[MUSIC PLAYING]

**Investigate** Architecture. In this module, we'll identify how our templates gets parsed and transformed into a PDF document. We'll also deepen our ability to recognize which part of the template is FreeMarker or BFO by looking at several code samples. By the end of this module, you should be able to understand what are the steps involved in printing PDF and identify what are the technologies involved in building templates, recognize what are the different parts of the template, identify which part is the header, body, and footer section.

In this illustration, we see how the WYSIWYG mode template is translated into code once we switch into the source code mode. Looking at the code snippet, it is enclosed inside of a table tag, followed by a tr tag which indicates the start of a table row. Inside of the tr tag are multiple th tags which serve as our table headers, while td access the data containers for your field values. In the following slides, we're going to look into several code samples, and highlight different parts of the template, and isolate which part is BFO and FreeMarker.

In this slide, codes highlighted in red are BFO tags. Remember that most of these tags you see on an advanced PDF template are this kind of tag. Having a mental note of "if it looks like HTML, it's BFO" might help you remember.

Now this shows the same code, but this time, the FreeMarker interpolations are highlighted in red. The first cluster are the table headers populated with field labels. Remember the @ sign? Meanwhile, the section below are your table values.

This is a new chunk of code in here, and the highlighted section is BFO. Again, remembering our mental note of "if it looks like HTML, it's BFO." We are showing the same code, and you might be wondering why the highlighted BFO section did not include the tag on the first two lines.

Yes, I did mention that if it looks HTML tag, it's BFO. But there are very few exceptions, the list tag and the if tag. These are called FreeMarker directives. But for a quick brush over, the list tag is used for looping over arrays like your transaction line item, while the if tag allows us to evaluate the condition and based on the results, display or hide a certain section of the PDF. Now we go behind the scenes and look at what happens once we click the Print button on any NetSuite record where advanced PDF is applicable.

The generation of a PDF is a fairly short process. First, FreeMarker interpolations are parsed wherein values are fetched from the data sources. It might be from a record, subsidiary, or company information. Afterwards, the BFO tags are then parsed to start building the structure of the document, i.e. render table structures, apply cascading-style sheets, and finally, produce a PDF ready for download.

In the tail end of this module, we'll look closely on different sections of the template and explain what's the purpose of each. The PDF tag is the very first one you'll see once you look inside the source code mode. This signals the start of the document and acts as a parent tag of all elements.

Related to the PDF tag, we also have the PDF set tag which then encloses one or more PDF tags. Through the use of this tag, we can print multiple PDF documents in one go. We'll look into this deeper once with review the BFO documentation.

If you're looking to modify your headers and footers, it's best to get accustomed with the macro list tag. This tag houses both the headers and footers sections. Inside the macro list tag, we have the macro tags, and the content of these tags can be printed on multiple pages.

In this illustration, we see the macro tag in action, and it contains the header and footer section separately. These are then printed out to every page of the PDF. Meanwhile, the main section of the PDF is contained inside of the body tag. This is where you put all of your contents like your document ID, transaction date, and line items.

For a quick reference on what are the BFO equivalents of frequently used HTML tags, here are the following. For HTML tags, we replace it with a PDF tag. Same with the HTML tag, we end it with a closing PDF tag.

Meanwhile, for line breaks br, it is highly recommended in BFO to put a slash at the end of the tags, which doesn't have a closing tag. While the paragraph tag, in HTML, we can always get away leaving out the closing p tag. But in BFO, it will return a compilation error. Lastly, for aligning text, we need to enclose the alignment value inside of a double quotation. This also applies to other aspects of BFO like applying in-line styles to your HTML elements.

In this walkthrough, we'll go full circle and observe how the source code mode template represent parts of the WYSIWYG mode interface then vice versa. In this walkthrough, we're finally switching to the source code mode. We'll contextualize what we're seeing in the code by going back and forth between the source code mode and the preview mode.

So let's start by going back to the template. Open the SuiteDreams Expense Report template again. But this time, switch into source code mode by clicking the Source Code button. On the Warning window, just click Yes. And immediately, we see the back end code that runs behind the curtains of the WYSIWYG mode.

And every time I go from WYSIWYG mode to source code mode for the very first time, I always make sure to format my code. And we can do that by first copying all of the code by pressing Command-A or Control-A on your Windows machine, then Command-C to copy. Then inside of the Pretty Diff website, make sure to use the Beautify function. Now paste your code on the left-hand side, then press Execute, and a newly formatted code is generated on the right side. Now if you do a quick comparison, I would say that the code on the right side is much more readable, easy to understand than what we have on the left.

Now let's copy this code, Control-A, and then right click Copy. And the code that we have originally here, let's delete that. And once we have deleted the old code, Control-V or Command-V on a Mac to paste your newly formatted code. Then click Preview to validate that we have not broken any part of the code during the reformatting process. OK. Still good.

Next up, we'll determine where we can find inside of the code the interpolation format for the SuiteDreams logo, for the SuiteDreams company name, as well as the company address. Also look for the expense report title and the transaction number interpolation. Go back to the template. Scroll a little bit further down.

Inside of the macro list tag, we have two macro tags. First is the nlheader and the nlfooter. Inside of the nlheader tag, we see an image tag with the interpolation expression for the logo URL. We also find a company name accessing the company information datasource, as well as the address text. Also in here is your record@title to print the transaction name as well as the record.tranid.

Then go back again to the PDF. Let's identify where we can find this expenses total and the data due section below it. Scroll down. We're going to brush through the style tag, which contains all of your CSS codes.

Then in here, we are inside the body tag. And this right here is your interpolation for the record.total. And below that as well, you have the record.duedate@label. Print the label and interpolation expression for the exact record.duedate.

One more table down. Let's see where we can find the complete supervisor's email, last modified date, and the posting period sections. Go down further on the template.

And in this section, this table right here, we have the record.complete@label, supervisor's email, last modified date, and label interpolation for record.postingperiod. And below that, you have the interpolation for printing out the value; record.complete, record.entity.custentity, last modified date in the posting period. Then finally, let's see inside of the code where we can find the reimbursable and non-reimbursable expenses, and how these line items are segregated into two categories.

In here, this is a comment line which signifies the start of the expense line item processing. And the list here, this is a FreeMarker directive. It allows us to perform iterations on all transaction line items so that we can process them one by one. And this code here checks if the current transaction line item or processing is reimbursable or not. If it's reimbursable, then print that line in this section.

And we can access each transaction line items field by typing in expense.category, or expense.customer, or expense.amount. And below that, we have the non-reimbursable expenses section. This section has the same logic, but this time, it expects each line to be non-reimbursable. If that's true, then print that line into this section.

Now it's your turn. Please complete your hands-on exercises for this module. Refer to your student guide to complete the required exercises.

This concludes the Investigate Architecture Module.