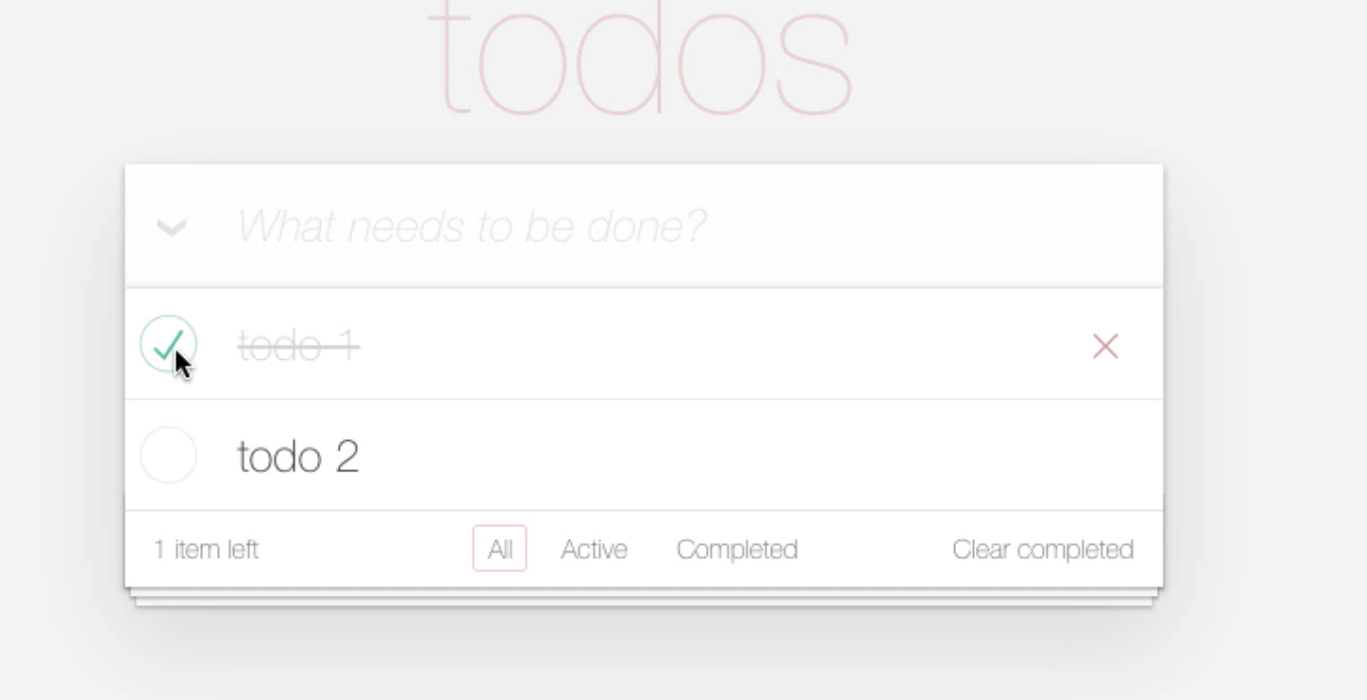
So, we've seen the theory for how to select things on screen; now it's time for us to figure out what we do with that, because we want to interact with these elements.

We want to be able to do things like simulating the "toggle all", deleting a todo, toggling a todo as to complete or uncomplete, and clearing all the completed todos. Filtering the todos, creating a todo, and amending a todo.

And we want to be able to do all that from code, within the console, because then that opens up possibilities for us for creating test data, for trimming down parts of the test data, but not all. Simulating other users interacting at the same time as us.

We want to be able to have the option for doing all these things without introducing other tools into our equation, just by doing everything form the browser.

One thing we have to understand is that when we're interacting with the application manually, we are clicking a mouse button, then the browser interprets that on the page somewhere and issues a click event and then JavaScript captures the click event and does something.

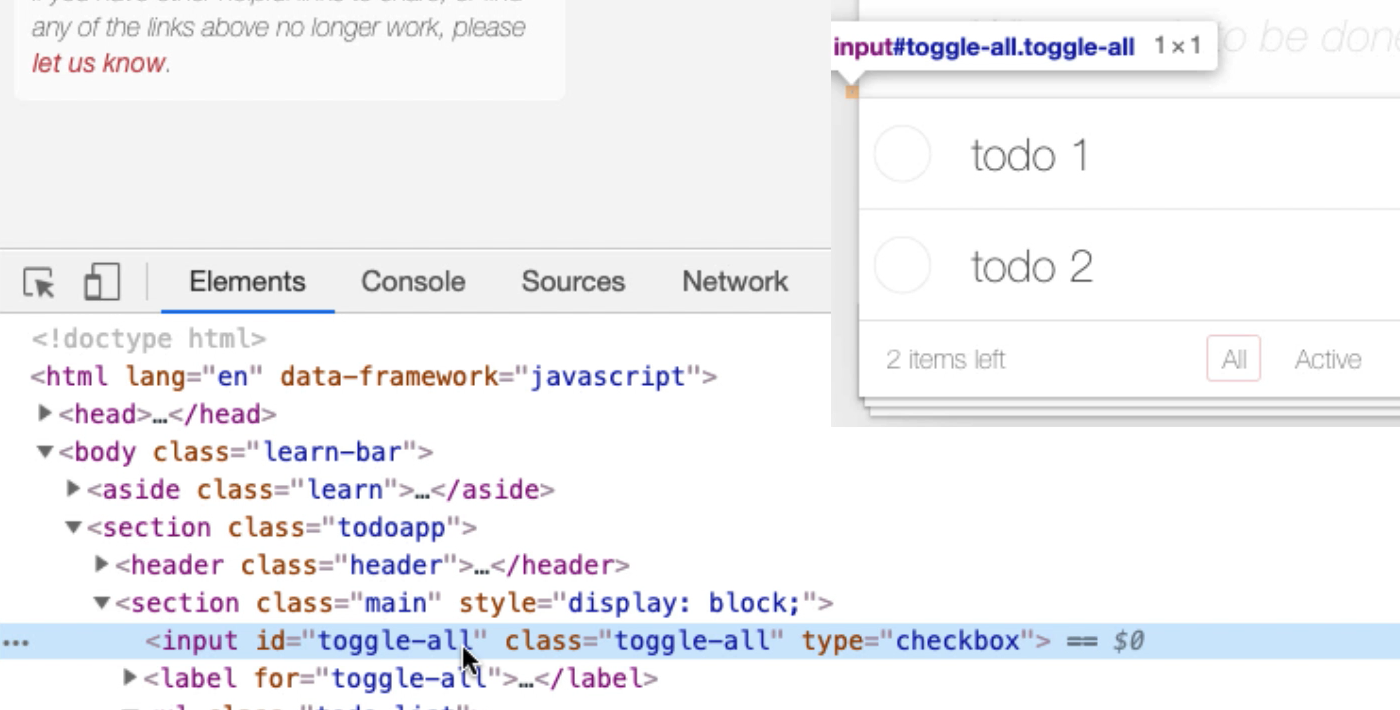


This a "click" event.

**Some of the events in JavaScript are very simple to simulate from the console, so let me show you.**

So, I first of all, want to be able to ... let's work with the toggle all. That's very simple, "toggle all," has an Id, so let's do that.

I'm going to *Inspect* this, and I'm going to have the application itself help me.



I want to find this input. So, I'm just going to say — Copy > Copy JS path — because that way, when I come into the console ... So, I'll paste that into the console.

And you can see here — document.querySelector(‘#toggle-all’) — it's found the input that I want. I want to issue a click event there.

Now, I can do that very simply by typing, "click":

document.querySelector(‘#toggle-all’).click()

We can't do this with all events, it'd be great if we could do this with all events. I can't do this with double-click, there's a whole bunch of things that I can't, but click, I can. So, click, there we go.

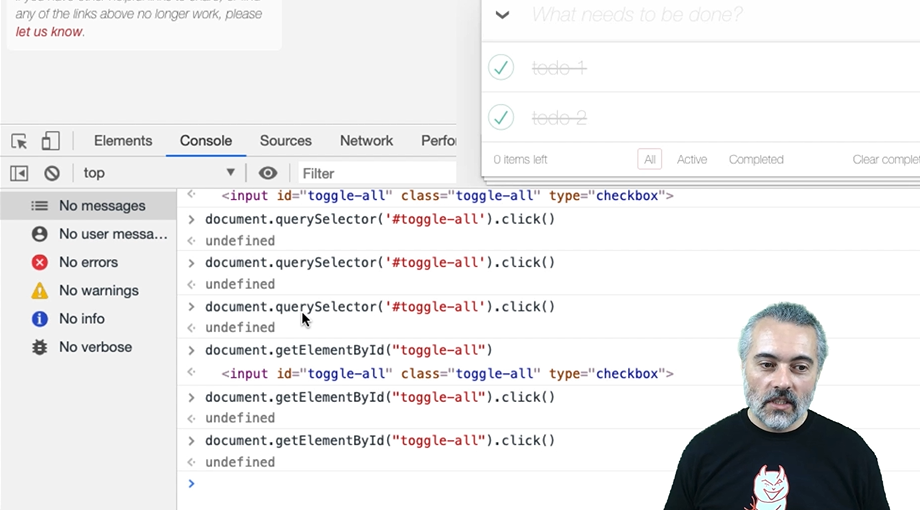
I have just, from the JavaScript console, selected all the items there. I simulated the click event and you can see I used the querySelector for the Id, "toggle-all."

I could've done that with a getElement:

document.getElementById(“toggle-all”)

Does it find it? Yes, it does.

document.getElementById(“toggle-all”).click() — There we go.



So, you can see how there's also the different ways of accessing the element: getElementById because that has an Id, or just the very general querySelector.

If you can learn how to write CSS queries, all you will need to find things on the page is querySelector or querySelectorAll.

Sometimes, using things like getElementById or getElementsByClass can help make your code a little bit easier for other people to read initially, but querySelector or querySelectorAll will do everything that you need.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.1.html#interacting-with-app-selecting-an-item)**Interacting with App – Selecting an Item**

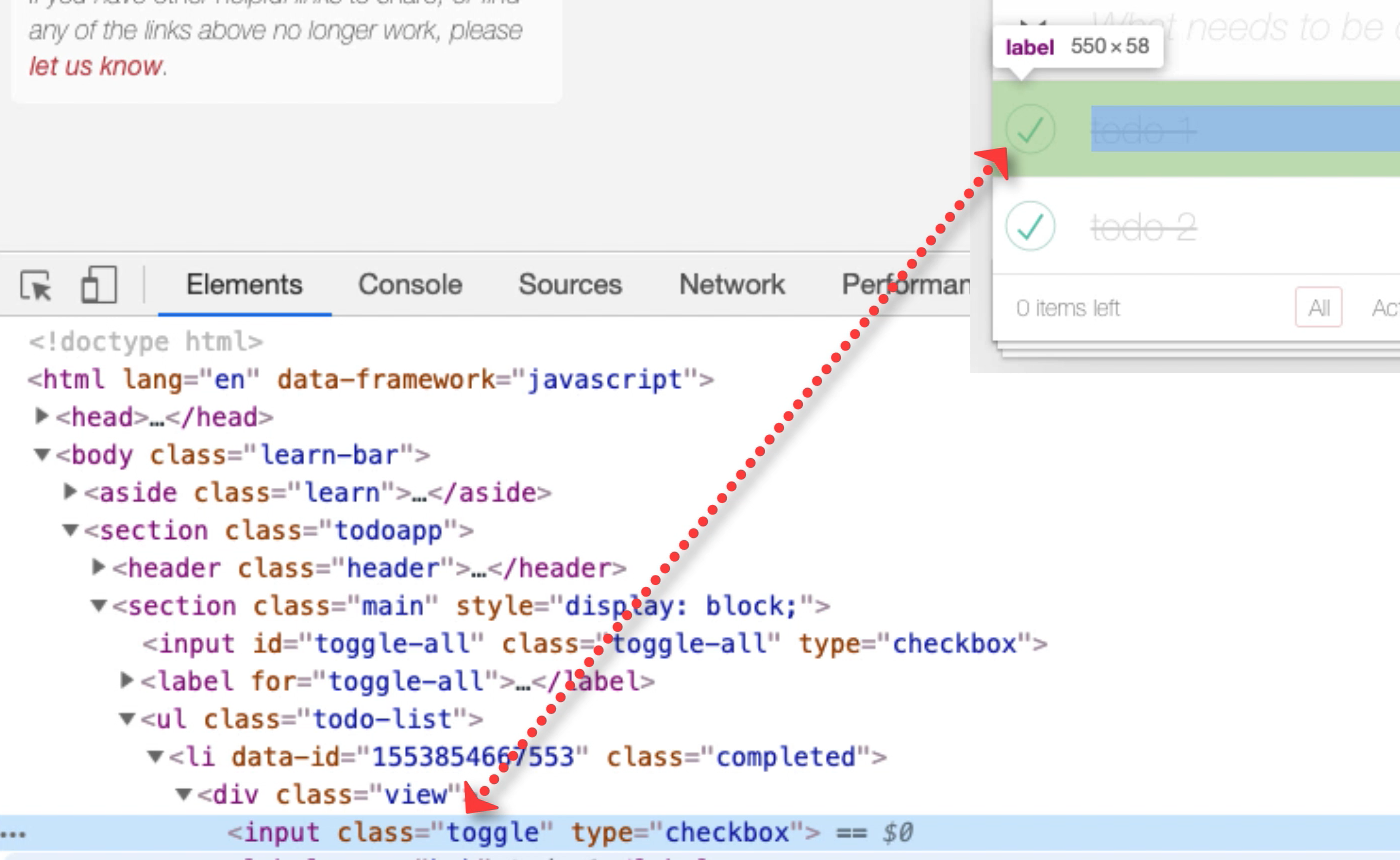
Let's try and select an item.

We've selected this here [todo 1 & todo 2], so I'm going to *Inspect* this.

**Useful Tip**

What I tend to do when I'm testing is I will have some way of capturing the work that I'm doing. That is very useful when I'm doing "query selectors" like this because then I'll just take that code and I'll paste it into a file so that I remember what it is that I've done. And then later on I can just copy and paste code from here and reuse it back into the console. That's just a tip. So, I'm going to try and, as I type in the commands here, remember to copy and paste it in here so that I can reuse it later on.

The next thing we want to be able to do is to toggle these, and that is using the "toggle" checkbox, so I need to be able to find that, "toggle checkbox."



So, let me do a *Copy JS path*.

There we go: document.querySelector(‘body > section > section > ul > li:nth-child(1) > div > input’)

And then I'll simplify that, because I know that the ul is actually “todo-list”: document.querySelector(‘ul.todo-list> li:nth-child(1) > div > input’)

There we go, it's found it.

What I really want is the input “toggle," so I'll say:

document.querySelector(‘ul.todo-list> li:nth-child(1) > input.toggle’)

That finds it, and then I don't really need this, "div," in the middle. So, I've simplified the CSS.

Now I have the ability to say I want the second one:

document.querySelector(‘ul.todo-list> li:nth-child(2) > input.toggle’)

So that's fairly good query for selecting that.

What I want to do, then, is click on it.

document.querySelector(‘ul.todo-list> li:nth-child(1) > input.toggle’).click()

And there we go.

Now I have the ability to click on and toggle any of the todos in that list. That's a very useful code snippet, so I want to capture that.

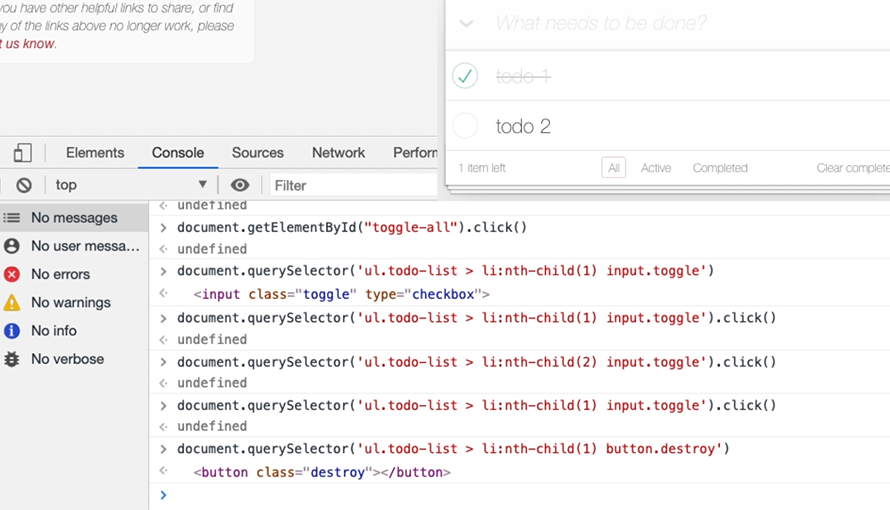
[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.1.html#interacting-with-app-deleting-items)**Interacting with App – Deleting Items**

Let's look at how we can delete items.

We know that there's this "destroy” button, so I could just say:

document.querySelector(‘ul.todo-list > li:nth-child(1) button.destroy’)

Does that find it?



There it is, and it isn’t enabled.

So, I'm wondering, is that a problem? Because when I click here, in order to click on the button, I have to hover. Here, I haven't hovered anything.

Let me try this and see if it works:

document.querySelector(‘ul.todo-list li:nth-child(2) button.destroy’).click()`

It does.

So, I can delete things, but I have to be aware that I'm not simulating the full user experience here. What I'm doing is I'm triggering the event to delete the todo, but I'm not necessarily doing the hover. Now I don't see that as a big risk because the hover is controlled by a CSS. We looked at that earlier.

**We always have to be aware of when we're automating, are we doing things differently than the user?**

If we are, that might be expedient in our automated execution, but that gap between what the user actually does and what we've automated might be a risk. So, here I'm looking for expediency in terms of automating to help me build up triggering the functionality.

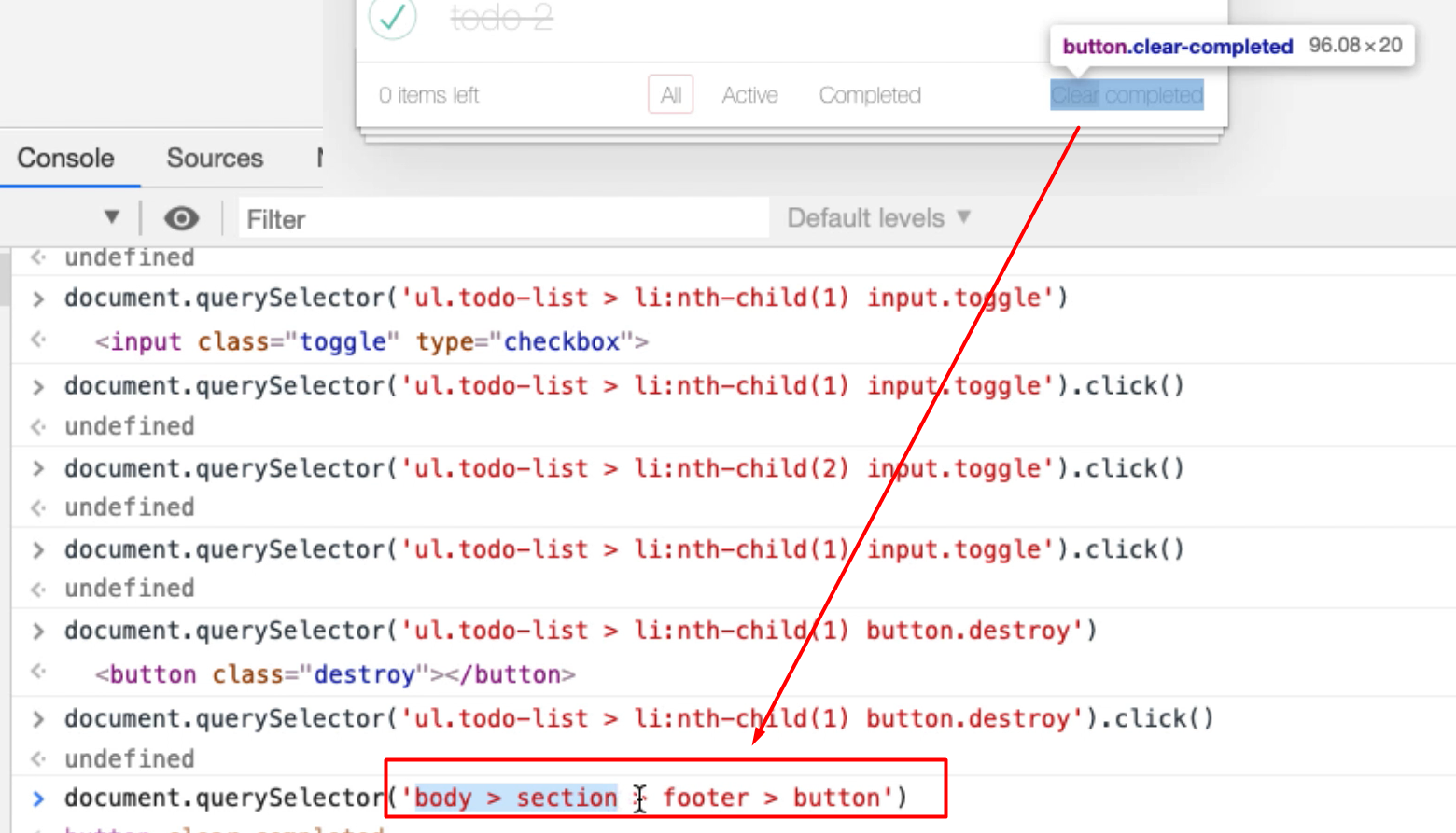
But if I was actually testing this, I'd want to make sure I'd bridged this gap and not rely completely on me doing a click here when I haven't done a hover first. Because that might impact the user experience, if it never, ever shows, that would impact the user experience because they would never be able to click that button.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.1.html#interacting-with-app-clearing-items)**Interacting with App – Clearing Items**

Okay, so, if I complete this, let's have a look at the "Clear Completed."

This is a button with the class clear-completed and there's only 1 of those.

If I \_Copy JS path \_in the console ... so, that's just picked a button.



That's not a very good selector as far as I'm concerned.

So, I'm going to say:

document.querySelector(‘button.clear-completed’)

Does that match it? Yes, it does. That's a much better selector for me. It finds the right one.

Check it comes back, then I want to trigger a "click" event on that:

document.querySelector(‘button.clear-completed’).click()

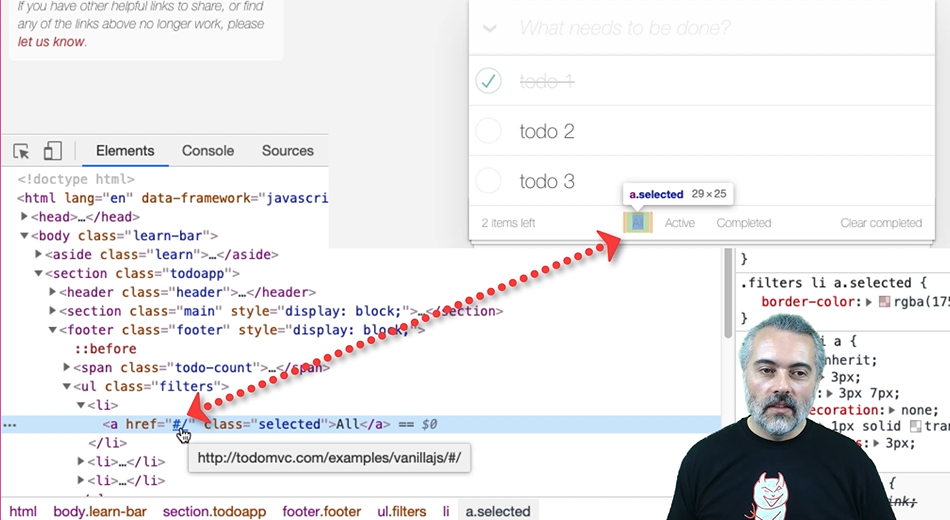
Okay, that works, too. So, the click function that I'm calling there is doing a pretty good job of helping me do my basic functionality.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.1.html#interacting-with-app-filtering-items)**Interacting with App – Filtering Items**

How can I filter items?

Let me create some todos. I put my todo items into a particular set of states, so we got 3 todos here: 1 completed, 2 active. There's the 2 active. There's the 1 completed.

How can I click on these filters? Let's have a look.



This has got an href so it looks almost as though it's going to reload the page, but it doesn't.

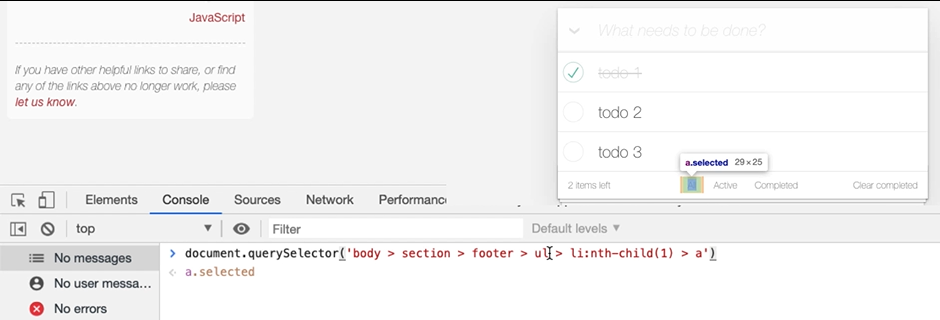
**Now one of the things that's interesting in here that we haven't shown you yet is if I look at "event listeners" [EventListener].**

I can see what "event listeners" are on each of these tags. Now this has a hashchange event. I'm thinking that I can do this with a click event.

Let's look at both of those.

First, I'm going to start with a click event.

I think I am going to have to do a list item with a position to find this, so let's see what we get with a *Copy JS path*.



Let’s simplify that:

document.querySelector('ul.filters > li:nth-child(1) > a')

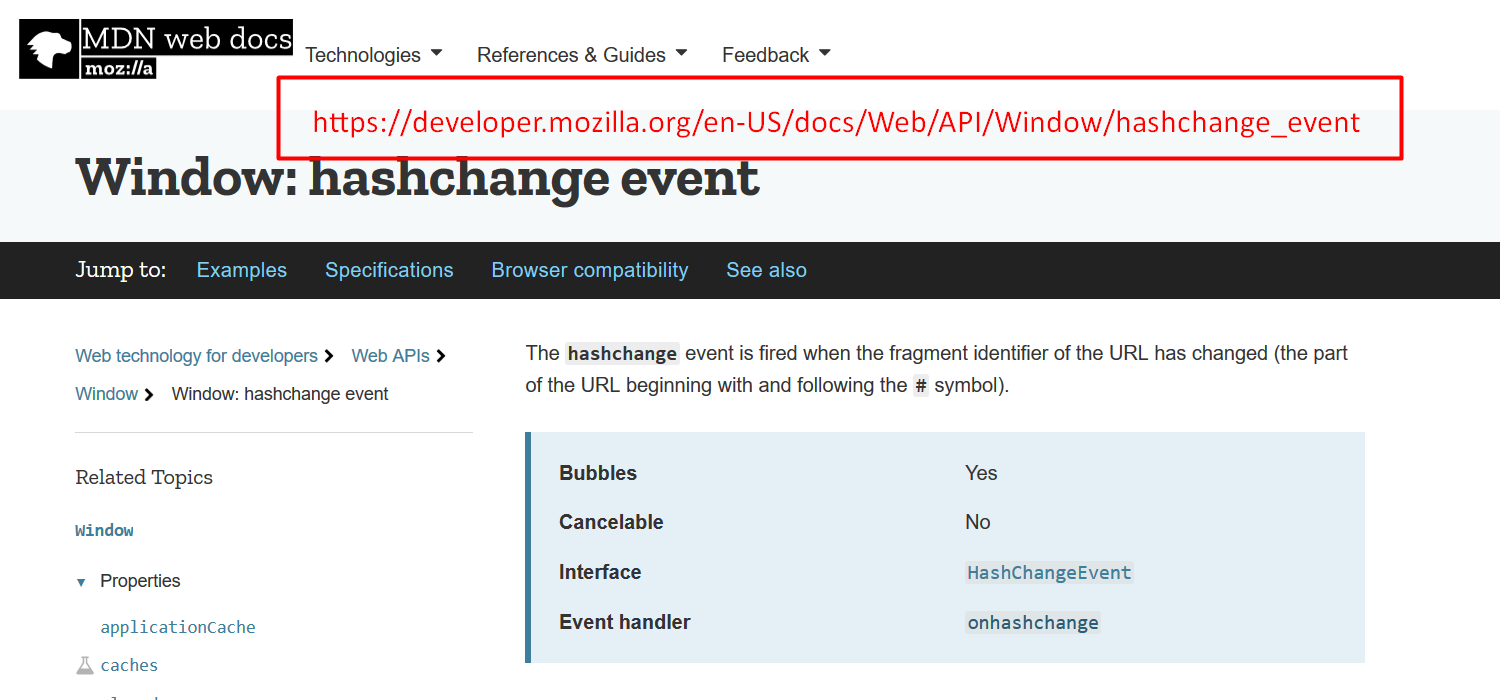
There we go. So, I can be specific and say, "for the filters, find the list item which is the nth child with the anchor and click on that."

* That's the first one, which is "show everything": document.querySelector('ul.filters > li:nth-child(1) > a').click()
* The second one is, "show me just the active ones": document.querySelector('ul.filters > li:nth-child(2) > a').click()
* Third one is, "show me the completed ones”: document.querySelector('ul.filters > li:nth-child(3) > a').click()

So, I can use the click just to trigger the filters, but what I'm interested in is, in the elements there.

We saw the hashchange event. Now, this is a good opportunity to learn a little bit more about JavaScript and the DOM and how things change. So, I'm going to have a look at the hashchange event.

On MDN., I'm going to just search for "hashchange" and there this is.



The hashchange event is fired when the fragment identifier to the URL. has changed.

So, we're learning a little bit about how browsers work here.

If I say location.hash and I bring this in in my console, so location.hash is completed: “#/completed”

What if I say, location.hash and then just, "slash?": location.hash = “/”

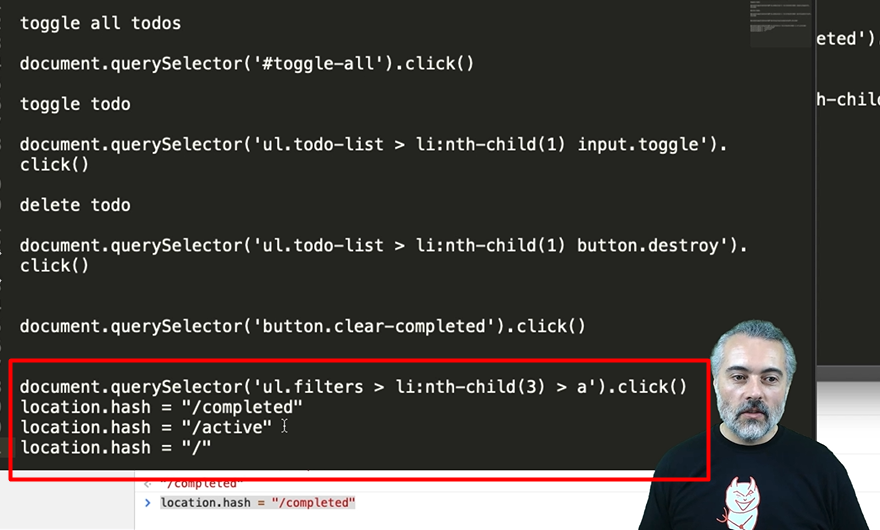
Then I'm triggering the hashchange event: “/”

So, I could say, "slash active": hashchange event: “/active”`

And we'd get all the active.

Completed: hashchange event: “/completed”`

So, now I have a choice. I can trigger the event directly with the hashchange or I can click on it to find out which one it is. Now, I don't have to make a decision about which one of these I'm going to standardize on yet, but knowing I have options is useful for me.



So, there I've just made a record of all the things I knew about the changing filters.

**Resources**

* [MDN EventListeners Documentation](https://developer.mozilla.org/en-US/docs/Web/API/EventListener)
* [MDN hashchangeDocumentation](https://developer.mozilla.org/en-US/docs/Web/API/Window/hashchange_event)