GlideRecord Cheat Sheet

gliderecord, cheat sheet

Query Copy

```
var gr = new GlideRecord('incident');
gr.addQuery('active', true);
gr.query();
while(gr.next()) {
	gs.log('Category is ' + gr.category);
}
```

Get Copy

If other than 'sys_id', format is gr.get('number', 'INC0010114');

```
var gr = new GlideRecord('incident');
gr.get('sys_id'); //Now use it directly
if(gr.category == 'software') {
	gs.log('Category is ' + gr.category);
}
```

OR Copy

Or conditions can be stacked orGr.addOrCondition('state', 6).addOrCondition('state', 7);

```
var gr = new GlideRecord('incident');
var orGr = gr.addQuery('state', 6);
orGr.addOrCondition('state', 7);
gr.query();
while(gr.next()) {
	gs.log('Category is ' + gr.category);
}
```

GetRefRecord Copy

```
var gr = new GlideRecord('incident');
```

```
gr.get('sys_id');
var caller = gr.caller_id.getRefRecord(); //Gets the sys_user GlideRecord for the caller
caller.first_name = 'New name';
caller.update();
```

Insert Copy

```
var gr = new GlideRecord('incident');
gr.initialize();
gr.short_description = 'Printer is broken again';
gr.description = 'There is something wrong with the printer again, should we get a new one?';
gr.insert();
```

Update Copy

```
var gr = new GlideRecord('incident');
gr.get('sys_id');
gr.short_description = 'Printer needs a restart';
gr.update();
```

Delete Copy

```
var gr = new GlideRecord('incident');
gr.get('sys_id');
gr.deleteRecord(); //Delete the record
```

DeleteMultiple Copy

We don't use gr.query() here

```
var gr = new GlideRecord('incident');
gr.addQuery('active', false);
gr.deleteMultiple(); //Delete all the queried records
```

AddEncodedQuery Copy

Encoded query strings can be copied directly from a filter, by right-clicking on the breadcrumbs

```
var gr = new GlideRecord('incident');
```

```
gr.addEncodedQuery('active=true^state=2');  //Encoded query
gr.query();
while(gr.next())  {
        gs.log('Category is '  +  gr.category);
}
```

GlideAggregate Copy

Other aggregates include COUNT, SUM, MIN, MAX, AVG

```
var  count  =  new  GlideAggregate('incident');
count.addQuery('active',  'true');
count.addAggregate('COUNT');  //Select COUNT as our aggregate
count.query();
var  incidents  =  0;
if(count.next())  {
        incidents  =  count.getAggregate('COUNT');  //Get the aggregate
        gs.log('Found '  +  incidents  +  ' incidents.');
}
```

OrderBy Copy

.orderByDesc() can be used in the same way

```
var  gr  =  new  GlideRecord('incident');
gr.addQuery('active',  true);
gr.orderBy('number');  //Order this query by number
gr.query();
while(gr.next())  {
        gs.log('Category is '  +  gr.category);
}
```

AddNullQuery Copy

.addNotNullQuery() can also be used to find non-empty fields

```
var  gr  =  new  GlideRecord('incident');
gr.addNullQuery('caller_id');
```

```
gr.query();
```

AddActiveQuery Copy

Same as adding gr.addQuery('active', true);

```
var gr = new GlideRecord('incident');
gr.addActiveQuery();
gr.query();
```

GetRowCount Copy

Note that GlideAggregate is better performance wise

```
var gr = new GlideRecord('incident');
gr.addQuery('caller_id', 'Jesper_ServiceHow');
gr.query();
gs.log('Jespers incidents: ' + gr.getRowCount());
```

SetLimit Copy

Works with orderBy

```
//Limit the query to 10 incidents
var gr = new GlideRecord('incident');
gr.setLimit(10);
gr.query();
```

ChooseWindow Copy

ChooseWindow 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

```
//Limit the query to 10 incidents
var gr = new GlideRecord('incident');
gr.chooseWindow(10, 20);
gr.query();
```

SetWorkflow Copy

This will enable/disable the triggering of business rules

```
var gr = new GlideRecord('incident');
```

```
gr.addNullQuery('short_description');
gr.query();
while(gr.next()){
        gr.short_description = 'Must be the printers that are broken again...';
        gr.setWorkflow(false); //Disable business rules for this query
        gr.update();
}
```

AutoSysFields Copy

Will not update fields such as sys_updated_on, sys_updated_by, etc.

```
var gr = new GlideRecord('incident');
gr.addNullQuery('short_description');
gr.query();
while(gr.next()){
        gr.short_description = 'Must be the printers that are broken again...';
        gr.autoSysFields(false); //Disable the update of system fields
        gr.update();
}
```

SetForceUpdate Copy

This will force an update, even when no fields are changed

```
var gr = new GlideRecord('incident');
gr.query();
while(gr.next()) {
        gr.setForceUpdate(true); //Force an update even with no changes
        gr.update();
}
```

SetAbortAction Copy

Set the next database operation (create, update, delete) to be aborted

```
//Standard date comparison of a start date that must be set before an end date
//Copy/pasted from http://wiki.servicenow.com/index.php?title=GlideRecord#setAbortAction
if ((!current.u_date1.nil()) && (!current.u_date2.nil())) {
```

```
        var start = current.u_date1.getGlideObject().getNumericValue();
        var end = current.u_date2.getGlideObject().getNumericValue();
        if (start > end) {
                gs.addInfoMessage('Start must be before end');
                current.u_date1.setError('Start must be before end');
                current.setAbortAction(true);
        }
}
```

Operators Copy

addQuery defaults to an equals (=) operator, but it is possible to include a 3rd parameter to the addQuery, with a different operator

```
// =
//Equals (this is the same as not including the 3rd parameter)
addQuery('priority', '=', 1);

// >
//Greater than
addQuery('priority', '>', 1);

// <
//Less than
addQuery('priority', '<', 1);

// >=
//Greater than or equals
addQuery('priority', '>=', 1);

// <=
//Less than or equals
addQuery('priority', '<=', 1);

// !=
```

```
//Not equals
addQuery('priority', '!=', 1);

// STARTSWITH
//Field must start with value
addQuery('short_description', 'STARTSWITH', 'Printer');

// ENDSWITH
//Field must end with value
addQuery('short_description', 'ENDSWITH', 'Printer');

// CONTAINS
//Field must contain value somewhere
addQuery('short_description', 'CONTAINS', 'Printer');

// DOES NOT CONTAIN
//Field must not contain value anywhere
addQuery('short_description', 'DOES NOT CONTAIN', 'Printer');

// IN
//Field must be found somewhere in the value
addQuery('sys_id', 'IN', '8d641046c0a80164000bc7c0d3ed46a0,a9a16740c61122760004fe9095b7ddca');

// INSTANCEOF
//Return only records that are instances of an extended table (like incident is of task in this example)
addQuery('sys_class_name', 'INSTANCEOF', 'incident');

//For more operators for encoded queries and filters see:
// http://wiki.servicenow.com/index.php?title=Operators_Available_for_Filters_and_Queries
```