Week 4

Introduction to User Event Scripts (UES) and Client Scripts (CS)

Week 4 Overview

You will learn:

1. User Event scripts
    a. Execution contexts
        i. beforeLoad
        ii. beforeSubmit
        iii. AfterSubmit
2. Client Scripts
    a. Execution contexts
3. How to deploy scripts to your NetSuite account
4. Hands on exercises for both User Event Scripts and Client Scripts

Understanding User Event Scripts and Their Triggers

What Are User Event Scripts

UES are server-side scripts, meaning they are executed on NetSuite's backend servers.

They are triggered by specific record events, such as when a record is opened (loaded), submitted (created or updated), or deleted.

These events allow UES to handle both real-time and transactional operations without requiring user input or manual intervention.

3 Types

1. beforeLoad
2. beforeSubmit
3. afterSubmit

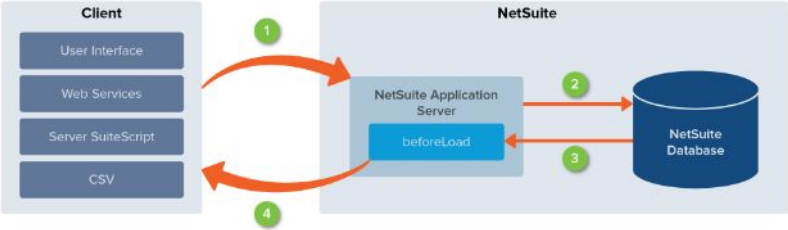
beforeLoad

This event is triggered when a record is loaded from the server (i.e., before it’s displayed to the user or integrated into other processes). In this phase, UES can modify the record’s contents or user interface (UI) elements. Common use cases include pre-populating fields, setting default values, or applying field-level security or restrictions based on user roles.

- **Example Use Case:** Modify the appearance of the form based on the user’s role. For instance, for sales reps, hide certain financial fields they shouldn’t have access to.

beforeLoad

The following diagram shows an overview of what occurs during a beforeLoad operation:



1. The client sends a read operation request for record data. This request can be generated from the user interface, SOAP web services, REST web services, CSV import, or server SuiteScript (except other user event scripts).
2. Upon receiving the request, the application server performs basic permission checks on the client.
3. The database loads the requested information into the application server for processing. This is where the beforeLoad operation occurs – before the requested data is returned to the client.
4. The client receives the now validated/processed beforeLoad data.

beforeSubmit

The beforeSubmit event fires just before the record is submitted to the NetSuite database. This is a critical stage for performing validation, data transformation, or applying business logic. At this stage, UES can intercept and modify data to ensure the record adheres to required standards or rules before saving.

- **Example Use Case:** Validate that a sales order contains a valid customer email or that the item quantity is within a certain threshold before allowing the order to be processed.

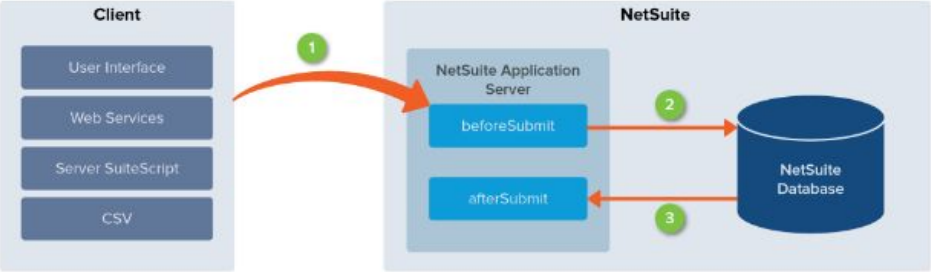
afterSubmit

- This event occurs after a record has been successfully submitted to the database. It’s typically used for actions that depend on the final, committed state of the record. For example, after a sales order is submitted, an afterSubmit script might trigger an external system notification or update related records.

  - **Example Use Case:** Send an email notification to the finance team when a customer credit limit is updated or initiate a workflow that triggers an approval process after an invoice is submitted.

beforeSubmit and afterSubmit

The following diagram shows an overview of what occurs on beforeSubmit and afterSubmit operations:



1. The client performs a write operation by submitting data to the application server. This request can be generated from the user interface, SOAP web services, REST web services, server SuiteScript calls, CSV imports, or XML. The application server:
    1. performs basic permission checks on the client
    2. processes the submitted data and performs specified validation checks during a beforeSubmit operation

    The submitted data has **NOT** yet been committed to the database.
2. After the data has been validated, it is committed to the database.
3. If this (newly committed) data is then called by an afterSubmit operation, the data is taken from the database and is sent to the application server for additional processing. Examples of afterSubmit operations on data that are already committed to the database include, but are not limited to:
    1. sending email notifications (regarding the data that was committed to the database)
    2. creating child records (based on the data that was committed to the database)
    3. assigning tasks to employees (based on data that was committed to the database)

Use Cases for UES

Automating Processes

UES scripts are often used to automate mundane tasks, saving time and ensuring consistency in workflows. For instance: Automatically assign a sales rep to customer record based on geographical region.

Validating Data

Enforcing data integrity is one of the most common use cases for UES. The beforeSubmit trigger can ensure records meet specific criteria before they are saved.

Modifying Records After Submission

Post-processing is critical when external actions need to be triggered based on record submission. afterSubmit can be used for updating related records, triggering notifications, or initiating workflows. For example: After a customer record is updated, send a notification to the customer success team.

Introduction to Client Scripts and Their Application in NetSuite Forms

What Are Client Scripts

Client Scripts are browser-side (client-side) JavaScript functions that run in response to user actions on a NetSuite form.

Unlike server-side User Event Scripts (UES), which operate on the backend, Client Scripts are executed in real-time within the user's browser, allowing for instant interaction with the user interface.

Client Scripts enhance the user experience by making forms more interactive, validating data dynamically, and providing real-time feedback before the record is submitted to the server.

When and Where Client Scripts are Used in NetSuite

Client Scripts are used on record forms (e.g., Sales Orders, Invoices, Customers) to manipulate data or control user actions based on events like loading a form, changing a field, saving a record, or clicking a button.

They are highly useful for making the NetSuite user interface more responsive and interactive by applying logic that directly affects the form's usability and behavior.

Client Script Events

List of all Client Script Entry Points

Common entry point events:

1. fieldChanged - executed when a field is changed on a record
2. saveRecord - executed after the submit button is pressed but before the form is submitted to the server
3. pageInit - Executes when page completes loading or form is reset

Client Scripts Use Case 1 - fieldChanged

Dynamic Field Population Based on Other Field Changes

- **Scenario:** When a user selects a customer on a Sales Order, auto-fill related information such as the customer's default sales rep, shipping address, and payment terms.

- **Action:** Write a `fieldChanged` script that listens for changes in the `customer` field and, once changed, dynamically populates the appropriate fields (sales rep, terms, etc.) by fetching this information from the selected customer record.

- **Benefit:** This reduces manual data entry and ensures consistent and accurate information across records.

Client Script Use Case 2 - saveRecord

Prevent Save If Mandatory Fields Are Missing

- **Scenario:** Ensure that all required fields are filled out before allowing the user to save the record.

- **Action:** In the `saveRecord` script, perform a final check to verify that fields like "Customer Email", "Billing Address", or "Item Details" are not blank. If any are missing, display a message to the user and prevent the save.

- **Benefit:** This prevents incomplete or invalid records from being submitted to the database, improving data quality.

Client Script Use Case 3 - pageInit

Pre-Fill Fields Based on User Role

- **Scenario:** When a sales rep opens a new Sales Order form, automatically pre-fill their name into the "Sales Rep" field.

- **Action:** Use the `pageInit` script to detect the user's role and, if they are a sales rep, pre-populate the "Sales Rep" field with their name.

- **Benefit:** This reduces data entry and ensures the right sales rep is assigned to every order.

Working with Form Objects in a Client Script

Interacting with Form Fields Dynamically
Client Scripts in SuiteScript 2.1 provide an API to interact with form fields dynamically, allowing developers to access, modify, and control the form's structure. Key methods include:

- **getFieldValue()/setFieldValue():** Retrieve or set the value of a specific field.
- **getFieldText()/setFieldText():** Retrieve or set the display text of a field (often used for dropdown fields).
- **getSublistValue()/setSublistValue():** Work with sublist fields by retrieving or setting their values based on the line number and column.

Understanding Script Deployment

Before an entry point script will run in your NetSuite account, it must be deployed. You can deploy a script when you create a script record, or you can deploy it later. The deployment settings available vary depending on the script type and on how you deploy the script.

When you deploy a script, NetSuite creates a script deployment record. Script deployment records are listed at *Customization > Scripting > Script Deployments*. Deployments are also listed on the Deployments subtab of the script record.

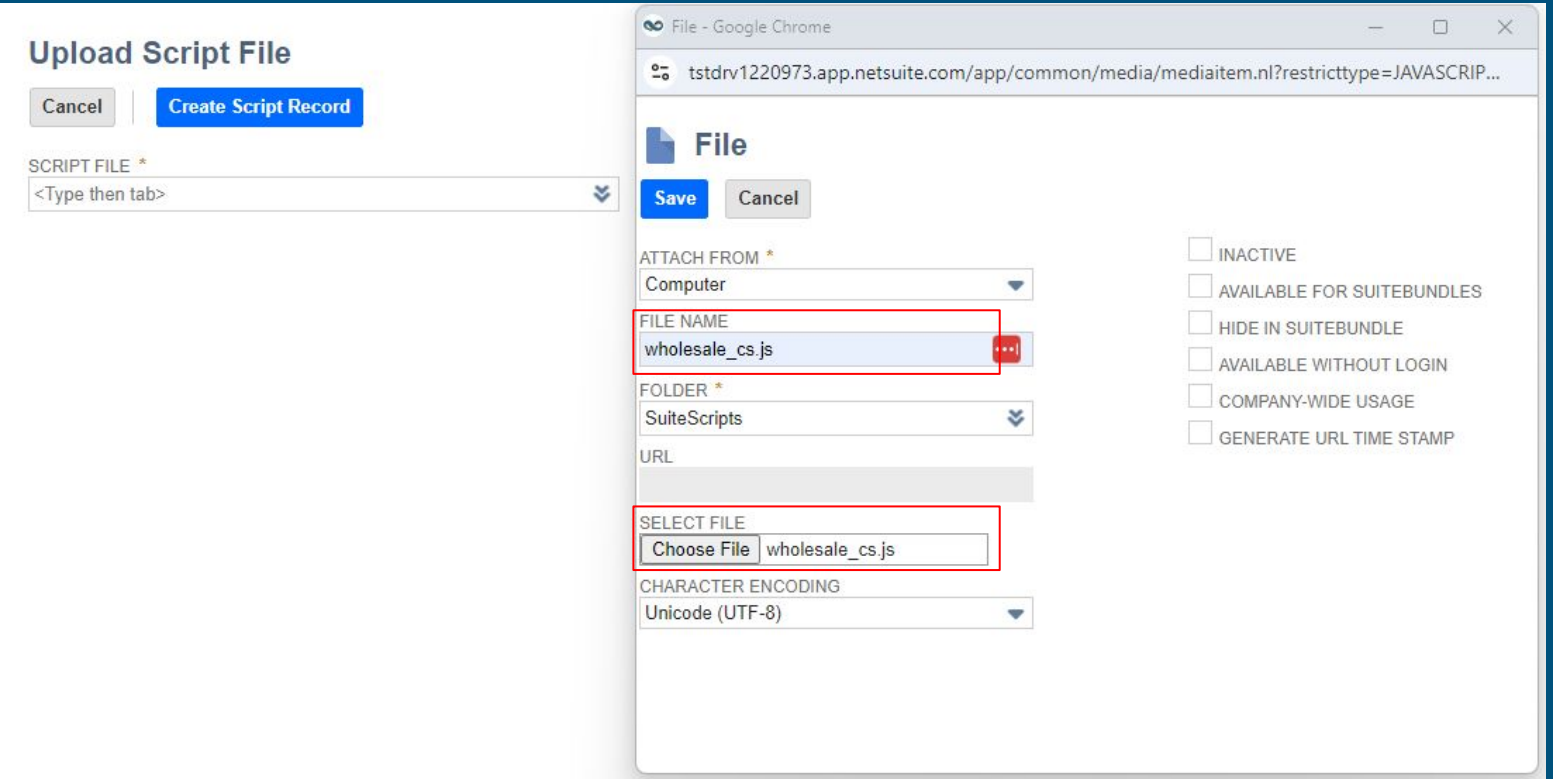
Multiple deployments can be created for the same script record. When multiple deployments exist, they are executed in the order in which they are listed on the Deployments subtab. This sequence typically corresponds with the order in which the deployments were created.

Step 1

Customization > Scripting > Scripts > new

Upload the script file > *save* > *Create Script Record*

Filename should match the name of selected file

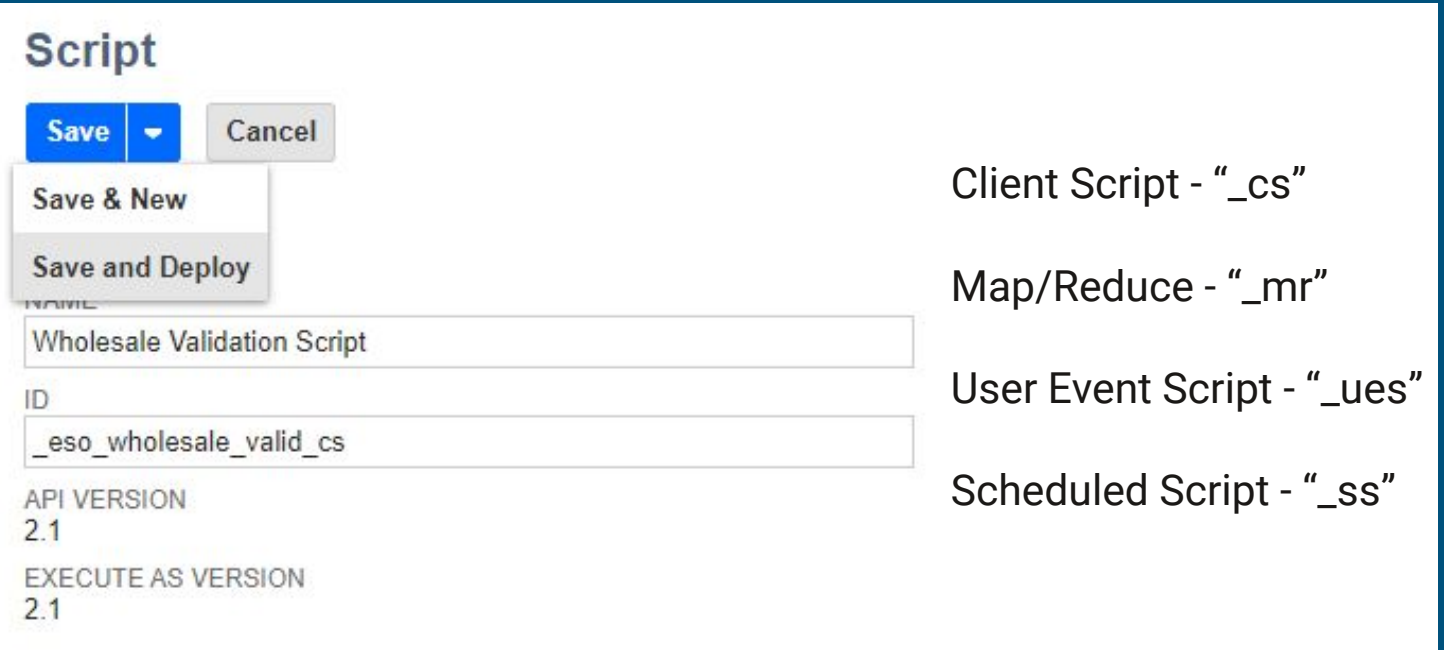


Step 2

Give your script a name and ID.

Save and Deploy

Best practice is to give a descriptive ID and include script type suffix



Client Script - “_cs”

Map/Reduce - “_mr”

User Event Script - “_ues”

Scheduled Script - “_ss”

Step 3: Deployment

Script Deployment

Save | Cancel

SCRIPT
Wholesale Validation Script

APPLIES TO *
Sales Order

Select the record type you want the script to run on

ID
eso_whlsl_valid_cs_dply

Give your deployment a unique ID

☑ DEPLOYED

You can turn each deployment on/off with the 'DEPLOYED' checkbox

Change status to 'Released'

STATUS *
Released

EVENT TYPE

LOG LEVEL
Debug

Set log level depending on log type used in your code

Audience • | Scripts • | Context Filtering • | Execution Log

Choose the audience - who /what this script will run for

ROLES ☐ Select All
- 01: Senior Executive
- 02: Engineering
- 03: Inside Sales
- 04: VP Sales

DEPARTMENTS
- Admin
- Consumer Electronics
- Engineering
- Executive Team
- Marketing

SUBSIDIARIES
- HH Inc.
- HH Inc. : Honeycomb Canada
- HH Inc. : Honeycomb EU
- HH Inc. : Honeycomb Mexico
- HH Inc. : Honeycomb Mfg.

GROUPS
- 3D Printer
- Assembly Cell
- Corporate East
- Corporate West
- Finishing

EMPLOYEES ☐ Select All
- Access User
- Amy Nguyen
- Anna Jackson
- Ashley Colbert
- Ben Saralegui

PARTNERS ☐ Select All
- Jasper Supply

Save | Cancel

Practical Exercises: Writing UES and Client Scripts

Exercise 1: Writing a User Event Script

Use Case:

Whenever a sales order is created for a customer, an email should be sent to that customer and all contacts associated with that customer record.

User Event Template

```
/**
* @NApiVersion 2.1
* @NScriptType UserEventScript
*/
define([], function() {
    function afterSubmit(context) {
    }

    return {
    afterSubmit: afterSubmit
    };
});
```

Identify which modules you need to use

N/record - record manipulation

N/email - Sending emails

N/search - gathering contact information

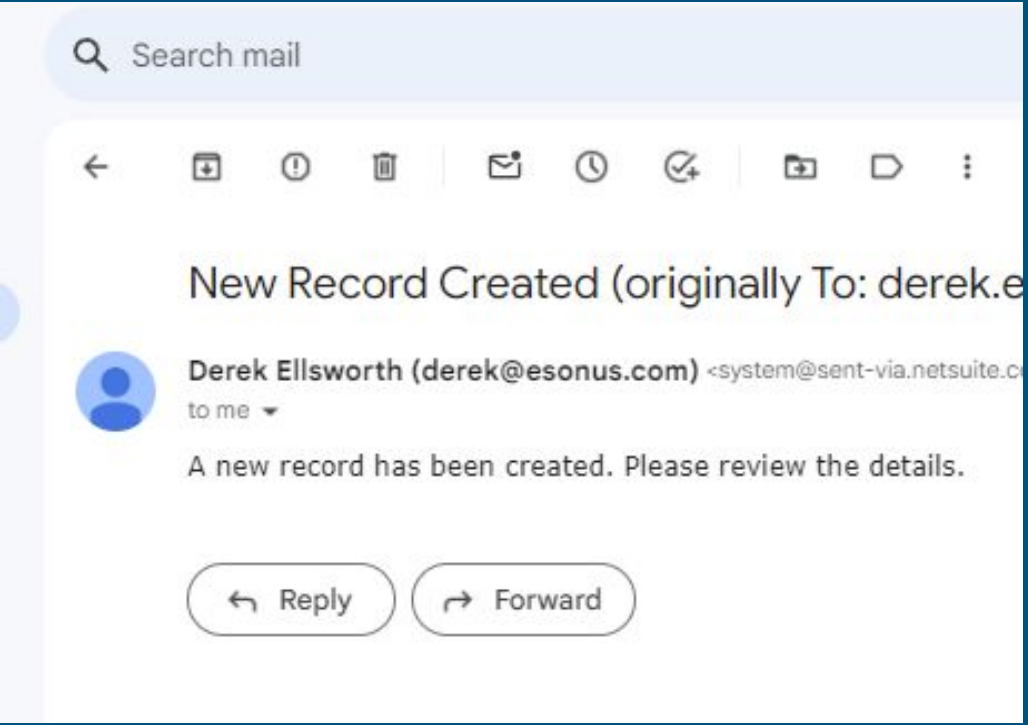
N/runtime - for sending the email from the currently logged in user

Try it yourself

It’s okay if you struggle! This is a more involved script than what we’ve covered before. Take this chance to try to come up with a solution yourself!

Continue to the next page for a link to our solution.

End Result Upon Transaction Creation



Solution: https://github.com/DerekEsonus/SampleSuiteScripts/blob/main/SampleUserEvent.js

Exercise 1: Writing a Client Script

Use Case:

Employees keep forgetting to enter their name and today's date on the memo field! Let's help them out by automating the process with a Client Script that runs on page initialization (pageInit).

Memo (memo): Employee name - Date

Client Script Template

```
/**
* @NApiVersion 2.1
* @NScriptType ClientScript
*/
define([], function() {
    function pageInit(context) {
    }


    return {
        pageInit: pageInit
    };
});
```

Identify which modules you need to use

- N/currentRecord - client side record manipulation
- N/runtime - getting information about the currently logged in user
- N/format - formatting the date for easier readability

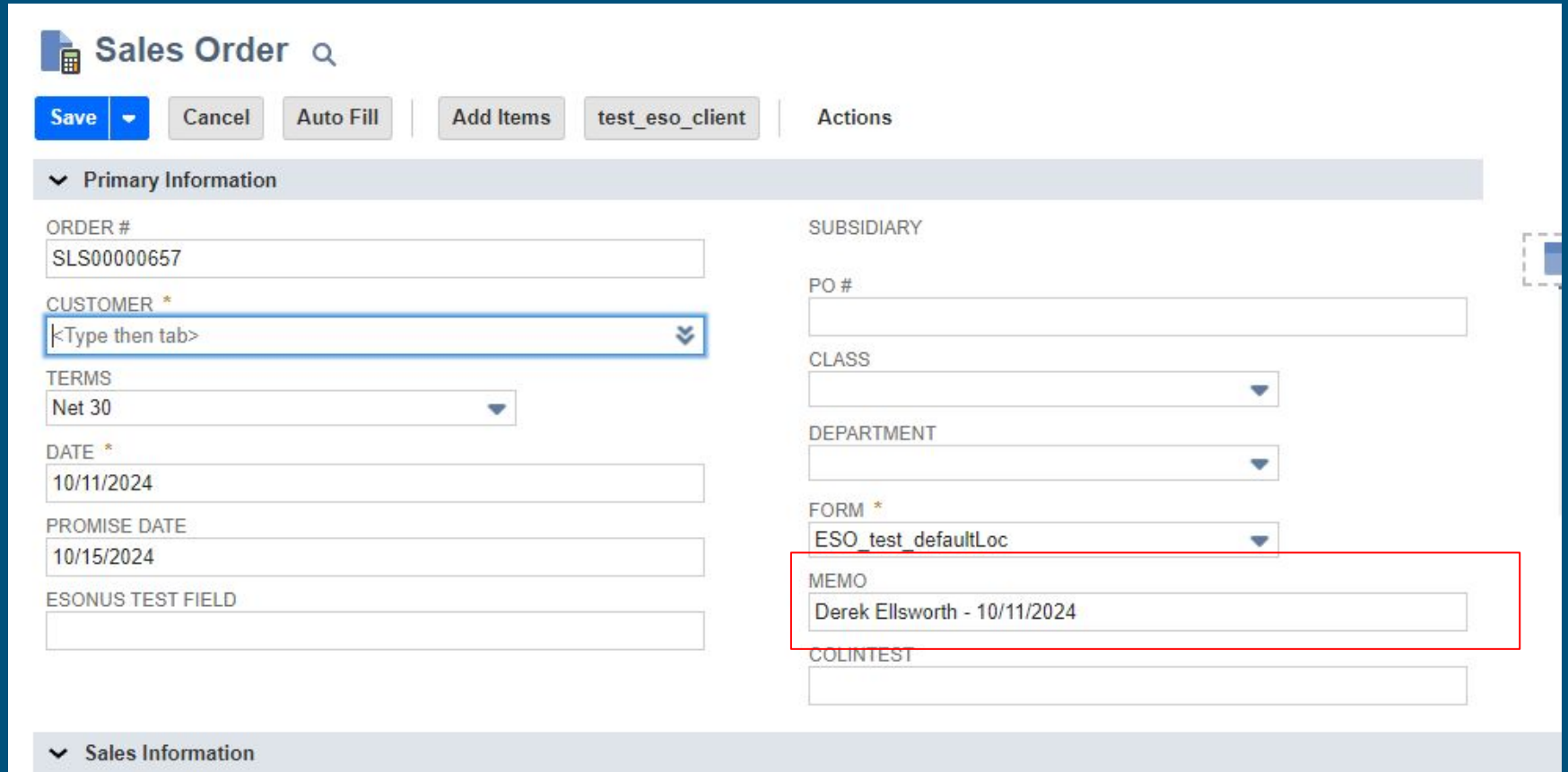
Try it yourself

It’s okay if you struggle! This is a more involved script than what we’ve covered before. Take this chance to try to come up with a solution yourself!

Continue to the next page for a link to our solution.

End Result

Solution: https://github.com/DerekEsonus/SampleSuiteScripts/blob/main/SampleClientScript.js



Recap + Review

This week we covered:

1. User Event scripts
    a. Execution contexts
        i. beforeLoad
        ii. beforeSubmit
        iii. AfterSubmit

2. Client Scripts
    a. Execution contexts

3. How to deploy scripts to your NetSuite account
4. Hands on exercises for both User Event Scripts and Client Scripts