When I am automating from the console with JavaScript, I have two main ways of triggering code running in the background.

I've got setTimeout which runs a function, or a block of code, ***after*** a set amount of time — you saw that when we were creating todos in the for loop.

And then I have another one called setInterval — setInterval runs a block of code, or a function, ***every*** “x” milliseconds.

* setTimeout says, "I will run this block of code in this number of time, and I'll do it once."
* setInterval keeps going. Every 500 milliseconds, if that's what we set it to, it keeps running the same block of code.

That could be incredibly useful for us.

This is used a lot in game programming to create the main loop, so it keeps looping around, checking the status, polling the keys, things like that.

**And we can use that for writing autonomous bots.**

Let me just show you how this works.

In the console here, when I write a setInterval, I always assign it to a variable, because if I don't, I can't stop it.

I'm going to say setInterval. Now I'm going to give it some code to run. I'll say it's the function; and this function will say something to the console. And I want to execute this every, let's say 1000 milliseconds.

var sayHello = setInterval(function(){console.log("hello")},1000);

So, every second it will write to the console, "hello."

There we go.



And you can see next to the text, "hello," a little number that's increasing.

That's because the console doesn't keep showing you the same log message, it just says, "You've seen this log message 13 times, 14 times, 15 times." And it just keeps going like that, so we know it's running.

**Now, if I went to stop it, I could refresh the page, or I can use the clearInterval command and I just say, clearInterval(sayHello).**

Very often my variable will say things like, createTodoBot or playTheGameBot because then I'm thinking in my head, it's a little bit of code that's running in the background doing its thing.

So, given that we used setTimeout to create 100 todos, I'm going to use setInterval to create 100 todos.

Let me just \_Control A > Control C \_[select all, copy & paste] > create a *New Snippet* > *Rename* this and call it “create 100 todos setInterval”.

With setTimeout, we had to use a for loop, because setTimeout runs once.

**With setInterval I'm not going to need the for loop because my code is going to iterate around.**

So really what I need is I need to say, is var botTodoCount and how many things have I created? I'm going to mark that zero.

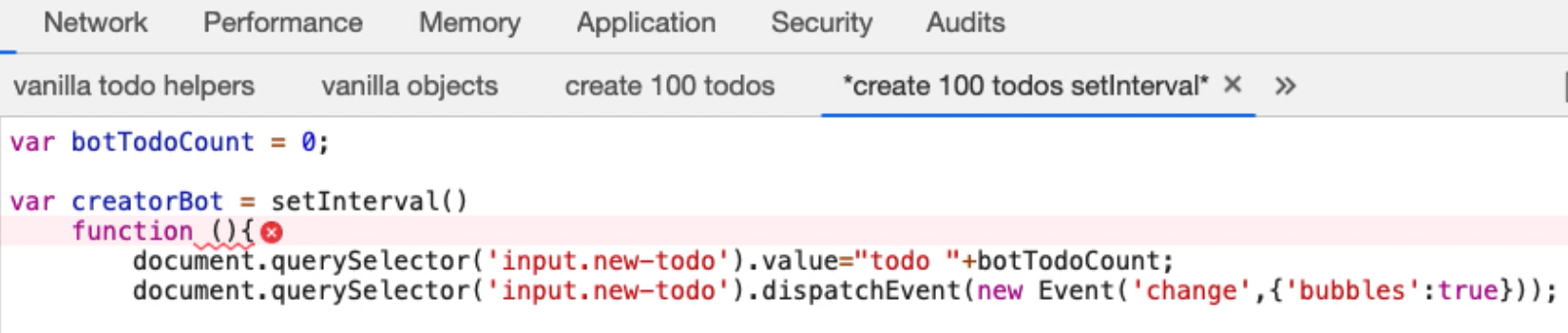
Now, I could count down from 100, but I'm going to leave that as an exercise for you.

Instead, I'm going to count up from 0. So, I'm going to say, var creatorBot, because I need a variable in order to set this, and this is going to be setInterval.

And then it's going to run the code.

So, I now have a setInterval, which is going to take a function. It's not going to take a “name” as a parameter because we're just going to use the variable here.

I will then create a todo, so I will say — "todo " +botTodoCount — that will be the name of the todo that I'm going to add.



Then we've got the dispatch event, so that will create it.

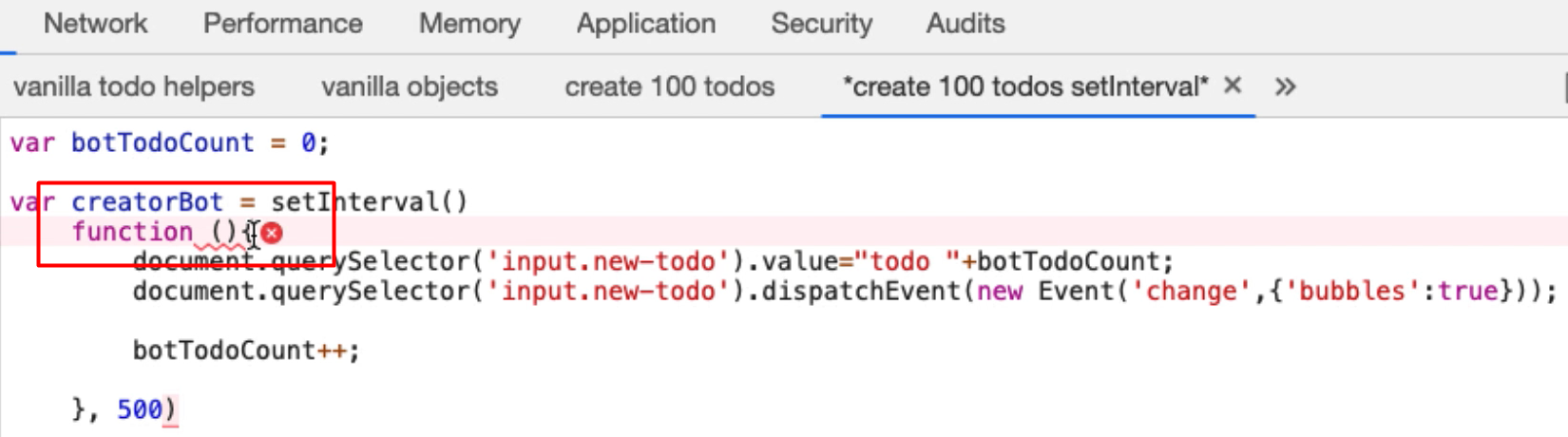
Then what I need to do is botTodoCount and then increment it.



Now, if I did this — x\*100, “todo ”+x [what we used for setTimeout]— it would essentially run forever.

So, my setInterval is going to run every, let's make it 500 milliseconds.

We've got a syntax error here because I closed the setInterval.



Let me format this correctly [removing the extra parenthesis].

What this is saying, is that my creatorBot is a setInterval — it's an interval that is going to run a function every 500 milliseconds.

And the function it’s going to run, is going to create a todo with the name “todo” and the current botTodoCount added to the back.

So, it will say — todo 0, todo 1, todo 2… — then it will increment the botTodoCount. Now, if I do this, it will just keep going forever.

I'm going to put a hard stop in place and say, "If botTodoCount is greater than or equal to 100, then I will clear the interval."



So, it's got a self-destruct mechanism in here.

**This probably looks a little bit more complicated than the for loop with a setTimeout, but this is actually much more flexible for us because I can write whatever code I want in here.**

In reality, I could just put this like this because I always consider the function as part of the setInterval creation.

And then I always have the end point in that separate.

var botTodoCount = 0;

var creatorBot = setInterval(

function(){

document.querySelector('input.new-todo').value="todo " +botTodoCount;

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

botTodoCount++;

if(botTodoCount>=100){

clearInterval(creatorBot);

}

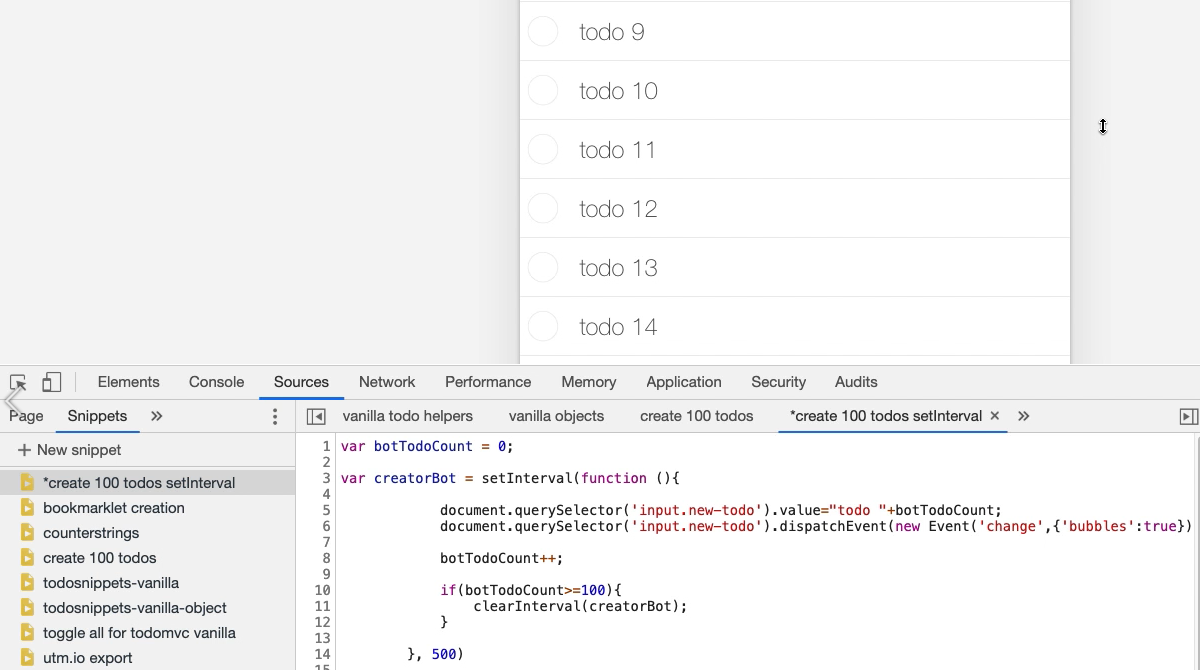
}, 500);

Now this is relatively readable.

So, this should work. This should create a setInterval that will create 100 todos.

Let me get in the application again. Let me just clear down the app, refresh it, there we go.

Back to my snippet with the todo setInterval bot, run the code, and there we go.



So now every 500 milliseconds, it's starting to create a todo.

Now, you can imagine that this would be very useful if it was a multi-user application, but there's only one person testing it. We could have a bot running in the browser and we could be testing it in another browser with a different user, and we'll see the interactions.

And we're going to extend this and build on this to make a bot that will actually use the entire application.

When I write bots to play games and automate, I just wrap a setInterval to a set of functions or objects that will do all the work that is required. And we can achieve many of the same things that we would do with an external tool, but from within the console. And we don't have to interact with it or monitor it, it would just keep doing the work.

If you ever needed to keep refreshing a page so you don't time out, a setInterval can be very useful for that.

There we go. It's created our 100 todos.

And as an exercise, you might want to try writing this so that instead of counting up from 0 to 100, it counts down from 100 to 0.

**Resources**

* [setInterval Documentation](https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setInterval)
* [clearInterval Documentation](https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/clearInterval)
* [Example Usage of setInterval](https://developer.mozilla.org/en-US/docs/Games/Anatomy)

**Quiz: +100 credits**

**1. If we create a `setInterval` and do not assign it to a variable we can stop it by:**

refreshing the browser page

running the code \*.clearInterval()

**2. If we create a setInterval and assign it to a variable we can stop it using the command: var sayHello = setInterval(function(){console.log("hello")},1000);**

stopInterval(sayHello)

clearInterval(sayHello)

clearInterval("sayHello")

sayHello.clearInterval()

**3. When we use `setInterval` we need to wrap it in a for loop:**

true because setInterval runs once after X milliseconds

false because setInterval runs every X milliseconds

**4. A `setInterval` will run:**

forever, until page is refreshed or it is cleared

a maximum of 100 times

Submit