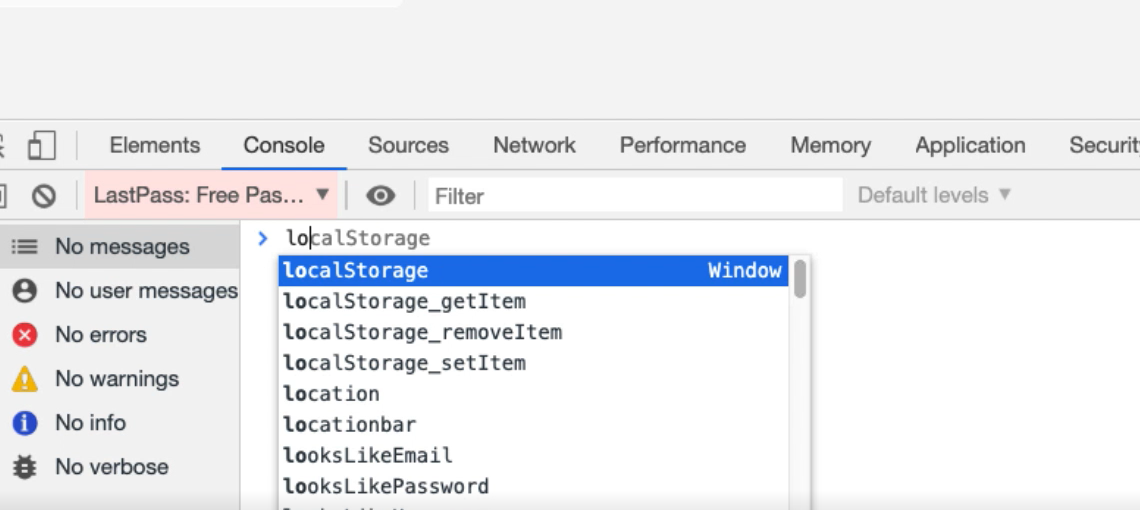
Now we are going to build some code that's going to be really useful if we were testing this.

I'm going to create some code to help me create 100 todos, and setup test data in this environment to save my time. Or when I am interacting with it and I want to just create a whole bunch more data in there, I'm just going to have some code to help me do that.

Before we start, I'm just going to show you one thing, because what we've been doing, I've been coming in here into the application [*Sources > Local Storage*], right click and clearing this.

**Now that we're getting used to working from the console and we're getting used to writing code, I can clear the *Local Storage* from the *Console*.**

If I just start typing localStorage there's the code completion.



So, it's "local" and "Storage," with an uppercase.

localStorage.clear()

And it appears that nothing happened, but that's not true.

The application now is empty of data. It hasn't refreshed, so we're just going to refresh the screen there, and I have no data in the application.

So that localStorage.clear() is what I used in order to set the environment up.

**Okay, so I want to create 100 todos — I'm going to create a snippet to do that and rename this "create 100 todos".**

I would say:

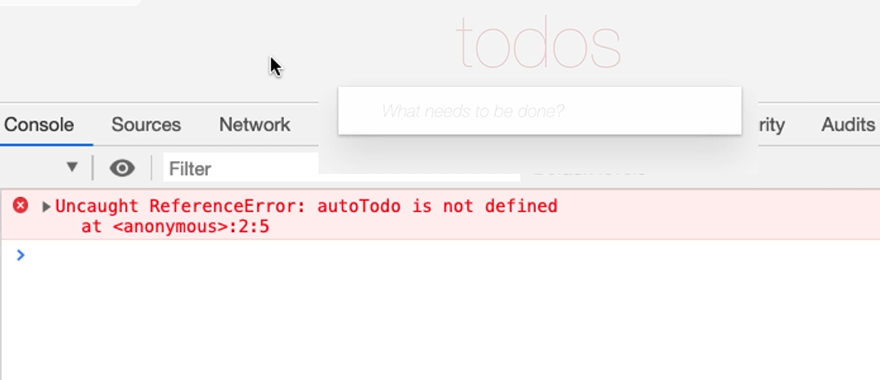
for(x=1;x<=100;x++){

autoTodo.createTodo("todo "+x)

}

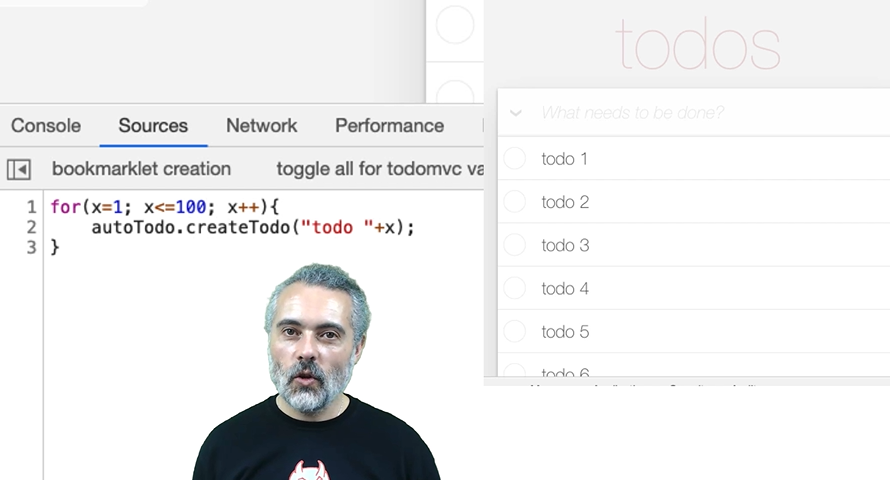
That should create 100 todos using the autoTodo object.

Run that snippet.



I forgot to run the snippet that creates autoTodo, because we refreshed the page, so all the code has gone.

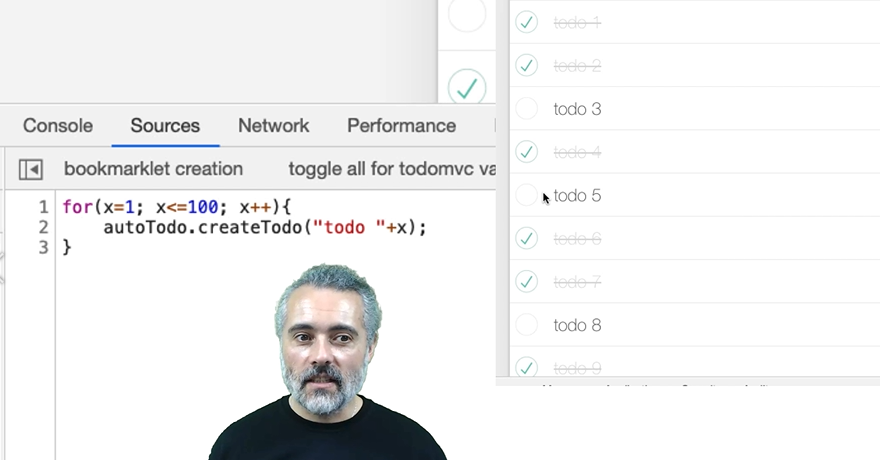
Let me create my “autoTodos” using the autoTodo snippet that we had earlier. Then, if I run my "create 100 todos", look at that 100 todos.



Excellent.

But something went wrong, and it's not obvious that something has gone wrong here. Let me show you.

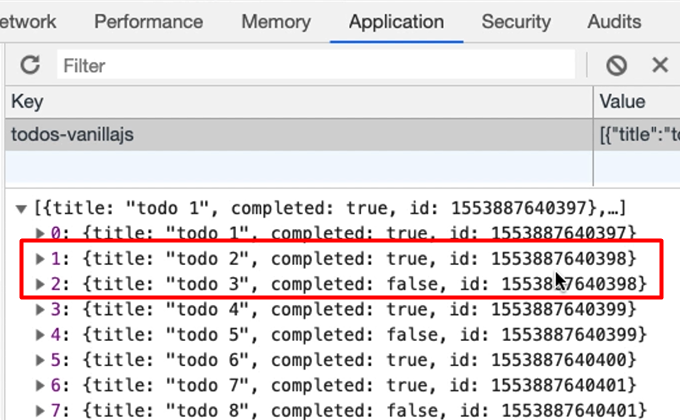
This is the easiest way to demonstrate the problem. I'm going to select all.



That doesn't look like it selected all. That selected some, and now I can't unselect. I've broken the application.

Let's have a quick look here.

If I look at the data, I can see that todo 1, todo 2, and todo 3. So todo 2 and todo 3 have the same id.



**That's because in the code the id is actually the milliseconds when the item was created.**

So, because I'm running in a for loop, and it's doing it within the same millisecond, and it's not the right way.

Remember I said, "When we are automating the applications and we start bypassing the way the application normally works there's a risk." We've just triggered that risk.

Now we could say, "this is a bug," because we shouldn't be able to ever create id’s that are the same. That should just not be possible.

We should have a good id that is uniquely created, but no user is ever going to create something in milliseconds. There's no automated API. It's not an application that is designed to handle this. So, this is a perfectly viable, functional approach. It's just that our way to automate it is incorrect.

We need a different way to automate this application to create 100 todos.

I better fix up the Local Storage. Fortunately, we know how to do that now — localStorage.clear() — refresh the screen.

And we better figure out a different way of creating 100 todos.

**The issue is really a typical issue that we have whenever we automate anything, which is synchronization or waiting until the application is ready for us to do some functionality.**

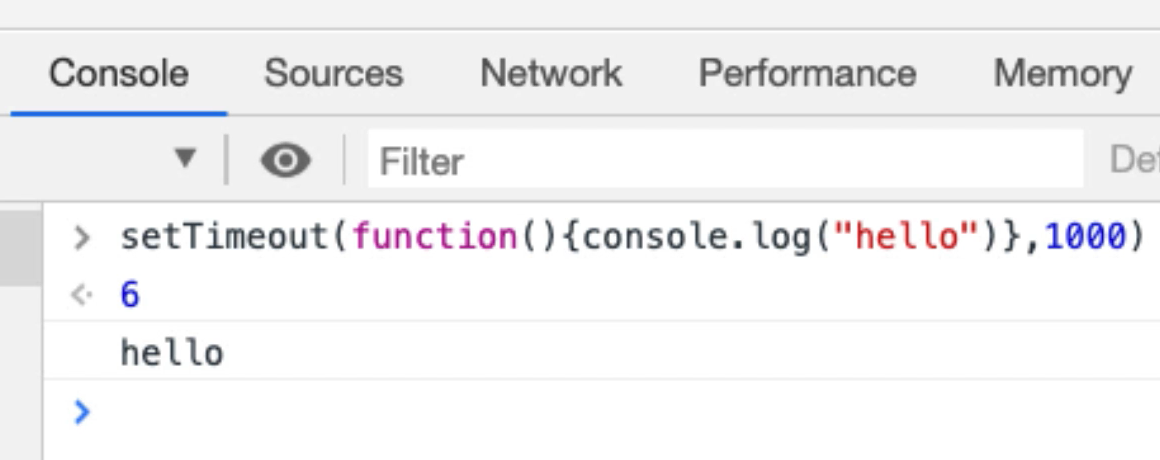
Now if you look up online, "JavaScript how do I wait," you're going to get 2 functions coming back. One of them is a setTimeout, and one of them is setInterval.

Let's have a quick look at setTimeout.

setTimeout(function){console.log(“hello”)},1000)

You can look this up on MDN, basically it says, "I'm going to run a function. This function is going to say, ‘hello` to the console.”

It takes 2 arguments: set time, it takes a function as an argument, and it takes a timeout. I'm going to say 1,000 milliseconds, which basically means in 1,000 milliseconds time it will run the function that writes that console.log.



There we go.

So, there was a gap before it said, "hello," and the gap was 1,000 milliseconds long — setTimeout lets me push off the execution of code that's going to function into the future.

**I might be able to use that to create 100 todos.**

If I had a for loop that pushed off into the future the creation of a todo, and each time I pushed it into the future it was 100 milliseconds in advance of the last time I'd be building, a queue of function calls that might allow me to create todos without having the same id.

Let's just test this out with our autoTodo object.

I'm going to put the autoTodo object into memory. I should now be able to say:

**autoTodo.createTodo("hello");**

Let's make sure that we can actually run this from the setTimeout:

setTimeout(autoTodo.createTodo("hello again"), 1000)

Okay, so that didn't work.

Now what this is doing is, when I run this command it actually does it immediately. Why is that?

What this is saying is, “ set the time out and run this function.”

But this isn't actually a function; this is a function call. So, we call the function to create the todo; then, in 1,000 milliseconds time we will call the return value from that function. There wasn't one.

I don't want to actually write my code like that — what I want to do is see not a function call, but an actual function.

**How do I pass in the arguments?**

Well, with setTimeout, after the millisecond timeout I can add in arguments, and these will be passed into that function when it is called.

setTimeout(autoTodo.createTodo,1000, "hello again again again")

This is saying, "Create a setTimeout so that 1,000 milliseconds in the future we will call the createTodo function on autoTodo with this parameter."

Let's try that.

Alright, so we got a 1,000-millisecond gap. I'm going to try that again.

Just to make it a bit more obvious, I'm going to put 5 seconds .

setTimeout(autoTodo.createTodo,5000, "hello again again again")

Can you handle 5 second wait? 1, 2, 3, 4, 5.

I'm going to edit that to make it look good. We've got a 5 second wait before we run it.

**Now I can put code in the future to execute, so I might be able to wrap that in a for loop.**

I'm going to take this code so I can remember it, go to my *Sources* where I've got "create 100 todos", and we've got our for loop.

Rather than "hello again", I'm going to say, ``"todo "+xandrather than 5,000 what I want is “x” times 100..

for(x=1;x<=100;x++){

setTimeout(autoTodo.createTodo, x\*100, "todo "+x)

}

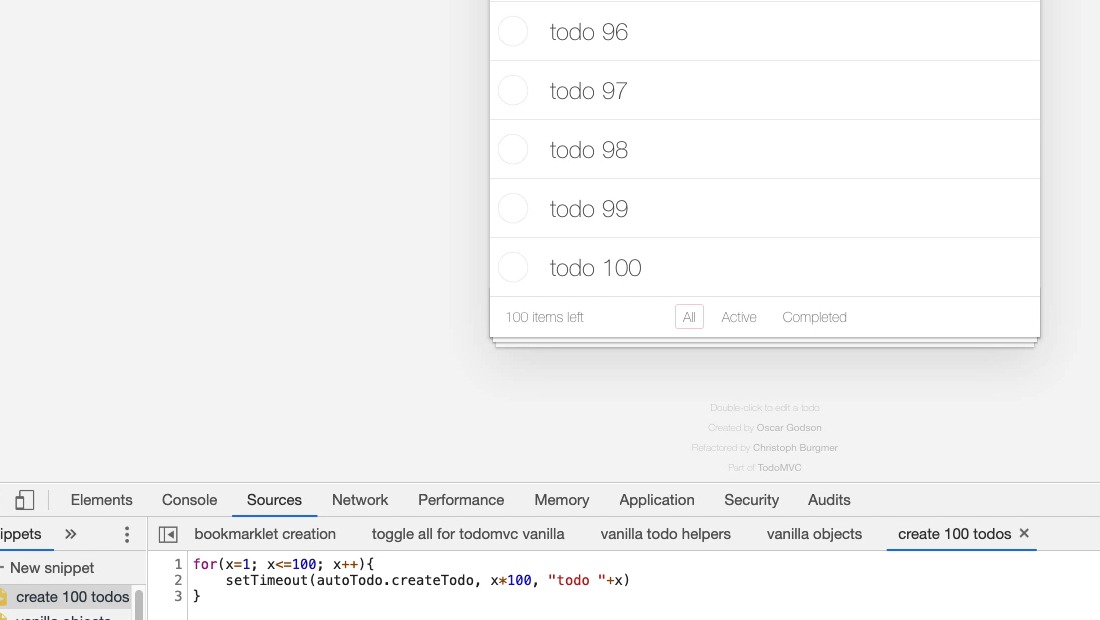
What this is basically saying is: go through this loop 100 times.

* When it's 1 it will say: 100 milliseconds in the future, call createTodo with “todo 1”.
* The next time we come to the loop when it's 2 it says: call createTodo 200 milliseconds in the future with “todo 2”.

I'm going to run this and see what happens.

Let's clear this down first. I'm going to do it manually just so that we don’t change this screen. Now run this.

Look at that.



How cool is that?

We are creating a todo every 100 milliseconds, so now are populating the data.

But I don't know if it's cool enough yet because, how does my select all work? That looks good.

The real test is [when I try to reset the data]... Oh, there we go. I haven't broken the application.

My items here are going up in hundreds of milliseconds, so they've all got unique ids.

**Now I have the ability to create 100 todos using a setTimeout and a for loop.**

Sometimes if I'm writing code and I'm interacting with the application like this, I may not use the helper methods all the time. I may not use the helper object all the time. I'm just going to have the code here, use the createTodo function itself, and then that just makes it a little better of a self-contained snippet, so I don't have to run 2 snippets at the same time.

function createTodo(name){

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

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

}

for(x=1;x<=100;x++){

setTimeout(createTodo, x\*100, "todo "+x)

}

Let me just see if this works.

Let me just do a *Console > localStorage.clear()*, refresh this, none of my code is left lying about. I've gotten my unique snippet.

Let me run this, and there we go.

Now I've got a very self-contained snippet. That might not seem that useful now, but when we come on to do bookmarklets this is very useful to have it all in a single place.

In fact, what I can do is I don't actually need to have it in a function here, because remember setTimeout can use a function.

I can say function, which takes “name”:

for(x=1;x<=100;x++){

setTimeout(

function (name){

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

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

}

, x\*100,"todo "+x)

}

Now I have an all self-contained within one block of code itself.

Of course, I say that, but I haven't run it, so let's just double check that it runs. There we go.

Sometimes having helper functions, helper objects is useful.

Sometimes having it completely self-contained is useful, but because it's all in different snippets I'm not losing any code. It's just that this is going to be so much easier for me to add into a bookmarklet later on.

**Resources**

* [setTimeout Documentation](https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/setTimeout)
* [clearTimeout Documentation](https://developer.mozilla.org/en-US/docs/Web/API/WindowOrWorkerGlobalScope/clearTimeout)
* [setTimeout in Game Development](https://developer.mozilla.org/en-US/docs/Games/Anatomy)

**Quiz: +100 credits**

**1. In order to clear the localstorage from the console we use the command:**

localStorage.empty()

localStorage.clear()

console.clear(localStorage)

**2. Using a for loop to create test data is:**

never possible because it runs too fast

possible for some applications but not the todoMVC application

not possible because it is single threaded

**3. The `setTimeout` function:**

runs a function every X milliseconds

runs a function after X milliseconds has passed

**4. The following command waits 1000 milliseconds before running the `createTodo` code: setTimeout(autoTodo.createTodo("make jello"),1000)**

True

False

**5. The following command waits 1000 milliseconds before running the `createTodo` code: setTimeout(autoTodo.createTodo, 1000, "make jello")**

True

False

Submit