Debugging Client Scripts

Lab 2.3 10-15 minutes

Lab Objectives

You will perform the following in this lab:

- Debug a Client Script using ServiceNow messages and the JavaScript alert() method.

A. Preparation

1. Locate and open the **Lab 2.3 Client Script Debugging** Client Script for editing.
2. Select the **Active** checkbox to make the script active.
3. Examine the pseudo-code for the script:
    - When the value of Impact changes, store the current value of State in the variable incState.
        - If the value of incState is 1
            - Add a decoration to the State field.
            - Have the State label flash the color aqua.
        - If Impact has the value High or Medium
            - Remove the State choice list options On Hold, Resolved, Closed and Canceled.
            - Log a message to the top of the form confirming the State options removal.
        - Else if the form is not loading and Impact does not have the value High or Medium
            - Clear the State choice list.
            - Add options back to the State choice list.
            - Set the State field to the value of incState.
4. Consider saving the record as a favorite as you will be opening it a few times throughout the lab. Select **Create Favorite** on the form's Context menu.
5. Select **Update**.

B. Observe the Current Execution of the Script

1. Create a new Incident.
2. Without changing the value of the State field, open the drop-down choice list and make a mental note of the available options.
3. Update the value of Impact to **1-High**.
4. Confirm the **Lab 2.3 Client Script Debugging** Client Script executed as expected.

Expected Behavior	Occurred?
A decoration appeared to the left of the **State** field.	No
The **State**'s label flashed aqua for four seconds.	No
The **State** choice list options now only include 'New' and 'In Progress'.	✓
An Information Message appears at the top of the form stating, "The State options have been removed".	✓

C. Use the *alert()* method to confirm Variable Values and Script Execution

1. Open the **Lab 2.3 Client Script Debugging** Client Script.
2. Use the alert() method in strategic places to debug your script. If you require assistance, you can use this script as an example of how to include alert() messages in your script.

```
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || newValue == '') {
        return;
    }

    //Document the current value of State, if New, use a decoration
    var incState = g_form.getValue('incident');
    alert("<your initials> - the value of incState is: " + incState);

    if (incState == 1) {
        alert("<your initials> LINE 11 executed");
        g_form.addDecoration('state', 'icon-edit','Gathering initial details');
        g_form.flash('state', "aqua", -4);
    }

```

3. Select **Update**.

D. Force the Updated Script to Execute

1. Create a new Incident. Save the record to stay on the form.
2. Update the value of Impact to **1-High**.
3. Review the alert messages.
    - What is the value currently in the **incState** variable? _____
    - What should it be? _____
    - Does the string "LINE 11 EXECUTED!" appear as an alert? _____
4. What can you conclude regarding the debugging output currently in the alert?

    _____

5. At this point, you probably know something is wrong with the value in the incState variable. Before leaving the Incident form, double-check the **State** field name by right-clicking the field's label. What is the exact spelling of the **State** field name? _____
6. Open the **Lab 2.3 Client Script Debugging** Client Script.
7. Locate the statement at the beginning of your script that reads:

    var incState = g_form.getValue('incident');

    Review the **State** field name in this statement. Does it match the field name you previously documented? _____
8. Update the statement to:

    var incState = g_form.getValue('state');
9. Select **Update**.
10. Force your updated script to execute by creating a new Incident.
11. Update the value of Impact to **1-High**.
12. Review the messages in the alert.
    - Is the value of *incState* **1**? _____
    - Does the string "LINE 11 EXECUTED!" appear as an alert? _____

13. Confirm the Lab 2.3 Client Script Debugging Client Script executed as expected.

Expected Behavior	Occurred?
A decoration appeared to the left of the **State** field.	✓
The State's label flashed aqua for four seconds.	✓
The **State** choice list options now only include 'New' and 'In Progress'.	✓
An Information Message appears at the top of the form confirming the State options removal.	✓

14. If you answered **No** to any of these questions in steps 12 or 13, debug, and re-test.

15. Review the **InfoMessage** at the top of the form. Are you the only person who can see this script output?

    _____

16. Form messages and alerts are good debugging strategies as the results are instantly presented at the top of the form you are testing on. Would this type of debugging strategy be best in a development or production instance?

    _____

17. Update the value of Impact to **3-Low**.

18. Examine the built-in debugging provided by the platform below the Impact field. Try correcting any errors in the script and re-test until no errors remain.

Lab Completion

Well done! You have successfully debugged a script using client-side debugging strategies and the alert method.