knowledge15

Everything as a Service

# Technical Best Practices 201: Avoid Pitfalls

Chuck Tomasi
Sr. Services Enablement Program Manager
ServiceNow

Jonatan Jardi
Sr. Technical Consultant
ServiceNow



servicenow

# Agenda

Business Rules Best Practices

Client Script Best Practices

Coding Best Practices

Excerpt from
TBP Workshop

# Chuck Tomasi

## Sr. Services Enablement Program Manager

- 30 years IT experience
- ServiceNow customer 2008-10
- First Innovation of the Year Award @ K10
- ServiceNow since 2010
- Lots of deployments, custom apps, and special projects
- Technical Best Practices owner
- Co-host of ServiceNow web series *TechNow*
- Other stuff:
  - Author, Podcaster, golfer, downhill skier, photographer, sci-fi geek, martial artist

# Jonatan Jardi

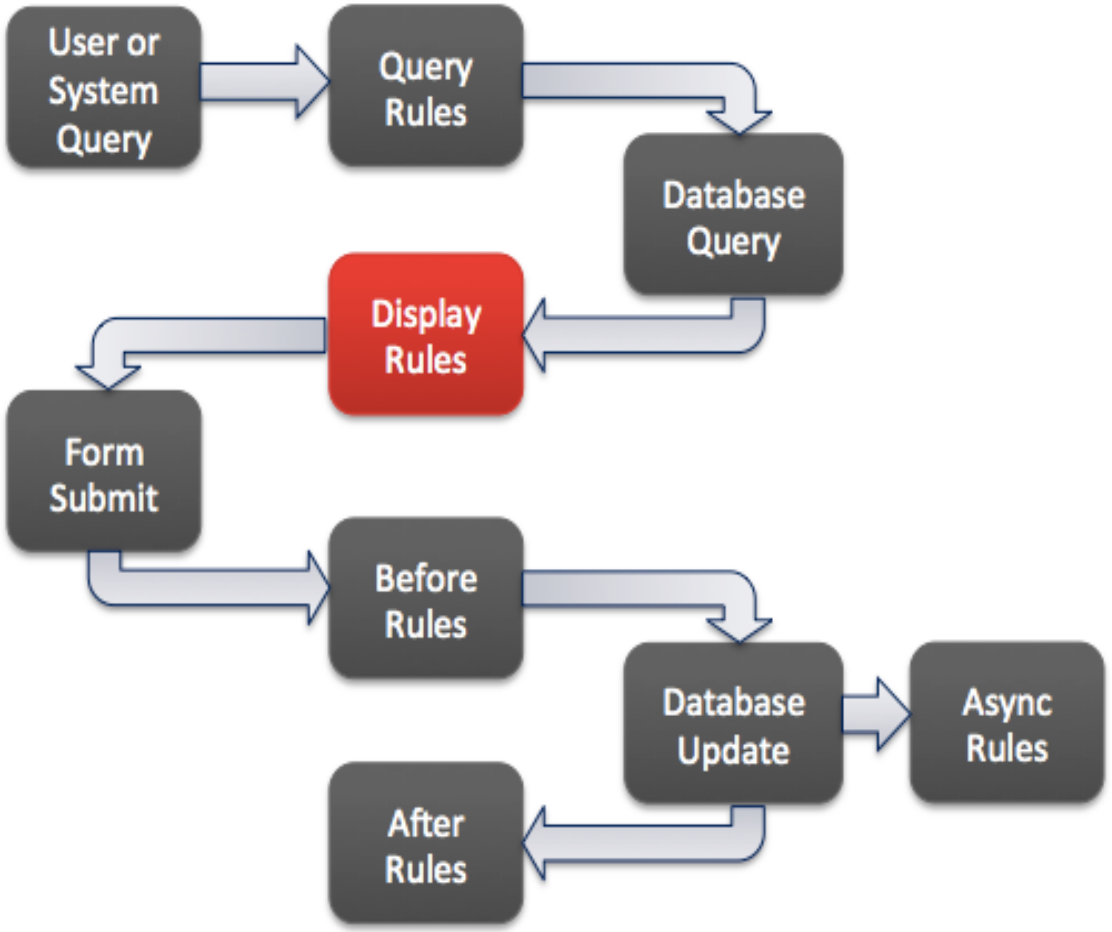
## Sr. Technical Consultant

- 10 years IT experience
- 5 years working **with** ServiceNow
- 2.5 years working **for** ServiceNow
- Technical Best Practices Workshop Leader
- Enjoy anything to do with Integrations, SSO, Discovery & Orchestration
- Other stuff:
  - **Love** football (soccer), going to the movies, reading sci-fi books.
  - Fluent in Spanish (Argentina) and Brazilian Portuguese

# Business Rules Best Practices

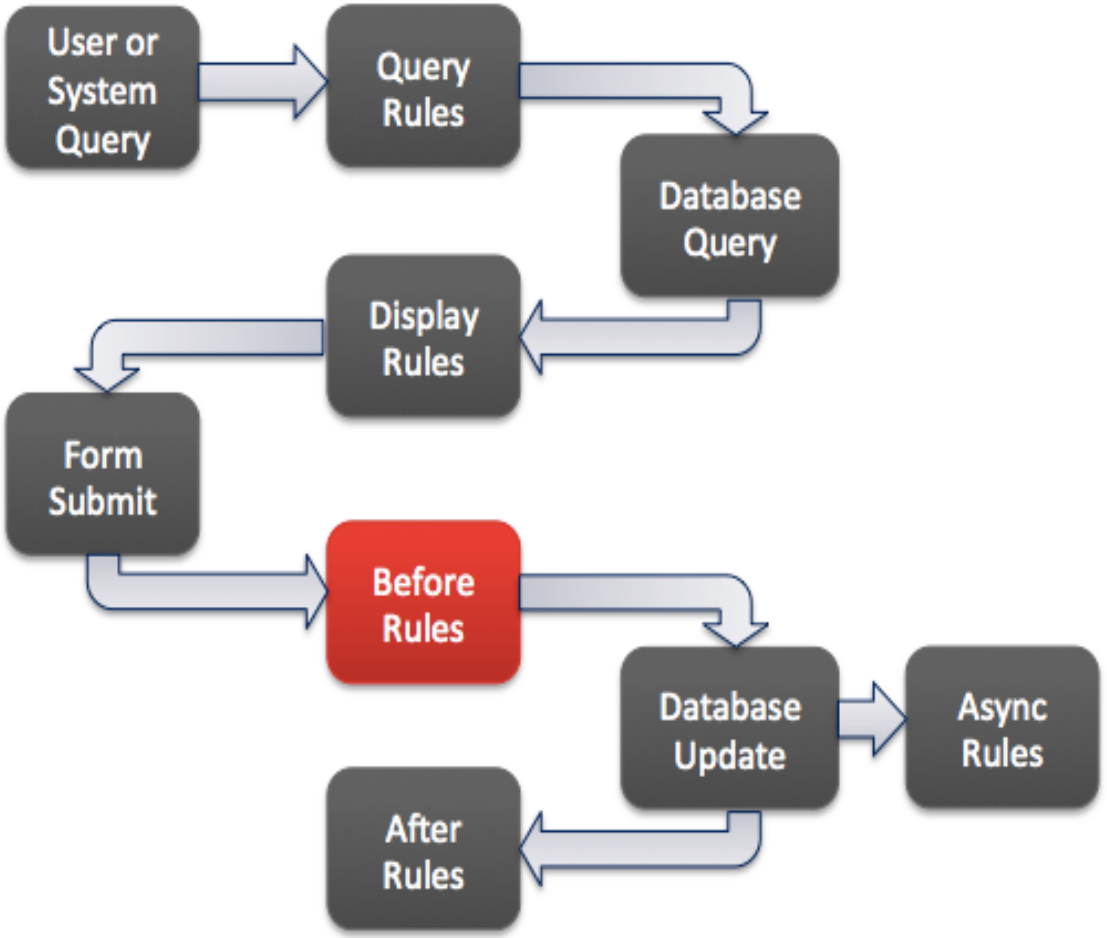
# Business Rules

- Display
  - Run just before the form is loaded
  - Typical use: passing g_scratchpad information to client scripts



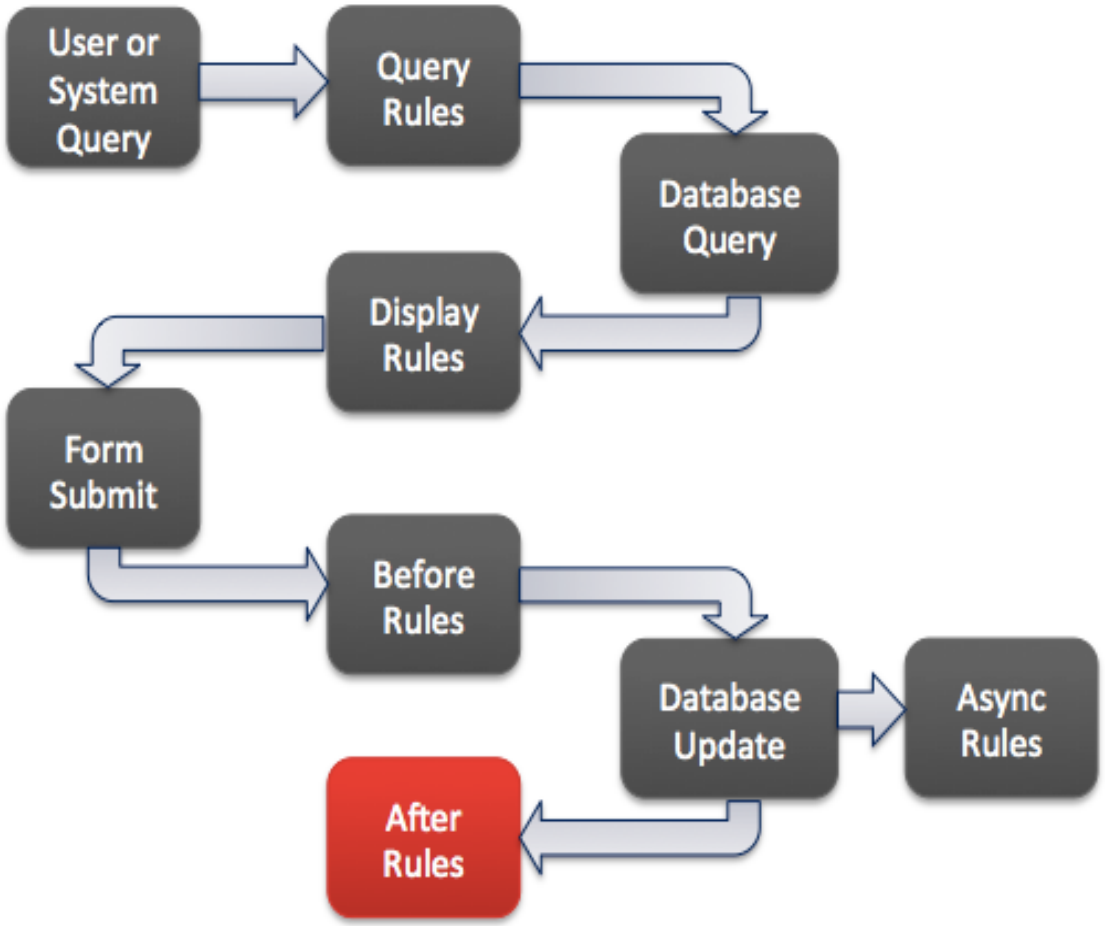
# Business Rules

- Before
  - Implicit update – **don't use current.update()**
  - Generally used for manipulating the current record

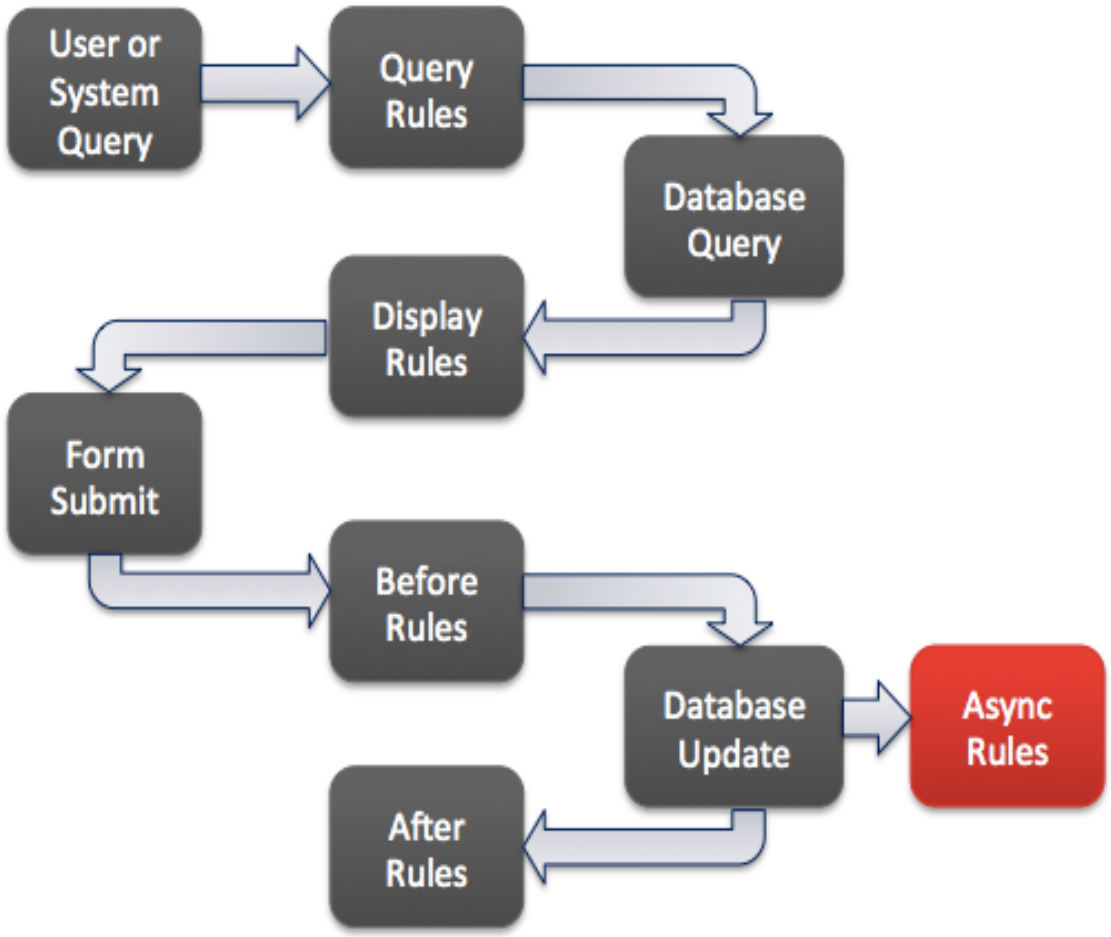
# Business Rules

- After
  - Generally used for managing related tables/fields

# Business Rules

- Async
  - Consider as a replacement for after rules
    - Key indicator: Is the updated info required on the screen immediately?
  - Examples
    - Processing Email
    - Calculating SLAs
    - Generating report fields



# Avoid Global Business Rules

- GBRs are loaded for every page
- Script includes only get loaded when required
- Converting is easy

It is not possible to create global business rules in Fuji!

Script Include

Name: u_recent_active_poll

Application: Global

Client callable: ☐

Accessible from: This application scope o

Active: ☑

Description: Reads the poll [u_poll] table and returns the most recent active poll

Script

```
function u_recent_active_poll(pid) {

    var result = '';
    var pr     = new GlideRecord('u_poll');

    pr.addQuery('u_active', true);
    pr.orderByDesc('sys_updated_on');
    pr.query();

    if (pr.next())
        result = pr.getValue('sys_id');

    return result;
}
```

# Encapsulate Code in Functions

## Particularly True for Business Rules

- By default, all variables and functions are global!
- Several business rules running with the statement…

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

- …can have unpredictable conflicts
- My favorite… go through a list of five items and get hundreds of output statements!

```
for (i = 0;
    processField(list[i]);
    gs.print(i + '=' + list[i]);
}
```

"i" is global unless you say "var i"

# Encapsulate Code in Functions (pre-Fuji)

- The variable ‘gr’ is only known to the function doStuff()
- Now, the “doStuff()” function is in the global name space…
- Use good function naming standards and conditions to mitigate this
  - Ex: acme_chg_closure();

```
doStuff(); // Do 1 thing - call the function

/*
 * doStuff - a simple example of something
 *
 * @param - None
 * @return - None
 *
 */
function doStuff() {

    var gr = new GlideRecord('incident');

    gr.addQuery('active', true);
    gr.query();

    while (gr.next()) {
        // Do something important here
        // with each active incident record
    }
}
```

# Fuji – Business Rules Templates

- Automatically encapsulates code in functions

Condition

!current.start_date.nil() && !current.end_date.nil()

Script

```
function onBefore(current, previous) {

    var start = current.start_date.getGlideObject().getNumericValue();
    var end = current.end_date.getGlideObject().getNumericValue();
    if (start > end) {
        var msg = gs.getMessage('loaner_error_end_before_start');
        gs.addInfoMessage(msg);
        current.start_date.setError(msg);
        current.setAbortAction(true);
    }
}
```

# Anonymous Functions

- If you're not good at naming functions and want to avoid the issue of the global name space, you can also use "anonymous functions" with format similar to the following example:

```
(function() {

   var gr = new GlideRecord('u_my_table');

   gr.state = 3; // Closed complete
   gr.active = false;
   gr.update();
})();
```

# Other Business Rules Best Practices

- In general, make business rules small and specific
  - Easier to debug and maintain
- Use conditions to prevent unnecessary execution and easier to debug

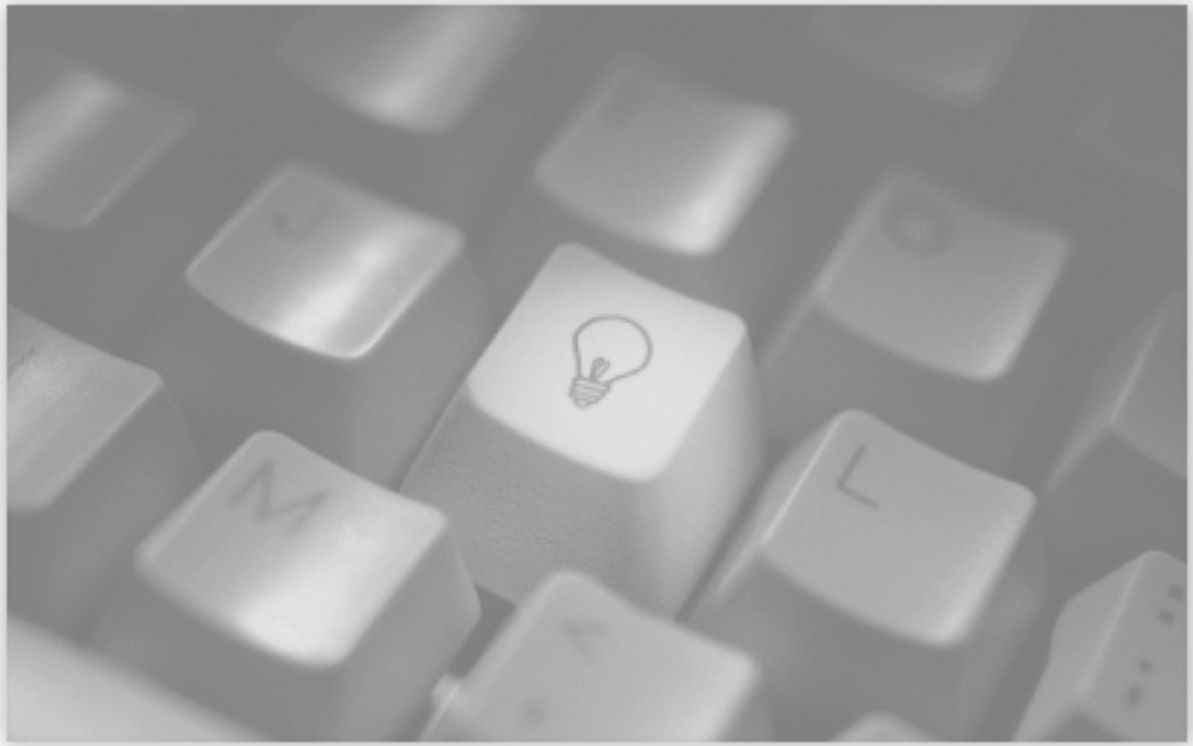
Condition: gs.hasRole('itil') && gs.getProperty('glide.ui.risk_calculate_rule') == "business_rule"

Script

```
// This business rule checks Risk Conditions and returns risk and impact specified by ri.risk and ri.impact
// Labels for risk and impact are returned as ri.riskLabel and ri.impactLabel
// Other values that are returned include Name of the rule (ri.name) and order (ri.order)
// For validation with a form button, deactivate this business rule and activate the Calculate Risk UI Action

var scr = new SetChangeRisk();
scr.setRisk(current);
```

```
15:22:30.346: ==> 'Attach Delivery Plan' on change_request:CHG0000015
15:22:30.347: <== 'Attach Delivery Plan' on change_request:CHG0000015
15:22:30.347: === Skipping 'Calculate Risk' on change_request:CHG0000015; condition not satisfied: Condition: gs.hasRole('itil') && gs.getProperty('glide.ui.risk_calculate_rule') == "business_rule"
15:22:30.347: === Skipping 'mark_closed' on change_request:CHG0000015; condition not satisfied: Filter Condition: state>2^EQ
15:22:30.347: ==> 'Outside Maintenance Schedule' on change_request:CHG0000015
15:22:30.359: <== 'Outside Maintenance Schedule' on change_request:CHG0000015
```

# Lab 1:



*1.1 Review Business Rules*
*1.2 Global Business Rules*

# Client Script Best Practices

# Reduce Client Side Logic

- Client side = JavaScript code that runs on the browser
- When well designed, it can reduce the time it takes to complete a form
- Try using server side whenever possible
  - Not everything needs to be a client script
- Client scripts and UI policies run on forms only
  - Disable list editing (with an ACL) if you must use client side scripting
- Minimize server lookups

# Getting Server Data

## Options to Get Server Side Data to the Client

Most Preferred

- g_scratchpad
- GlideAjax
- g_form.getReference()
- GlideRecord Query

Least Preferred

# g_scratchpad

- Great for passing information to your form (on load)
- No round trip required
- Does not update fields dynamically (onChange)

# g_scratchpad – Business Rule

- Server Side (display business rule)

Condition: current.isNewRecord()

Script:

```
if (!JSUtil.nil(current.question.sys_id)) {
    g_scratchpad.order = getNextHundred(current, 'order', 'question', 'question='
 + current.question);
}
```

# g_scratchpad – Client Script

- Client side

Script

```
1 function onLoad() {
2     if (g_form.getValue('order') == '') {
3         g_form.setValue('order', g_scratchpad.order);
4     }
5
6 }
```

# GlideAjax

- Pros
  - Callback (async/background) functionality
  - More flexibility in what's returned
- Cons
  - A little more complex to create
    - Client and Server side scripts to write/maintain
  - Requires a round trip from the client to server

Key differentiator between g_scratchpad and GlideAjax:

- GlideAjax can be used for dynamic updates
- g_scratchpad only passes information when the form is loaded

# GlideAjax Example

- Server

```
var HelloWorld = Class.create();
HelloWorld.prototype = Object.extendsObject(AbstractAjaxProcessor, {
    alertGreeting: function() {
        return "Hello " + this.getParameter('sysparm_user_name') + "!";
    }
});
```

- Client

```
var ga = new GlideAjax('HelloWorld');
ga.addParam('sysparm_name','alertGreeting');
ga.addParam('sysparm_user_name','Bob');
ga.getXML(HelloWorldParse);

function HelloWorldParse(response) {
    var answer = response.responseXML.documentElement.getAttribute("answer");
    alert(answer);
}
```

# setValue() with Reference Fields

- Usage: g_form.setValue(fieldName, value, displayValue);
- Incorrect:

```
var id = '5137153cc611227c000bbd1bd8cd2005';
g_form.setValue('assigned_to', id);
```

CAUSES A SYNCHRONOUS AJAX CALL TO GET THE DISPLAY VALUE

- Correct

```
var id = '5137153cc611227c000bbd1bd8cd2005';
var name = 'Fred Luddy';
g_form.setValue('assigned_to', id, name);
```

# GlideAjax - Example

## Passing Back a Reference and Display Name

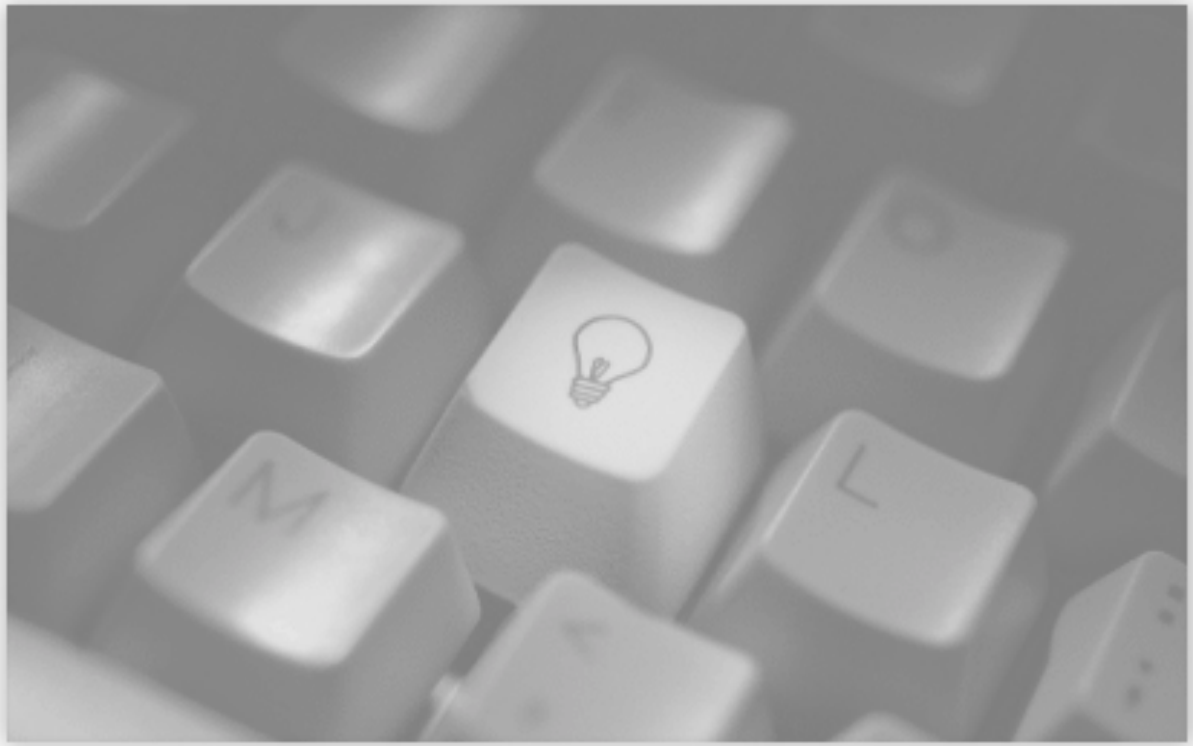
```
var GetUserStuff = Class.create();
GetUserStuff.prototype = Object.extendsObject(AbstractAjaxProcessor, {
   getLocation : function() {

      var usr    = new GlideRecord('sys_user');
      var usr_id = this.getParameter('sysparm_caller_id');

      if (usr.get(usr_id)) {
         return usr.location + ':' + usr.location.getDisplayValue();
      } else {
         return '';
      }
   },
});
```

# Avoid DOM manipulation

- Avoid getElementById() or gel() calls
- Use g_form methods such as:
  - g_form.getControl()
  - g_form.setValue()

g_form insulates you from browser compatibility issues

# Lab 2:



*2.1 Convert getReference to GlideAjax*
*2.2 Use g_scratchpad*

# Coding Best Practices

# Comment Your Work

- Get into the habit now
- You **will** forget why you did something

```
// a single line comment
var gr = new GlideRecord('incident'); // here, too!
```

```
/*
 * multi line comments
 */
```

```
//////////////////////////////
// BOXES ARE EASY TO SPOT //
//////////////////////////////
```

# Avoid Hard Coded Values

- Avoid using hard coded values in scripts

```
var taskID = '0b74a[illegible]';
```

```
var groupName = '[illegible]';
```

- Alternative

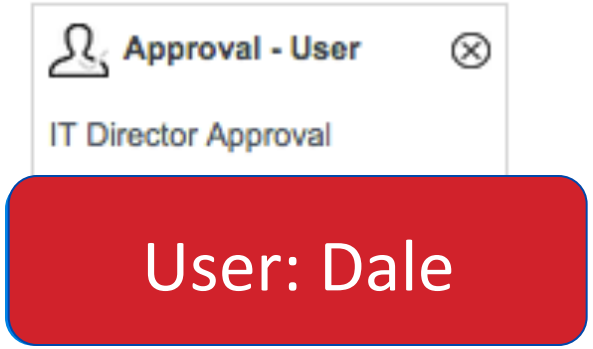
```
var taskID = gs.getProperty('acme.task.default');
var groupName = gs.getProperty('acme.group.name');
```

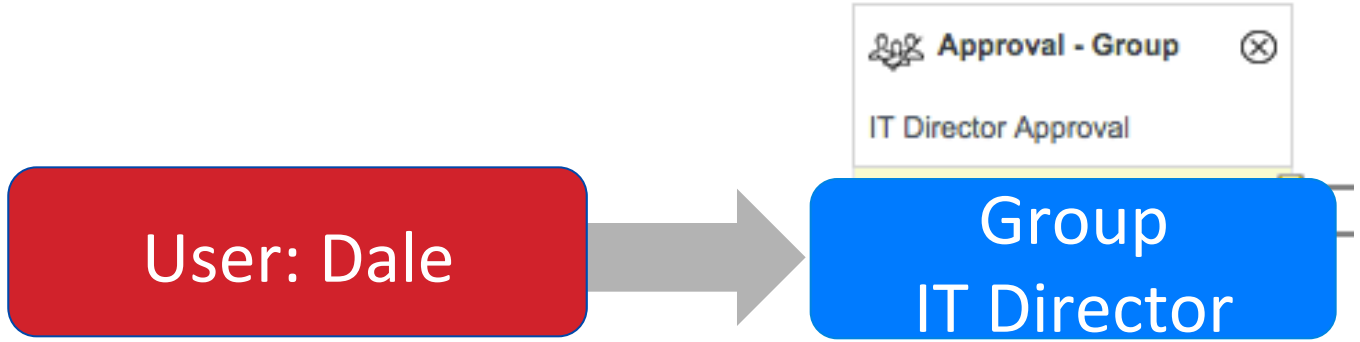
As you code, ask yourself "*What if this thing in quotes changes?*"

# Avoid Hard Coded Values

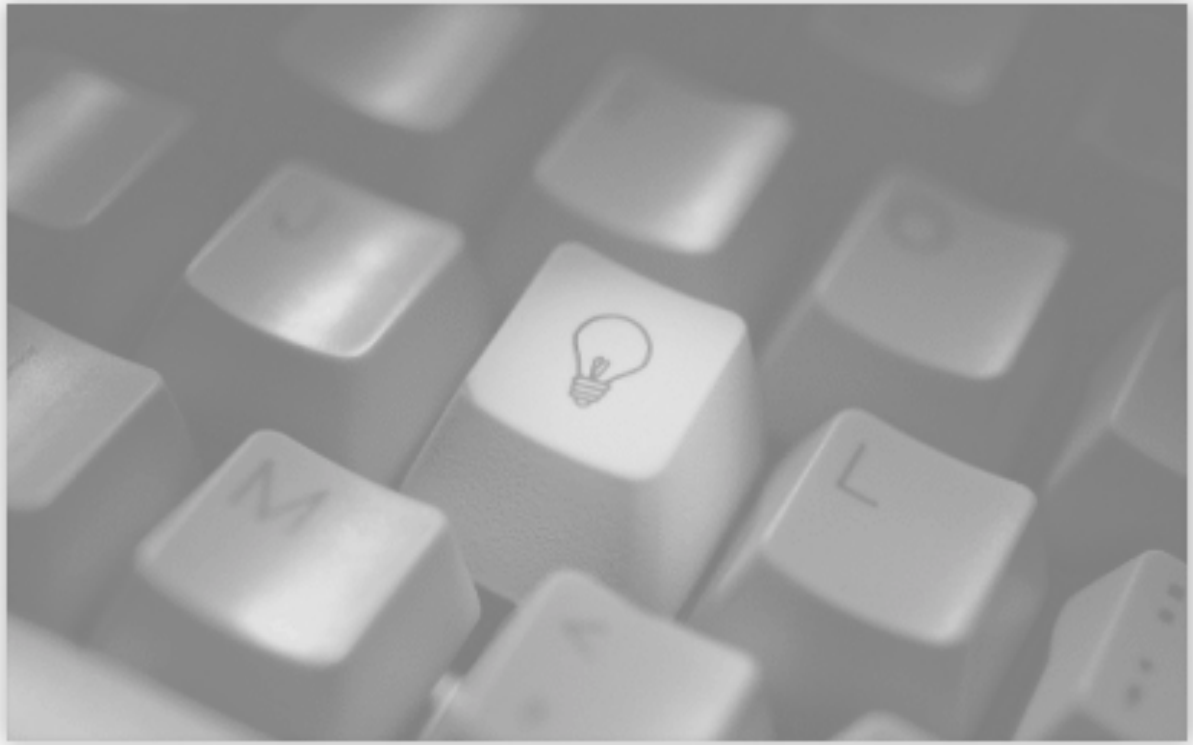
- Initial Approach
  - Create a user approval activity for Sara
  - Challenge: Dale is the new IT Director
  - Requires you to update the workflow!

- Better Approach
  - Use Group Approval Activity
  - Create a group "IT Director"
  - Make Sara a member
  - When Dale becomes the new IT Director, just update the group membership, not the workflow



# Lab 3:



*3.1 Hard Coded sys_ids*

# Counting Records

- Avoid using GlideRecord.getRowCount() when counting an undetermined number of records

| | Pros | Cons |
|---|---|---|
| **GlideRecord** | • Easy to write | • Slower performance.<br>• Not scalable (more records = more time.) |
| **GlideAggregate** | • Fast<br>• Scalable (leverages a database function to count records) | • 3 more lines of code |

Exception: If you are going to be reading/writing the records anyway…

# Row Count Example

## Worst Case

- GlideRecord – don't use: while (gr.next())

```
gs.print('>>>DEBUG: ' + countEm() + ' records read');

function countEm() {

    var gr = new GlideRecord('sys_audit');
    var count = 0;

    gr.query();
    while (gr.next()) {
        count++;
    }

    return count;
}
```

- 490K records: 1m 40s

# Row Count Example

- GlideRecord – getRowCount() is not as bad, but…

```
gs.print('>>>DEBUG: Starting row count...');
gs.print('>>>DEBUG: ' + countEm() + ' records read');

function countEm() {

    var gr = new GlideRecord('sys_audit');

    gr.query();
    return gr.getRowCount();
}
```

- 490K records: >2.1 seconds

# Row Count Example

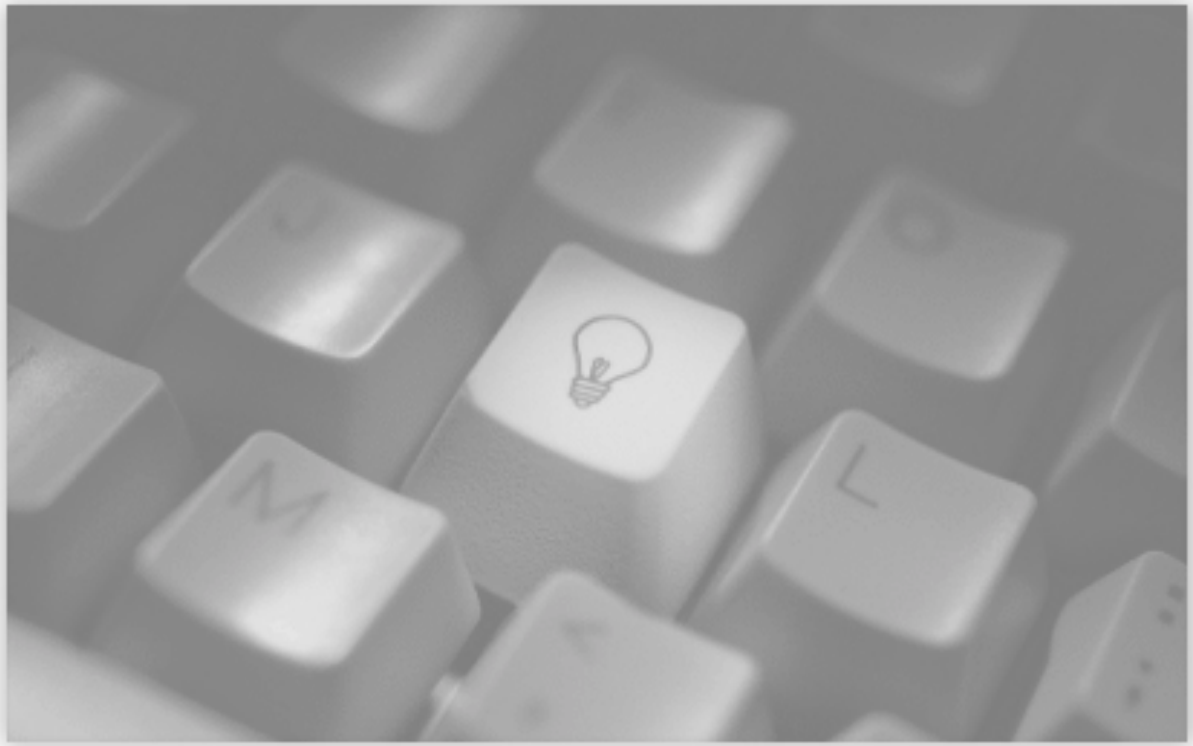
## BEST WAY!

- GlideAggregate

```
gs.print('>>>DEBUG: Starting row count...');
gs.print('>>>DEBUG: ' + countEm() + ' records read');

function countEm() {

    var count  = new GlideAggregate('sys_audit');
    var retVal = 0;

    count.addAggregate('COUNT');
    count.query();

    if (count.next()) {
        retVal = count.getAggregate('COUNT');
    }

    return retVal;
}
```

- 490K records: 1-3 ms

# Lab 4:



*4.1 Row Counting*

# More Information

## ServiceNow Wiki

servicenow

**CATEGORIES**

- **Get Started**
- Use
- Administer
- Script
- Build
- Deliver
- Integrate
- Release Notes
- Technical Support
- Books
- Video Tutorials

### Technical Best Practices

Learn to increase the effectiveness of your ServiceNow experience.

System Performance Best Practices
Debugging Tools Best Practices
Customizing the UI Appearance
[more]

# Top Takeaways

1 Use before/after/async business rules effectively.

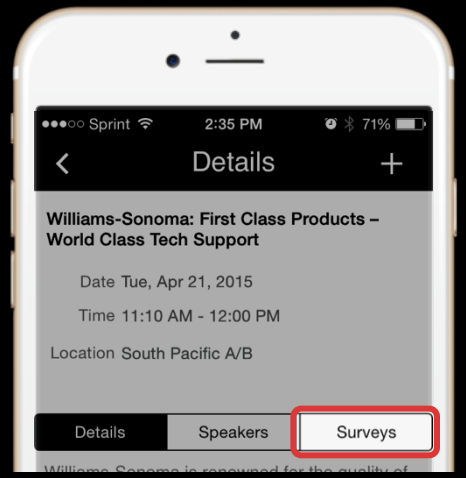
2 Use asynchronous GlideAjax in client scripts to retrieve data from the server.

3 Use System Properties and avoid hard coded values.

## 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


**Chuck Tomasi**

*Sr. Services Enablement Program Manager*

ServiceNow

chuck.tomasi@servicenow.com

**Jonatan Jardi**

*Sr. Technical Consultant*

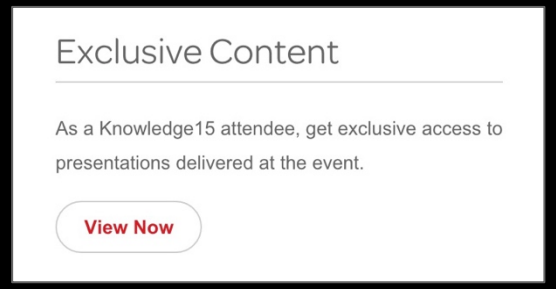
ServiceNow

jonatan.jardi@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

@ctomasi | @enojardi | #Know15