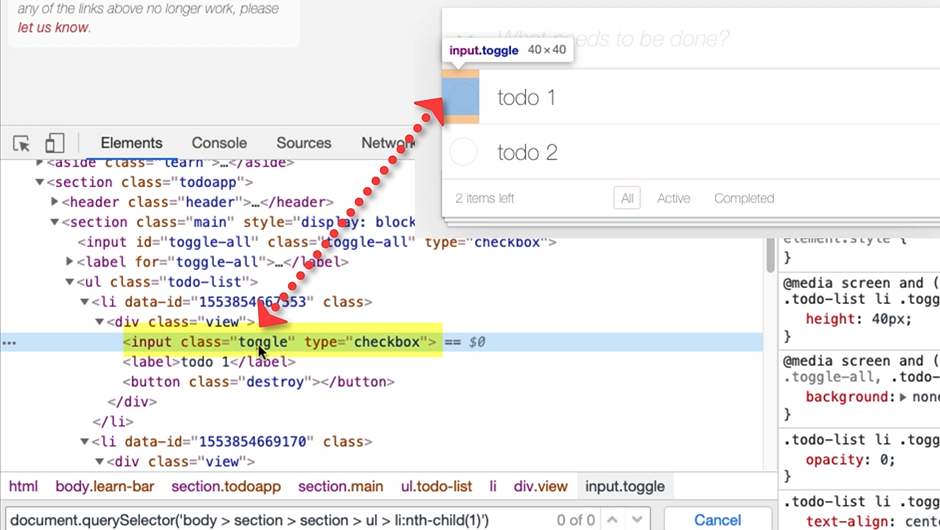
This says getElementsbyClassName, so this would find all the elements that have got class name so I'm expecting to see a collection coming back which has got 1 thing.



Yep, there we go. Which is the toggle-all.

Now the toggles, I'm expecting to see 2 which is one for each of these todo items because each of these to do items has got a class toggle. Let me inspect this.

There we go.



That input field there which matches the right button that I click on to make it complete has got class toggle.

So, we've seen getById, getElementById, getElementsByClassName, what else have we got?

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-getelementsbyname)**Finding Elements - document.getElementsByName**

Now what names do we have in here?

I don't really think that there are any names in this app, so I'm just going to show you a quick cheat.

What I'm going to do is I'm going to inspect this todo. Now in the DOM view, not only can we view the DOM, we can amend it. So I'm going to say this label has a name equal, I'll call it Bob.

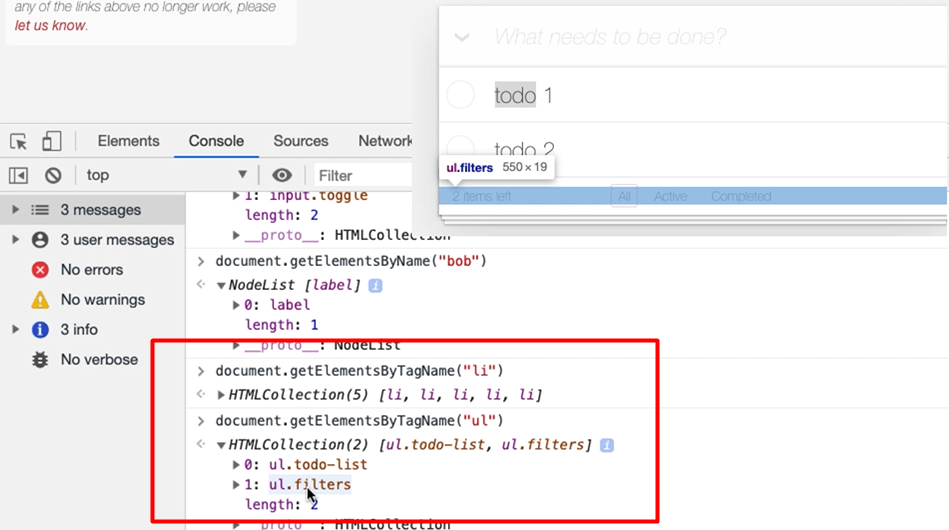
So now, if I go into my console and I say: document.getElementsByName(“bob”)

There we go. It's found that node, the label that we added the name to.

Now we can manipulate the DOM, we can change the actual application. That can be useful when you're doing technical testing, but I've only just done that there to represent, to demonstrate the getElementsByName command.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter3.2.html#finding-elements-document-getelementsbytagname)**Finding Elements - document.getElementsByTagName**

We can also getElementsByTagName.



I know that there's a bunch of list items in there.

document.getElementsByTagName("li") — There we go, there's 5.

I know there's a bunch of unordered lists.

document.getElementsByTagName("ul") — There we go, 2 have returned which is the todo-list and the filters.

**So, you can see that we can use these commands to return elements from the DOM that we can then manipulate and we're going to learn how to do that in just a second.**

**Resources**

* [Mozilla Developer Network](https://developer.mozilla.org/en-US/docs/Web)
* [W3 Schools](https://www.w3schools.com/)