Okay, so it's time to learn a little bit more about JavaScript functions.

Now we've actually been using JavaScript functions. This click is a function, and it's on the web element that's returned when we iterate through the collection of query selectors.

We can write our own JavaScript functions.

If I take some of the code that I've already had written, what I'm going to do here is I'm going to create a *New snippet*. I'm going to call this “vanilla todos helpers”. And this is going to be a set of functions that I can use to help me work with this todos application. Fortunately, I already have all the code that I need in here.

**I'm going to start with a very simple function to click on something.**

We've already got the basic code here — find the query selector and click it.

So, if I did item.click() and if I somehow was able to pass in an item to a function. So, I'm going to say I want a function. And I want it to be called “clickItem”.

function clickItem(item){

item.click();

}

Now if I run this, I see an undefined, because item has not been defined.

I need to pass item in, and then I can call my clickItem function from here.

If I do — document.querySelector('#toggle-all')— to get the element, that finds the toggle all.

If I use my clickItem function and pass in that as the parameter — clickItem(document.querySelector('#toggle-all'))— it will click it.

**Now, from the console, I am able to interact with the application in a relatively readable format.**

So, I'm going to turn all the code that we've been writing up until now into functions to help me out.

I'm just going to show you a couple of these, and then you can study the source code when we finish. I'm going to show you the different variations.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-toggling-all-items)**Function for Toggling All Items**

There's the code, and that will toggle all — so I'm going to create a function called “toggleAll”.

function toggleAll() {

document.querySelector('#toggle-all').click();

}

It doesn't take any parameters and the body of the function is the code that we wrote. Let me format this.

And I have a tendency, again, to put in the semicolons just to make my code clear. Other JavaScript programmers won't do that.

If I run it, it won't do anything, because these are functions and they're not being called. What running it does is it puts it into memory so that I can now say toggleAll and it toggles all. I can now start writing relatively readable scripts at the command line.

**I'm just going to show you one thing with the clickItem.**

Here, if I turn this into .toggle and do a querySelectorAll, that will find all the toggles. Yes, we've seen that before. So, I'm going to turn this into a variable, as we've done before.

var toggles=document.querySelectorAll('.toggle')

**Now, I'm going to show you another way to do a forEach loop.**

Now that we've got the click item functionality, I can say —toggles.forEach(clickItem)— and this basically says, “for each item that's in that toggles array, call the clickItem function”.

Right, so that's a little bit easier than writing a for loop in some circumstances. I tend not to use that very much simply because I remember for loops and it's very easy for me. But because you are learning from scratch now, you now know that if you have a collection of things...

Let me get a collection of things. There we go.



I can forEach over that collection, and call a function for each item. Okay. So that's quite handy.

So, back to the functions.

I've now got a function to toggle all. What other code do we have in here?

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-selecting-items)**Function for Selecting Items**

I can select an item.

function selectItemX(x){

document.querySelector("ul.todo-list > li:nth-child(" + x + ") input.toggle").click()

}

Let me take this code. I'm going to create a function called “selectItemX”. I'll just put “x”in there, and “x” is going to represent a position in the list. So, to do this, this is a String. There's two ways of building strings in JavaScript. I can either use single quotes or I can use double quotes. For the purposes that we're using, it's doesn't really matter which one.

In order to join two strings together, I use a plus sign.

function selectItemX(x){

document.querySelector("ul.todo-list > li:nth-child(" + ") input.toggle").click()

}

If I take this, put this in the console, see what we get.



And the console is very good for just ad hoc running code when we're writing. So, you can see here that if I add this String to this String, I get another string. But what I want in there is a number. A value.

And I'm actually passing that value in there. So, I'm going to say “plus x”, and that will click it.

function selectItemX(x){

document.querySelector("ul.todo-list > li:nth-child(" + x + ") input.toggle").click()

}

This is the same code that we had here, but I've just parametrized this one in the function.

If I run this, so now in the console, if I do — selectItemX(1) — and choose item 1.

There we go. So now I can choose the first one in the list.

**So now any code that I write is going to be a little bit more readable.**

* My “delete item” is going to be exact the same as the toggle. I'm just going to put this code in, pass in a parameter.
* Clear is going to be a function that doesn't take a parameter.
* I'm going to write a function for each one of these filters — so I'll write a filter completed, filter active, filter all.
* When I create an item, I'm going to do “createTodo” and pass in the name variable.
* And when amending an item, I'm going to pass in a position and a name to amend it.

So, I'm just going to write that code.

Then when we come back, I'll walk you through what I've written. I'm basically just going to copy and paste the code from here and turn it into function.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-deleting-items)**Function for Deleting Items**

This is the deleteItemX — I'm passing in deleteItemX.

function deleteItemX(x){

document.querySelector('ul.todo-list > li:nth-child(' + x + ') button.destroy').click()

}

Let me just check if that works. There we go.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-clearing-items)**Function for Clearing Items**

I'm going to write a clearCompleted function that doesn't take any arguments.

So that's going to be a straight copy and paste in here. Now that one's giving me a syntax error because I've spelled “function” incorrectly. This is why the IDE is useful. It doesn't let you make mistakes.

There we go.

function clearCompleted(){

document.querySelector('button.clear-completed').click();

}

So, I better create another todo.

It would be handy to have a function that created todos. Might have to write one of those.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-filtering-items)**Function for Filtering Items**

I'm going to use the location.hash for my filtering.

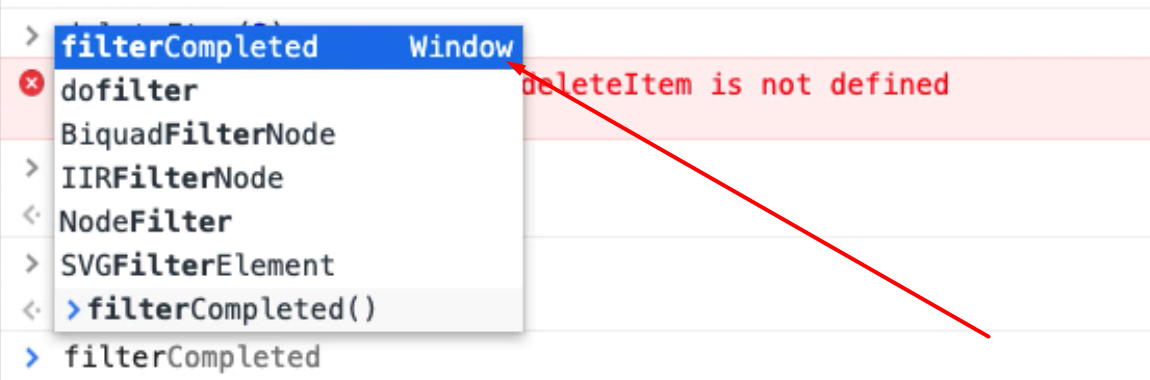
Function filterCompleted:

function filterCompleted(){

location.hash = "/completed";

}

Now, can you see here that it says window?



That means that this function is in the actual memory space that we're working on.

filterCompleted, we've done that. Good.

**Note about Copy/Pasting Code**

I'm going to do what you shouldn't really do; and copy and paste the code. Now, the reason you shouldn't do this is that it's very easy to make mistakes.

So I'm going to say filterAll and that's just the slash..

function filterAll(){

location.hash = "/";

}

Double check. There it is

And filterActive:

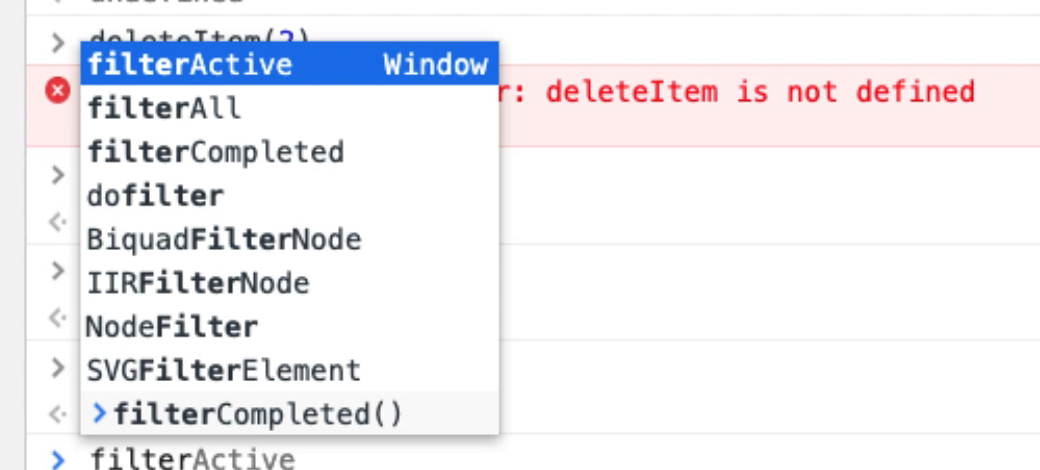
function filterActive(){

location.hash = "/active";

}

And that says active. There we go. Run that.

In the console, now I can filterActive.



And if I do tab, I get the code completion.

Let's make this a little bit more obvious. So, all should show me everything. Active should only show me those that are active. Completed should show me those that are completed. All should show me everything. All right.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html" \l "function-for-creating-items)**Function for Creating Items**

I want to be able to create something. I'm going to say function createTodo and I want to pass in the “name” parameter.

Let me copy and paste this in — instead of value “hello”, I want value “name”.

function createTodo(name){

document.querySelector('input.new-todo').value=name;

document.querySelector('input.new-todo').dispatchEvent(new Event('change',{'bubbles':true}));

}

Let's run that: Console > CreateTodo(“hello”)

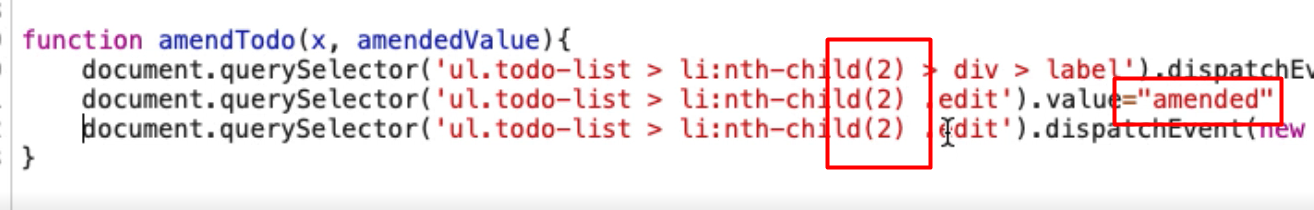
Right, now I can create a todo.

[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter8.1.html#function-for-amending-items)**Function for Amending Items**

I think last one is amend an item.

I want function amendTodo and I need to pass it a position and a name.

So, you can see why the snippets view is very useful. It would be really hard to do this kind of stuff in the console.



So x is the 2. Let's put that in. I've just edited that to concatenate the strings. But instead of 2, I want x.

Instead of “amended” I want amendedValue.

And because this is now code, I'm just putting the semicolons in.

function amendTodo(x, amendedValue){

document.querySelector('ul.todo-list > li:nth-child(' + x + ') > div > label').dispatchEvent(new Event('dblclick', {'bubbles':true}));

document.querySelector('ul.todo-list > li:nth-child(' + x + ') .edit').value=amendedValue;

document.querySelector('ul.todo-list > li:nth-child(' + x + ') .edit').dispatchEvent(new Event('blur'));

}

So now, if I run this, I should be able to amendTodo(1) (2, 3, 4, 5): amendTodo(5, “bob”)

There we go.

All the codes that I'd previously written to do stuff earlier on, I now have in a snippets; and I can run that and load that in here, which makes it a lot easier for me to interact with the application from the console.

So, if I'm doing testing, and I want to toggleAll, now it's pretty easy to click that. But I'm just setting this up so that when we write code to do things — like create lots of test data, interact with the application as a bot, to set up lots of todos and click lots of things — we can use the functions that we've just set up here in order to do them.

This is one of the probably most important things that you will learn here, because we're creating an abstraction layer that models the application, and this can support us in our testing.

I could create an item, a function for toggling every second item if I wanted to. That would be easy to create a function for.

But I'll leave that as an exercise for you.

**Quiz**

The quiz for this chapter can be found in section 8.2