

SERVICENOW WEEK-4

Scripting in Servicenow

In ServiceNow, scripting refers to writing and using code to customize and automate processes, tasks, and business logic within the platform. It enhances the functionality and behavior of ServiceNow applications beyond the out-of-the-box capabilities.

ServiceNow primarily uses two types of scripts:

- **Client-side scripting:** Runs on the user's browser. It is used to control how forms, fields, and user interfaces behave. Common client-side scripts include:
- **Server-side scripting:** Runs on the server and is used to control data operations and business logic. Common server-side scripts include: These scripts use JavaScript as the programming language and allow developers to manipulate data, automate tasks, validate data, create integrations, and more.

Client-side vs. Server-side

Scripts in ServiceNow execute either on the client (user's browser) or in the ServiceNow back end. It is important to know where a script will execute as there are different APIs for the client and server-side scripts.

Client-side

- Manage forms and form fields:
- UI Policies
- Client Scripts

Server-side

- Manage database and back-end:
- Business Rules
- Scheduled Jobs
- Script Actions
- Script include

The script types listed here are examples; this is not an exhaustive list of script types in ServiceNow.

Client Scripts

- Onload()
- Onsubmit()
- Onload()
- Onchange()

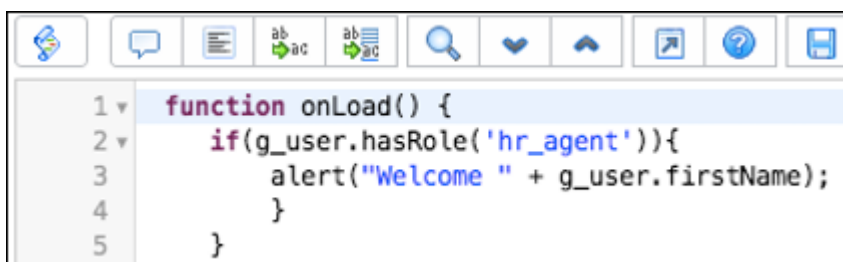
GlideForm



```
1 function onSubmit() {
2     var tempDesc = g_form.getValue('short_description');
3     tempDesc += ": " + g_form.getValue('number');
4     g_form.setValue('short_description', tempDesc);
5     alert("All submissions are reviewed by the Claims team.");
6
7 }
```

- g_form.showFieldMsg() - prints a message on a blue background below the field passed in the method call.
- g_form.addInfoMessage() - prints a message on a blue background to the top of the current form.
- g_form.addErrorMessage() - prints a message on a red background to the top of the current form.
- alert is a JavaScript method that opens a dialog with an OK button.
- confirm is a JavaScript method that opens a dialog with OK and Cancel buttons.
- g_form.setMandatory()
- g_form.SetReadOnly()
- g_form.setdisplay()
- g_form.set_value(' feild_name" , 'value') - sets the value of the field
- g_form.get_value('feild_name) - returns the value in the field

GlideUser



```
1 function onLoad() {
2     if(g_user.hasRole('hr_agent')){
3         alert("Welcome " + g_user.firstName);
4     }
5 }
```

- g_user.firstname
- g_user.lastname
- g_user.userId
- g_user.username
- g_user.getfullname
- g_user.hasrole('role_name')
- g_user.hasroleexactly('role_name')
- g_user.hasroleFromList('role_name')
- g_user.hasroles()

Scripting assistance-Press ctrl-spacebar

UI policy

Client-side logic governing form and field behavior. **UI Policies dynamically change the behavior of information** on a form and control custom process flows for tasks. you can use UI policies to make the number field on a form.

- read-only,
- field mandatory,
- hide other fields.

The image shows two parts of the UI Policy configuration interface. The top part, titled "When to Apply", shows conditions for when the policy should be active. It has two buttons: "Add Filter Condition" and "Add 'OR' Clause". Below these, it states "All of these conditions must be met". There are two conditions listed: "Police report filed" is "true" and "Active" is "true". The bottom part, titled "UI Policy Action", shows the action to be taken. It is named "Hide Police Report Number Field". The application is "Claims", and the table is "Claims [x_snc_claims_claims]". The field name is "Police report number". The action is "Hide". The application settings are: "Mandatory" is "True", "Visible" is "True", and "Read only" is "Leave alone". A green arrow points from the "Police report number" field in the action settings to the "Police report number" field in the form below.

When to Apply

Conditions

All of these conditions must be met

Police report filed is true AND OR X

Active is true AND OR X

UI Policy Action
New record

UI policy Hide Police Report Number Field

Table Claims [x_snc_claims_claims]

* Field name Police report number

Application Claims

Mandatory True

Visible True

Read only Leave alone

Number TASK0020520

Assigned to Beth Anglin

Active ☒

Accident 2017-01-10 15:50:51

Police report ☒

* Police report number

UI Policy Scripts

The image shows the "Script" tab of the UI Policy configuration interface. It has two tabs: "When to Apply" and "Script". The "Script" tab is active. It shows the "Run scripts" checkbox is checked. The "Run scripts in UI type" is set to "All". There are two sections for scripts: "Execute if true" and "Execute if false". Each section has a toolbar with icons for different actions. The "Execute if true" section contains a script:

```
1 function onCondition() {
2   g_form.setSectionDisplay('insured',true);
3 }
4
```

 The "Execute if false" section contains a script:

```
1 function onCondition() {
2   g_form.setSectionDisplay('insured',false);
3 }
```

When to Apply Script

Run scripts ☒

Run scripts in UI type All

Execute if true

```
1 function onCondition() {
2   g_form.setSectionDisplay('insured',true);
3 }
4
```

Execute if false

```
1 function onCondition() {
2   g_form.setSectionDisplay('insured',false);
3 }
```

Server-side scripts

- GlideForm and GlideUser, GlideAjax classes are part of the client-side API.
- GlideRecord and GlideSystem are part of server side API.

Data Policy

- A Data Policy is a rule that enforces data consistency by setting fields as mandatory and/or read-only.
- Data Policy controls are similar to UI Policies, but UI Policies are only enforced on data entered into a form (passing through the UI).
- Data Policies are applied to all data entered into the platform: form (UI), Import Sets, or Web Services.

Business Rule

A Business Rule in ServiceNow is a **server-side script** that runs when a record is inserted, updated, deleted, displayed, or queried from a database table.

The screenshot shows the configuration page for a Business Rule named 'problem_reopen'. The page has a header with navigation icons and 'Update' and 'Delete' buttons. A blue informational box at the top explains that a business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Below this, the 'Name' field is set to 'problem_reopen' and the 'Table' dropdown is set to 'Problem [problem]'. The 'Application' is set to 'Global'. There are checkboxes for 'Active' and 'Advanced', both of which are checked. Below these fields are tabs for 'When to run', 'Actions', and 'Advanced'. The 'When to run' tab is selected, showing a blue box with instructions to specify when the rule should run. The 'When' dropdown is set to 'before' and the 'Order' is set to '100'. There are checkboxes for 'Insert', 'Update', 'Delete', and 'Query'. The 'Update' checkbox is checked and highlighted with a red box. Below these are 'Filter Conditions' with buttons to 'Add Filter Condition' and 'Add "OR" Clause'. The filter conditions are set to 'All of these conditions must be met'. The first condition is 'Problem state' is one of 'Closed/Resolved', 'Fix in Progress', 'Resolved', and 'Closed'. The second condition is 'Active' is 'false'. There are 'AND' buttons between the conditions. At the bottom, there is a 'Role conditions' section with a pencil icon and the text 'problem_admin, problem_manager'.

Business Rule configuration page for **problem_reopen**.

Name: problem_reopen
Table: Problem [problem]
Application: Global
Active: ☒
Advanced: ☒

When to run: before
Order: 100

Specify whether the business rule should run on Insert or Update. Use Filter Conditions to specify under which conditions the business rule should run.

When: before
Order: 100

Insert: ☐
Update: ☒
Delete: ☐
Query: ☐

Filter Conditions: Add Filter Condition Add "OR" Clause

All of these conditions must be met

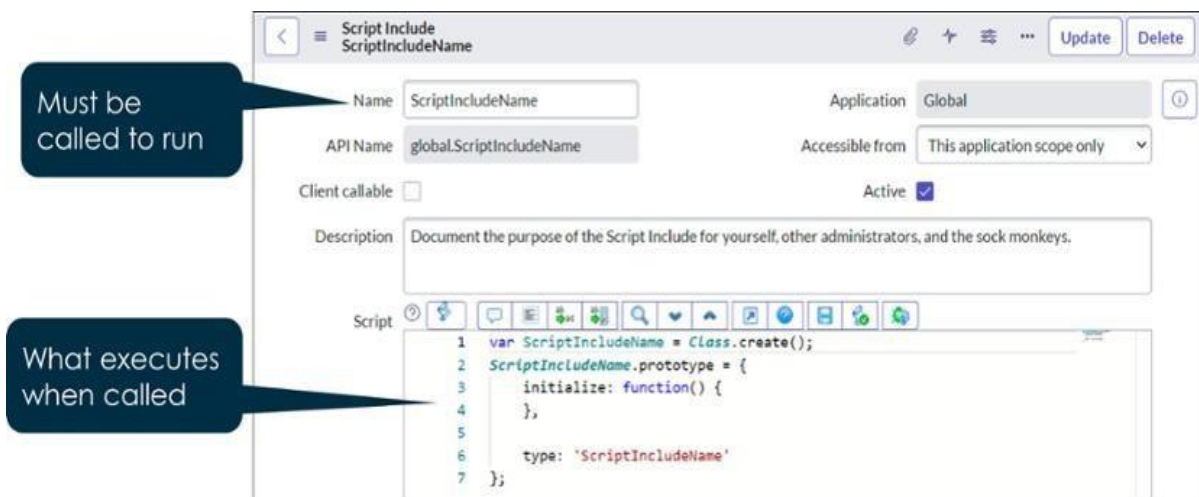
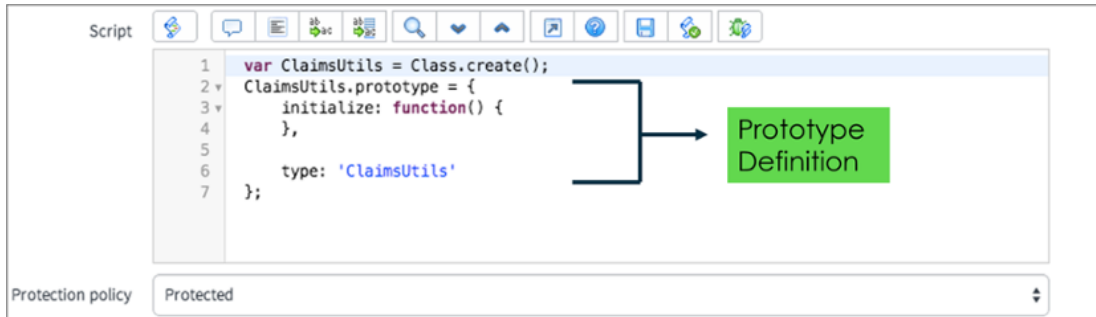
Problem state is one of Closed/Resolved, Fix in Progress, Resolved, Closed

Active is false

Role conditions: problem_admin, problem_manager

Script Include

A Script Include in ServiceNow is a reusable server-side JavaScript class or function that can be called from other server-side scripts like Business Rules, Workflow scripts, or even from the client-side via GlideAjax.



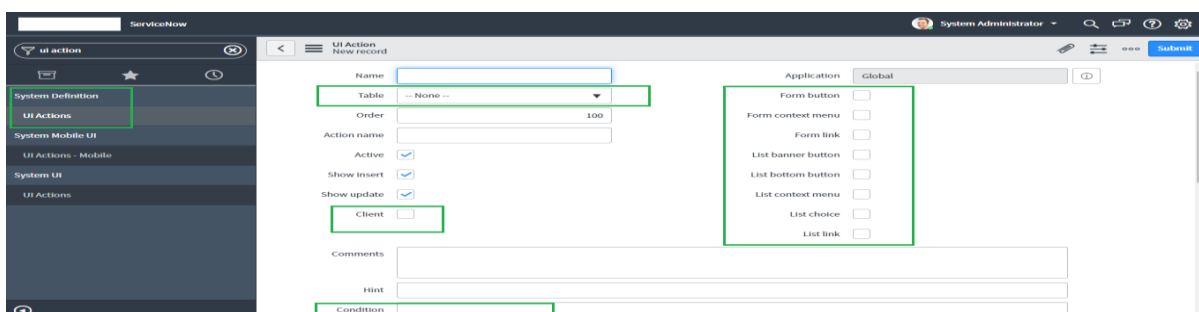
Types of Script Include

- Classless Script Include (Function-Based Script Include)
- Class-Based Script Include (Object-Oriented Script Include):
- Extend an existing class

UI Actions

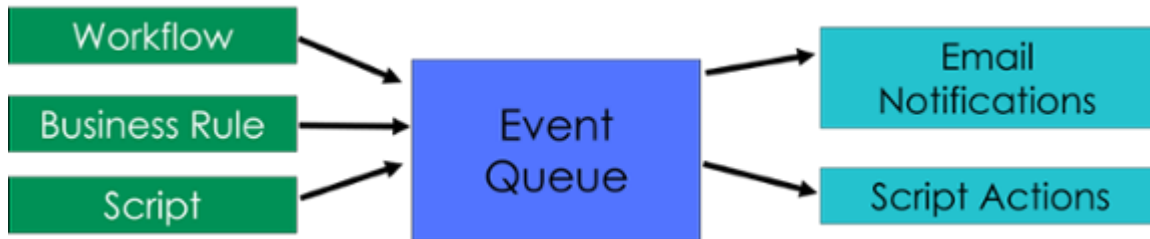
User Interface (UI) Actions add buttons, links, and context menu items on forms and lists, making the UI more interactive, customizable, and specific to user activities.

The New button on the existing list of All Incidents is an example of a UI Action.



Script Actions

Script Actions in ServiceNow are server-side scripts that are triggered by system events. They are often used in conjunction with event-driven architecture, where they respond to specific events in the system by executing a script.



Key Features of Script Actions:

- Event-driven
- Server-side
- Linked to Events

Methods in Script Actions:

- `gs.eventQueue()`
- `gs.addErrorMessage()`
- `Current`
- `event`

-----END-----

