JSON Object

Lab 8.5 — 25-30 minutes

Lab objectives

You will perform the following in this lab:

- Use the Script Include and a copy of the Client Script you wrote in the previous lab.
- Modify the Script Include to return a JSON object containing an array of Group members.
- Modify the Client Script to dynamically select a group member from the JSON object to assign to an Incident.
- Complete a Cloud Dimensions' Phase II requirement.

Business Problem

A few IT Analysts are reporting the same individuals are continually being assigned Incidents on their teams. They would like to ensure support responsibilities are spread across the team, as everyone is busy working on other items in addition to their support responsibilities. After some discussions, management has agreed to implement a strategy that will randomly assign Analysts to Incidents, ensuring everyone has a turn.

Project requirement

To implement this strategy, two scripts will be required:

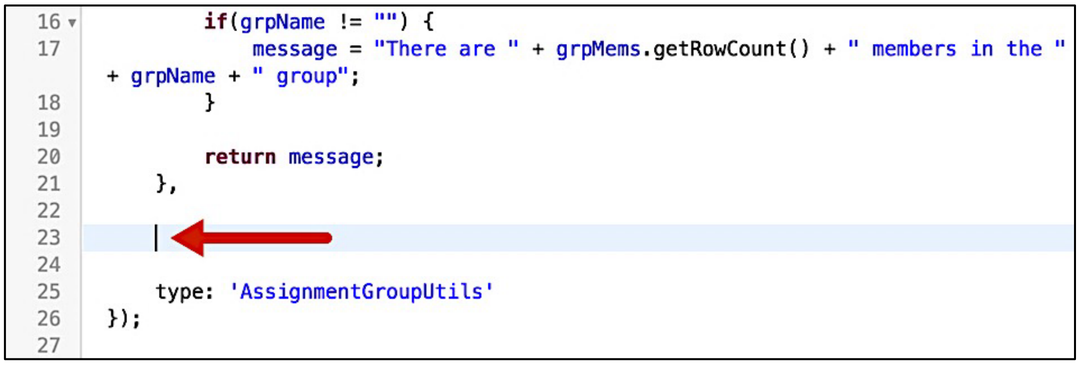
1. First, create a Script Include to randomly select an Analyst from the identified Assignment group and return that name to the calling Client Script.
2. Create a Client Script that calls the Script Include method whenever the value in the Assignment group changes. The name of the randomly selected Analyst returned from the server should then populate the Assigned to field.

> **Lab Dependency:** *You must complete the scripts in **Lab 8.4 Number of Group Members** to do this lab.*

A. Add a New Method to an Existing Script Include

1. Open the **AssignmentGroupUtils** Script Include.

2. Place your cursor after the comma at the end of the countGroupMembers function. Press **Enter <return>** on your keyboard two times to create some blank lines. *(The blank lines will leave some space between functions for easy readability.)*

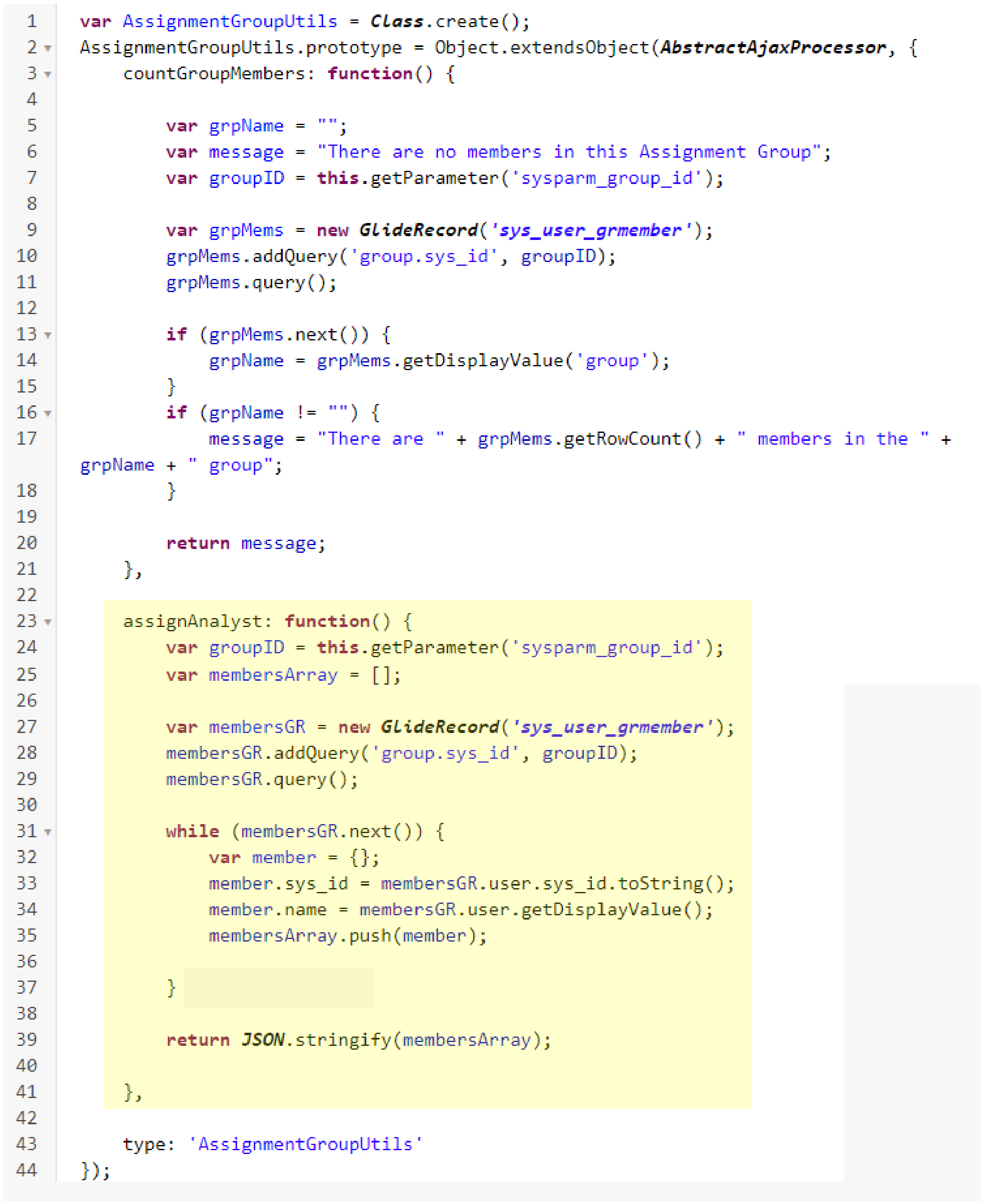


```
        if(grpName != "") {
            message = "There are " + grpMems.getRowCount() + " members in the "
+ grpName + " group";
        }

        return message;
    },

    |

    type: 'AssignmentGroupUtils'
});
```

3. Examine the pseudo-code for the method you will ADD to the Script Include:

- Add a method to the object called assignAnalyst.
    - Store the sys_id of a group from a Client Script in the variable groupID.
    - Create a new array.
    - Query the sys_user_grmember table to return records for all members of the group passed in from a Client Script.
    - While there are returned records
        - Create a new object to store member information.
        - Add the sys_id for the member to the new object.
        - Add the Display Value for the member to the new object.
        - Push the object into the array.
    - Return the array of members as a JSON object.

4. Write the highlighted script after the blank lines you inserted in step 2.



```
var AssignmentGroupUtils = Class.create();
AssignmentGroupUtils.prototype = Object.extendsObject(AbstractAjaxProcessor, {
    countGroupMembers: function() {

        var grpName = "";
        var message = "There are no members in this Assignment Group";
        var groupID = this.getParameter('sysparm_group_id');

        var grpMems = new GlideRecord('sys_user_grmember');
        grpMems.addQuery('group.sys_id', groupID);
        grpMems.query();

        if (grpMems.next()) {
            grpName = grpMems.getDisplayValue('group');
        }
        if (grpName != "") {
            message = "There are " + grpMems.getRowCount() + " members in the " + grpName + " group";
        }

        return message;
    },

    assignAnalyst: function() {
        var groupID = this.getParameter('sysparm_group_id');
        var membersArray = [];

        var membersGR = new GlideRecord('sys_user_grmember');
        membersGR.addQuery('group.sys_id', groupID);
        membersGR.query();

        while (membersGR.next()) {
            var member = {};
            member.sys_id = membersGR.user.sys_id.toString();
            member.name = membersGR.user.getDisplayValue();
            membersArray.push(member);

        }

        return JSON.stringify(membersArray);

    },

    type: 'AssignmentGroupUtils'
});
```

5. Select **Update**.

B. Call the Script Include from a Client Script

1. Open the **Lab 8.4 Number of Group Members** Client Script.

2. Select **Insert and Stay** on the form's Context menu to make a copy of the record and remain on the form.

3. Re-configure the new Client Script trigger:

   Name: **Lab 8.5 Assign Analyst**
   Table: **Incident [incident]**
   Active: **Selected (checked)**

4. Examine the pseudo-code for the Client Script:

   - Create an instance of the GlideAjax object called AssignmentGroupUtils.
   - Add a param to call the assignAnalyst method.
   - Add a param to pass the sys_id of the Assignment group value to the Script Include.
   - Use the getXMLAnswer method and a callback function in your Client Script to execute the Script Include.
   - Pass the response returned from the Script Include to the callback function.
     - Locate the response variable in the returned XML and store the value in a variable.
     - Create an object from the JSON formatted string and store it in the variable named members.
     - Find the length of the array.
     - If the array has elements
       - Generate a random number between zero and one less than the number of array elements.
       - Set the assigned_to field to the array element selected by the random number.
     - Else
       - Clear any value currently in the assigned_to field.
       - Generate an alert that says the selected Assignment group has no members.

5. Most of the code for this Client Script already exists from the previous lab. Use the image on this page as a guide to make the following modifications.

   a) Modify the statement calling the method name to call the **assignAnalyst** method.

   b) Change getXML(membersParse) to getXMLAnswer(membersParse) on line 9.

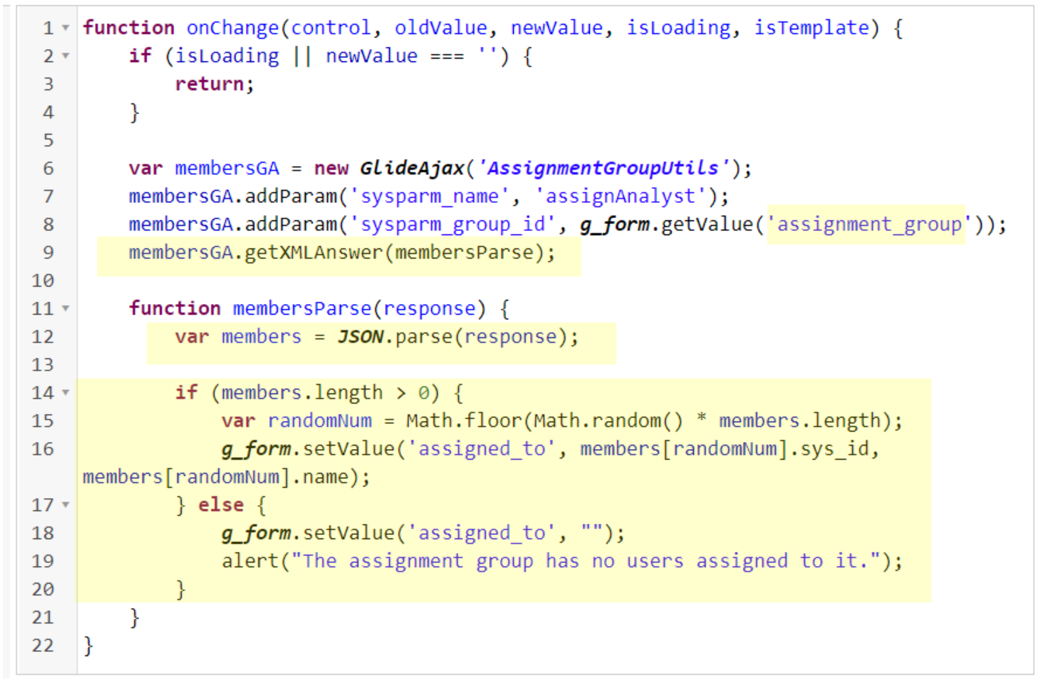
c) In the callback function…

   i. Delete the var myObj = response.responseXML.documentElement.getAttribute('answer'); and alert(myObj); statements.

   ii. Use the JSON.parse() method to create an object from the returned JSON formatted string.

   iii. Add an 'if' statement to check if the array has elements. Generate a random number and make the assignment.

   iv. Add an else statement to clear the assigned_to field and advise if the group has no members.



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

    var membersGA = new GlideAjax('AssignmentGroupUtils');
    membersGA.addParam('sysparm_name', 'assignAnalyst');
    membersGA.addParam('sysparm_group_id', g_form.getValue('assignment_group'));
    membersGA.getXMLAnswer(membersParse);

    function membersParse(response) {
        var members = JSON.parse(response);

        if (members.length > 0) {
            var randomNum = Math.floor(Math.random() * members.length);
            g_form.setValue('assigned_to', members[randomNum].sys_id,
members[randomNum].name);
        } else {
            g_form.setValue('assigned_to', "");
            alert("The assignment group has no users assigned to it.");
        }
    }
}
```

6. Select **Update**.

C. Test Your Work

1. Open an Incident record with a populated **Assigned to** field.

2. Examine the value of the **Assigned to** field. Record the value here:

   ____

3. Change the value in the **Assignment group** field.

4. Was the **Assigned to** field populated with the name of an Assignee from that group? If not, debug and re-test.

5. Record the name of the Analyst currently in the **Assigned to** field here: ________________________

6. Change the value in the Assignment group field to **IT Securities**. (The IT Securities group has no members baseline.)

7. Did the value in the **Assigned to** field clear? If not, debug and re-test.

8. Did an alert advise there are no members in this group? If not, debug and re-test.

9. Make the Lab 8*.5 Assign Analyst* Client Script for this lab **inactive.**

Lab Completion

Fantastic! You have successfully added a second method to an existing Script Include. The assignAnalyst() method uses the passed in Assignment group's sys_id in a GlideRecord query to build an array containing all the sys_ids of the group members. It then returns the object to the client.

You used the GlideAjax class in a Client Script to call the new method and use the returned object to randomly assign incidents to group members ensuring the group's work is shared fairly. Although is anything really random in this world?