User Object Cheat Sheet

N

o matter what system you’re working in, it is always critical to be able to identify information about the user who is accessing that system. Being able to identify who the user is, what their groups and/or roles are, and what other attributes their user record has are all important pieces of information that allow you to provide that user with a good experience (without giving them information they don’t need to have or shouldn’t have). ServiceNow gives administrators some pretty simple ways to identify this information in the form of a couple of user objects and corresponding methods. This article describes the functions and methods you can use to get information about the users accessing your system.  
  
  
  
GlideSystem User Object

The GlideSystem (gs) user object is designed to be used in any server-side JavaScript (Business rules, UI Actions, System security, etc.). The following table shows how to use this object and its corresponding functions and methods.

| Function/Method | Return Value | Usage |
| --- | --- | --- |
| gs.getUser() | Returns a reference to the user object for the currently logged-in user. | var userObject = gs.getUser(); |
| gs.getUserByID() | Returns a reference to the user object for the user ID (or sys\_id) provided. | var userObject = gs.getUser().getUserByID('employee'); |
| gs.getUserName() | Returns the User ID (user\_name) for the currently logged-in user. e.g. 'employee' | var user\_name = gs.getUserName(); |
| gs.getUserDisplayName() | Returns the display value for the currently logged-in user. e.g. 'Joe Employee' | var userDisplay = gs.getUserDisplayName(); |
| gs.getUserID() | Returns the sys\_id string value for the currently logged-in user. | var userID = gs.getUserID(); |
| getFirstName() | Returns the first name of the currently logged-in user. | var firstName = gs.getUser().getFirstName(); |
| getLastName() | Returns the last name of the currently logged-in user. | var lastName = gs.getUser().getLastName(); |
| getEmail() | Returns the email address of the currently logged-in user. | var email = gs.getUser().getEmail(); |
| getDepartmentID() | Returns the department sys\_id of the currently logged-in user. | var deptID = gs.getUser().getDepartmentID(); |
| getCompanyID() | Returns the company sys\_id of the currently logged-in user. | var companyID = gs.getUser().getCompanyID(); |
| getCompanyRecord() | Returns the company GlideRecord of the currently logged-in user. | var company = gs.getUser().getCompanyRecord(); |
| getLanguage() | Returns the language of the currently logged-in user. | var language = gs.getUser().getLanguage(); |
| getLocation() | Returns the location of the currently logged-in user. | var location = gs.getUser().getLocation(); |
| getDomainID() | Returns the domain sys\_id of the currently logged-in user (only used for instances using domain separation). | var domainID = gs.getUser().getDomainID(); |
| getDomainDisplayValue() | Returns the domain display value of the currently logged-in user (only used for instances using domain separation). | var domainName = gs.getUser().getDomainDisplayValue(); |
| getManagerID() | Returns the manager sys\_id of the currently logged-in user. | var managerID = gs.getUser().getManagerID(); |
| getMyGroups() | Returns a list of all groups that the currently logged-in user is a member of. | var groups = gs.getUser().getMyGroups(); |
| isMemberOf() | Returns true if the user is a member of the given group, false otherwise. | Takes either a group sys\_id or a group name as an argument.  if(gs.getUser().isMemberOf(current.assignment\_group)){ //Do something... }  var isMember = gs.getUser().isMemberOf('Hardware');  To do this for a user that isn't the currently logged-in user...  var user = 'admin'; var group = "Hardware"; if (gs.getUser().getUserByID(user).isMemberOf(group)){ gs.log( gr.user\_name + " is a member of " + group); }  else{ gs.log( gr.user\_name + " is NOT a member of " + group); } |
| gs.hasRole() | Returns true if the user has the given role, false otherwise. | if(gs.hasRole('itil')){ //Do something... } |
| gs.hasRole() | Returns true if the user has one of the given roles, false otherwise. | if(gs.hasRole('itil,admin')){ //If user has 'itil' OR 'admin' role then Do something... } |
| hasRoles() | Returns true if the user has any roles at all, false if the user has no role (i.e. an ess user). | if(!gs.getUser().hasRoles()){ //User is an ess user... } |

It is also very simple to get user information even if the attribute you want to retrieve is not listed above by using a ‘gs.getUser().getRecord()’ call as shown here…

*//This script gets the user's title*  
gs.getUser().getRecord().getValue('title');

g\_user User Object

The g\_user object can be used only in UI policies and Client scripts. Contrary to its naming, it is not truly a user object. g\_user is actually just a handful of cached user properties that are accessible to client-side JavaScript. This eliminates the need for most GlideRecord queries from the client to get user information (which can incur a fairly significant performance hit if not used judiciously).

| g\_user Property or Method | Return value |
| --- | --- |
| g\_user.userName | User name of the current user e.g. employee |
| g\_user.firstName | First name of the current user e.g. Joe |
| g\_user.lastName | Last name of the current user e.g. Employee |
| g\_user.userID | sys\_id of the current user e.g. 681ccaf9c0a8016400b98a06818d57c7 |
| g\_user.hasRole() | True if the current user has the role specified, false otherwise. ALWAYS returns true if the user has the 'admin' role.  Usage: g\_user.hasRole('itil') |
| g\_user.hasRoleExactly() | True if the current user has the exact role specified, false otherwise, regardless of 'admin' role.  Usage: g\_user.hasRoleExactly('itil') |
| g\_user.hasRoles() | True if the current user has at least one role specified, false otherwise.  Usage: g\_user.hasRoles('itil','admin') |

It is often necessary to determine if a user is a member of a given group from the client as well. Although there is no convenience method for determining this from the client, you can get the information by performing a GlideRecord query. Here’s an example…

*//Check to see if assigned to is a member of selected group*  
**var** grpName = 'YOURGROUPNAMEHERE';  
**var** usrID = g\_form.userID; *//Get current user ID*  
**var** grp = **new** GlideRecord('sys\_user\_grmember');  
grp.addQuery('group.name', grpName);  
grp.addQuery('user', usrID);  
grp.query(groupMemberCallback);  
     
**function** groupMemberCallback(grp){  
*//If user is a member of selected group*  
    **if**(grp.next()){  
        *//Do something*  
        alert('Is a member');  
    }  
    **else**{  
        alert('Is not a member');  
    }  
}

To get any additional information about the currently logged-in user from a client-script or UI policy, you need to use a GlideRecord query. If at all possible, you should use a server-side technique described above since GlideRecord queries can have performance implications when initiated from a client script. For the situations where there’s no way around it, you could use a script similar to the one shown below to query from the client for more information about the currently logged in user.

*//This script gets the user's title*  
**var** gr = **new** GlideRecord('sys\_user');  
gr.**get**(g\_user.userID);  
**var** title = gr.title;  
alert(title);