

# ServiceNow 201:

## *9 - Becoming A Scripting Master*



# Course Outline

---

1 Course Introduction

2 Development Overview

3 Scripting Locations

4 GlideRecord

5 GlideSystem

6 GlideForm & GlideUser

7 GlideAjax

8 Exploring Other APIs

9 Becoming A Scripting Master

10 Creating A Custom App

# Section Outline

---

1 General Best Practices

2 Server-Side Best Practices

3 Client-Side Best Practices

4 Debugging

5 Validating JavaScript

6 Additional JavaScript Tools

7 Debugging Demo

8 Version Control

9 Tips

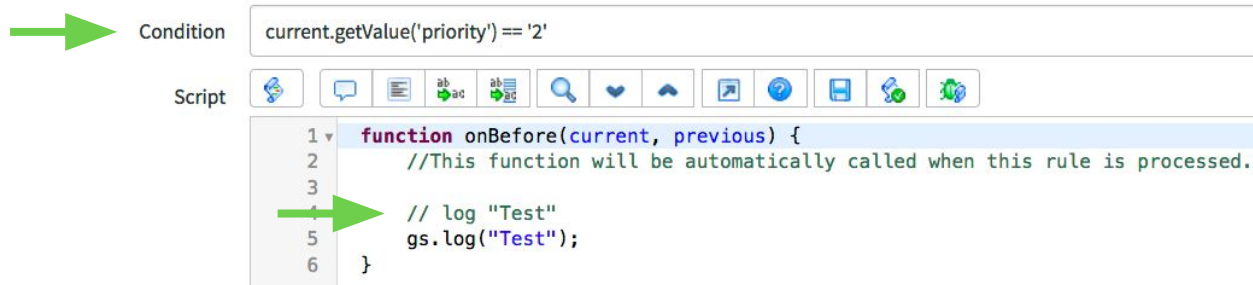
10 Help!

11 Version Control Demo

# Best Practices: General Scripting

---

- Add a short description to scripts when available
- Use condition statements if condition field is available depending on scripting location
- Comment your code! ([source](#))
- Follow a convention - especially for debugging
- Wrap code in functions to prevent polluting the global namespace ([source](#))



# Best Practices: General Scripting (cont.)

- Do not use hardcoded values ([source](#))
- Use descriptive variables and function names ([source](#))
- Verify a value exists before using it ([source](#))
- Let the database do the work ([source](#))



```
116 ▾ SPCart.prototype.getItem = function(cartItemID) {  
117 ▾   function showQuantity() {  
118 →     var roles = gs.getProperty('glide.sc.allow.quantity');  
119     var showItemQuantityByRole = true;  
120     if (roles == null || roles == '')  
121     ;  
122 ▾     else {  
123         var hasRole = gs.hasRole(roles);  
124         if (hasRole == false)  
125             showItemQuantityByRole = false;  
126     }  
}
```

```
220  
221 ▾   if (gr.active == true) {  
222 →     if (!gr.end_text.nil())  
223         current.setValue("state", gr.end_text);  
224         if (type != 'automatic' && type != 'manual')  
225             return;  
226  
227     var executedScript = this._executeScript(gr, type + "_script", current);  
228 ▾     if (this._needsUpdate(gr, type, executedScript, gr.end_text.nil())) {  
229 ▾         if (!!this._isNewRecord(current, type) || (type == "manual")) {  
230             current.update();  
231         }  
}
```

# Best Practices: Server-Side



- Use GlideAggregate over GlideRecord when dealing with aggregates like *counts*
- Log records before deleting
- Use GlideRecordSecure where appropriate
- Use Script Includes over global Business Rules



```
102     var count = new GlideAggregate(tableName);
103     count.addQuery('sys_created_by', userName);
104     count.addQuery('sys_created_on', '>=', gs.hoursAgo(24)); // check for # records
created in last 24 hours
105     count.addAggregate('COUNT');
106     count.query();
107
108     var totalCount = 0;
109     if (count.next())
110         totalCount = parseInt(count.getAggregate('COUNT'));
```

# Best Practices: Client-Side

---



- Make as few calls as possible
- Do not make synchronous calls
  - `GlideRecord` on client-side
  - `g_form.getReference` w/o callback
- Use `GlideAjax` when passing data between server-side & client-side
  - Use JSON when passing data from server-side to client-side
- Debug using `console.log`
- Avoid direct DOM manipulation
- Use UI Policies over Client Scripts when available



# Best Practice Resources

---

- ServiceNow
  - [Coding Best Practices](#)
  - [Client Script Best Practices](#)
  - [Business Rule Best Practices](#)
  - [Commenting Best Practices](#)
  - [Update Set Best Practices](#)
- Other
  - [servicenowguru.com](#)
  - [servicenowelite.com](#)
  - More...

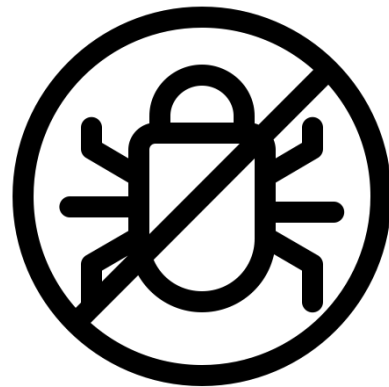




# Debugging

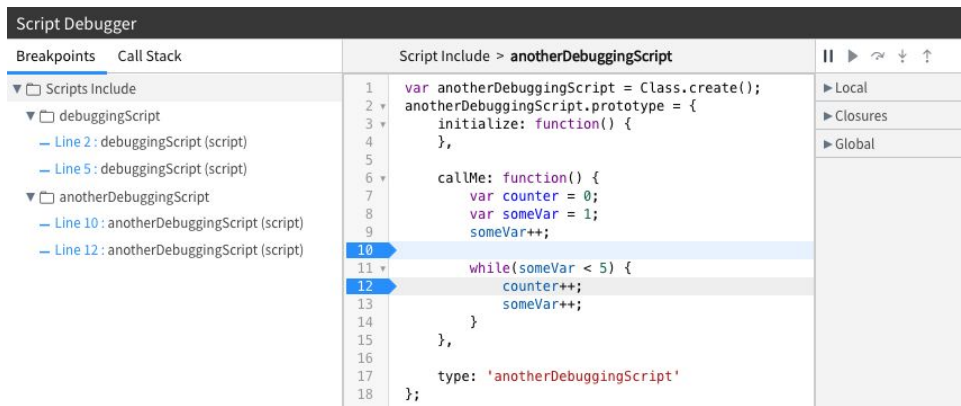
---

- You will run into bugs, it's ok
- Is the issue reproducible? Why not?
- Debug the smart way, know your tools
- Is the error/bug due to:
  - ServiceNow API
  - JavaScript problem
  - Scoping issue
  - Etc.
- Oftentimes simply logging values will be enough



# Debugging Tools

- Server-side
  - Script Debugger
  - Session debuggers
- Client-side
  - Browser console
  - JavaScript Executer
- Other resources:
  - [Debugging Tools Best Practices](#)
  - [Script Debugger Docs](#)
  - [Using Chrome to Debug Client Side Errors](#)



# Debugging Tools: Script Debugger

- Set breakpoints
- View variables
- Examine call stack

The screenshot displays the Script Debugger interface. On the left, the 'Breakpoints' tab is active, showing a tree view of 'Scripts Include'. Under 'debuggingScript', there are two breakpoints: 'Line 2: debuggingScript (script)' and 'Line 5: debuggingScript (script)'. Under 'anotherDebuggingScript', there are two breakpoints: 'Line 10: anotherDebuggingScript (script)' and 'Line 12: anotherDebuggingScript (script)'. The main pane shows the script 'Script Include > anotherDebuggingScript' with the following code:

```
1 var anotherDebuggingScript = Class.create();
2 anotherDebuggingScript.prototype = {
3   initialize: function() {
4   },
5
6   callMe: function() {
7     var counter = 0;
8     var someVar = 1;
9     someVar++;
10
11     while(someVar < 5) {
12       counter++;
13       someVar++;
14     }
15   },
16
17   type: 'anotherDebuggingScript'
18 };
```

Lines 10 and 12 are highlighted with blue arrows, indicating breakpoints. On the right, the 'Local' tab is active, showing an empty list of local variables. The 'Closures' and 'Global' tabs are also visible.

# Debugging Tools: Session debugging

- Oftentimes a last resort
- Can log just about everything
- Business Rule orders, Security, UI policies, etc.
- Checkout this [wiki page](#) for more info

```
✓ 10:43:58.631: Global TIME = 0:00:00.000 PATH = ui/context_menu.view/read CONTEXT = null RC = true Rule =  
✓ ui/context_menu.view/read  
✓ 10:43:58.635: Time: 0:00:00.001 for: [glide.12] SELECT ... FROM sys_attachment sys_attachment0 WHERE sys_attachment0.`table_name`  
sys_attachment0.`encryption_context` IS NULL ORDER BY sys_attachment0.`sys_created_on` /*...*/  
✓ 10:43:58.637: TIME = 0:00:00.000 PATH = record/incident/write CONTEXT = INC0020188 RC = true RULE = Evaluated from cache  
✓ record/incident/write Global  
✓ record/incident/write Global  
✓ 10:43:58.638: TIME = 0:00:00.001 PATH = record/incident.number/read CONTEXT = INC0020188 RC = true RULE = Evaluated from cache  
✓ record/incident/read App:Security Incident  
✓ record/incident/read Global  
✓ record/incident/read Global  
✓ record/incident/read Global  
✓ 10:43:58.640: TIME = 0:00:00.000 PATH = record/incident.number/write CONTEXT = INC0020188 RC = true RULE =  
✓ record/incident/write Global ✓ record/task.number/write Global  
✓ record/incident/write Global  
10:43:58.644: Attempted script access to inaccessible member denied - com.glide.script.ElementDebugMessage:getType:()Ljava/lang/String;  
10:43:58.660: Time: 0:00:00.003 for: [glide.15] SELECT ... FROM (sys_script sys_script0 INNER JOIN sys_metadata sys_metadata0 ON sys  
sys_script0.`action_query` = 1 AND sys_script0.`active` = 1 AND sys_script0.`when` = 'before' ORDER BY sys_script0.`order`, sys_script0.`name` /*  
10:43:58.674: Time: 0:00:00.006 for: [glide.16] SELECT ... FROM sys_illegal_member sys_illegal_member0 WHERE sys_illegal_member0.  
AND sys_illegal_member0.`signature` = '()Ljava/lang/String;' /*...*/  
10:43:58.683: Attempted script access to inaccessible member denied - com.glide.script.ElementDebugMessage:getType:()Ljava/lang/String;  
10:43:58.686: Time: 0:00:00.000 for: [glide.18] SELECT ... FROM sys_illegal_member sys_illegal_member0 WHERE sys_illegal_member0.  
AND sys_illegal_member0.`signature` = '()Ljava/lang/String;' /*...*/
```

## System Diagnostics

Script Debugger

▼ Session Debug

Enable All

Disable All

Debug Business Rule

Debug Business Rule (Details)

Debug Log

Debug SQL

Debug SQL (Detailed)

Debug Security

Debug Escalations

Debug Text Search

Debug UI Policies

Disable UI Policies Debug

Debug Data Policies

# Validating JavaScript with JSBin

---

- Test JavaScript logic with JSBin's JavaScript runtime environment

JavaScript ▾

```
var currentHour = 1;

var gs = {};
gs.getUserDisplayName = function() {
  return 'Mark Miller';
}

var greetingsMessage = '';

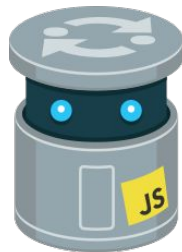
if(currentHour >= 3 && currentHour < 11) {
  greetingsMessage = 'Good morning ';
} else if(currentHour >= 11 && currentHour < 17) {
  greetingsMessage = 'Good afternoon ';
} else {
  greetingsMessage = 'Good evening ';
}

greetingsMessage += gs.getUserDisplayName();

console.log(greetingsMessage);
```

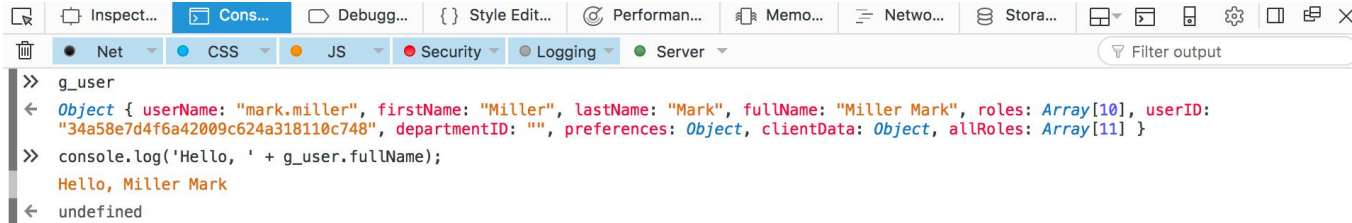
Console

"Good evening Mark Miller"

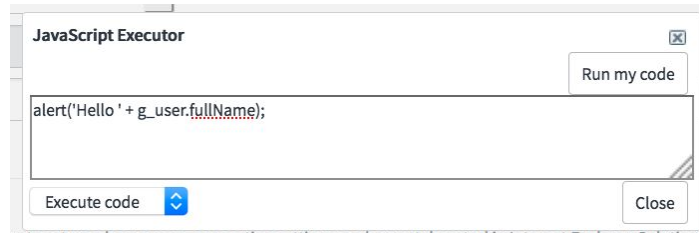


# Browser Console & ServiceNow JavaScript Executor

- If testing/debugging client-side scripts, try your browser's JavaScript console or ServiceNow's JavaScript Executor



*Firefox Developer Tools*



*ServiceNow JavaScript Executor*

# Exploring Client-Side JavaScript Scope

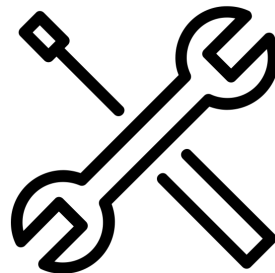
- Use browser's `console.dir()` method



# Additional JavaScript Tools

---

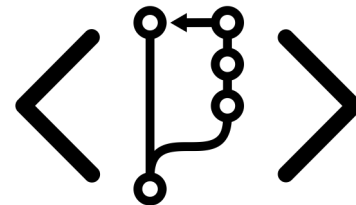
- [loupe](#) - JavaScript Event Loop
- [metajs](#) - JavaScript AST
- [JSBin](#) - JS debugging
- [JSFiddle](#) - JS debugging
- [Plunker](#) - JS debugging
- [JSLint](#) - Syntax checker & more
- [RegExr](#) - Regular Expressions builder





# Version Control With Update Sets

- Find the difference between 2 script versions
- Similar to DiffMerge or other diff tools



Compare to Current - [sys\_script\_include] - ServiceNow201ScriptInclude

Revert to Selected Version Save Merge

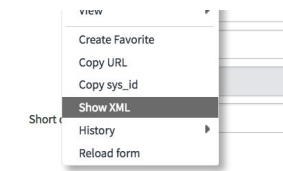
Compare to Current - Script

Selected Version	Current Version
<pre>1 var ServiceNow201ScriptInclude = Class.create(); 2 ServiceNow201ScriptInclude.prototype = Object.extend(Object.prototype, { 3   getIncidentStatus: function() { 4     var incidentNumber = this.getParameter('sysparm_incident'); 5     if(incidentNumber) { 6       var incidentGR = new GlideRecord('incident'); 7       incidentGR.get('number', incidentNumber); 8       return incidentGR.state.getDisplayValue(); 9     } else { 10      return 'No incident was found'; 11    } 12  }, 13 14  type: 'ServiceNow201ScriptInclude' 15 });</pre>	<pre>1 var ServiceNow201ScriptInclude = Class.create(); 2 ServiceNow201ScriptInclude.prototype = Object.extend(Object.prototype, { 3   getIncidentStatus: function() { 4     var incidentNumber = this.getParameter('sysparm_incident'); 5     if(!gs.nil(incidentNumber)) { 6       var incidentGR = new GlideRecord('incident'); 7       incidentGR.get('number', incidentNumber); 8       return incidentGR.state.getDisplayValue(); 9     } else { 10      return 'No incident was found'; 11    } 12  }, 13 14  sayHelloToTheWorld: function() { 15    return 'Hello world!'; 16  }, 17 18  type: 'ServiceNow201ScriptInclude' 19 });</pre>

Cancel OK

## Tips

- Application Navigator
  - *table\_name*.list
  - *table\_name*.do
  - cache.do
  - stats.do
- *Show XML* UI action
- View object properties
- Right-click a form field to view additional info
- When locating scripts, use the *contains* keyword



```
1 // obj = JavaScript object
2 for(property in obj) {
3     gs.log(property + ': ' + obj[property]);
4 }
```

**Client Scripts (Client Scripts)** New for text Search Grid Split

All > Class = Client Script > Script contains GlideAjax

Load Filter Save Filter Add Sort Clear All Run

### CLIENT SCRIPT CONDITIONS

All of these conditions must be met

Class	is	Client Script	-	OR	AND
Script	contains	GlideAjax	-	OR	AND

or

New Criteria

### RELATED LIST CONDITIONS ?

Name	Active	Table	Application	View	Type
Search	Search	Search	Search	Search	Search
onLoadHello	true	incident	Global		onLoad

# Help!

---

- What to do when you're stuck

- Give debugging a try
- Google
- ServiceNow community
- Stackoverflow
- reddit.com/r/servicenow
- Other 3rd-party websites
- Course Q&A



Overview

Course Content

Q&A

Bookmarks

Announcements

Options ▾

Search for a question

or

Ask a new question

# Section Recap

---

- Give debugging a try; play around with the tools
- Try Googling for answers
- Post to the community, Stackoverflow or course Q&A
- Validate your JavaScript logic
- Review ServiceNow best practices

