knowledge15

Everything as a Service

Scripting 202:

Server-side JavaScript in ServiceNow

Andrew Venables
Principal Technical Consultant
ServiceNow

Chris Terzian
Principal Solution Architect
ServiceNow

© 2015 ServiceNow All Rights Reserved

servicenow

© 2015 ServiceNow All Rights Reserved

Agenda

Introductions

JavaScript Context and Naming Collision (20 minutes)

Debugging Business Rules and Script Includes (40 minutes)

Unified Code through Script Includes (35 minutes)

Use GlideAggregate for Faster Queries (10 minutes)

© 2015 ServiceNow All Rights Reserved

Introductions

Your Presenters

© 2015 ServiceNow All Rights Reserved

Your Presenters

- Andy Venables
  - Principal Technical Consultant
  - Professional Services UK
  - With ServiceNow for 2+ years
  - Development Background
- Chris Terzian
  - Principal Solution Architect
  - Professional Services US
  - With ServiceNow for 4 Years
  - IT Consulting Background

© 2015 ServiceNow All Rights Reserved

Connectivity & Lab Environment

- Ethernet: Just plug in!
- WIFI: Knowledge15 / servicenow2015

- Instance URL will be: https://script202-<###>.lab.service-now.com
- Please raise your hand if you have difficulties logging into your instance

- Lab Guide available from: http://bit.ly/1El81q9

© 2015 ServiceNow All Rights Reserved

Server-side Scripting

JavaScript Context and Naming Collision

© 2015 ServiceNow All Rights Reserved

Global vs. Local vs. IIFE (Immediately Invoked Function Expression)

- How do the three examples differ?
- Is one better/worse?
- Self-document with descriptive variable names

```
var c = 'my comment';
current.comments = c;

//////////////////////

setComment();

function setComment() {
    var c = 'my comment';
    current.comments = c;
}

//////////////////////

(function() {
    var c = 'my comment';
    current.comments = c;
})();
```


© 2015 ServiceNow All Rights Reserved

Lab 1 Business Rule Scope

- Explore the impact of Business Rule scope

Lab Goal

This lab explores techniques for writing robust business rules and preventing pollution of the global scope.

Lab 1
Business
Rule Scope

© 2015 ServiceNow All Rights Reserved

Business Rule Scope

New feature in Fuji, all business rules now automatically encased in function

Name: MyNewRule

Table: Incident [incident]

Actions | **Advanced**

Condition:

Script:

```
1 function onBefore(current, previous) {
2    //This function will be automatically called when this rule is processed.
3
4 }
```


© 2015 ServiceNow All Rights Reserved

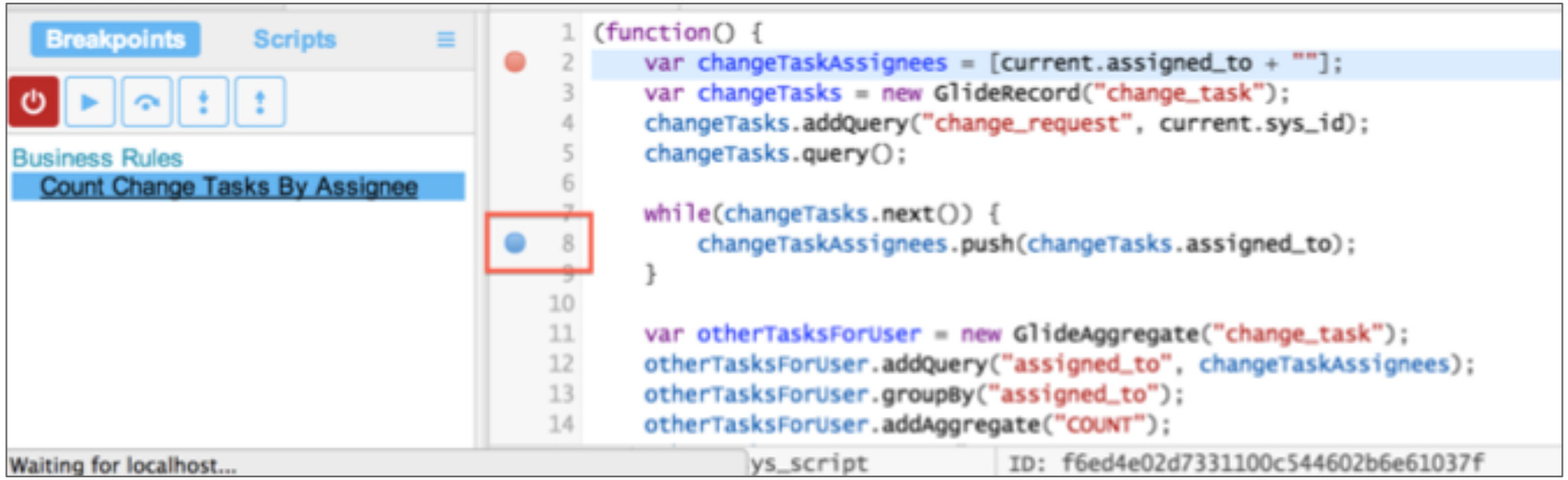
Server-side Scripting

Debugging Business Rules and Script Includes

© 2015 ServiceNow All Rights Reserved

JavaScript Debugger

Dublin feature to observe and modify server-side code at run-time



© 2015 ServiceNow All Rights Reserved

Lab 2 Server-Side JavaScript Debugger

- Use the server-side JavaScript debugger

Lab Goal

This lab explains how to leverage the Server-Side JavaScript Debugger to effectively debug a faulty business rule.

Lab 2
Server-Side
JavaScript
Debugger

© 2015 ServiceNow All Rights Reserved

Referenced vs. De-referenced Values

In our previous lab, we found that our array was misbehaving.

What was happening?

```
▼ Local
▼ changeTaskAssignees: Array
    0: 5137153cc611227c000bbd1bd8cd2005
    1: f298d2d2c611227b0106c6be7f154bc8
```

```
▼ Local
▼ changeTaskAssignees: Array
    0: 5137153cc611227c000bbd1bd8cd2005
    1: 9ee1b13dc6112271007f9d0efdb69cd0
    2: 9ee1b13dc6112271007f9d0efdb69cd0
```

```
7   while(changeTasks.next()) {
8       changeTaskAssignees.push(changeTasks.assigned_to);
9   }
```

```
7   while(changeTasks.next()) {
8       changeTaskAssignees.push(changeTasks.assigned_to + "");
9   }
```


© 2015 ServiceNow All Rights Reserved

Server-side Scripting

Unified Code through Script Includes

© 2015 ServiceNow All Rights Reserved

Script Includes Enable Re-usable Code

Move complex or duplicated condition logic

Condition	gs.hasRole("itil") && current.caller_id.company.u_critical_account && current.caller_id.vip

into a well named Script Include!

Condition	new RelatedTaskCounts().checkConditions(current, "caller_id")

- **Could also be a function in a Script Include named "checkConditions" rather than a class.**
- **Benefits:**
  - More readable
  - Shorter condition (max length 254 characters for the field)
  - Changes to condition in one place for multiple Business Rules

© 2015 ServiceNow All Rights Reserved

Script Includes Enable Re-usable Code

Avoid duplication of code. Similar functions can be unified to increase maintainability.

```
(function() {
      var incidentsBySameCaller = new GlideRecord("incident");
      incidentsBySameCaller.addQuery("caller_id", current.caller_id);
```

```
(function() {
      var problemsBySameUser = new GlideRecord("problem");
      problemsBySameUser.addQuery("opened_by", current.opened_by);
      problemsBySameUser.query();

      gs.addInfoMessage(current.opened_by.getDisplayValue() +
            " has " + problemsBySameUser.getRowCount() +
            " Problems(s).");
})();
```

```
(function() {
      new RelatedTaskCounts().getRecordCount(current, "opened_by");
})();
```

```
getRecordCount: function(current, field) {
      var recordsBySameUser = new GlideRecord(current.sys_class_name);
      recordsBySameUser.addQuery(field, current[field]);
      recordsBySameUser.addActiveQuery();
      recordsBySameUser.query();

      gs.addInfoMessage(current[field].getDisplayValue() +
      " has " + recordsBySameUser.getRowCount() +
      " active " + current.getClassDisplayValue() + "(s) .");
},
```


© 2015 ServiceNow All Rights Reserved

Lab 3.1 DRY – Don’t Repeat Yourself

- Create reusable code in Script Includes

Lab Goal

This lab explains how to use Script Includes to eliminate duplicate code and centralize business rule conditions.

Lab 3.1
Do Not
Repeat
Yourself

© 2015 ServiceNow All Rights Reserved

Lab 3.2 Code Re-usability

Lab Goal

This lab explores decomposing complex code to extract common routines. You learn to move record counting code into its own script include so that it can be re-used for incident reassignment.

Lab 3.2
Code
Reusability

© 2015 ServiceNow All Rights Reserved

Server-side Scripting

Use GlideAggregate for Faster Queries

© 2015 ServiceNow All Rights Reserved

GlideAggregate

- Uses SQL “group by” and “having” clauses to calculate the aggregate data at the database rather than in the JVM
- Improves calculation time over GlideRecord
- Reduces data sent to the JVM
- Much more efficient than getRowCount(); scales better as row count increases

```
var count = new GlideAggregate('incident');
count.addAggregate('MIN', 'sys_mod_count');
count.addAggregate('MAX', 'sys_mod_count');
count.addAggregate('AVG', 'sys_mod_count');
count.groupBy('category');
count.query();
while (count.next()) {
   var min = count.getAggregate('MIN', 'sys_mod_count');
   var max = count.getAggregate('MAX', 'sys_mod_count');
   var avg = count.getAggregate('AVG', 'sys_mod_count');
   var category = count.category.getDisplayValue();
   gs.log(category + " Update counts: MIN = " + min + " MAX = " + max + " AVG = " + avg);
}
```


© 2015 ServiceNow All Rights Reserved

getRowCount is evil

- **Instead of using:**

```
if (gr.getRowCount() < 1) {
    // do something
}
```

- **Use simply:**

```
if (gr.hasNext()) {
    // do something
}
```

- **Ever been asked to write a report that finds duplicates in the CMDB?**

```
var ciGA = new GlideAggregate('cmdb_ci');
ciGA.addNotNullQuery('name');
ciGA.groupBy('name');
ciGA.groupBy('sys_class_name');
ciGA.addAggregate('COUNT');
ciGA.addHaving('COUNT', '>', 1);
ciGA.query();
while (ciGA.next()) {
  gs.print(ciGA.getAggregate('COUNT')+'x '+ciGA.name+' ['+ciGA.sys_class_name+']');
}
```


© 2015 ServiceNow All Rights Reserved

Lab 3.3 Glide Aggregate

Lab Goal

This lab illlustrates the benefits of modular code by swapping the **RecordCounter** implementation with another that uses **GlideAggregate**.

Lab 3.3
Glide
Aggregate

© 2015 ServiceNow All Rights Reserved

Challenge!

- **How would you run a Server-side script on your instance right now?**
  - Scheduled Job?
  - Business Rule?
  - Script Include?

- **Use a Background script!**
  - Scripts are added to the log after being run
  - gs.log is printed to the screen (and saved to the log)
- **Add your own custom scripts!**
  - Upload js file and appears in the list
  - http://bit.ly/1P6MyD8

© 2015 ServiceNow All Rights Reserved

Scope!

- In Fuji everything now has a scope!

- Background scripts can be run against any scope, but make sure you have the correct scope!

http://wiki.servicenow.com/index.php?title=Scripting_in_Scoped_Applications

Check out the Application Creator course / lab

© 2015 ServiceNow All Rights Reserved

Top Takeaways

1 Wrap scripts in functions to protect against variable name collision.

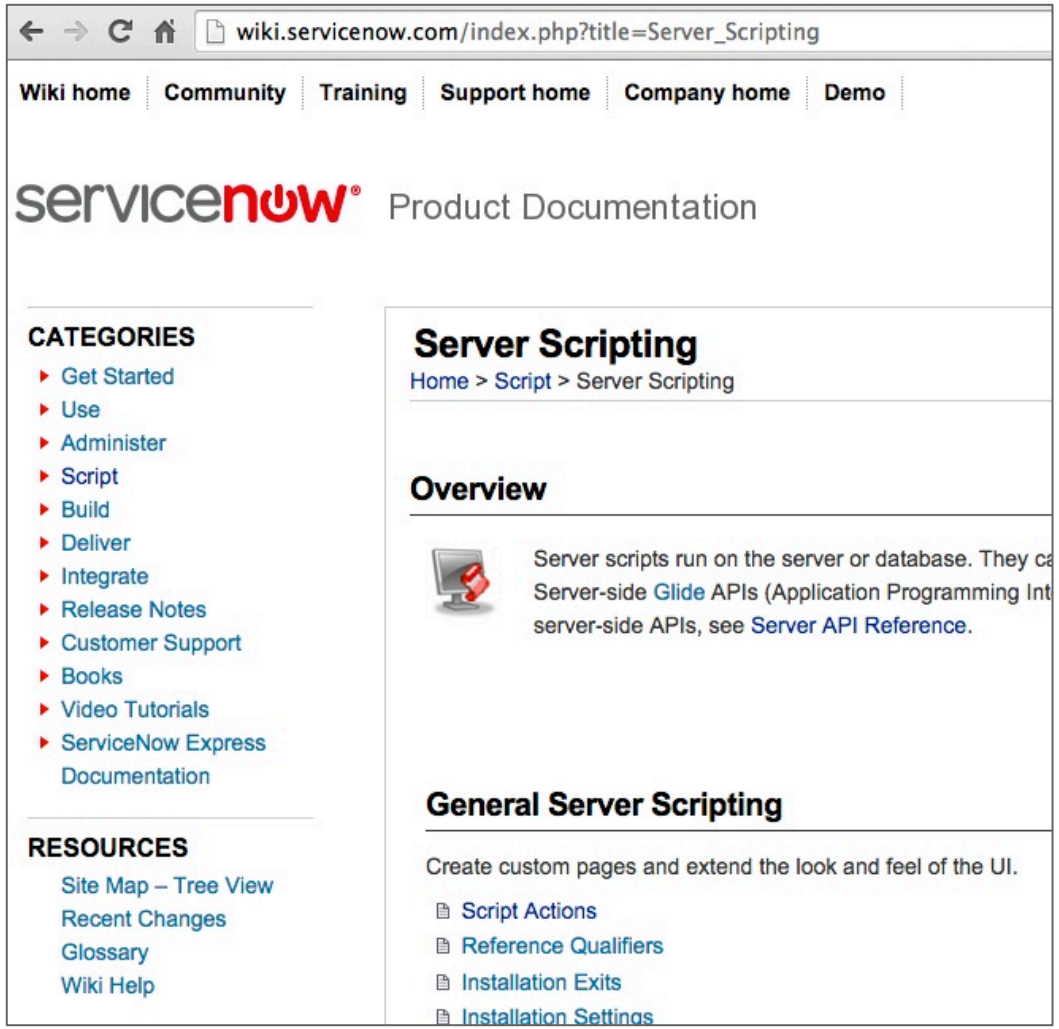
2 The JavaScript debugger allows you to hook in to server-side code at run-time to observe variables and process flow.

3 Use Script Includes to de-duplicate code.

© 2015 ServiceNow All Rights Reserved

Additional Resources

- http://wiki.servicenow.com/index.php?title=JavaScript_Debugger
- http://lmsnwtfy.com/?q=script includes

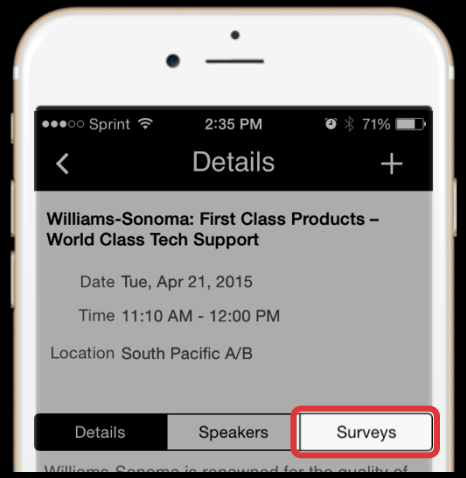


© 2015 ServiceNow All Rights Reserved

How Did We Do?

Your feedback on this session helps us deliver great content.

Please take a moment to complete a session survey in the Knowledge15 app or use the survey forms at the back of the room.



Everything as a Service

Thank You

Andrew Venables

Principal Technical Consultant

ServiceNow

andrew.venables@servicenow.com

Chris Terzian

Principal Solution Architect

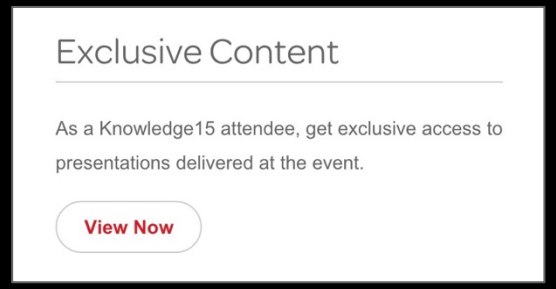
ServiceNow

chris.terzian@servicenow.com

Get Presentations

As a Knowledge15 attendee, you have exclusive access to breakout and lab session content from the event.

1. Go to knowledge.servicenow.com
2. Log into the community
3. Click on **View Now** button



knowledge.servicenow.com

© 2015 ServiceNow All Rights Reserved