## Contents

1 Copying Data
   1.1 Worknotes ................................................................. 3
   1.2 Currency Fields .......................................................... 3

2 CORS Error
   2.1 How to get around CORS Exception .................................. 5

3 Date Time Servicenow
   3.1 Update Review Date ....................................................... 7
   3.2 Comparing Due Date with Start Date ................................. 7
   3.3 Comparing Months ......................................................... 8
   3.4 Comparing Dates when days doesn’t matter (untested) .......... 8

4 Design Resources
   4.1 Icon Library ............................................................... 9
   4.2 Instance Style Guide ...................................................... 9

5 Emails
   5.1 Parsing out Hidden Watermarks ....................................... 11

6 GlideAjax
   6.1 Returning multiple values ............................................. 13

7 Glide Conditionals
   7.1 Matching records using conditions ................................. 15

8 Hiding non-readable Rows in List View
   8.1 Separating Readable Projects By Department .................... 17

9 Metrics
   9.1 Metric from Assignment Group to Close ........................... 21

10 Modules
   10.1 Open new window from module .................................... 23

11 OAuth
   11.1 Manual Oauth ............................................................ 25


1.1 Worknotes

Copying the full set of work notes to another area

```javascript
// Gets all worknotes associated with current
var notes = current.work_notes.getJournalEntry(-1);

var newNotes = "Copied notes ::\n" + notes;
```

1.2 Currency Fields

Copying old funds in current to new funds in another field

```javascript
bCase.new_funds = bCase.new_funds.getReferenceCurrencyCode() + '; ' + current.old_funds.getReferenceValue();
```
CHAPTER 2

CORS Error

2.1 How to get around CORS Exception

```
chrome.exe --user-data-dir="C:/Chrome dev session" --disable-web-security
```

Depending on location of Chrome, file can be run with single line below

```
"C:\Program Files (x86)\Google\Chrome\Application\chrome.exe" --user-data-dir="C:\ 
--Chrome dev session" --disable-web-security
```

**Warning:** Goes without saying !! be careful of security
### 3.1 Update Review Date

```javascript
(function executeRule(current, previous /*null when async*/) {
    var gdt = new GlideDateTime(current.ends);
    gdt.addDays(90);
    current.u_review_date_custom = gdt;
})(current, previous);
```

**Warning:** `.addDays()` will add days to the time, but does not return a glideDateTime

current.ends is a Date (not DateTime) field

### 3.2 Comparing Due Date with Start Date

**On Change Client Script for Due Date**

```javascript
function onChange(control, oldValue, newValue, isLoading, isTemplate) {
    if (isLoading || g_form.getValue("expected_start") == ") {
        return;
    }

    var start_date = g_form.getValue("expected_start");

    // Calls serverside script to check if start date is earlier than due date
    var ajax = new GlideAjax('TransferOrderDateTimeAjax');
    ajax.addParam('sysparm_name','compareDatesAjax');
    ajax.addParam('sysparm_startDate', start_date);
}
```
ajax.addParam('sysparm_endDate', newValue);
ajax.getXML(warnIfFalse);

//Callback will display error if start date is earlier than due date
function warnIfFalse(response){
    var answer = response.responseXML.documentElement.getAttribute("answer");
    if (answer == "false") {
        g_form.setValue("due_date", "");
        g_form.showErrorBox("due_date", "Due Date must be after Start Date");
    }
}

3.3 Comparing Months

This code is used to find the difference between two dateTime in months

```javascript
var start = profileGR.getValue("employment_start_date");
var nowDate = GlideDateTime().toString();

var year1 = start.split("-")[0];
var month1 = start.split("-")[1];

var year2 = nowDate.split("-")[0];
var month2 = nowDate.split("-")[1];

if (year1 == year2) {
    return month2 - month1;
} else {
    var yearDiff = year2 - year1;
    var monthDiff = month2 - month1;
    return (yearDiff * 12) + monthDiff;
}
```

3.4 Comparing Dates when days doesn’t matter (untested)

Uses a milisecond timestamp, works for either Date or Date/Time

```javascript
newDate(g_form.getValue("expected_start")).getTime() > newDate(g_form.getValue("due_date")).getTime()
```
4.1 Icon Library

http://styleguide.servicenow.com/#/style/icon-library

4.2 Instance Style Guide

https://<instancename>/service-now.com/styles/heisenberg/styleguide/docs/index.html
5.1 Parsing out Hidden Watermarks

5.1.1 Business Rule on sys_email table

```javascript
(function executeRule(current, previous /*null when async*/) {
  //RegEx to find remove Re: in subject and anything after In-Reply-To:
  current.subject = String(current.subject).replace(/Re:+/i, "");
  current.headers = String(current.headers).replace(/In-Reply-To:<[^>]*>/ig, "");
})(current, previous);
```

5.1.2 Summary

We had a client where an email would continually be marked as a reply with a watermark, even after the watermark had been removed. We found in this case, we needed to remove the “RE” in the subject and the “In-Reply-To” in the email header.
6.1 Returning multiple values

You can return more than one value easiest if you return it as a JSON string.

```javascript
var GetUserRecord = Class.create();
GetUserRecord.prototype = Object.extendsObject(AbstractAjaxProcessor, {

  getForCorrectiveItem: function () {
    var userID = this.getParameter("sysparm_user_id");

    var userGR = new GlideRecord("sys_user");
    userGR.addQuery("sys_id", userID);
    userGR.query();

    var userProps = {};

    if (userGR.next()) {
      userProps.name = userGR.name.toString();
      userProps.manager = userGR.manager.toString();
      userProps.department = userGR.department.toString();
      userProps.location = userGR.location.getDisplayValue();
    }

    var json = new JSON();
    var data = json.encode(userProps);

    return data;
  },

  type: 'GetUserRecord'
});
```
7.1 Matching records using conditions

The business rule below runs when a record is inserted into the table. It orders the approval_conditions record and checks to see if the current inserted record matches the condition set in the approval_condition record.

```javascript
(function executeRule(current, previous /*null when async*/) {

    var approvalConditionsGR = new GlideRecord("approval_conditions_table");
    approvalConditionsGR.orderBy('order');
    approvalConditionsGR.query();

    while (approvalConditionsGR.next()) {
        if (GlideFilter.checkRecord(current, approvalConditionsGR.approval_condition) === true) {
            var approver = approvalConditionsGR.getValue("approver");
            current.approver.setValue(approver);
            return;
        }
    }
})(current, previous);
```
Hiding non-readable Rows in List View

Row Level Security and On Query Business Rules

8.1 Separating Readable Projects By Department

If you ever try placing security by record (or by row), your users may start seeing lists views with a count of Rows removed by Security constraints. Even worse they may think there are no more records, when records can easily be on the first and last page with any number of removed rows in between.
8.1.1 Creating a Before Query Business Rule

The query created in the Business Rule can be thought of as a filter. So let’s say one requirement is to allow anyone from the InfoSec Department to see any Contract record. We would start the Business Rule, by wrapping the filter with an if statement.

```java
if (gs.hasRole('admin') || userIsInInfoSec()) {
    // Add filter query here
}

function userIsInInfoSec() {
    // Normal Glide Record query to see if user's Dept is InfoSec
}
```
Next, we need to apply the query. Let’s say there’s a department field on each contract and if a user is in the same
department as the one listed in contracts then they should be able to view it.

```javascript
if (gs.hasRole('admin') || userIsInInfoSec()) {
    var uID = gs.getUserID();
    var uGR = new GlideRecord("sys_user");
    uGR.get(uID);

    current.addQuery('department', uGR.department);
}
```

What the query is doing here is returning only records that match department with the current user’s department. 
These will be the only records that will show in list view and will remove the security constraints, as long as they have 
security rights. Below, we are going to add another query where all contracts in the “Guest” department can be viewed 
as well.

```javascript
if (gs.hasRole('admin') || userIsInInfoSec()) {
    var uID = gs.getUserID();
    var uGR = new GlideRecord("sys_user");
    uGR.get(uID);

    var q = current.addQuery('department', uGR.department);
    q.addOrCondition('department', 'Guest');
}
```
9.1 Metric from Assignment Group to Close

This will require two metric definitions. One will track the assignment group field and the other will track the state field. When the assignment group is filled the metric begins and when the state turns to close, the metric will find the assignment group metric and end the duration.

Metric Definition

Name: Assignment Group has been populated
Type: Script calculation

```javascript
if (!current.assignment_group.nil()) {
  createMetric();
}

function createMetric() {
  var mi = new MetricInstance(definition, current);
  if (mi.metricExists())
    return;

  var gr = mi.getNewRecord();
  gr.field_value = current.assignment_group;
  gr.start = current.sys_updated_on;
  gr.calculation_complete = false;
  gr.insert();
}
```

Metric Definition

Name: Stop Assignment duration on close
Type: Script calculation
// Close Durations on Closed Complete, Closed Incomplete, or Cancelled
if (current.state == 3 || current.state == 4 || current.state == 7) {
    closeDurations(current);
}

function closeDurations(current) {
    var gr = new GlideRecord('metric_instance');
    gr.addQuery('id', current.sys_id);
    gr.addQuery('calculation_complete', false);
    gr.addQuery('definition.name', 'Time to closure');
    gr.query();
    while (gr.next()) {
        var definition = new GlideRecord('metric_definition');
        definition.get(gr.definition);
        var mi = new MetricInstance(definition, current);
        mi.endDuration();
    }
}

**Warning:** Both Metrics need to be placed as “Script calculation” or it will not process correctly
10.1 Open new window from module

This will require two metric definitions. One will track the assignment group field and the other will track the state field. When the assignment group is filled the metric begins and when the state turns to close, the metric will find the assignment group metric and end the duration.

Module

Link Type : URL (from Arguments:)

In the arguments field use the following:

```javascript
top.window.open('/sys_user_list.do', '_blank', 'height=700,width=1000');
```
11.1 Manual OAuth

OAuth profile needs to be first set up on ServiceNow. After set up, code below can run OAuth for each REST call. This is taken form a Salesforce Integration.

```javascript
var oAuthClient = new sn_auth.GlideOAuthClient();
var params = {grant_type:"password", username: username, password: password};
var json = new global.JSON();
var text = json.encode(params);
var tokenResponse = oAuthClient.requestToken("nameless-dev-ed.my.salesforce.com", text);
var token = tokenResponse.getToken();
gs.info("AccessToken:" + token.getAccessToken());
gs.info("AccessTokenExpiresIn:" + token.getExpiresIn());
gs.info("RefreshToken:" + token.getRefreshToken());

var response, responseBody, status;
try {
    var restMessage = new sn_ws.RESTMessageV2();
    restMessage.setHttpMethod("get");
    restMessage.setEndpoint("https://nameless-dev-ed.my.salesforce.com/services/data/v39.0/query/?q=SELECT+FirstName,+LastName,+Company,+MobilePhone+from+Lead")
    restMessage.setRequestHeader("Authorization", "Bearer " + token.getAccessToken());
    response = restMessage.execute();
    responseBody = response.haveError() ? response.getErrorMessage() : response.getBody();
    status = response.getStatusCode();
}

catch(ex) {
    responseBody = ex;
}
```
status = '500';
} finally {
    requestBody = restMessage ? restMessage.getRequestBody():null;
}

gs.info("Request Body: " + requestBody);
gs.info("Response: " + responseBody);
gs.info("HTTP Status: " + status);
12.1 Creating a Dynamic Qualifier

I decided to create a catalog item to request a parking space. One feature I wanted was to only show parking spots depending on which type of spot you want to reserve (i.e. VIP, Motorcycle, Normal).

To accomplish that requirement, I decided to go with a dynamic qualifier that changes depending on type selected.

12.1.1 The Script Include

This is probably up for debate, but I decided to keep the same object format for my script include using Class.create() and .prototype.

---

**Script Include Fields**

- Name: `parkingByParkingType`
- Client callable: `True`

```javascript
var parkingByParkingType = Class.create();
parkingByParkingType.prototype = {
    initialize: function() {
    },

    getSpots: function() {
        var result = [];

        var res = new GlideRecord("u_parking_spot");
        res.addQuery("u_assigned", false);
        res.addQuery("u_type", current.variables.parking_type);
        res.query();

```
The function of interest is the `getSpots()` function. It’s a basic query in the `u_parking_spot` table, but with a Dynamic Query we have access to `current` which we can use to add to query. In this example we’re using the `parking_type` variable, which is a selection.

### 12.1.2 The Dynamic Filter Option


#### 12.2 Qualifier that uses multiple queries

```javascript
while (res.next() ) {
    result.push(res.sys_id.toString());
}

var au = new ArrayUtil();

return au.unique(result);

```

28 Chapter 12. Reference Qualifiers
13.1 Replace with callback

Regex can be used for replacing a string, but if you want to use the string still or store it, you can use a callback function.

The code below will find a string, store it in a table, and replace it with the sys_id of the record:

```javascript
var body = textWithURLs;
var newBody = "";

var trackGR = new GlideRecord("url_table");

//Here we see the "url" variable will hold each matched string
// {string} The return string of each replace will replace the matched string
var newBody = body.replace(/href\s*=[\s*]*"([^"\n]*)"/ig, function(url) {
  trackGR = new GlideRecord("url_table");
  trackGR.initialize();
  trackGR.u_email_message = current.sys_id.toString();
  var formattedUrl = url.toString().split('/"\n').join();
  trackGR.u_url_link = formattedUrl;
  var trackID = trackGR.insert();

});

textWithURLs = newBody;
```
13.2 Example 2: replace <<EMPLOYEE>>

```javascript
var parsedForm = htmlForm.replace(/&lt;&lt;EMPLOYEE NAME&gt;&gt;|&lt;&lt;EMPLOYEE NAME&gt;&gt;/ig, employeeGR.name);
parsedForm = parsedForm.replace(/&lt;&lt;EMPLOYEE FIRSTNAME&gt;&gt;|&lt;&lt;EMPLOYEE FIRSTNAME&gt;&gt;/ig, employeeGR.first_name);
parsedForm = parsedForm.replace(/&lt;&lt;DATE&gt;&gt;|&lt;&lt;DATE&gt;&gt;/ig, gs.nowDateTime().split(" ")[0]);
```
CHAPTER 14

14.1 External Cart Orders

There’s three pieces I had to use to allow cart orders from an external source

1. Creating the staging table
2. Creating the business rule after insertion into the staging table
3. Setting up the REST Message

14.1.1 Create the Staging Table

I created a table and named it Cart Order (u_cart_order). This will hold four fields used for calling the Cart API to create an order, more on that later.

<table>
<thead>
<tr>
<th>Column Label</th>
<th>Type</th>
<th>Max length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Req_Id</td>
<td>String</td>
<td>100</td>
</tr>
<tr>
<td>Catalog Item</td>
<td>String</td>
<td>40</td>
</tr>
<tr>
<td>Errors</td>
<td>String</td>
<td>500</td>
</tr>
<tr>
<td>Variables</td>
<td>String</td>
<td>3000</td>
</tr>
</tbody>
</table>

14.1.2 Sending the REST Message

From within another ServiceNow Instance

This can be run however you would want, but I’ve only tested this using a background script. I don’t see why it would fail anywhere else.

```javascript
var request = new sn_ws.RESTMessageV2();
request.setEndpoint('https://dev22614.service-now.com/api/now/table/u_cart_order');
request.setHttpMethod('POST');
```
// Eg. UserName="admin", Password="admin" for this code sample.

var user = 'admin';
var password = 'admin';

request.setBasicAuth(user, password);
request.setRequestHeader("Accept","application/json");
request.setRequestHeader('Content-Type','application/json');
request.setRequestHeader('u_catalog_item':"Blackberry", +
  'u_variables':"original^299-999-9991|'
  'replacement^Yes"');
var response = request.execute();
gs.log(response.getBody());

In the request body there’s two key value’s to set up

- **u_catalog_item** Name of the catalog item should be the value
- **u_variables** The value should be a pipe \| separated list of variables, with the name of the variable the value separated by a caret ^

### 14.1.3 Creating the Business Rule

When the REST message is sent, it will populate the Cart Order table that we had created above. What we want to do is create a **BEFORE** business rule to be called on **INSERT** so it can go into the **Cart API** and make an order depending on what was sent in the REST call.
15.1 Overview

This will allow a system to receive a base64 encoded string and set it to the attachment table.

15.2 Staging Table

A table should be created that extends the Import Set Row table with the following added fields:

<table>
<thead>
<tr>
<th>Column label</th>
<th>Type</th>
<th>Max length</th>
</tr>
</thead>
<tbody>
<tr>
<td>attachment</td>
<td>String</td>
<td>1,000,000</td>
</tr>
<tr>
<td>name</td>
<td>String</td>
<td></td>
</tr>
<tr>
<td>ticket number</td>
<td>String</td>
<td>1,000</td>
</tr>
</tbody>
</table>

15.3 Create a Transform Map

<table>
<thead>
<tr>
<th>Name</th>
<th>Source table</th>
<th>Target table</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attachment Stage to Queue</td>
<td>Attachment Staging Table</td>
<td>ecc_queue</td>
</tr>
</tbody>
</table>

15.3.1 Field Maps

<table>
<thead>
<tr>
<th>Source field</th>
<th>Target field</th>
</tr>
</thead>
<tbody>
<tr>
<td>u_attachment</td>
<td>payload</td>
</tr>
<tr>
<td>u_name</td>
<td>name</td>
</tr>
<tr>
<td>u_ticket_number</td>
<td>source</td>
</tr>
<tr>
<td>[Script]</td>
<td>agent</td>
</tr>
<tr>
<td>[Script]</td>
<td>topic</td>
</tr>
</tbody>
</table>
Agent Scripts

answer = (function transformEntry(source) {
    // Add your code here
    return "AttachmentCreator"; // return the value to be put into the target field
    return source;
})

Topic Scripts

answer = (function transformEntry(source) {
    // Add your code here
    return "AttachmentCreator"; // return the value to be put into the target field
    return source;
})

15.4 Sending the REST Message

The rest message should be sent with the fields below

{  
  "u_name":"Something.jpg:image/jpeg",  
  "u_ticket_number":"incident:83bff1fdddb2d220003ee793ebf961957",  
  "u_attachment":"e490quigikelkabvrohatge4",  
}

15.5 Example Business Rule Script for Attachment

```javascript
var target = new GlideRecord('sys_attachment');
target.addQuery('table_name', 'incident');
target.addQuery('table_sys_id', current.sys_id);
target.query();

while(target.next()) {
    var sa = new GlideSysAttachment();
    var binData = sa.getBytes(target);
    var base64Data = GlideStringUtil.base64Encode(binData);

    //Send Attachments
    var requestAttachment = new sn_ws.RESTMessageV2();
    requestAttachment.setEndpoint('https://XXXXX.service-now.com/api/now/u_attachment_staging_table');
    requestAttachment.setHttpMethod('POST');
    requestAttachment.setBasicAuth(user,password);
    requestAttachment.setRequestHeader("Accept","application/json");
    requestAttachment.setRequestHeader('Content-Type','application/json');
}
```
```
var requestAttachmentJSONBody = createRequestBody("u_name", target.file_name + ":" + target.content_type);
requestAttachmentJSONBody += addToRequestBody("u_ticket_number", 'incident:' + sysid);
requestAttachmentJSONBody += addToRequestBody("u_attachment", base64Data);
requestAttachmentJSONBody += closeRequestBody();

requestAttachment.setRequestBody(requestAttachmentJSONBody);

var responseAttachment = requestAttachment.execute();
```

The createRequestBody() functions are just used to create a JSON Object with for example `{u_name: "targetinfo", u_ticket_number: "incident:39024903284"}`
16.1 Overview

This will discuss using Angular and coding widgets to create a Service Portal with as much, if not more, features as the Content Management System.

16.2 Understanding How Server Script Works

Server script globals

<table>
<thead>
<tr>
<th>input object</th>
<th>This is the data object as sent from the client script.</th>
</tr>
</thead>
<tbody>
<tr>
<td>options object</td>
<td>The options that were used to invoke the widget on the server.</td>
</tr>
<tr>
<td>data object</td>
<td>An object of stuff you want to send to the client.</td>
</tr>
</tbody>
</table>

Client script globals

<table>
<thead>
<tr>
<th>data object</th>
<th>The serialized data object from the server script.</th>
</tr>
</thead>
<tbody>
<tr>
<td>options object</td>
<td>The options that were used to invoke the widget on the server.</td>
</tr>
</tbody>
</table>

The Flow of Data from ServiceNow Docs

The Server Script on a widget is executed `onLoad` and can be used to transfer a `data` object from the server to the client. On the client, this data object is very important for fleshing out your portal page.

Let’s say you want a widget of vip users and their picture. The server would send the data with that information and the client receives that data object and then displays it the way you desire.

```javascript
data.names = [];

var userGR = new GlideRecord("sys_user");
userGR.addQuery("vip", true);
userGR.query();
```
while (userGR.next()) {
    data.names.push(userGR.name.toString());
}

Once the code reaches the bottom the data object will be sent. So now you have an array of VIP names that you can use.

### 16.2.1 The Client can send Data also

So let’s say you want to give users the ability to change what kind of users they want to see. For example, the user chooses to query for, let’s say, all ITIL users.

On the client side code, create an object literal called `input` and fill in the information the server would need. For this example, we can send the string “itil” in the input object. Once we are ready we use `server.update()` to send the input object to the server.

```javascript
var c = this;
var input = {};

$scope.onItilButtonClick = function() {
    input.button = "itil";
    c.server.update();
}
```

After `c.server.update()`; the process then repeats sending whatever data object the server now has back to client, and re-renders if anything changes.

### 16.2.2 Processing the input sent from client

The server client script gets processed in whole after a `c.server.update()`, so to prevent the original query from being processed, we’ll have to wrap what we want in an `if (input) {}` statement. The only difference from the first `onLoad` and the runs from the client is the appearance of the input object.

```javascript
data.names = [];

if (input) {
    var userGR = new GlideRecord("sys_user");
    userGR.addQuery('roles=' + input.button);
    userGR.query();

    while (userGR.next()) {
        data.names.push(userGR.name.toString());
    }
} else {
    var userGR = new GlideRecord("sys_user");
    userGR.addQuery("vip", true);
    userGR.query();

    while (userGR.next()) {
        data.names.push(userGR.name.toString());
    }
}
```
The else statement here is defaulted to searching for vip. When an input is sent to the server it will start a new query looking for whatever role is in input.button.

Code that you want to run on both `onLoad` and when a client re-renders, just put them in the script normally. Such as the initialization of the data.names array on the first line.

```
if (input) {
    data.foo = input.foo
}
```

**Warning:** Only data will get sent to the client from the server. If you want to send input back you must put it back into the data object.
17.1 Sphinx codes

/*
.. code-block:: Javascript

    var someCode = "some more code"

.. warning::

    Some warning block
*/
18.1 Location

System Definition > Syntax Editor Macros
19.1 Generate Time Cards based on Resource Allocations

Currently timecards are only generated for the table planned_task and only for the current week

19.1.1 Goal

- Generate timecards when the current user is assigned as a resource allocation
- Generate timecards for upcoming weeks (not just the current week)

19.1.2 Script Include

The only script we need to edit is the TimeCardAjax script include. Since this is OOB, it’s best to insert-and-stay then mark the original as inactive.

The only OOB function we will be editing is ajaxFunction_generateTaskCards(), the rest we will be creating ourselves.

```javascript
ajaxFunction_generateTaskCards: function(){
    // get request info
    this.weekStart = this._getWeekStart();
    this.user = this.getParameter('sysparm_user');
    this.weekEnd = this._getWeekEnd();
    this.newCards = [];// list of cards created
    this.existingCards = this._getExistingCards();
    this.plannedTasks = this._getPlannedTasks();
    this._generateMissingCards(this.plannedTasks);
    this._generateMissingCards(this.tasks);
    // added two functions to be run for Resource Allocations
    this.resourceAllocations = this._getResourceAllocations();
    this._generateMissingAllocations();
}
```
This is the function that gets run initially, when TimecardAjax is called. What I added were two lines between `this._generateMissingCards(this.plannedTasks)` and `this._buildResult();`.

Both functions should be self explanatory. The `_getResourceAllocation()` grabs all resource allocations for the user and `_generateMissingAllocations()` will generate timecards if there are no timecards associated.

### 19.1.3 Unit Tests

```javascript
var alloc = new GlideRecord("resource_allocation");
alloc.user = "0ac0a7f7db20220003ee793ebf961970";
var date = new GlideDate();
date.setValue('2016-11-05');
alloc.start_date = date;
var date2 = new GlideDate();
date2.setValue('2016-11-23');
alloc.end_date = date2;
alloc.requested_hours = "30";
alloc.task = "48b1d30d95a20003ee793ebf961942"
alloc.resource_plan = "e3bc5091db19a20003ee793ebf961943"
var test = alloc.insert();
gs.log(test);
```

```javascript
var newTask = new GlideRecord("cert_follow_on_task");
newTask.assigned_to = "0ac0a7f7db20220003ee793ebf961970";
newTask.short_description = "Unit Test 08171131";
var taskId = newTask.insert();
gs.log("taskId = " + taskId);

var newRP = new GlideRecord("resource_plan");
newRP.task = taskId;
newRP.start_date = "2015-01-01";
newRP.end_date = "2019-01-01";
newRP.user_resource = "0ac0a7f7db20220003ee793ebf961970";
newRP.resource_type = "user";
newRP.planned_hours = 0;
var resourceId = newRP.insert();
gs.log("resourceID = " + resourceId);

var alloc = new GlideRecord("resource_allocation");
alloc.user = "0ac0a7f7db20220003ee793ebf961970";
var date = new GlideDate();
date.setValue('2016-11-05');
alloc.start_date = date;
var date2 = new GlideDate();
date2.setValue('2016-11-23');
alloc.end_date = date2;
alloc.requested_hours = "30";
alloc.task = taskId;
alloc.resource_plan = resourceId;
```
```javascript
var allocationId = alloc.insert();
gs.log("allocationId = " + allocationId);

var alloc2 = new GlideRecord("resource_allocation");
alloc2.user = "0ac0a7f7db20220003ee793ebf961970";
var date = new GlideDate();
date.setValue('2016-11-16');
alloc2.start_date = date;
var date2 = new GlideDate();
date2.setValue('2016-12-02');
alloc2.end_date = date2;
alloc2.requested_hours = "30";
alloc2.task = taskId;
alloc2.resource_plan = resourceId;

allocationId = alloc2.insert();
gs.log("allocationID2 = " + allocationId);
```
20.1 Triggering on change for select box

To prevent using gel, we can pass in the current value of the select box using this.value

20.1.1 HTML

```html
<g:evaluate>
  var choiceList = new GlideChoiceList();
  choiceList.add('', 'None--');
  choiceList.add('123', 'OneTwoThree');
  choiceList.add('FourFiveSix', '456');
</g:evaluate>

<select id="document_select" name="document_select" onchange="myFunction(this.value)"
  <g:options choiceList="${choiceList}" choiceValue="" />
</select>
```

20.1.2 Client Script

```javascript
function myFunction(val) {
  alert(val);
}
```
20.2 Using a passed parameter in HTML field

20.2.1 HTML

```html
<g:evaluate var="jvar_document_array" expression="RP.getWindowProperties().get('sysparm_document_array')" />
<script>
    alert('"${jvar_document_array}"');
</script>
```

20.3 Using a passed parameter in the Client field

20.3.1 Client script

```javascript
if ("${sysparm_checked_out}" == "true") {
    alert('"${sysparm_checked_out}"');
}
```
21.1 Variable Iteration

Thanks to Andrew Sainz for finding this

```javascript
var item = new GlideRecord('sc_req_item');
item.addQuery('sys_id', current.sysapproval);
item.query();

if (item.next()) {
    for (var key in item.variables) {
        var v = item.variables[key];
        if (v != '' && !v.nil()) {
            current.u_approval_summary += v.getGlideObject().getQuestion().getLabel() + ' : ' + v.getDisplayValue() + '\n';
        }
    }
}
```
CHAPTER 22

Workflows

22.1 Canceling and Reopening Workflow

```javascript
function server_cancel_business_case()
{
    // Cancels all approvals for Business Case and delete workflow
    new WorkflowApprovalUtils().cancelAll(current, "Placed on hold by " + gs.getUser().name);
    new Workflow().deleteWorkflow(current);

    current.u_ppm_bc_state = 'closed_cancelled';
    current.update();
    action.setRedirectURL(current);
}
```

// Cancels all approvals for Business Case and delete workflow
new WorkflowApprovalUtils().cancelAll(current, "Re-Opened by " + gs.getUser());
new Workflow().deleteWorkflow(current);

22.2 Sending in workflow values with StartFlow

UI Action / Business Rule

```javascript
var wf = new Workflow(); // Get the workflow id
var wfId = wf.getWorkflowFromName("Print Screen");
var vars = {};
vars.u_task_sys_id = current.sys_id.toString();
vars.variable2 = "This is var2";
wf.startFlow(wfId, current, "Update", vars);
```

Within Workflow Script

53
var readValue = workflow.inputs.u_task_sys_id;
// workflow.variables_u_task_sys_id ... will also work

var projectTaskGR = new GlideRecord("pm_project_task");
projectTaskGR.get("sys_id", readValue);
projectTaskGR.state = 3;
projectTaskGR.update();
CHAPTER 23

Indices and tables

- genindex
- modindex
- search