**Implement** FreeMarker Use Cases. In this module, we'll go into the Source Code mode for the very first time, focused on FreeMarker directives and how to implement them. We'll also look on ways on how to store values on a variable in Prism afterwards.

By the end of this module, you should be able to enter the Source Code mode and see how the template layout is translated into code, use built-ins that require parameters to operate, and understand how the if and list directives work, and lastly, differentiate use cases for FreeMarker directives to efficiently construct and test FreeMarker codes.

This slide shows again what are the different operations you can perform inside a FreeMarker expression. You can access the record data source, perform calculations, use built-ins-- this is one of our main focus points of this module-- and use special FreeMarker fields like .now.

Now let's look into more FreeMarker components like built-ins with parameters and FreeMarker directives. In FreeMarker, we have several useful built-ins for string manipulation. One of them is the string built-in. The notation for using these built-ins is similar to how we use the standard ones-- we append it with a question mark then the built-in function name. This time, it is followed by a parenthesis pair.

These parentheses require one or more parameters depending on the built-in. What's happening here at the second line is that the string built-in evaluates the Boolean value of the field. If record.complete is true, display Done. Otherwise, display Not Done. You can check out FreeMarker's website for complete documentation on all FreeMarker built-ins.

FreeMarker NetSuite, we have a couple of field data types that we use frequently. And in this chart, we'll see how it is represented in FreeMarker.

Record.memo-- in NetSuite, it is a free form text and this represented a string in FreeMarker. Record.total-- it is displayed as currency in NetSuite but treated as number FreeMarker. Record.complete-- it is rendered as checkbox in NetSuite, but in FreeMarker, It is evaluated as a Boolean value. That's why when we did record.complete, question mark, string, we were able to extract trueness or falseness of the field.

Record.trandate-- dates are represented the same way in NetSuite and FreeMarker. Lastly, record.expense. In NetSuite, it is represented in sublists, while in FreeMarker, they are treated as sequences, and sequences are values where we could perform iterations on.

Now we add the FreeMarker directives into the mix and use them alongside all of the PDF elements we have used so far. Each directive has different use cases, and we'll see them in action in the following slides.

Directives are part of the template that don't show up in the WYSIWYG mode. That's why during the introduction of the course, it was mentioned that the Source Code mode offers better flexibility for template customizations.

The "if" directive allows us to control use cases, show or hide certain sections of the PDF, depending on one or more conditions. Here's the syntax of the "if" directive. Open with the less than sign, the number sign, the word "if," then followed by the expression that needs to be evaluated. Then close with a greater than sign. The end of the conditional is indicated by a closing "if" tag. This means that all FreeMarker or BFO codes inside of the sphere will only execute if the expression is evaluated to be true.

This is an example of the "if" directive usage. We see the familiar opening of the "if" statement then the expression, "total gt 100." This line checks if the value of the total field is greater than 100. If true, display the table structure below.

We took the code from the previous slide and compared to two more variants. Let's see what makes the other two return an error. The first one uses the actual greater than sign instead of using the gt sign. Remember, the greater than sign is one of the few restricted strings inside the FreeMarker.

For the second one, we don't have to use the FreeMarker expression inside of the "if" directive. Note that we are already on the scope of FreeMarker when using directives so no need to use a separate expression using a dollar sign.

Now we look at two other directives that are equally useful when dealing with Advanced PDF templates. The "assign" directive gives us a way to store data either for later use or for calculations. We can make assignments by typing in, opening less than sign, the number sign, followed by the word "assign," then the variable name. So this can be any variable name you want, except for numbers and special symbols. Followed by an equal sign. Then the expression, then closed by a greater than sign. For the expression, it could be record.total or record.trandate, or whatever data you might want to extract and store from any data sources.

All transaction records in NetSuite have sublist, and we can access this list by using the "list" directive. This directive gives us a way to access all of the line item values of a certain record on a per line basis. It is typed in as follows-- opening less than sign, then the record sublist ID, followed by the word "as," then an alias for the line. This alias can be anything as long as it's a valid string. Then finally close by a greater than sign.

Similar with the "if" directive, it is terminated by a closing "list" tag. And all of the BFO and FreeMarker tags will be executed for how many times, depending on the number of line items in a transaction.

In this walk-through, we'll be introducing built-in with parameters to our template. We'll also take on the concept of conditionals, iterations, and value assignments.

In this walk-through, we're going to do two things. First, use the FreeMarker string built-in and change the representation of some text inside a PDF as it gets printed. And second, we'll create a new subtotal line in the PDF that computes for all breakfast, lunch, and dinner-related line items.

In the expense report list, let's open one expense report. And we can see that this expense report is currently a completed one. Let's try to print it out.

For this expense report, the complete value is "Yes." Meanwhile, let's open an incomplete expense report just to verify that we're seeing the right value. Print this out. And the complete value is equal to "no."

Now what we're going to do is represent "yes" as Done and "no" as Not Done. In order to do that, let's go back to the list of templates and open the SuiteDreams expense report template.

So I get this list for SuiteDreams expense report. Click Edit. And inside the Source Code editor, let's search for record.complete.

And we see here the record.complete at label interpolation and the record.complete. Let's add the question mark string both in. And for the first parameter, type in "done" to replace the word "yes" in the PDF and replace the word "no" with "not done." And then scroll up and save the template. And after saving, let's go back to the complete expense report and see how it affected the printout. Click Print. And now we see that the complete value is "done" instead of "yes."

Now let's go to the incomplete expense report and we should expect that the word "not done" is printed instead of "no." Click Print, and there it is. "Not done" is displayed instead of "no."

Now let's go one step further by chaining another FreeMarker built-in into the string built-in. Let's go open again the SuiteDreams expense reports template and look for a record.complete.

And right after the closing parentheses of the string built-in, let's add a question mark uppercase built-in. Then click Save, then open again the expense report record. Click Print.

And now, not only the "no" word was replaced by "not done." It's also printed in all capital letters. Let's go back again to the complete expense report, and we also see the "done" printed in all capitalized letters.

Now next up, we'll create a new subtotal line and compute for all Breakfast, Lunch, and Dinner-related line items. And in the example here. We have Lunch, Breakfast, Meals and Entertainment, and the fifth line for Dinner. But the Dinner here is a non-reimbursable expense, which is still all right and will be included into the calculation while ignoring and excluding the Meals and Entertainment line.

Let's click Print to see a preview of the expense report. And we see the Lunch. Two lunches. Breakfast and Meals, Entertainment printed under the reimbursable expenses, while the Dinner is under the non-reimbursable expenses.

And under the Advance to Apply, we're going to create a new line here and compute for all breakfast, lunch, and dinner-related expense report lines.

So let's go to the template. Forms, Advanced PDF/HTML Templates. Edit SuiteDreams expense report and look for the word "advance" inside of the template. In here, we see the record.advance@label and a record.advance interpolation.

Basically, what we need to do here is to make a copy of this row line. Let's do that. Press Enter and create a new table row. Type in tr for the new row and td colspan of 4, and type in ampersand nbsp semicolon, which is technically a white space placeholder. Then followed by td align equals to right to align all of the values to the right side of the cell.

Then inside of it, let's type in the meal's total for the label of the field. And another cell to the side of it. And type in the placeholder for the value of the meal's total. Temporary meal's total of 0.0. Then click Preview. And now we see a new line added under the Advance to Apply line with a Temporary Meal Total of 0.0.

Now next up, let's go to the template and start building our conditionals. In here, we see the reimbursable expenses, the non-reimbursable expenses sections. And on top of that, we have an iteration that's going on. And we'll take advantage of that and evaluate each line whether or not they are lunch, breakfast, or dinner-related or not.

But for the meantime, let's type in, assign, meals total for us to have a variable with the store our meals total value. For now, let's type in "100" and copy the meal's total variable name. Scroll down to our subtotal line. And in here, replace the temporary meal total of 0.0 with a FreeMarker interpolation to print the meal's total value. And that is done by typing in dollar sign, open and curly braces, then typing in the meal's total. Let's click Preview and scroll down. And now meals total is equal to 100, printing the stored inside the meal's total variable.

So up next, let's reset the meal's total variable to 0 and start working on the conditionals. 100, 0. Directly under this first iteration, let's open up our first conditional. If, expense, dot, category. So expense is the data source ID for this line. The category is the column field ID. The expense category is lunch. Then do the following.

So press Enter to add a couple of lines. And what we want to do is capture the line item's amount and add that to the current value of the meals total variable. So assign meals total equals meals total, its current amount plus the expense.amount, which is the way to access the current line item amount and delete automatically assigned tag for the closing. And let's click Save.

After saving, let's open the expense report. And click Print. And in here, we see two lunch-related line items, one with 30, the second one with $20, which is equivalent to $50 in total.

And that is reflected on the meals total line here at the bottom. Let's expand the condition and add a breakfast and dinner conditions. Open again the SuiteDreams Expense Report template and search for the Lunch section.

There we go. And let's expand the condition by typing in the or symbol, which is a double pipe expense.category equals to breakfast or expense.category is equal to dinner. Remember that it only takes one condition here to be satisfied for that line to qualify for the inclusion for the calculation. Click Save, and click Print.

And in here, if we add 30 plus 20 plus 15, ignoring the 15 for the Meals and Entertainment, plus 50 is equal to $115. But the only problem here is that it's not in dollars and it's just represented as a simple integer.

To fix that, let's go back to the template one last time. Open the SuiteDreams expense report. Click Edit. And let's look for the Meals Total section.

Here in the Meals Total interpolation, let's add the question mark string, dot, currency to render this value as in currency. Let's go the extra mile and add alignment to put everything in good position.

Click Save. Then finally, click the Expense Report Print button. And now, 115 is represented in dollar figures. This ends this module's walk-through.

Now it's your turn. It is now time for you to complete your hands-on exercises for this module. Please refer to your student guide to complete the required exercises.

This concludes the Implement FreeMarker Use Cases module.