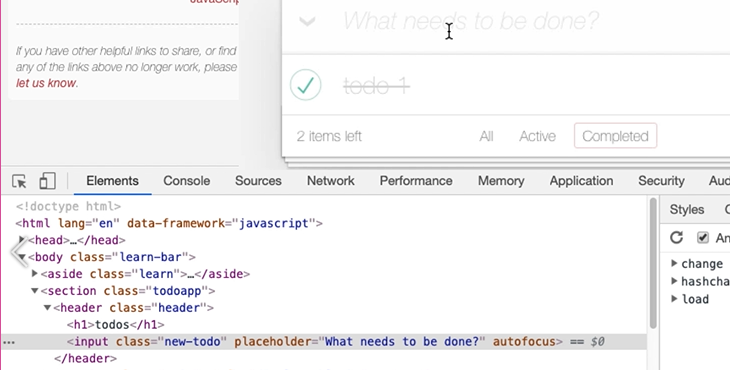
Creating todos — this is interesting.

If I *Inspect* here, I've got an input.



So, all I essentially have to do is find the input, put a new value in the input, and then...

Well, how do I trigger the completion of the input?

Let's have a look.

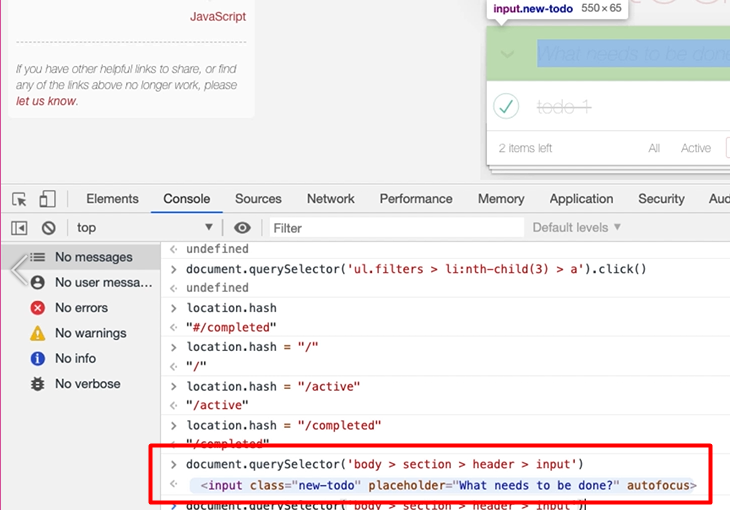
[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.2.html#interacting-with-app-creating-items)**Interacting with App – Creating Items**

Here in the input we've got hashchange. I probably don't want to do that.

There's a change event — input.new-todo — so I need to somehow trigger that.

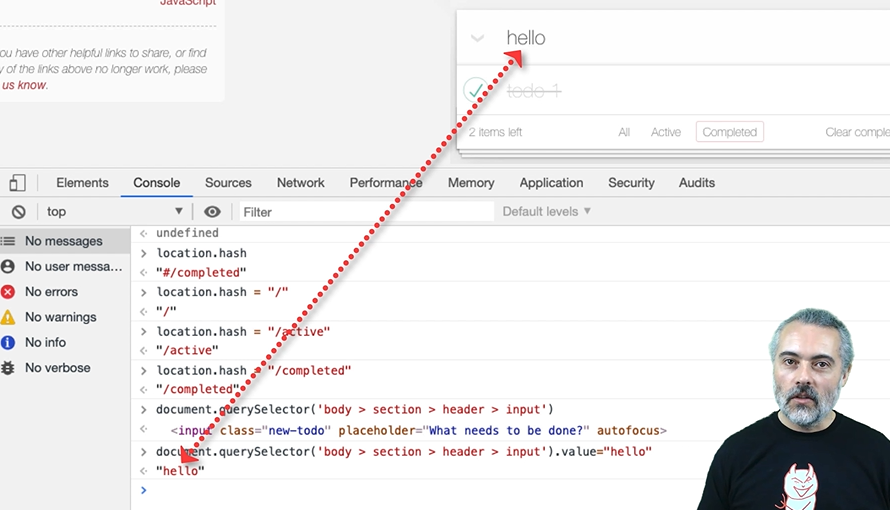
Let's first of all find the input. *Inspect* that, *Copy JS path* in the *Console*.

So, there's the input.



**With input fields the real magic is controlled with the value field.**

So, there we go: document.querySelector(body > section > header > input).value="hello"



I've just typed a value into that field. But it hasn't created a todo yet, because I haven't triggered the change event.

What if I just do change? Okay, so now we've got one of the events that we can't trigger just with a function.

**I need to actually learn how to trigger an event in JavaScript.**

To do that, I need to use the “dispatch event” command — the dispatchEvent function.

Then I need to create a new event to dispatch in there. So, it's going to be “new Event”. Then I have to create a change event. Then, I have to tell it what parameters I want to pass into that change event.

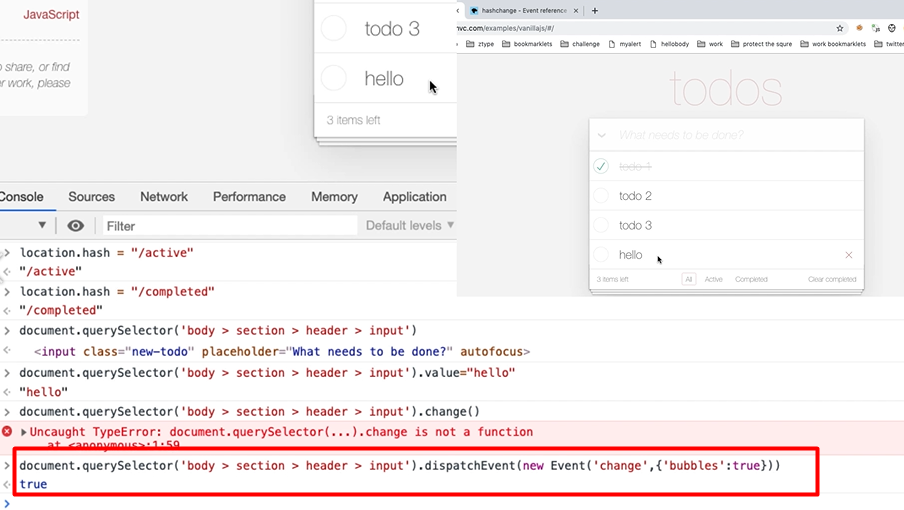
Now, for the change event, we could [look all this up in MDN](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/change_event). All I really want to say is fire the change event and make sure that it bubbles up.

**Event Bubbling**

What “[bubbles up](https://developer.mozilla.org/en-US/docs/Web/API/Event/bubbles)” means is it doesn't just get sent directly to the element. It will also be sent to anything up the hierarchy, which is what would happen if a user actually clicks on it or hits the return key.

So, we're going have a change event and say: document.querySelector(body > section > header > input).dispatchEvent(new Event('change', {'bubbles': true}))

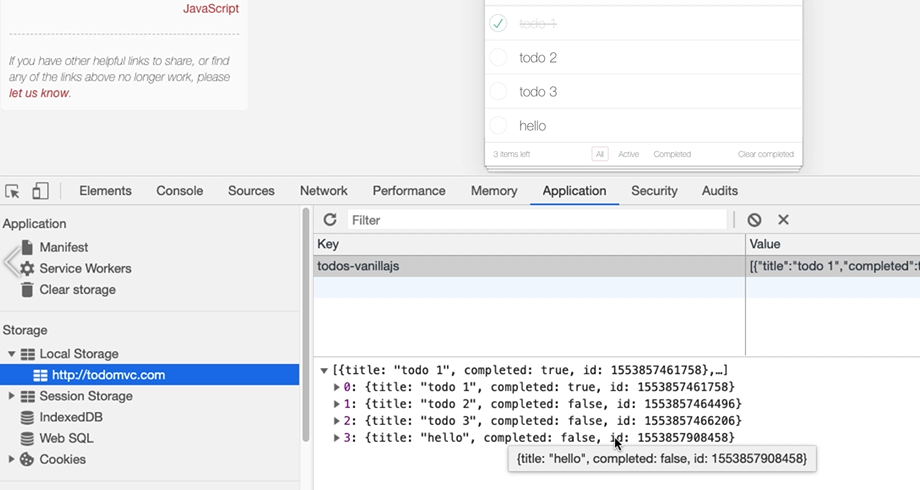
So that should be good enough. Let's see what happens now.



Why didn't that work? Because I'm in a filter.

So, there it is. There's the “hello”. And just to double check, I'm going to look at the data in here.

In the *Application* tab in *Local Storage*, there's the “todoMVC” and there's the “hello”.



I have to do 2 things to trigger this — I have to set the value and I have to dispatch the event.

Let me, now that I know how to do this, find a decent CSS selector.

If I *Inspect* this, it's the input with class new todo. Let me copy this.

So, if I say:

document.querySelector(".new-todo").value="hello"

I'm just going to try that out in the find here and it only matches one.

That is good enough for me.

So, I'm going to use that in here:

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

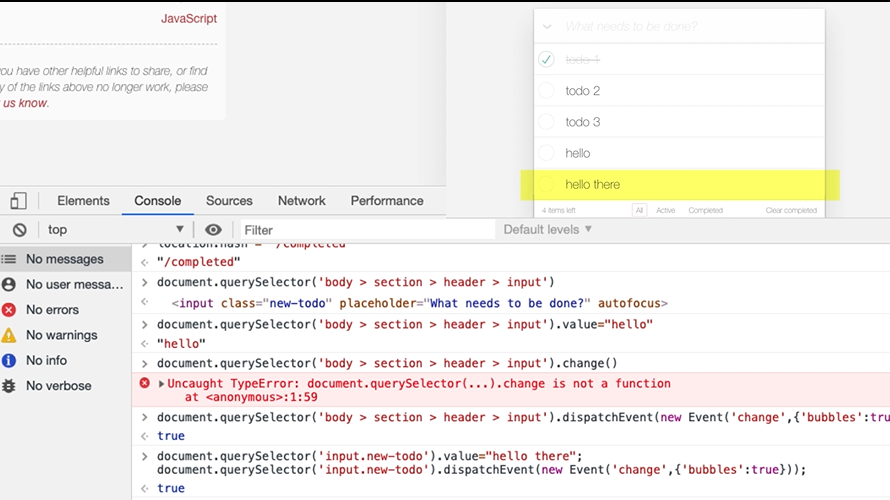
If I copy this into my scratch pad. Now later on, we're going to see some built-in functionality in Chrome that can take the place of the scratch pad, but just to start with, let's get everything working nice and simple.

**Callout-title**

Now, with JavaScript, you don't have to put semicolons on the end of the lines most of the time. But very often, I will do just to make my code clearer, because I kind of grown up with JavaScript for a while and I'm used to that.

If I copy both these lines into the console, paste, I can run both lines at the same time.

Let's just change the “hello” to “hello there”, because we don't have that.



There we go.

I've added a todo, so now I know how to do that.

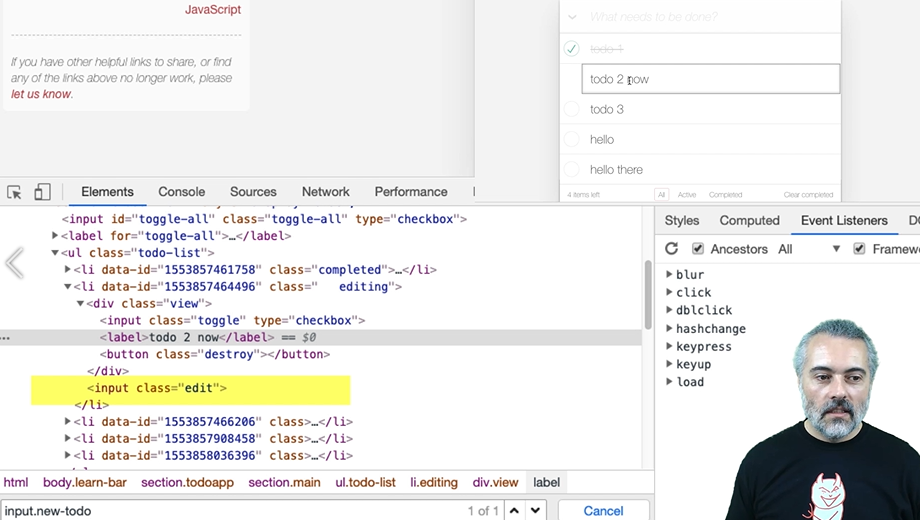
[#](https://testautomationu.applitools.com/automating-in-the-browser-using-javascript/chapter4.2.html#interacting-with-app-amending-items)**Interacting with App – Amending Items**

Now I think the last thing we have to figure out how to do is how to amend one of these todos.

So, to do that interactively, I don't just click on it. I double click, then I can change the value — “todo 2 now” — and hit return, then it changes.

Let's have a look at that in the elements. So, let me *Inspect* this.

There we go. Double click.



Now, did you see that?

In the DOM now, we have a new element that was added. So, I have to double click on the label, then there will be an input field, which I can then use. To change the value of an input field, it's just the .value again.

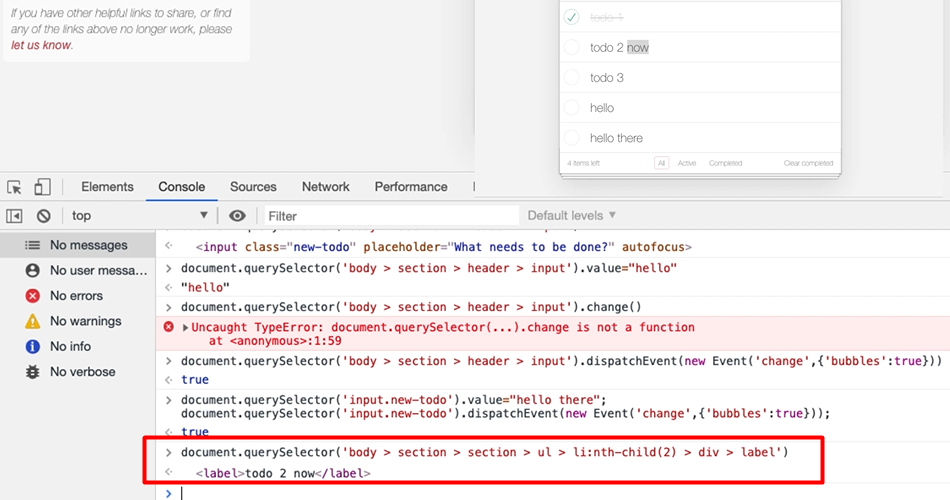
And then I have to ...

Now, this is one of the trickiest things we're going to do in here. There's a bunch of events and that I could potentially trigger here. And I think it's the blur that I'm interested in.

I'm going to trigger the blur event once I've changed the value. So, let's do this in chunks.

Let me find this, so *Inspect* it — I want the label.

*Copy JS path* into the *Console*. So that's found that.



I'm just going to trim this down: document.querySelector('ul.todo-list > li:nth-child(2) > div > label')

So, I need to double click.

Well, let's try and see if we can double click:

document.querySelector('ul.todo-list > li:nth-child(2) > div > label').dblclick()

Nope.

What about double click capital [.dblclick]? Nope.

**So, I can't, I don't have a function. I need to do the dispatchEvent trick.**

And, again, that's going to be bubbles true:

document.querySelector('ul.todo-list > li:nth-child(2) > div > label').dispatchEvent(new Event('dblclick', { 'bubbles': true }));

Let's try this. All right, so now we're in edit mode.

Then I need to find the edit.

I'm going to say, “list child dot edit” which has found it, and then I'll say:

document.querySelector(‘ul.todo-list > li:nth-child(2) .edit’) .value="amended"

We've changed the value, but it hasn't triggered the event that's going to cause it to persist yet.

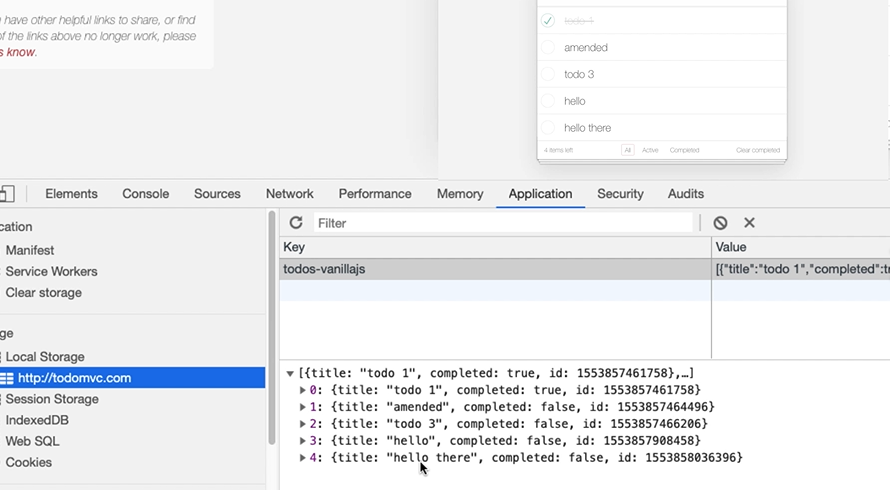
So then, in this field that we've just edited, I'm going to say:

document.querySelector(‘ul.todo-list > li:nth-child(2) .edit’) .dispatchEvent(new Event(‘blur’))

And that's it. I don't need to bubble this up. I don't need to have any other parameters in there.

Let's go.

All right. So now, hopefully in the application, amended.



So, we've simulated the amendment of a todo.

**Important Reminder**

Remember, we've simulated it. We didn't do proper double clicks. We didn't hit the return key. We've simulated the events that the application is monitoring in the back end to work with. So, again, if this was proper strategic automated execution, we'd have to be aware that there's a risk between how we have automated it and what the user will actually do.

Let me just keep a record of what I actually did.

Now, what you have seen there is the result of years of amending and manipulating applications online. An amendment is the hardest thing, probably, you have to do in the todo. But the clues that we used for this were all in the *Elements* tab. And a lot of the automating of applications from the console is about the clues that you're going to pick up here.

The events that you need to trigger from the eventListeners. Seeing the change in the DOM that happens as a result of doing things. This is the view that ties things together.

Then, in order for us to work out how to trigger the blur events, I would just type into a search engine “how to trigger blur event JavaScript”. Then, there'll be a bunch of answers that come back, and I have to figure out which one I'm actually going to use and copy and paste into the console to see exactly how this is going to work. And it would take time for me to read through the answers to figure out which one is actually going to be useful for me.

To learn this stuff, I just have to do lots of web searches, be very specific, look through the answers that come back, and try them out.

But what we're doing in this course is we're shortcutting your learning. When you need to trigger an event, you're looking for the new event, then it's pretty much how do I trigger this event?

An art to this is doing web searches that give you the information you need, but I'm just trying to shortcut as much as possible the techniques that I use and if you learn how to fire events like this, you'll probably be able to trigger most events.

**Resources**

* [MDN Event.bubbles Documentation](https://developer.mozilla.org/en-US/docs/Web/API/Event/bubbles)
* [MDN change event Documentation](https://developer.mozilla.org/en-US/docs/Web/API/HTMLElement/change_event)