Welcome to Module 10, Script Parameters. In this module, we're going to look at what script parameters are and how we can use script parameters and how that can potentially save us from having to modify our code. Now, script parameters are different from the parameters that you pass on to your function. But this is actually a NetSuite functionality that would allow us additional configuration for our script.

So in this module, we're going to deal with script parameters. And here we want to understand how to use script parameters, what are the different kinds of script parameters, and when to use those script parameters. Now, let's start with defining what a script parameter is. Now, script parameter, for one, is not the parameter that you pass to your script. So that's a little bit different.

Script parameters are actually fields that you create on your form that is unique to your particular script. So for example, if you want to pass data to your script, what you can do is that you can use a script parameter. Now, with a script parameter, you can use that to pass data via configuration made by your administrator. And if that's the case, the administrator value would affect all users on that account or you can create a parameter where that is unique to a certain person.

And each individual account owner or each individual user on the account would have their own data for that particular field. That would be useful if you want to create some sort of customization that would be unique to users. Maybe you want to create your own user interface for your account. And maybe you want to have like colors change or something like that. You can have like each individual user have a different color. But, again, that's just an example for it.

Basically, as an overview, with a script parameter, those are just fields that are assigned to a script. So far when we've created a field, we've assigned fields to records. Say if I want to create like our coupon code value, we assign that to an entity record, which is our customer. Or for my case, I created an entity field and attached that to my employee record.

But with script parameters, those are fields that you actually attach to a script instead of attaching that to a record. Now, we're going to see later why attaching to a script would be useful. But that's basically the general idea of a script parameter, having a field that is attached to a script.

Now, what are the advantages of using script parameters? So for script parameters, you can use that to change the behavior of the script without having to change your code. One example for that would be what if you have a certain value that would affect the execution of your code? Say you have a static value of 20%. Let's say you use that 20% value for your discount.

And it's always going to be 20%. That doesn't really change often. So what you might have decided to do was to just hard-code that on the script because you can't always just attach that field to say a customer or an item. Because just imagine if you have like 100 customers. You would have a field that has the exact same value, 20%.

Now, if the value changes, you have to go to each individual field or each individual customer and then have that value change. So it's not really practical. So I think the better solution for that would be hard-coding the value on your script.

Problem with that is that if you hard-code your value to your script, every time the value changes, you have to go back to the script, update the value, and then just go ahead and test everything. Now, for simple values like that, it might not affect your execution that much, but it might not always be the case. Remember during development, you have to do your own testing where you have your development lifecycle.

It's not just a changing a value and then that's that. You have to make sure everything is tested, nothing is breaking with the value change. And you go through the normal cycle. And so by using script parameters, it allows you to just change the value, because since you've already scripted that-- OK, this value can potentially change-- you can easily change the value in the NetSuite user interface without worrying that it would affect your code.

Another use case for using a script parameter is by allowing users to have their own preference settings. Again, this would be useful if you want to create some sort of user preference kind of customization where each individual user can turn on or turn off certain settings or maybe if you're allowing them to configure their own user interface of some sort. So that's something that you can use a script parameter for.

You can also have like admins control company values, similar to what I have mentioned in the first point. If you have a value to make sure that all of the company uses that particular value for something, then you can have just admin maintain that value instead of you having to hard-code that on your script and then modify that value each time the value changes.

Now, another very common use of a script parameter is to pass a value from one script to another. So if you want to pass values from one script to another, you have to create a script parameter. And then instead of just passing the value to the script, what you're going to do is that you're actually going to set that value through that script parameter so that the script that you're passing the data to would just pick that up.

So you can think of a script parameter as an intermediary, the repository of information. So you're not directly passing the information to the other script. But, instead, you're putting that into that location. And then the other script would just be picking that up from the location where you placed it. So that's kind of how it works.

And same thing with the workflow. So if you want to pass a value from a workflow to the script and a script to the workflow, then you would have to use a script parameter. So that's something that we're going to look at in the next module.

So that said, there are three different kinds of parameters that we can use in SuiteScript. We have a company-level parameter, a user-level parameter, and a deployment-level parameter. The difference between all of those three parameters is the scope of where that value is applicable to. For example, if you have a company-level parameter, that means that the value would be applicable to the whole company. So since this would be applicable to the entire account, only NetSuite administrators can set this value.

Regular users would not be able to set that. User-level parameter, as the name implies, would be a parameter that can be set by individual users. And since this parameter is set by an individual user, that means that this value can change depending on the user. For example, my value can be different from another person's value, which can also be different from another person's value.

And then, lastly, here we have the deployment-level parameter. And this parameter depends on the deployment record. If you have multiple deployment records on your account, then you can potentially have multiple values.

Now, deployment-level level parameters are typically used by developers. These are not used by users. And this would just allow you to have a value that you can set for a deployment. For example, instead of having multiple scripts, you decided to create one script. And then you deploy that script to say a customer record, a vendor record, and a partner record.

And for the most part, the processing is the same, except maybe there are certain values that are different depending on the record that's executing the script. Let's say if a partner record is executing this, it has to do certain processes. And a customer would do another process.

And a vendor would do another process. So about maybe 20% of the process is different. But 80% of the process is the same. So in that case, just create one big script. And then if there's a value that you can use to change the direction of the processing, you can put that inside your deployment parameter.

What we're going to do in this walkthrough is that we're going to create an automation so that we can change the default value of our Employee Code. Now, this default value would not be hard-coded in the system. But we're going to have the NetSuite administrator maintain that value.

So how are we going to do that? Now, what we want to learn is that we want to learn how we can create a script parameter field, how we can configure the value of a script parameter, and how we can retrieve the value inside of a script parameter. So let's go ahead and go to our walkthrough.

Before building our script, the first thing that we're going to do is that we're going to create our script parameter. Now, a script parameter is simply a field that is attached to a script, similar to a custom entity field, being a field that is attached to a record. So what I'm going to do now since we're going to add in a field for our default Employee Code value and our automation is on our client side script, I'm going to add a script parameter here.

Now, I've opened my script record. And I'm going to my Parameter subtab. And I'll just click on New Parameter.

Now, notice here that when I open the new parameter page, it looks very, very similar to my custom entity field. The only thing that's different is that it's not asking me where I want to apply this particular custom field. So remember I have an Applies To subtab on my custom entity field. So I don't have that because this would be applied to my record.

Now, for this, I'm going to create a default Employee Code field, Employee Code. For the ID, \_sdr\_default\_emp\_code. I'll just call that emp\_code. Now, for this one, I'm going to use a free-form text.

And, also, I have a special field here that I can only set on my script parameter. This is something that you don't have on a custom entity field. Now, you have three different values that you can use on your preference setting. You have a user preference, company preference, and also a blank setting that would refer to a deployment preference.

So what are these preference settings? If you set your preference to user, what that means that this parameter would be accessible to the user, meaning each individual user would have their own setting. If they want to change that setting, then that value would be unique to that user.

Now, if you want to create a value-- now, in our case, this would be maintained by the administrator and should affect all users, not just for one individual user-- then you're going to use company. Now, this company parameter, again, can only be set by the NetSuite administrator because of the nature of the setting. It's going to affect everybody.

Now, the last option that we have here is a blank setting. And that would mean that you are creating a deployment-level parameter. And for my deployment-level parameter, that could be a setting that would be different depending on the deployment.

Now in this case, I want a company-level parameter. So I'll create that. But I'll also create other parameters so that you can see where that's set. So here I'll create a user-level parameter.

So for the ID, I'll choose the same thing, \_sdr\_user\_param. And for this one, maybe instead of using a free-form text, I'll add like a checkbox. So I'll just set that to user.

And I'll create a new one. And then here I'll call this my Deployment Param and then \_sdr\_deployment\_param. That's it. And I'm going to leave my preference setting as blank.

Now, we're not going to use those settings. I just want to show you where you can change the values on those script parameters. So, anyway, so we have three settings here. We have the default Employee Code, something that we are going to use on the script. And then we have our user-level parameter and then also my deployment-level parameter.

Now let's start with my user-level parameter. If a user wants to change a setting on the account, what they would do is that they would go to the Home icon. Then click on Set Preferences. These options would be unique to a user.

So for example, if you want to display NetSuite in a different language, for example, Japanese, or Italian, or French, or Spanish, now you can use all of those settings. They can change the time zone here. But for us, what we're interested in would be the script parameter.

And once you add a script parameter, you would notice that there is an option right over here that says Custom Preferences. So if you go to Custom Preferences, that's where you would see your user-level parameter. So as you can see, I gave my user-level param here. And I can enable that or disable that if I want to. Again, this would be unique for your user.

All users would have access to this setting. So, again, it's just going to be set from the Home icon and then Set Preferences. If you are an administrator and you want to change settings that would affect the whole account, for example, if you want to set a time zone for all of the accounts on the record, what you can do is that you can go to Setup, Company, and the General Preferences. Again, that's Setup, Company, General Preferences.

Now, for these preferences, this would only be accessible by the administrator. If you're not an administrator, you probably wouldn't even have access to the Setup tab. So here as an administrator, I cab change like the date format, the number format if I want to. So there are a lot of settings that I can change here. So I can override certain values if I want to.

But similar to our user-level parameter, I also have a Custom Preferences subtab. And this is what I need. So if I open my Custom Preferences subtab, this is where you're going to see that I have my Default Employee Code.

Now, for the Default Employee Code, maybe I'll just set a value to that. And I'll just say E0001. So that's my Default Employee Code. Now, we want to get that value. Click on Save that. And we're going to grab that value from the script.

Now, script parameters would be very convenient to use if you want to store like a singular value because if you are going to store just one value, then it doesn't really make sense to attach that to a record. Say even if that record is related to my employee code or to my employee record, it doesn't mean that I should be creating an entity record and then attach that to my employee record because that would mean that each employee record would have a different default employee code value. And you don't want that. You want one value for all of the employee records.

You also don't want to put this in a customer record. Because if you put that in a customer record-- like you have a record. Then you have like one singular field of the custom record. It really doesn't make sense. It's like creating a whole house where you have a house and the only thing that you have in that house is a chair because that's the only thing that you need.

So it doesn't make sense. So it's like creating a whole table with one field. So for this one, it's just really convenient to just use that one field one script parameter.

But, anyway, going back to our script here. Let me just close this and reopen our sdr\_cs script. What we want to do is that we want to default the employee code value.

Now, we don't have anything that defaults the employee code value. And by the way, I'm going to comment this out because we don't need that alert anymore. And then if you want to get the script parameter value, you need to make sure that you load the runtime module. So that's the first modification that we're going to do here.

So we have our N/runtime module. And for the name of my runtime module, I'll just call that runtime. Also, I'm going to add in my parameter annotation, so @param. Then we have our runtime here. And then the variable name, which is pretty much the same name, which is runtime. So that's it.

So I've added my runtime module. So the next thing that I'm going to do here is that I'm going to try to grab the value of my employee code. Now, for this one, we're only going to change the employee code value if the employee code value is empty. So first thing empCode equals to employee.getValue.

So as you can see here, it's same thing that we've been using previously. Let me see if I have an employee code value. Or I can just go to my employee record to get the ID from there. Or maybe I should have created a new one. Let me just create a new one.

There you go. Let me get my employee code value or employee code ID. Grab that. Go to my script. Then add in a condition to say if my employee code value is empty, then let's go ahead and start scripting.

Now, to get the default employ code value, you need to get a reference to the script first. So here I can just name it script. That's fine. Then I have my runtime module. And then from my runtime module, I have a function called the getCurrentScript.

So this would give me references to the script object, not the actual script, but an object that references that script. So if you're running the getCurrentScript method or function on a client side script, you would get a reference to client side script. If you're running this on a scheduled script, then you would get the reference to the scheduled script, or a user event script, whatever. But it will give you reference to that object.

Now, once you have that reference to the object, next thing that you can do is that you can get the value. Well, in fact, we should be able to just change these goals together. So here what I'm going to do is that I'm going to get the default EmpCode. And I can do that by getting the getCurve or the object reference for that script. And then use the getParameter method inside my script reference object.

So far the getParameter method, I have one property, which is the name property. And this refers to my default employee code value. Now, this should be the ID of the value that you are trying to get. So for example, here-- let me go to my script parameter.

And since I want to get this particular value, I'm going to copy that and paste that right over here. So as you can see, it's just really simple to use a script parameter. Just use the runtime module, get a copy of the script. Once you have the copy of the script, use the getParameter method.

And then once you have that getParameter method, then you can just use the normal set field value. So here I have my employee.setFieldValue, not set feel value, but setValue. setFieldValue is 2.190.

So I want to set my employee code value. Let me get that ID. And then from here, I'll just say I'll get the default employee code value. That's it.

So as you can see very, very simple. You can use this for any kind of field. You can use free-form text, a list or record field, a checkbox or any kind of field on your script parameter setting.

So let me upload this. And let's see if this works. Let me go to my employee record. And I'm going to refresh this by pressing Control-F5. And if I scroll down, as you can see here, my default employee code was set to E0001.

Now, here's the beautiful part of using a script parameter. If I hard-coded this value and the administrator, or the company manager, or whoever manages this value decides to change the value, I would have to go back to my script, change the value, test if everything works. Now, for this one, this is really like small. So it's easy to test. But if you have a longer script and that value can potentially affect any other parts of your script, then that's going to be very problematic.

Now, if I code based on my script parameter and if I'm expecting this value, that it could potentially change, then I can already add some provisions for that change. For example, for here, I know that this default employee value can change. And maybe I can just add in some things that can adapt to that change.

Now, here on my setting, let's say the company decided to change that value and then maybe move that back to EMP001. If I go to my Custom Preferences, if I change that EMP001, then click on Save, without having to modify anything on my script-- and if I refresh this and wait for this to load a little bit. If I scroll down, you can see here that the value changed to EMP001.

Again, now you're free because you don't need to maintain the code. They're already there. It's already working. And if they need to change the value, they don't need to contact you anymore.

The NetSuite administrator would just have to go to the preferences, change the value, and everything is working fine. So as you can see, script parameters are a very important feature. And it's a very convenient way of maintaining a value without having to modify our code.

So that's pretty much it for this walkthrough. Things to remember here. Just make sure you understand where you are going to set the parameters. If you're dealing with a user-level parameter, you have to go through the Home icon, then Set Preferences.

If you are on a company-level parameter, that's going to be Setup, Company, General Preferences. And then, lastly, go to the deployment record if you want to set the deployment parameter. Also, remember that if you are going to use a company-level parameter, this can only be set by your administrator and will affect the rest of the company. User-level parameters can be set by individual users. And these users can have their own setting that might be different from another user.

Also, you can use script parameters to pass values from one script to another. Or you can use the script parameter to pass a value from your script to a workflow and vice versa. And that's something that we're going to look at, again, in the next module.

Now, it's your turn. Please complete the following activity. This conclude module 10, Script Parameters.