

# Service Now Development

## Scripting Locations in Service Now

1. Business Rules
2. Client Scripts
3. UI Actions
4. Script Include
5. UI policies
6. Workflow Scripting
7. Scheduled Jobs
8. API s
9. Where to customization in service now
10. Transform Maps
11. UI Pages & UI Macros
12. Web Services
13. Service Portal Widgets
14. More.....

## Glide Record

### Section Outline

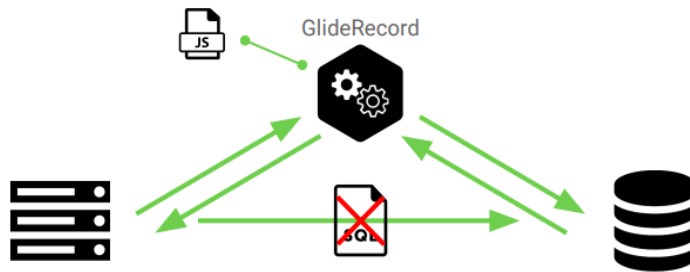
1	GlideRecord Introduction	7	Walking Through CRUD
2	Show Me The Code!	8	GlideRecord Demo
3	Concept: Dot-Walking	9	GlideRecordSecure
4	GlideRecord API Diagram	10	GlideAggregate
5	Common GlideRecord Methods	11	Where Can I Use This?
6	Stages Of A GlideRecord	12	Section Recap

## What is Glide Record?

Glide Record is a special Java class (GlideRecord.java) that can be used in JavaScript exactly as if it was a native JavaScript class.

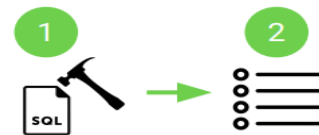
1. It is used for database operations instead of writing **SQL queries**.
2. It is an object that contains **zero or more records from one table**. Another way to say this is that a Glide Record is an ordered list.
3. A Glide Record contains both records (**rows**) and fields (**columns**). The field names are the same as the underlying database column names.

## Application Storage



## Glide Record Introduction

- Most common API
- Server side
- Used for database operations (CRUD)
- Generates SQL for you
- 2 stages
  - Building a query
  - Process records



## Show Me The Code!

- Print a list of all priority 1 incidents to the screen

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority' , 1);
3 incidentGR.query();
4 while (incidentGR.next()) {
5     gs.info(incidentGR.number)
6 }
```

- Print a list of all priority 1 incidents to the screen using by filter condition

<input type="checkbox"/>	<a href="#">i</a>	<a href="#">INC0000007</a>	2015-08-12 16:08:24	Need access to sales DB for the West	<a href="#">Joe Employee</a>	● 1 - Critical	2018-06-15 05:44:15
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">INC00000051</a>	2018-04-01 13:48:32	Manager can't access SAP Controlling application	<a href="#">Joe Employee</a>	● 1 - Critical	2018-06-15 05:44:15
<input type="checkbox"/>	<a href="#">i</a>	<a href="#">INC00000031</a>	2017-12-25 16:18:03	Need help with Remedy. Can we configure UI?	<a href="#">Joe Employee</a>	● 1 - Critical	2018-04-22 12:44:20

## Glide Record by Analogy: Grocery Shopping

1. Go to a grocery store
2. Grab a shopping cart
3. Place groceries in the shopping cart
4. Checkout at cashier



Table



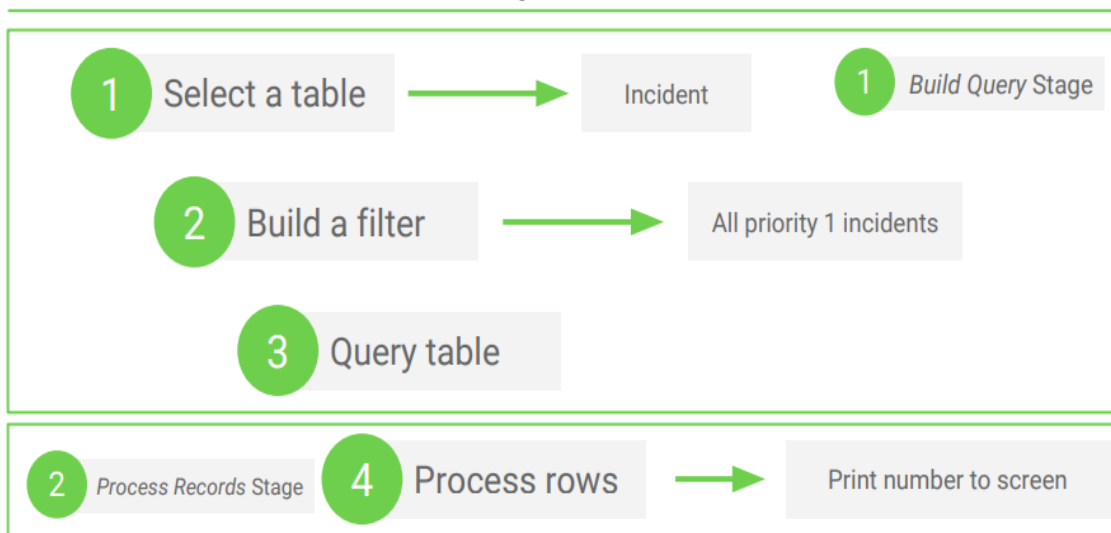
GlideRecord

Records



Action

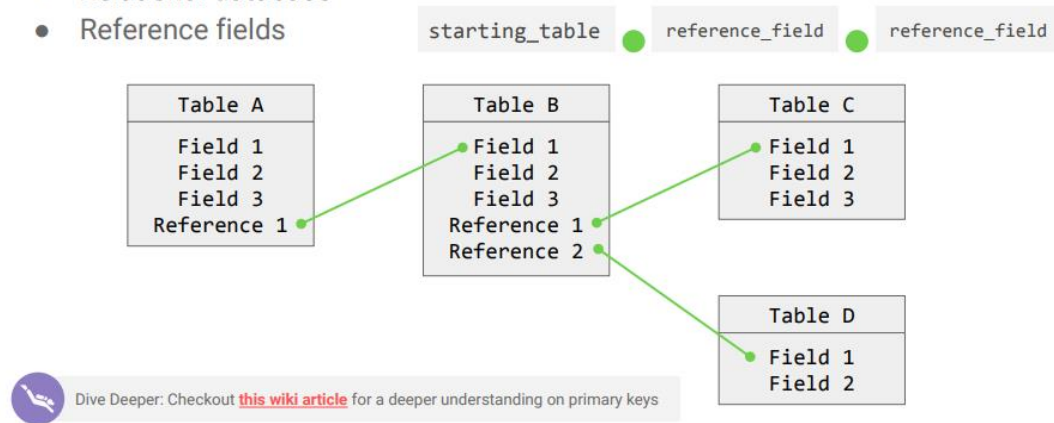
## General Glide Record Steps



Note: Any ServiceNow table may be queried with GlideRecord

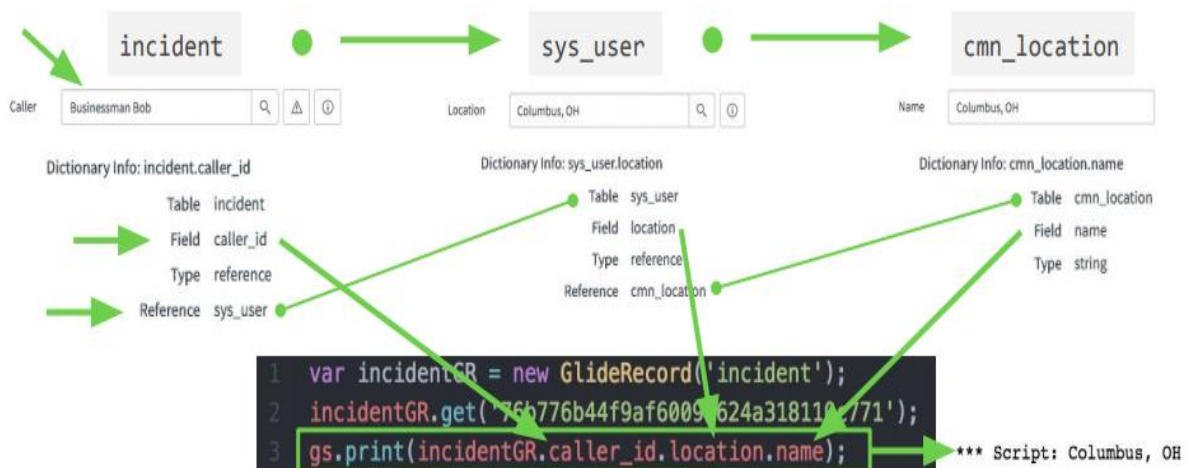
## Concept: Dot-Walking

- Relational database
- Reference fields



## Example: Dot-Walking

- Example:
  - For a specific incident, you would like to find the location of the caller



## Glide Record API Diagram



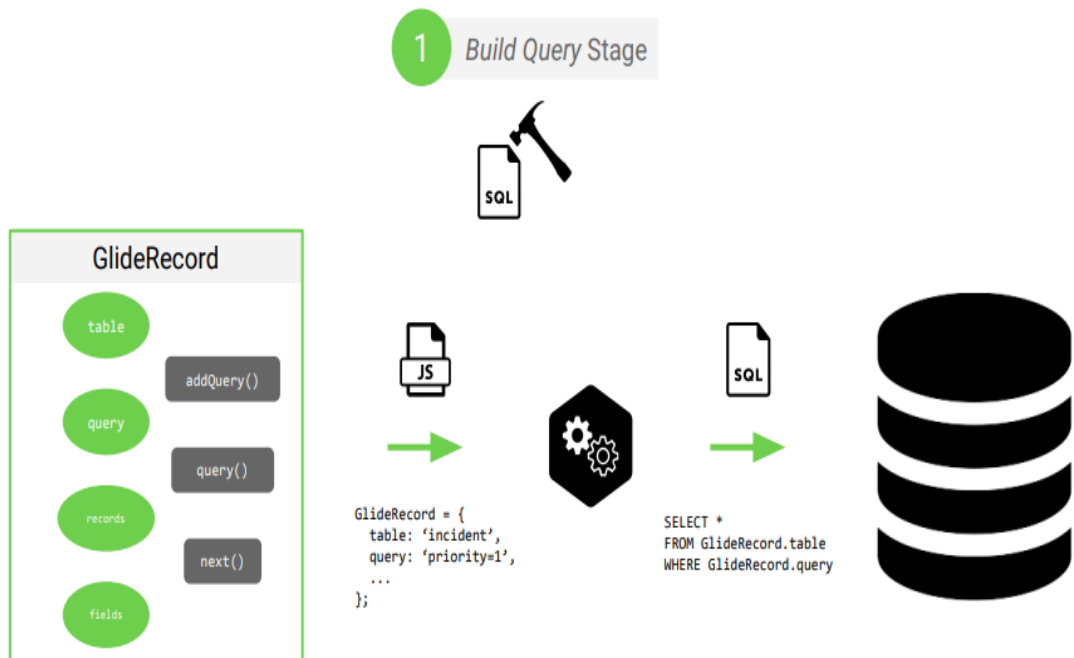
## Glide Record API Mapping



## Common Glide Record Methods

- query()
- newRecord()
- insert()
- update()
- deleteRecord()
- addQuery()
- addEncodedQuery()
- hasNext()
- next()
- get()
- orderBy()
- orderByDesc()
- canCreate()
- canWrite()
- canRead()

## Glide Record Stage 1: Building Query



## Glide Record Stage 1: Options to Build Queries

1

### Chain Methods

- `addQuery()`
- `addOrCondition()`
- `addNullQuery()`
- `addNotNullQuery()`
- `addActiveQuery()`
- `addInactiveQuery()`

2

### Encoded Query

- `addEncodedQuery()`

## Glide Record Stage 1: Option 1 - Chain Methods

1

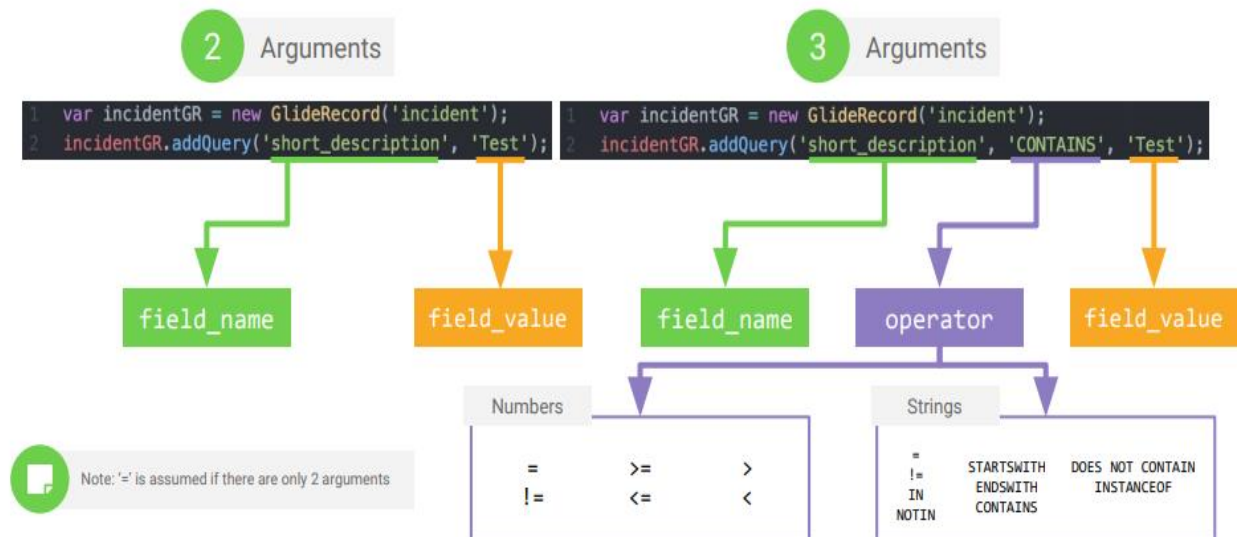
### Chain Methods

Add GlideRecord methods onto the current GlideRecord object

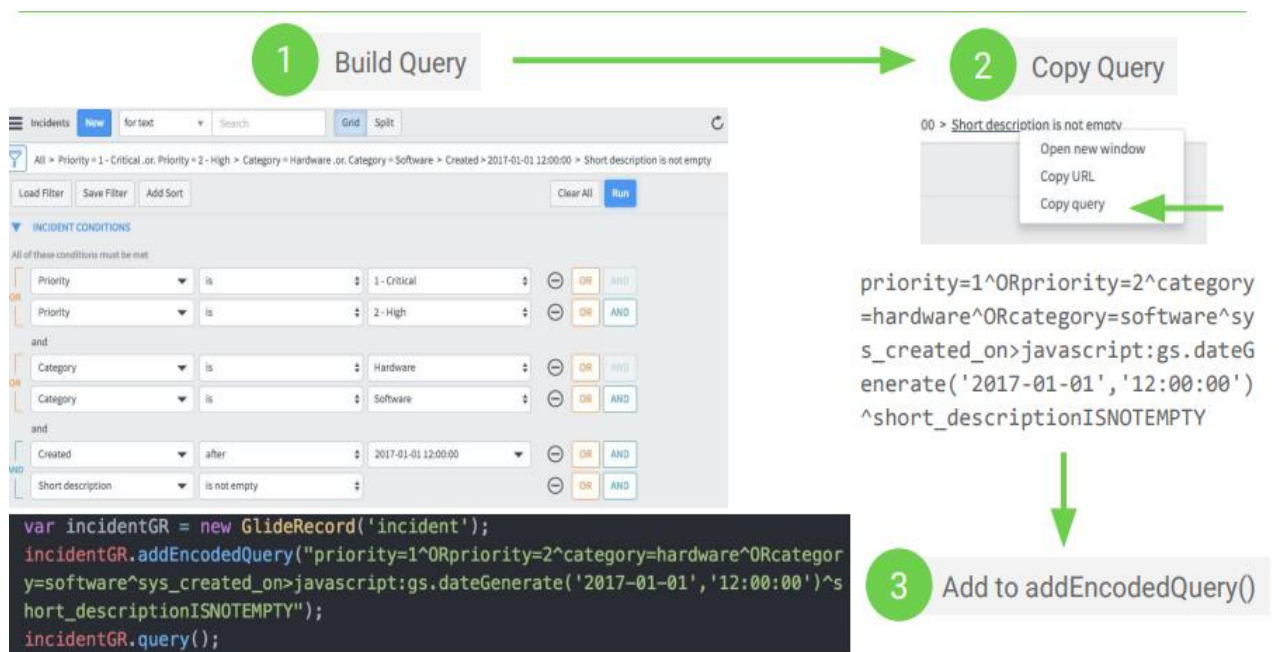
```
1 var incidentGR = new GlideRecord('incident');
2 var orCond1 = incidentGR.addQuery('priority', '1');
3 orCond1.addOrCondition('priority', '2');
4 var orCond2 = incidentGR.addQuery('category', 'hardware');
5 orCond2.addOrCondition('category', 'software');
6 incidentGR.addQuery('sys_created_on', '>', '2017-01-01 12:00:00');
7 incidentGR.addNotNullQuery('short_description');
8 incidentGR.query();
```

## Glide Record addQuery() Method

- Accepts 2 or 3 arguments

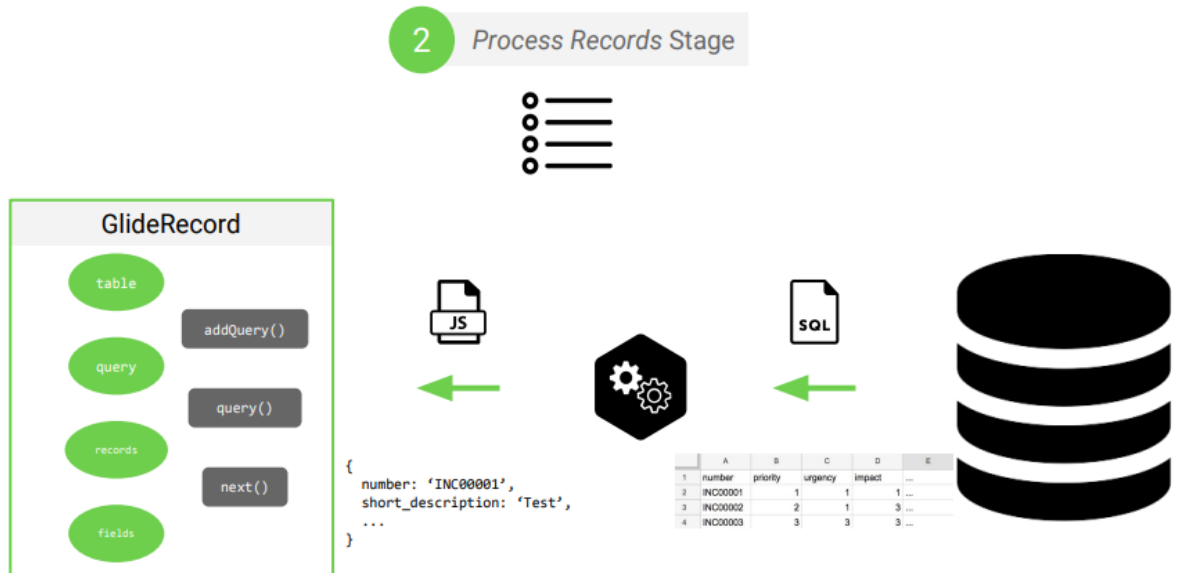


## Glide Record Stage 1: Option 2 - Encoded Query

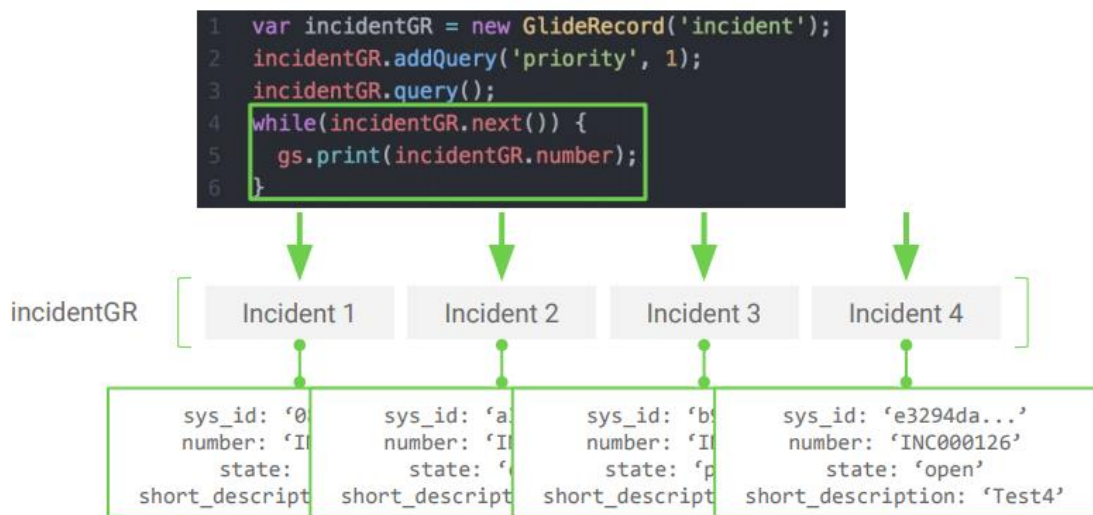




## Glide Record Stage 2: Process Records



## Glide Record next() Method & Iteration

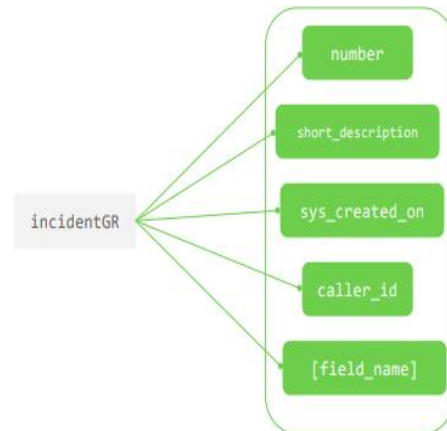


Dive Deeper: Checkout [this wiki article](#) and this [YouTube video](#) if you'd like to learn more on iterators

## Accessing a Record's Fields

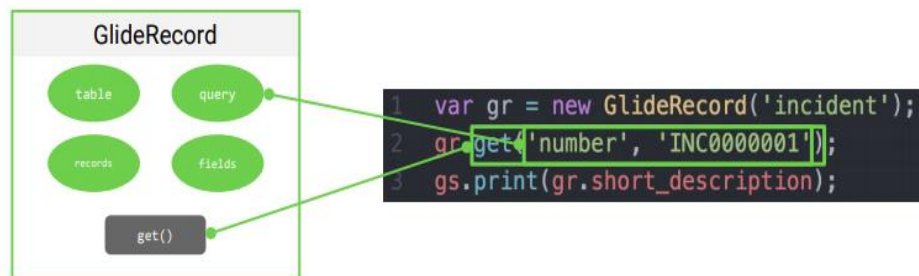
- Once query() method is executed and stage 2 begins, all fields are just a dot away
- Fields become GlideRecord properties

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority', 1);
3 incidentGR.query();
4 while(incidentGR.next()) {
5   gs.print(incidentGR.number);
6 }
```

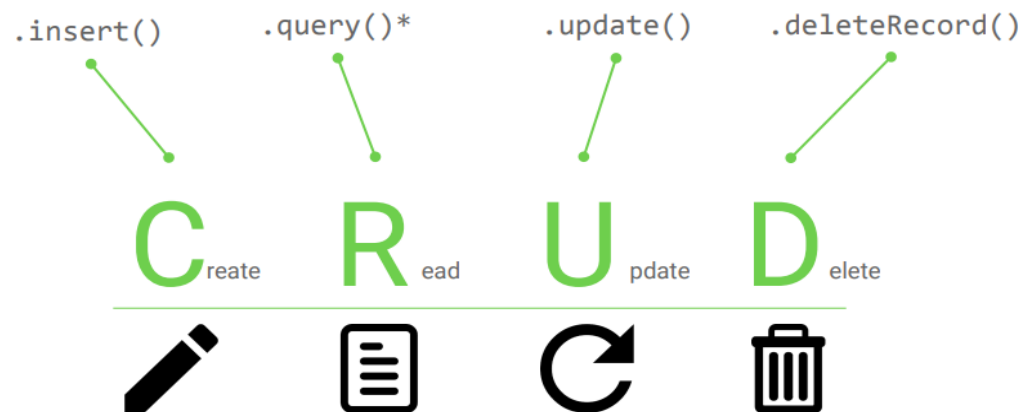


## Glide Record get() Method

- Shortcut
- Only grabs 1 record
- Commonly used with record sys\_id



## CRUD Glide Record Mapping



## CRUD - Create

1. Build GlideRecord
2. `query()`
3. `newRecord()`
4. Set field values
5. `insert()`

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 incidentGR.newRecord();
4 incidentGR.short_description = 'Testing 123'
5 incidentGR.insert();
```

**Note:** After executed this script check in incident table

## CRUD - Read

1. Build Glide Record
2. Add filter conditions (optional)
3. query()
4. next()
5. Print or copy variables

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority' , 1);
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.info(incidentGR.number);
6 }
```

## CRUD - Update

1. Build Glide Record
2. Add filter conditions (optional)
3. query()
4. next()
5. Set field values
6. update()

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority' , 1);
3 incidentGR.query();
4 while(incidentGR.next()); {
5     //updating the record
6     incidentGR.description = '2'
7     incidentGR.update();
8 }
```

## CRUD - Delete

1. Build GlideRecord
2. Add filter conditions (optional)
3. query()
4. next()
5. deleteRecord()

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('number', 'INC0000054');
3 incidentGR.query();
4 while(incidentGR.next()){
5     incidentGR.deleteRecord();
6 }
```

## Glide Record Secure

- Glide Record Secure class is inherited from Glide Record
  - ✓ Has all of the same methods
  - ✓ Performs ACL checking
- Used to secure Script Includes
- Replaces canWrite(), canRead(), canUpdate(), canDelete() GlideRecord methods



Dive Deeper: Watch [episode 15 of TechNow](#) to learn more about GlideRecordSecure

## Working with gs.print ()method

```
1 gs.print('Hello World')
```

**Result: Just print the Output is “Hellow World”**

## Adding two numbers a simple program:

```
1  var a =10;
2  var b =20;
3  var c = a+b;
4  gs.print(c);
```

**Result:** given the total = 30

## Working with query () method:

```
1  var incidentGR = new GlideRecord('incident');
2  incidentGR.query();
3  while(incidentGR.next()) {
4      gs.print(incidentGR.number)
5  }
```

**Result:** Display all the incident numbers in a specified table

## Working with addQuery () method

**Example 1:**

```
1  var incidentGR = new GlideRecord('incident');
2  incidentGR.addQuery('short_description', 'CONTAINS', 'Test');
3  incidentGR.query();
4  while(incidentGR.next()){
5      gs.print(incidentGR.number);
6  }
```

**Result:** Given the incident number where the description field is “Test” value

**Example 2:**

```

1  var incidentGR = new GlideRecord('incident') ;
2  incidentGR.addQuery('priority', '<=', 1);
3  incidentGR.query();
4  while (incidentGR.next()) {
5      gs.print(incidentGR.number);
6  }

```

### Example 3:

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.addQuery('active', true);
3  incidentGR.query();
4  while(incidentGR.next()) {
5      gs.log('Category is ' + incidentGR.category);
6  }

```

## Working addOrCondition () method

```

1  var incidentGR = new GlideRecord('incident');
2  var orincidentGR = incidentGR.addQuery('state', 6);
3  orincidentGR.addOrCondition('state', 7);
4  incidentGR.query();
5  while(incidentGR.next()) {
6      gs.log('Category is ' + incidentGR.category + ' : ' + incidentGR.number);
7  }

```

## Working with next () method

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.query();
3  gs.print(incidentGR.number)

```

**Result: Empty**

```

1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 incidentGR.next();
4 gs.print(incidentGR.number)

```

**Result: Given first incident number from incident Table**

## Working with addQuery () method

```

1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority' , 1);
3 incidentGR.query();
4 while(incidentGR.next()){
5     gs.print('priority 1 incidents: ' + incidentGR.number + ' : ' + incidentGR.priority);
6 }

```

**Result: Given all priority 1 incidents from incident Table**

```

1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority' , 1);
3 incidentGR.query();
4 while(incidentGR.next()){
5     gs.print('priority 1 incidents: ' + incidentGR.number + ' : ' + incidentGR.priority.getDisplayValue());
6 }

```

## Working with get () method

```

1 var incidentGR = new GlideRecord('incident');
2 incidentGR.get('097a1f22dbda13007fa185184b96190b');
3 gs.print(incidentGR.number + ' : ' + incidentGR.short_description)

```

**Result: Given incident number and short description of sys\_id**



```

1 var incidentGR = new GlideRecord('incident');
2 incidentGR.get('097a1f22dbda13007fa185184b96190b');
3 gs.print(incidentGR.number + ' has a sys_id of ' + incidentGR.sys_id);

```

## Working with addEncodedQuery () method

Step 1: Navigate to Incident list view

Step 2: Build the query using by filter condition shown below → Run

All of these conditions must be met

Prioritys	is	1 - Critical	AND	OR	X
Active	is	true	AND	OR	X
Assigned to	is	Fred Luddy	AND	OR	X

Step 3: Copy the query from list view

**Copy Query = priority=1^active=true^assigned\_to=5137153cc611227c000bbd1bd8cd2005**

```

1 var queryString = 'priority=1^active=true^assigned_to=5137153cc611227c000bbd1bd8cd2005';
2 var incidentGR = new GlideRecord('incident');
3 incidentGR.addEncodedQuery(queryString);
4 incidentGR.query();
5 while(incidentGR.next()){
6     gs.print(incidentGR.number);
7 }

```

**Result: Given incidents based on you condition query matched**

## Working with newRecord insert () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.newRecord();
3 incidentGR.short_description = 'This incident was created from background scripts';
4 incidentGR.insert();
```

**Result: A new record is created in Incident table and navigates to list view to confirm**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.newRecord();
3 incidentGR.short_description = 'This incident was created from background scripts';
4 var newIncidentSysId = incidentGR.insert();
5 gs.print(newIncidentSysId);
```

## Working with create multiple records insert () method

```
1 var newIncidents = [];
2 var counter = 1;
3 var incidentGR = new GlideRecord('incident');
4 while(counter <= 5) {
5     incidentGR.newRecord();
6     incidentGR.short_description = 'Incident #' + counter;
7     counter++;
8     newIncidents.push(incidentGR.insert());
9 }
10 gs.print(newIncidents);
```

**Result: Created multiple records then check in to your list view**

## Working with delete records () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('short_description' , 'Incident #1');
3 incidentGR.query();
4 while(incidentGR.next()){
5     incidentGR.deleteRecord()
6 }
```

**Result: Deleted a record then check in to your list view**

## Working with orderBy () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.orderBy('short_description');
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.short_description);
6 }
```

**Result: Returned all the incidents short\_description value in Order by Ascending**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.orderBy('short_description');
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.number + ' : ' + incidentGR.short_description);
6 }
```

## Working with orderByDesc () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.orderByDesc('short_description');
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.number + ' : ' + incidentGR.short_description);
6 }
```

**Result: Returned all the incidents short\_description value in Order by Descending**

## Working with setLimit () method

```
1 var problemGR = new GlideRecord('problem');
2 problemGR.setLimit(10);
3 problemGR.query()
4 while(problemGR.next()) {
5     gs.print(problemGR.number)
6 }
```

**Result: Returned 10 records from problem table for setLimit**

## Working with setLimit () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addEncodedQuery('priority=1')
3 incidentGR.setLimit(10);
4 incidentGR.query()
5 while(incidentGR.next()) {
6     gs.print(incidentGR.number)
7 }
```

### Example 2:

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addEncodedQuery('priority=1');
3 incidentGR.setLimit(10);
4 incidentGR.orderBy('short_description');
5 incidentGR.query();
6 while(incidentGR.next()) {
7     gs.print(incidentGR.number + ' : ' + incidentGR.short_description);
8 }
```

## Working with Access Control List canCreate, canWrite, canRead, canDelete

```
1 var problemGR = new GlideRecord('problem');
2 problemGR.query();
3 if(problemGR.canCreate() && problemGR.canRead() && problemGR.canWrite() && problemGR.canDelete() ) {
4     gs.print('I have access to ,Create ,Read Update, and Delete');
5 }
```

## Working with getRowCount () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 gs.print(incidentGR.getRowCount());
```

**Result: Returned total records in incident Table**

## Working with hasNext () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 gs.print(incidentGR.hasNext());
4
```

**Result: Returned bullion value as a True**

**Example 2:**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 gs.print(incidentGR.next());
```

**Example 3: This is not working properly the hasNext method**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 if(incidentGR.hasNext()){
4 gs.print(incidentGR.number);
5 }
```

**Example 4: This will work properly and returned first record from incident table**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.query();
3 if(incidentGR.next()){
4 gs.print(incidentGR.number);
5 }
```

**Example 5:**

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('priority',0);
3 incidentGR.query();
4 gs.print(incidentGR.hasNext());
```

## Working with get () method

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.get('number','INC0000039')
3  gs.print(incidentGR.number);

```

**Result: Returned incident number as provided in get method**

## Working with getLink () method

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.get('number','INC0000039')
3  gs.print(incidentGR.getLink());

```

**Result:**

incident.do?sys\_id=471bfbc7a9fe198101e77a3e10e5d47f&sysparm\_stack=incident\_list.do?sysparm\_query=active=true

## Working with deleteMultiple () method

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.addEncodedQuery('short_descriptionLIKEincident #');
3  incidentGR.deleteMultiple();

```

**Result: Deleted multiple records form expected table**

## Working with update () method to change urgency

**Example 1:**

```

1  var incidentGR = GlideRecord('incident');
2  incidentGR.get('number', 'INC0000018');
3  incidentGR.urgency = 2;
4  incidentGR.update();

```

**Result: Urgency value has been changed**

**Example 2:**

```

1  var incidentGR = new GlideRecord('incident');
2  incidentGR.addQuery('urgency', 2);
3  incidentGR.query();
4  while(incidentGR.next()) {
5      gs.print(incidentGR.number)
6  }

```

### Example 3:

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addQuery('urgency', 2);
3 incidentGR.query();
4 while(incidentGR.next()) {
5     incidentGR.urgency = 3;
6     incidentGR.update();
7 }
```

### Working with addNullQuery () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addNullQuery('short_description');
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.number);
6 }
```

**Result:** Return all records where the description field value is null

### Working with addNotNullQuery () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.addNotNullQuery('short_description');
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.number);
6 }
```

**Result:** Return all records where the description field contain the value

### Working with chooseWindow () method

```
1 var incidentGR = new GlideRecord('incident');
2 incidentGR.chooseWindow(10, 20);
3 incidentGR.query();
4 while(incidentGR.next()) {
5     gs.print(incidentGR.number);
6 }
```

**Result:** Choose Window will return all records between the first parameter(inclusive) and the second parameter(exclusive), so this example will return the 10 incidents between record 10-19 both inclusive. Works with orderBy

## Working with addInfoMessage () method

```
1  var gr = new GlideRecord('incident');
2  var queryString = "priority=1^ORpriority=2";
3  gr.addEncodedQuery(queryString);
4  gr.query();
5  while (gr.next()) {
6      gs.addInfoMessage(gr.number);
7  }
```

## Working with addJoinQuery() method

Example 1:

```
1  var prob = new GlideRecord('problem');
2  prob.addJoinQuery('incident');
3  prob.query();
4  while(prob.next()) {
5      gs.print(prob.number);
6  }
```

Example 2:

```
1  // Look for Problem records
2  var incidentGR = new GlideRecord('problem');
3  // That have associated Incident records
4  var grSQ = incidentGR.addJoinQuery('incident');
5  // Where the Problem records are "active=false"
6  incidentGR.addQuery('active', 'true');
7  // And the Incident records are "active=true"
8  grSQ.addCondition('active', 'true');
9  // Query
10 incidentGR.query();
11 // Iterate and print results
12 while (incidentGR.next()) {
13     gs.print(incidentGR.getValue('number'));
14 }
```



**Result: Find inactive problems with associated incidents**

**Just by you want to know caller is VIP or not methods**

```
1 var inc = new GlideRecord('incident');
2 inc.addQuery('number','INC0000019');
3 inc.query();
4 if(inc.next()){
5   if(inc.caller_id.vip == true)
6   {
7     gs.print("Caller is VIP");
8   }
9   else{
10    gs.print("Caller is not VIP");
11  }
12 }
```

**Secure canCreate , canCreate, canUpade, canDelete () methods**

**Example 1:**

```
1 var gr = new GlideRecord('incident');
2 gs.info(gr.canCreate());
```

**Example 2:**

```
1 var gr = new GlideRecord('incident');
2 gs.info(gr.canRead());
```

**Example 3:**

```
1 var gr = new GlideRecord('incident');
2 gs.info(gr.canDelete());
```

**Example 3:**

```
1 var gr = new GlideRecord('incident');  
2 gs.info(gr.canUpdate());  
3
```

```
1 var elementName = 'short_description';  
2 var incidentGR = new GlideRecord('incident');  
3 incidentGR.initialize();  
4 incidentGR.setValue(elementName, "My Net work cable is not working properly");  
5 incidentGR.insert();  
6 gs.info(incidentGR.getElement('short_description'));
```

# Glide Form

## Section Outline

1	GlideForm Introduction	7	GlideUser Introduction
2	Show Me The Code!	8	Show Me The Code!
3	Client Side Environment	9	GlideUser API Diagram
4	GlideForm API Diagram	10	GlideUser Methods
5	Common GlideForm Methods	11	GlideUser Demo
6	GlideForm Demo	12	Section Recap

## Glide Form Introduction

- Run from client-side
- Changes to form & fields
- Referenced by `g_form`



*The GlideForm API provides methods to customize forms... The global object `g_form` is used to access GlideForm methods. GlideForm methods are only used on the client... These methods are used to make custom changes to the form view of records.*

[ServiceNow Docs](#)



The Glide Form API provides methods to customize forms. **GlideForm.js** is the JavaScript class containing the methods. The global object **g\_form** is used to access Glide Form methods. Glide Form methods are only used on the client.

These methods are used to make custom changes to the form view of records. All validation of examples was done using Client Scripts.

Some of these methods can also be used in other client scripts (such as Catalog Client Scripts or Wizard Client Scripts), but must first be tested to determine whether they will work as expected.

**Note:** The methods **getControl()**, **getHelpTextControl()**, **getElement()**, and **getFormElement()** are deprecated for mobile devices. For information on using GlideForm for mobile, see [Mobile Client GlideForm \(g\\_form\) Scripting and Migration](#) .

## Show Me The Code!

- ✓ Set short description to mandatory when priority changes.

```
1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {  
2   if (isLoading || newValue === '')  
3     return;  
4   g_form.setMandatory('short_description', True);  
5 }
```

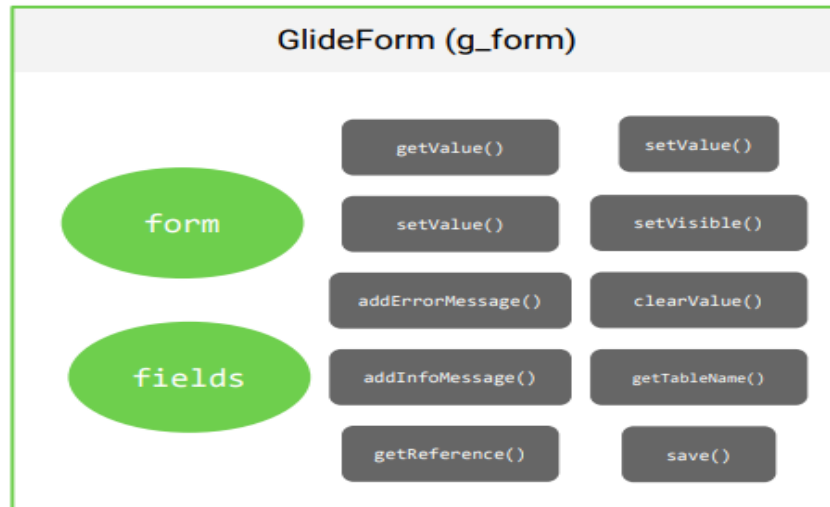
Type	onChange ▼
Field name	Prioritys ▼

## Client-Side Environment

- Access to client-side APIs
  - ✓ Most are accessible via global scope
- Ctrl + Shift + j
- Debug to browser console
  - ✓ console.log()
  - ✓ console.dir()

**Caution:** Avoid manipulating the DOM directly in Service Now w/ client scripts and UI actions, use `g_form` instead

## Glide Form API Overview



## Common Glide Form Methods

- addInfoMessage()
- addErrorMessage()
- addOption()
- clearOptions()
- clearValue()
- disableAttachments()
- enableAttachments()
- getLabelOf()
- getOption()
- getReference()
- hideRelatedLists()
- isMandatory()
- removeOption()
- setDisabled()
- setReadOnly()
- setVisible()
- setValue()

## Working with getValue() & setValue() Methods

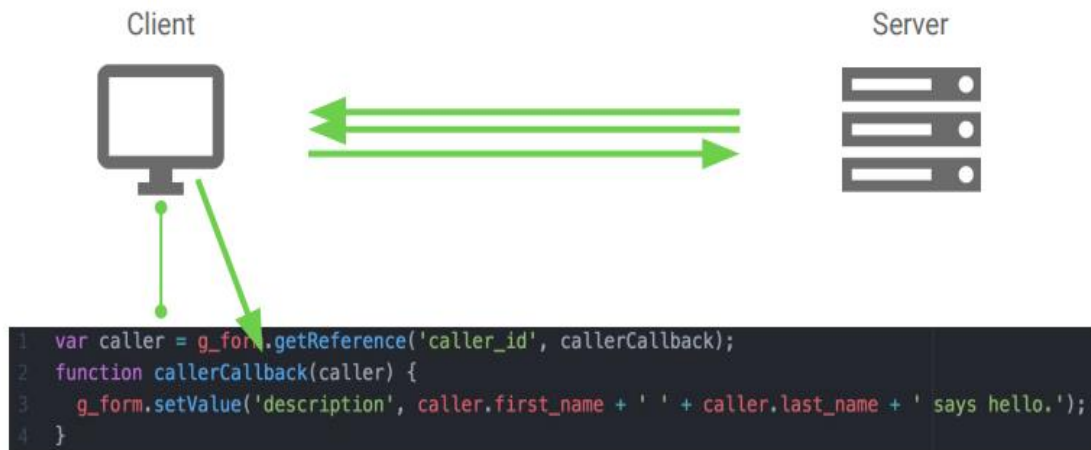
- ✓ Getter and setter methods for form fields
- ✓ getValue() accepts the field name as an argument
- ✓ setValue() accepts the field name and new value as arguments

```
1 var category = g_form.getValue('category');  
2  
3 var newCategory = 'software';  
4 g_form.setValue('category', newCategory);
```

Number	INC0020188
Caller	<input type="text"/>
Location	<input type="text"/>
Category	Inquiry / Help
Subcategory	-- None --
Configuration Item	<input type="text"/>

## Working with getReference() Method

- Form only loads fields associated with record on form
- Use getReference() to retrieve referenced field values
- Leverages JavaScript callbacks



## Working with getValue () Method

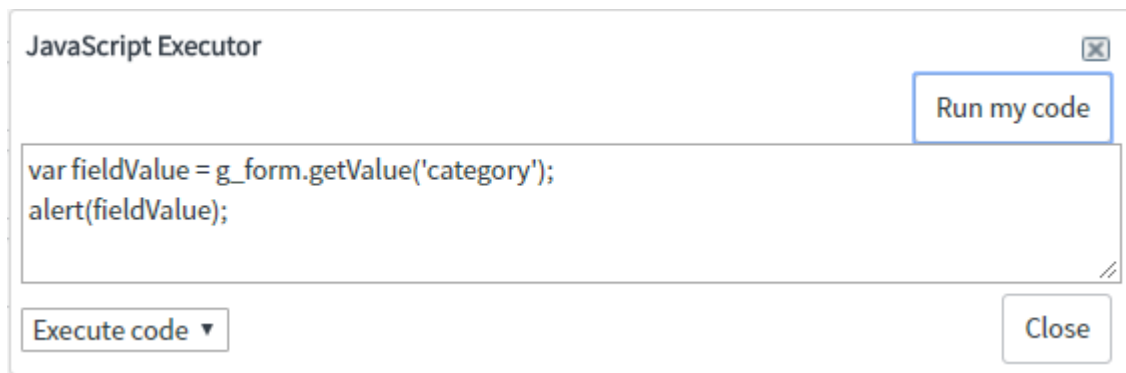
**Step 1: Ctr+Shift+j**

**Step 2: Open ATOM**

**Write code** `var fieldValue = g_form.getValue('category');`

`alert(fieldValue);`

**Step 3: Copy code and paste into JavaScript Excutor**



```
1 var fieldValue = g_form.getValue('category');
2 alert(fieldValue);
```

**Result: Get Current category field value on the form**

## Working with setValue () Method

```
1 g_form.setValue('category', 'network');
```

**Result: Set Current category field value on the form**

## Working with clearValue () Method

```
1 g_form.clearValue('category');
```

**Result: Clear the value in category field**



## Working with save() Method

```
1 g_form.save();
```

**Result: Saves current form**

## Working with setDisabled() Method

```
1 g_form.setDisabled('category', true);
```

```
1 g_form.setDisabled('category', false);
```

**Result: Disabled form fields**

## Working with hideRelatedLists() and showRelatedLists() Method

```
1 g_form.hideRelatedLists();
```

**Result: Hide related lists on form**

```
1 g_form.showRelatedLists();
```

**Result: Show related lists on form**

## Working with isMandatory() Method

Example 1:

```
1 alert(g_form.isMandatory('category'));
```

Example 1:

```
1 g_form.setMandatory('category', true);  
2 alert(g_form.isMandatory('category'));
```

Example 3:

```
1 g_form.setMandatory('category', true);  
2 alert(g_form.isMandatory('category'));  
3 g_form.clearValue('category');
```

## Working with isMandatory() Method

```
1 var isNewRecord = g_form.isNewRecord();  
2 alert('is New Record ?' + isNewRecord);
```

## Working with addInfoMessage() and addInfoMessage() Method

```
1 g_form.addInfoMessage('Hellow Thank you')
```

```
1  g_form.addErrorMessage('Not Valid')
```

## Working with clearMessage () Method

```
1  g_form.clearMessage();
```

## Working with getLabelOf () Method

```
1  alert(g_form.getLabelOf('category');
```

## Glide User

### Section Outline

---

7 GlideUser Introduction

8 Show Me The Code!

9 GlideUser API Diagram

10 GlideUser Methods

11 GlideUser Demo

12 Section Recap

### Glide User Introduction

- Client-side
- User information
- Referenced by g\_user
- Relatively small & simple API

“

*The GlideUser API provides access to information about the current user and current user roles. Using the GlideUser API avoids the need to use the slower GlideRecord queries to get user information.*

[ServiceNow Docs](#)

”

## Glide User

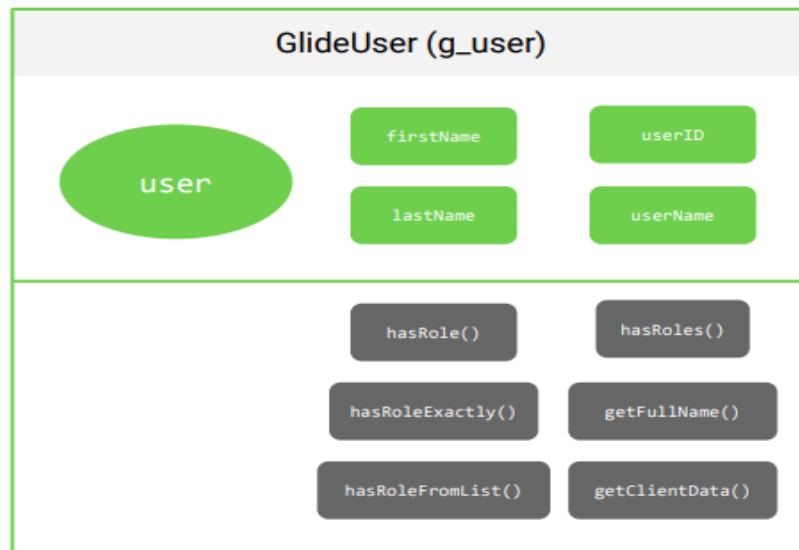
- Contains name and role information about the current user.
- It is typically used in client scripts and UI policies but is also found in UI actions that
- Cannot be used in business rules or UI actions that run on the server.
- Avoids the need for Glide Record queries to get user information.
- Session information about the current user
- And current user roles are contained in the client (web browser).
- All Glide User methods except **getClientData()** access the session information that is available by default.
- The **getClientData()** method requires setup on the server and use of **putClientData()** to make session information available.

## Show Me The Code!

- Check if the current user has the ITIL role

```
1 var hasITIL = g_user.hasRole('itil');  
2 if(!hasITIL) {  
3     alert('You dont have sufficient privilages.');
```

## Glide User API Overview



## Glide User Methods & Properties

- `firstName`
- `lastName`
- `userID`
- `userName`
- `getClientData()`
- `getFullName()`
- `hasRole()`
- `hasRoleExactly()`
- `hasRoleFromList()`
- `hasRoles()`

## Glide User hasRoleExactly() Method

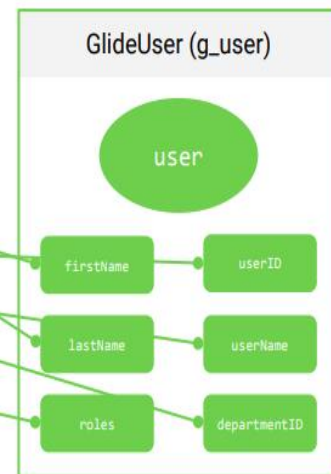
- Takes the name of a role as its argument
- Returns true only if user has the specified role, even if current user is admin

```
1 console.log(g_user.hasRole('approval_admin')); /* returns true */  
2 console.log(g_user.hasRoleExactly('approval_admin')); /* returns false */
```

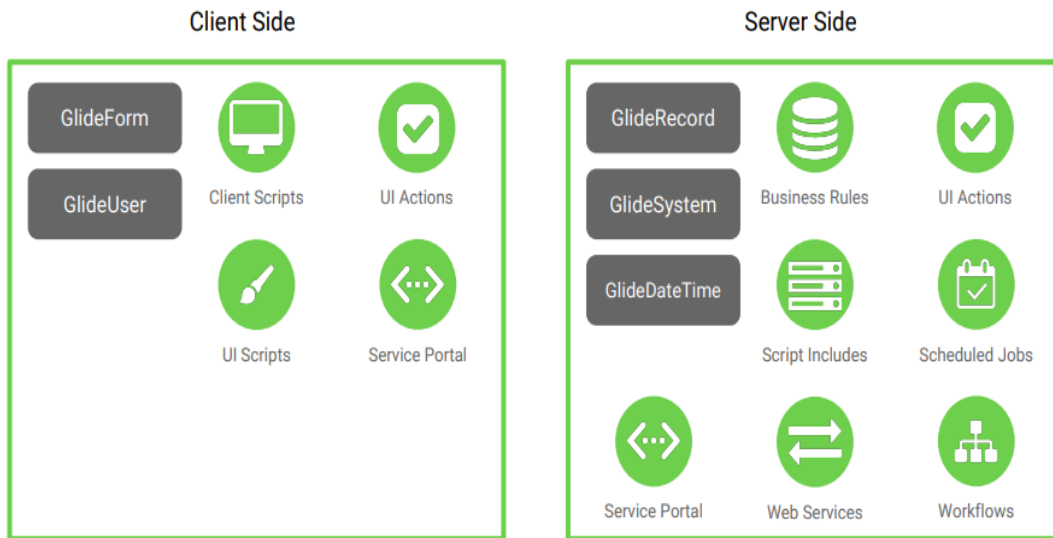
## Glide User Properties

- Object properties
- Don't need getter methods

```
1 console.log(g_user.firstName + ' ' + g_user.lastName);  
2 console.log('User ID: ' + g_user.userID);  
3 console.log('Username: ' + g_user.userName);  
4 console.log('Department: ' + g_user.departmentID);  
5 console.log('Roles: ' + g_user.roles);
```



## Where Can I Use This?



## Section Recap

- ✓ Use GlideForm (g\_form) to access information, or manipulate a form in ServiceNow
- ✓ Use GlideUser (g\_user) to access information about the current user
- ✓ Both GlideForm and GlideUser are client-side APIs and cannot be used on the server side
- ✓ Use ctrl + shift + j to access the Service Now client-side console or use your browser's web console





## Working with get first name and last name of user

```
1 alert('Hello' + g_user.firstName + ' ' + g_user.lastName + '. Your user ID is : ' + g_user.userID);
```

Result: Return the first and last name of current user and sys\_id

## Working with getFullName()

```
1 alert('Hello :' + g_user.getFullName());
```

Result: Return the first and last name of current user and sys\_id

## Working with hasRoles()

```
1 alert('Do have any Roles :' + g_user.hasRoles());
```

Result: Return the Boolean true or false

## Working with hasRoles('itil')

```
1 alert('Do have any Role ? :' + g_user.hasRole('itil'));
```

Result: Return the Boolean true or false

## Working with hasRole('admin')

```
1 var isAdmin = g_user.hasRole('admin');  
2 alert(isAdmin);
```

Result: Return the Boolean true or false

## Working with hasExactRole('itil')

```
1 var isItil = g_user.hasRoleExactly('itil');  
2 alert(isItil);
```

Result: Return the Boolean true or false

## Working with hasRoles('itil')

```
1 alert('User Name :' + g_user.userName);
```

Result: Return current logged in user name

## Working with hasRoleFormList('itil')

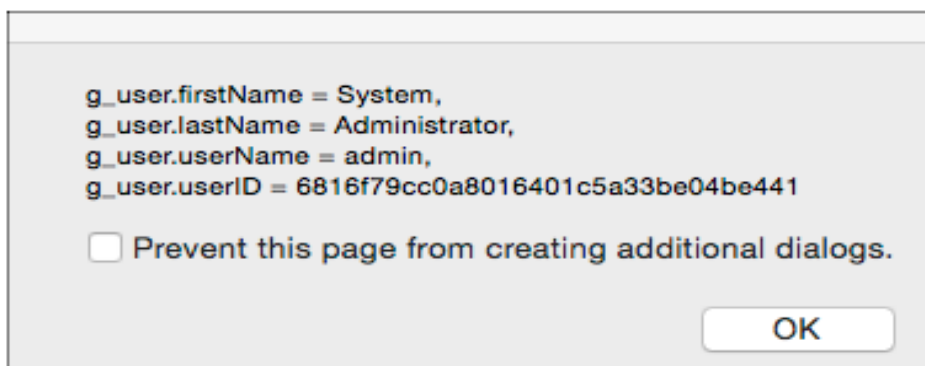
```
1 var contain = g_user.hasRoleFromList("itil, admin");  
2 alert(contain);
```

Result: Returns true if the current has a role in the list or the admin role

## Working with all methods in single program

```
1 alert("g_user.firstName = " + g_user.firstName  
2     + ", \n g_user.lastName = " + g_user.lastName  
3     + ", \n g_user.userName = " + g_user.userName  
4     + ", \n g_user.userID = " + g_user.userID);
```

Result:



## Glide User - Scoped

The scoped **Glide User API** provides access to information about the current user and current User roles. Using the scoped Glide User API avoids the need to use the slower **Glide Record Queries** to get user information.

### Working with getDisplayName()

```
1 var contain = g_user.hasRoleFromList("itil, admin");  
2 alert(contain);
```

Result: Returns the current user's display name.

### Working with getEmail()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getEmail());
```

Result: Returns User's email address

## Working with getfirstNamel()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getFirstName());
```

Result: Returns the user's first name.

## Working with getfirstNamel()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getID());
```

Result: Return Gets the sys\_id of the current user.

## Working with getFullName()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getFullName());
```

Result: Returns current logged in user full name

## Working with getFullNamel()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getUserRoles());
```

Result: List of roles explicitly assigned to the user

## Working with getUserRole()

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.getUserRoles());
```

Result: True if the user has the role.

## Working with isMemberOf(String group)

```
1 var currentUser = gs.getUser();  
2 gs.info(currentUser.isMemberOf('Capacity Mgmt'));
```

Result : Determines if the current user is a member of the specified group.

## Glide Date & Time

- The GlideDateTime class provides methods for performing operations on GlideDateTime objects, such as instantiating GlideDateTime objects or working with glide\_date\_time field
- In addition to the instantiation methods described below, a GlideDateTime object can be instantiated from a glide\_date\_time field using the getGlideObject() method (for example, var gdt = gr.my\_datetime\_field.getGlideObject();).
- Some methods use the Java Virtual Machine time zone when retrieving or modifying a date and time value. Using these methods may result in unexpected behavior. Use equivalent local time and UTC methods whenever possible.

### Working with GlideDateTime()

```
1  var gDate = new GlideDate();
2  gDate.setValue('2015-01-01');
3  gs.info(gDate);
4
5  var gDT = new GlideDateTime(gDate);
6  gs.info(gDT);
```

### Working with addDaysLocalTime()

```
1  var gdt = new GlideDateTime("2011-08-31 08:00:00");
2  gdt.addDaysLocalTime(-1);
3  gs.info(gdt.getLocalDate());
```

## Working with addMonthsLocalTime()

```
1 var gdt = new GlideDateTime("2011-08-31 08:00:00");
2 gdt.addMonthsLocalTime(2);
3 gs.info(gdt.getDate());
```

**Result: 2011-10-31**

## Working with addMonthsUTC()

```
1 var gdt = new GlideDateTime("2011-08-31 08:00:00");
2 gdt.addMonthsUTC(2);
3 gs.info(gdt.getDate());
```

**Result: 2011-10-31**

## Working with addWeeksLocalTime(-2)

```
1 var gdt = new GlideDateTime("2011-08-31 08:00:00");
2 gdt.addWeeksLocalTime(-2);
3 gs.info(gdt.getDate());
```

## Working with before ()

```
1 var gdt1 = new GlideDateTime("2016-05-09 10:11:12");
2 var gdt2 = new GlideDateTime("2017-06-12 15:11:12");
3 gs.info(gdt1.before(gdt2));
```



## Working with CompareTo(Object to)

```
1 var initDate = new GlideDateTime("2011-08-01 12:00:00");
2 var compDate1 = new GlideDateTime("2011-08-01 12:00:00");
3 var compDate2 = new GlideDateTime("2011-07-31 12:00:00");
4 var compDate3 = new GlideDateTime("2011-08-04 16:00:00");
5
6 gs.info(initDate.compareTo(compDate1)); // Equals (0)
7 gs.info(initDate.compareTo(compDate2)); // initDate is after compDate2 (1)
8 gs.info(initDate.compareTo(compDate3)); // initDate is before compDate3 (-1)
```

### Result:

---

```
*** Script: 0
*** Script: 1
*** Script: -1
```

---

## Working with getErrorMsg()

```
1 var gdt = new GlideDateTime();
2 gdt.setDisplayValue("2011-aa-01 00:00:00");
3 gs.info(gdt.getErrorMsg());
```

Result: Could not parse Date Time: 2011-aa-01 00:00:00

## Working with getLocalTime()

```
1 var gdt = new GlideDateTime();
2 gs.info(gdt.getLocalDate());
```

## Working with getWeekOfYearUTC()

```
1 var gdt = new GlideDateTime();  
2 gs.info(gdt.getWeekOfYearUTC());
```

## Glide Aggregate

### What is glide Aggregate?

The Glide Aggregate class is an extension of Glide Record and allows database aggregation (**COUNT, SUM, MIN, MAX, and AVG**) queries to be done. This can be helpful in creating customized reports or in calculations for calculated fields.

**Note:** This functionality requires knowledge of JavaScript.

## Working with gets a count of the number of records in a table with category

```
1 var incidentGA = new GlideAggregate('incident');
2 incidentGA.addQuery('active','true');
3 incidentGA.addAggregate('COUNT','category');
4 incidentGA.query();
5 while(incidentGA.next()){
6     var category = incidentGA.category;
7     var categoryCount = incidentGA.getAggregate('COUNT','category');
8     gs.log("We have currently "+ categoryCount +" incidents with a category of "+ category);}
```

---

```
*** Script: We have currently 4 incidents with a category of :
*** Script: We have currently 1 incidents with a category of : database
*** Script: We have currently 4 incidents with a category of : hardware
*** Script: We have currently 11 incidents with a category of : inquiry
*** Script: We have currently 4 incidents with a category of : network
*** Script: We have currently 7 incidents with a category of : software
```

---

## Working with MIN, MAX and AVG Values

```
1 var count = new GlideAggregate('incident');
2 count.addAggregate('MIN','sys_mod_count');
3 count.addAggregate('MAX','sys_mod_count');
4 count.addAggregate('AVG','sys_mod_count');
5 count.groupBy('category');
6 count.query();
7 while(count.next()){
8     var min = count.getAggregate('MIN','sys_mod_count');
9     var max = count.getAggregate('MAX','sys_mod_count');
10    var avg = count.getAggregate('AVG','sys_mod_count');
11    var category = count.category.getDisplayValue();
12    gs.log(category +" Update counts: MIN = "+ min +" MAX = "+ max +" AVG = "+ avg);}
```

## Working with MIN, MAX and AVG Values

```
1 var agg = new GlideAggregate('incident');
2 agg.addAggregate('count','category');
3 agg.orderByAggregate('count','category');
4 agg.orderBy('category');
5 agg.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(2)');
6 agg.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(2)');
7 agg.query();
8 while(agg.next()){
9     var category = agg.category;
10    var count = agg.getAggregate('count','category');
11    var query = agg.getQuery();
12    var agg2 = new GlideAggregate('incident');
13    agg2.addAggregate('count','category');
14    agg2.orderByAggregate('count','category');
15    agg2.orderBy('category');
16    agg2.addQuery('opened_at','>=','javascript:gs.monthsAgoStart(3)');
17    agg2.addQuery('opened_at','<=','javascript:gs.monthsAgoEnd(3)');
18    agg2.addEncodedQuery(query);
19    agg2.query();
20    var last = "";
21    while(agg2.next()){
22        last = agg2.getAggregate('count','category');}
23    gs.log(category+": Last month:"+ count +" Previous Month:"+ last);
24
25 }
```

```
*** Script: software: Last month:3 Previous Month:
*** Script: inquiry: Last month:2 Previous Month:
*** Script: : Last month:1 Previous Month:
*** Script: hardware: Last month:1 Previous Month:
*** Script: network: Last month:1 Previous Month:
```

---

## Working with Distinct count of a field on group query

```
1 var agg = new GlideAggregate('incident');
2 agg.addAggregate('count');
3 agg.addAggregate('count(distinct','category');
4 agg.addQuery('opened_at', '>=', 'javascript:gs.monthsAgoStart(2)');
5 agg.addQuery('opened_at', '<=', 'javascript:gs.monthsAgoEnd(2)');
6 //
7 agg.groupBy('priority');
8 agg.query();
9 while (agg.next()) {
10 // Expected count of incidents and count of categories within each priority value (group)
11   gs.info('Incidents in priority ' + agg.priority + ' = ' + agg.getAggregate('count') +
12         ' (' + agg.getAggregate('count(distinct','category') + ' categories)');
13 }
```

---

```
*** Script: Incidents in priority 1 = 5 (3 categories)
*** Script: Incidents in priority 2 = 1 (1 categories)
*** Script: Incidents in priority 3 = 2 (2 categories)
```

---

# Glide System

## Section Outline

1	GlideSystem Introduction	6	getMessage, getProperty, & nil Methods
2	Show Me The Code!	7	Concept: Events
3	GlideSystem API Overview	8	eventQueue Method
4	Common GlideSystem Methods	9	GlideSystem Demo
5	Message & Log Types	10	Section Recap

## What is Glide System ?

- The Glide System API provides methods for retrieving information.
- The Glide System (referred to by the variable name 'gs' in business rules) provides a number of convenient methods to get information about the system,
- The current logged in user, etc. For example, the method addInfoMessage() permits communication with the user.

## Glide System Introduction

- Server-side
- System and user data
- Referenced by gs
- Doesn't need to be invoked
- Helper methods

“

*The GlideSystem API provides a number of convenient methods to get information about the system, the current logged in user, etc.*


[ServiceNow Docs](#)

”

## Show Me The Code!


- Print a list of all incidents where the caller is the current user

```
1 var incidentGR = new GlideRecord('incident');  
2 incidentGR.addQuery('caller', gs.getUserID());  
3 incidentGR.query();
```

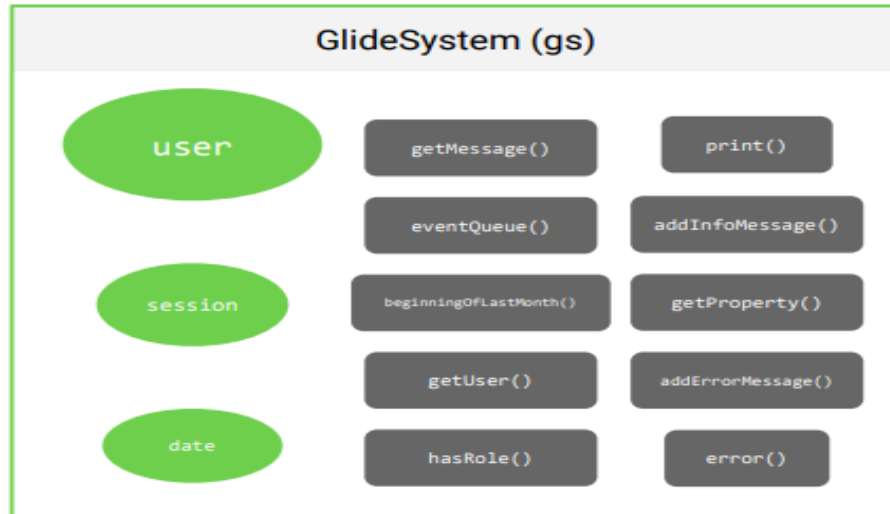


- Create a user greetings

```
1 var greetingsMessage = '';  
2  
3 if(currentHour >= 3 && currentHour < 11) {  
4   greetingsMessage = 'Good morning ';  
5 } else if(currentHour >= 11 && currentHour < 17) {  
6   greetingsMessage = 'Good afternoon ';  
7 } else {  
8   greetingsMessage = 'Good evening ';  
9 }  
10  
11 greetingsMessage += gs.getUserDisplayName();
```



## Glide System API Diagram



## Common Glide System Methods

- `addErrorMessage()`
- `addInfoMessage()`
- `eventQueue()`
- `getMessage()`
- `getProperty()`
- `getSession()`
- `print()`
- `error()`
- `log()`
- `debug()`
- `info()`
- `warn()`
- `generateGUID()`
- `getUser()`
- `hasRole()`
- `isLoggedIn()`
- `isMobile()`
- `nil()`
- `setRedirect()`
- `setProperty()`

## Message & Log Types



```
gs.addErrorMessage()  
gs.addInfoMessage()
```

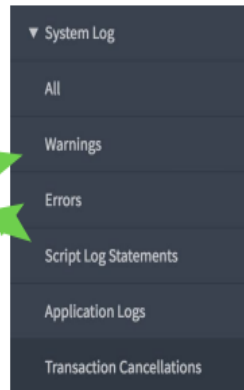
Welcome to ServiceNow 201: Development! This is an error message.

Welcome to ServiceNow 201: Development! This is an info message.

```
gs.log()
```

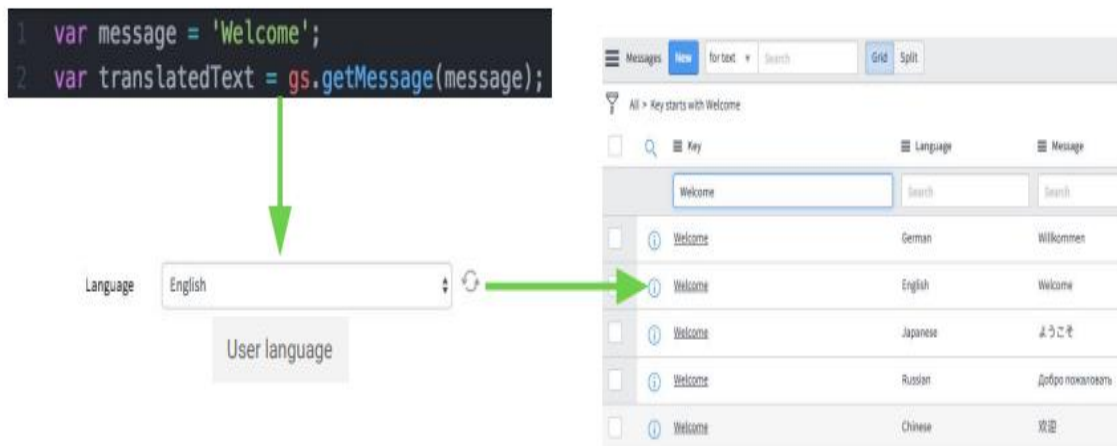
```
gs.warn()
```

```
gs.error()
```



## Glide System getMessage() Method

- Provides a way to translate text easily
- Uses the sys\_ui\_message table
- First argument is the string of the message
- System uses current user language as another parameter



## Glide System getProperty() Method

- Get a system property
- Best practice instead of hard-coding

`getProperty('glide.ui.buttons_bottom')`

System Properties	
<code>glide.ui.buttons_bottom</code>	<code>true</code>
<code>glide.ui.client.mandatory.check</code>	<code>true</code>
<code>glide.ui.incident_activity.image.comments</code>	<code>images/icons/edit.gifx</code>
<code>glide.ui.max.alt.chars</code>	<code>1000</code>

## Glide System nil() Method

- Every field in a Glide Record is an object
- Empty string comparison doesn't always work

```
1 var field = {  
2   value: '',  
3   displayName: ''  
4 };  
5  
6 console.log(field == ''); // returns false */
```

```
1 var incidentGR = new GlideRecord('incident');  
2 incidentGR.query();  
3 while(incidentGR.next()) {  
4   if(gs.nil(incidentGR.short_description)) {  
5     gs.print(incidentGR.number + ' has a nil short_description');  
6   }  
7 }
```

## Concept: Events



### User Events

- Mouse click
- Mouse movement
- Key press

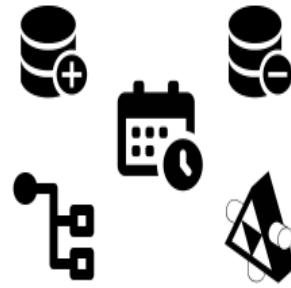


### System Events

- Time
- Database operation
- Network activity

## Events in Service Now




- Can be triggered by:
  - CRUD operations on tables (Business Rules)
  - Workflow activities
  - Scripts (Scheduled Jobs, Web Services, etc.)
- Must be registered
- Event components:
  - Registry
  - Event action (Script Action, Notification)
  - Event log



## Event Component: Registry

< ☰

Event Registration  
servicenow.201.hello.world

   Update Save Delete

\* Event name

Application

Table

Queue

Fired by

Description

Update Save Delete

Related Links

[Business Rules](#)

[Notifications](#)

# Event Component: Event Action

## Script Action

Script Action

Script name:  Application:  Active: ☒

Script:

```
1 // Do the following if the KB server just went down
2
3 trigger(fail);
4 var kb_server_name = 'kb_server';
5 var kb_server_ip = '192.168.1.1';
6
7 failover(kb_server_name, kb_server_ip);
8
9
10 function failover(kb_server_name, kb_server_ip) {
11     // Attempt to connect to the KB server
12     var result = true;
13     var result_msg = 'Success';
14     var result_code = 0;
15     var result_desc = '';
16     var result_err = '';
17     var result_status = 'Success';
18     var result_action = 'Success';
19     var result_time = 'Success';
20     // For each record for this KB server
21     while (true) {
```

## Notification

Notification - Knowledge Article Expiration - 1 Day (Advanced view)

Name:  Type:  Active: ☒

Table:

Description:

When to send:  Send when:  Event name:  Weight:

Conditions:

# Event Component: Event Log

Events (Event Log)

New

Created

Search

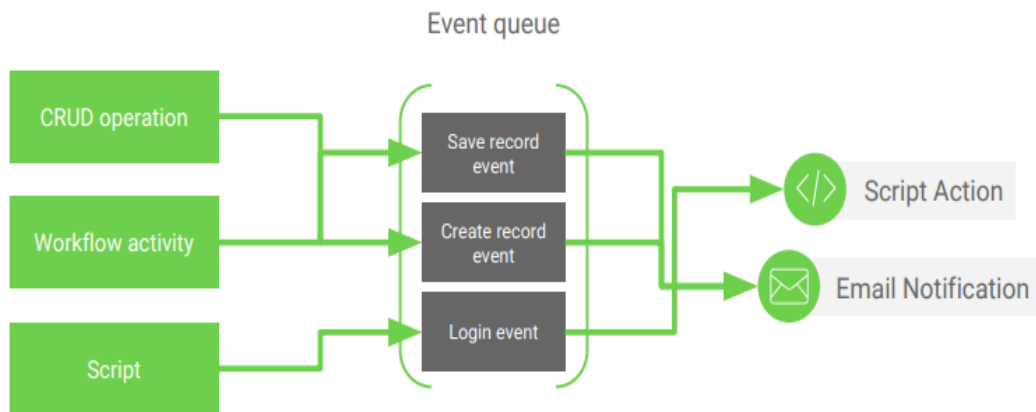
Grid

Split

All > Created on Today

		Created	Name	Parm1	Parm2	Table	Processed	Processing duration	Queue
	①	2017-05-17 20:47:45	incident.inactivity	34a17cb4c61122b7006b897258cbd702		incident	2017-05-17 20:47:55		2
	①	2017-05-17 20:10:18	incident.inactivity	34a17cb4c61122b7006b897258cbd702		incident	2017-05-17 20:10:29		1
	①	2017-05-17 20:10:08	incident.inactivity	34a17cb4c61122b7006b897258cbd702		incident	2017-05-17 20:10:29		2
	①	2017-05-17 20:10:08	incident.inactivity	34a17cb4c61122b7006b897258cbd702		incident	2017-05-17 20:10:29		1
	①	2017-05-17 20:05:04	user.view			syslog	2017-05-17 20:05:29		5
	①	2017-05-17 20:04:15	servicenow.201.hello.world				2017-05-17 20:04:29		6
	①	2017-05-17 20:02:51	text_index	[sys_class_name, script, name, event_name...	insert	sysevent_script_action	2017-05-17 20:03:00		92 text_index

## Process: Event Queue



## Script Actions

- Scripts invoked via registered events
- Server side
- Scope
  - current
  - event
    - parm1
    - parm2

Script Action  
Fall over MID server

Name: Fall over MID server

Event name: mid\_server.down

Order:

Condition script:

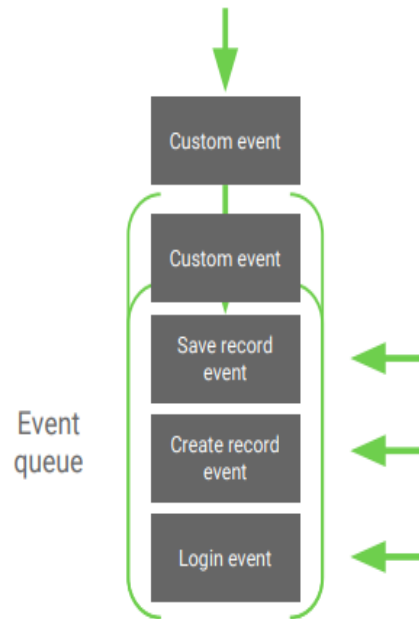
Script

```
1 /* Do the following if the MID server just went down
2
3 ArrayPolyfill;
4 var NO_MID_FOUND_EXCEPTION = 'NoSuitableMidServerFou
5 var LOG_SOURCE = 'MID fail over';
6
7 failOver();
8
```

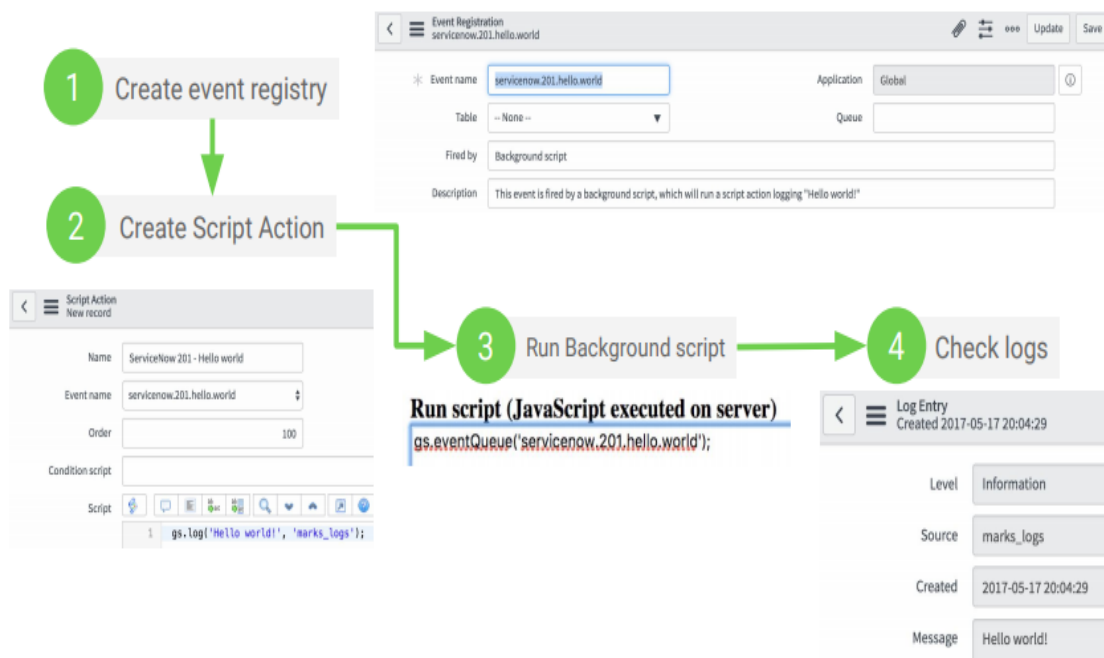
## Glide System eventQueue() Method

- Add events to the event queue
- Parameters
  - Event name
  - GlideRecord object
  - Optional parameter
  - Optional parameter

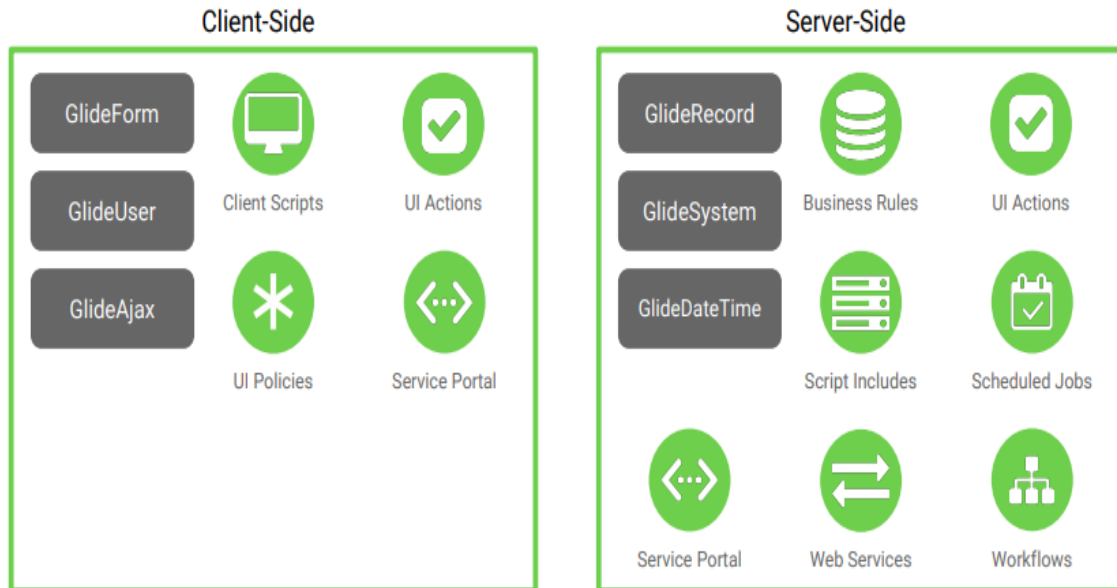
```
eventQueue('event.name', current, parm1, parm2)
```



## Demo: Log Hello world! Using Events



## Where Can I Use This?



## Section Recap

- Use the GlideSystem API to access helper methods and data about the current user and session
- Reference GlideSystem in any server-side script by using `gs`
- Events in ServiceNow have 3 main components
  - Event registry
  - Event action (Script Action, Notification)
  - Event log
- Use the `eventQueue()` method to add registered events to the event queue
- Use the `nil()` method when checking if a JavaScript value is empty





## Working with print method()

```
1 var helloText = 'Hello World';  
2 gs.print(helloText);
```

**Result: Hello World**

## Working with log()

```
1 gs.log('This is a log message' , 'marks_logs');
```

Navigate to **System Logs Application** → **Script log statement** → **Select filter Condition** → **Run**

All of these conditions must be met

Created	on	Today		AND	OR	X
Source	is	Marks_logs		AND	OR	X



**Log Record** → **Open Log Record and Check it**

	<u>2018-06-25 07:53:11</u>	Information	This is a log message	marks_logs
--	----------------------------	-------------	-----------------------	------------

## Working with Error()

```
1 gs.error('I am an error message');
```



Navigate to **System Logs Application→System Log→Error**

		<u>2018-06-25</u> <u>08:07:30</u>	Error	*** Script: I am an error message: no thrown error
---	---	--------------------------------------	-------	--

## Working with Warn()

```
1 gs.warn('I am an warning message');
```

Navigate to **System Logs Application→System Log→Warning**

		<u>2018-06-25 08:15:15</u>	Warning	I am an warning message	*** Script
---	---	----------------------------	---------	-------------------------	------------

## Working with beginningOfLastMonth()

```
1 gs.print(gs.beginningOfLastMonth());
```

**Result:**

```
*** Script: 2018-05-01 07:00:00
```

## Working with generateGUID()

```
1 gs.print(gs.generateGUID());
```

**Result:**

```
*** Script: ddebac67dbb21300a5adf28239961917
```

## Working with getMessage()

Step – 1: Open this list in left navigator **sys\_ui\_message.list**.

```
1 gs.print(gs.getMessage('One year ago'));
```

Result: Message will be display based on key

## Working with getMessage()

**Step 1:** Navigate and Open `sys_properties.list` table

**Step 2:** Create a new system property to click on **New**

**Step 2:** To fill the form provide **Name** and **Value**

**Name is :** Service Now ! Hellow World

**Value is :** Hellow World

```
1 gs.print(gs.getProperty('Service Now ! Hellow World'));
```

---

```
*** Script: Hellow World
```

**Result:**

---

## Working with getUser() and getDisplayName() And so on

**getUser ()**

```
1 gs.print(gs.getUser());
```

**getDisplayName ()**

```
1 gs.print(gs.getUser().getDisplayName());
```

**getFirstName()**

```
1 gs.print(gs.getUser().getDisplayName());
```

**getFullName()**

```
1 gs.print(gs.getUser().getFullName());
```

### getUserRoles()

```
1 gs.print(gs.getUser().getUserRoles());
```

### getUserID()

```
1 gs.print(gs.getUserID());
```

### hasRole()

```
1 if(gs.hasRole('itil') || gs.hasRole('admin')){  
2     gs.print('You have ITIL or Admin rolw')  
3 }
```

### getSession()

```
1 gs.print(gs.getSession());
```

### isLoggedIn()

```
1 gs.print(gs.getSession().isLoggedIn());
```

## Working with nil()

```
1 var incidentGR = new GlideRecord('incident');  
2 incidentGR.query();  
3 while(incidentGR.next()){  
4     if(gs.nil(incidentGR.short_description)){  
5         gs.print('This incident (' + incidentGR.number + ') has no short_description: ' + incidentGR.short_description);  
6     }  
7 }
```