# ServiceNow Scripting Fundamentals and Functions

## Week – 4 Hand Book

## My Learnings:

## What is Platform Scripting?

It is use of JavaScript to customize and extend the functionality of the platform. It allows developers to automate tasks, enforce business logic, and create complex workflows by writing custom scripts.

- **Business Rules** – Automating processes when records are inserted, updated, or deleted in a database.
- **Client Scripts** – Running scripts in the browser to manage user interactions with forms and lists.
- **Script Includes** – Reusable server-side code, often used for complex logic that can be called from multiple places.
- **UI Actions** – Adding buttons, links, or context menus that run scripts on the UI.
- **UI Policies** – Controlling form behavior dynamically, such as making fields mandatory or visible.
- **Workflows** – Automating processes, often using scripts to control the flow of tasks.
- **Scheduled Jobs** – Running server-side scripts on a schedule to perform background tasks.

- **Events and Notifications** – Using scripts to trigger

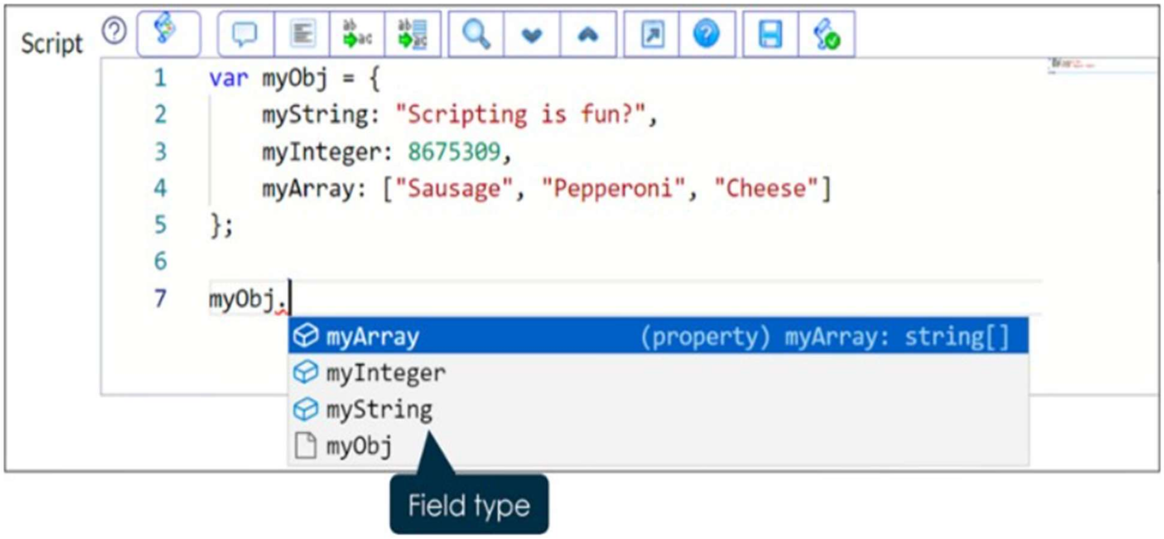
## Client Scripts

### Types of Client Scripts

- **OnLoad() -** Runs when a form loads (e.g., setting default values).
- **OnChange() -** Runs when a field value changes (e.g., updating related fields).
- **OnSubmit() -** Runs when a form is submitted (e.g., validating data).
- **OnCellEdit() -** Runs when a list cell is edited (e.g., adjusting other fields).

### When to Script

- To add functionality
- To extend the existing functionality

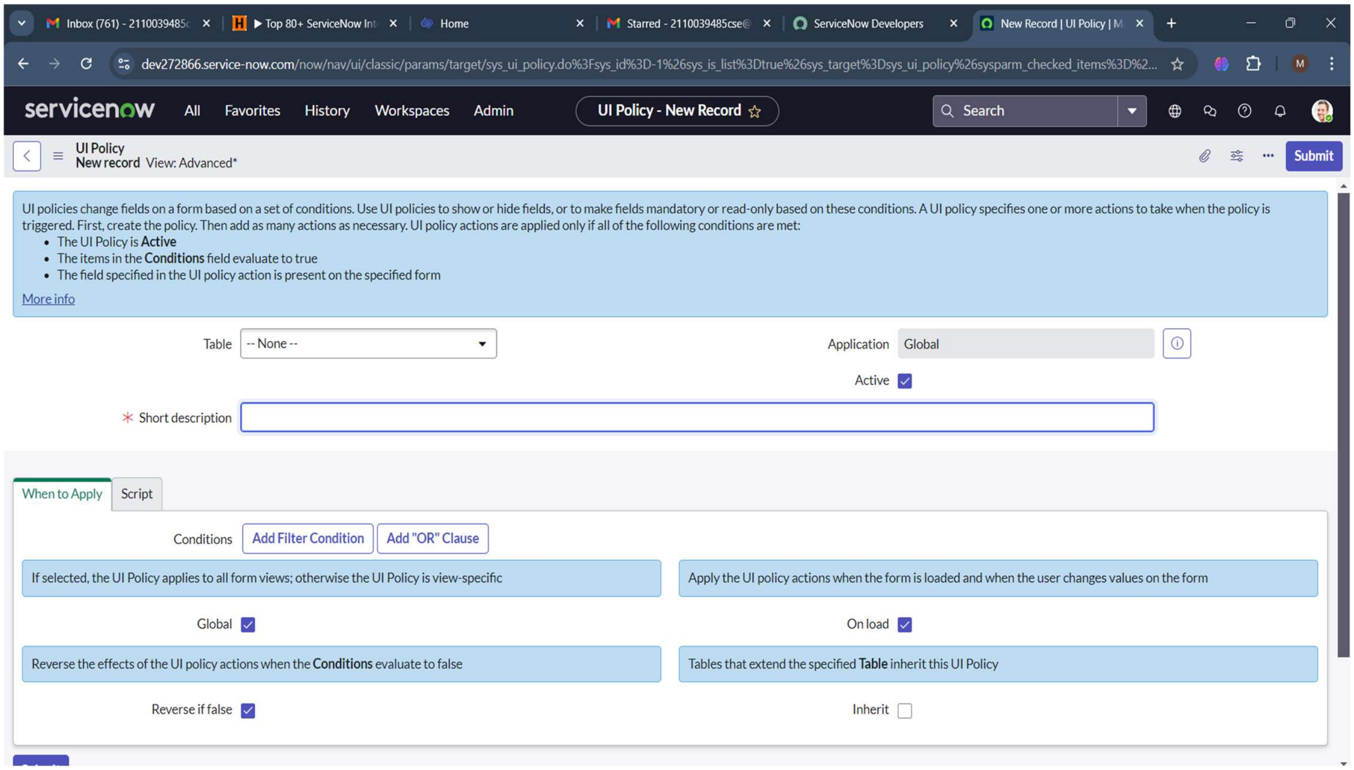


```
var myObj = {
    myString: "Scripting is fun?",
    myInteger: 8675309,
    myArray: ["Sausage", "Pepperoni", "Cheese"]
};

myObj.
```

Field type

# UI Policies

- It is the behavior and visibility of fields on a form
  - Mandatory
  - Visible
  - Read-only
- Conditions must be true for it to execute
- Executes after client Scripts.
- UI Policies take action based on scripts.
- These are majorly used for :
  - Customized Dashboards
  - Form Customization



dev272866.service-now.com/now/nav/ui/classic/params/target/sys_ui_policy.do%3Fsys_id%3D-1%26sys_is_list%3Dtrue%26sys_target%3Dsys_ui_policy%26sysparm_checked_items%3D%2...

UI Policy
**New record** View: Advanced*

UI policies change fields on a form based on a set of conditions. Use UI policies to show or hide fields, or to make fields mandatory or read-only based on these conditions. A UI policy specifies one or more actions to take when the policy is triggered. First, create the policy. Then add as many actions as necessary. UI policy actions are applied only if all of the following conditions are met:

- The UI Policy is **Active**
- The items in the **Conditions** field evaluate to true
- The field specified in the UI policy action is present on the specified form

More info

Table: -- None --

Application: Global

Active ☑

* Short description:

When to Apply | Script

Conditions: Add Filter Condition | Add "OR" Clause

If selected, the UI Policy applies to all form views; otherwise the UI Policy is view-specific

Global ☑

Apply the UI policy actions when the form is loaded and when the user changes values on the form

On load ☑

Reverse the effects of the UI policy actions when the **Conditions** evaluate to false

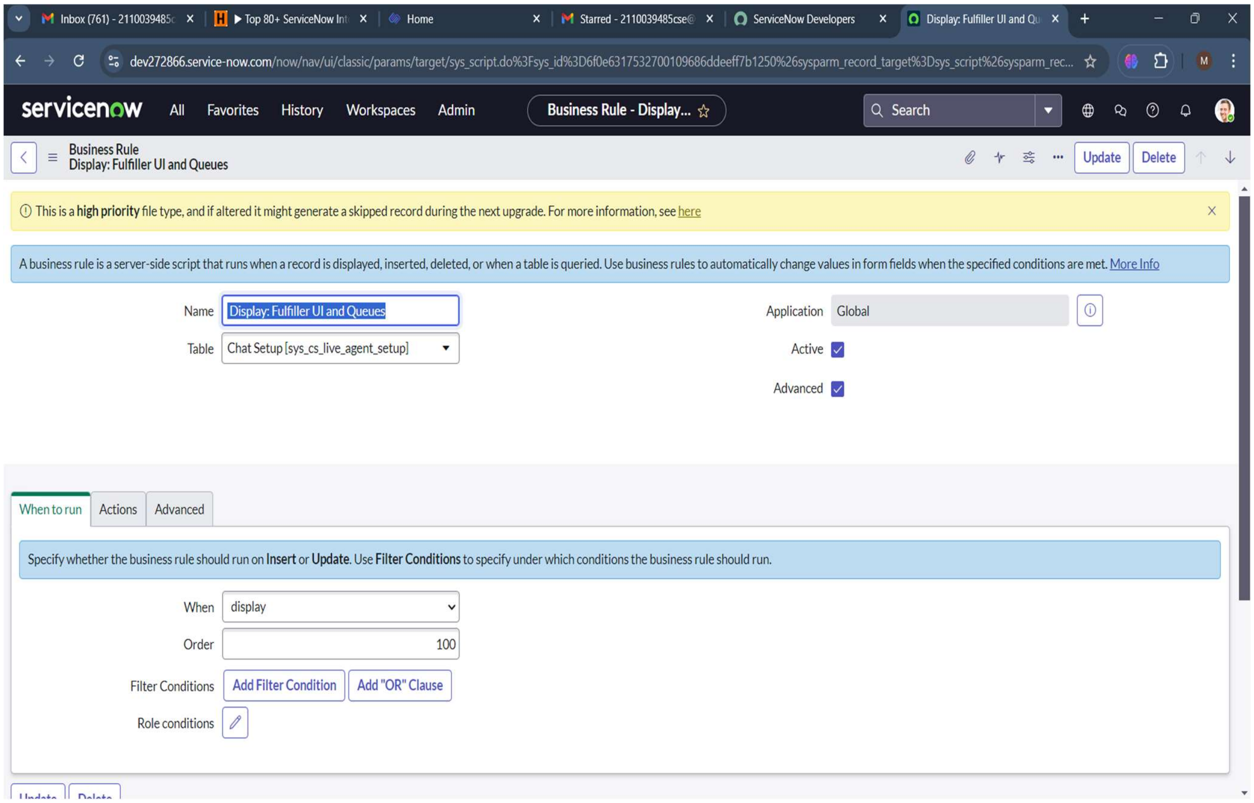
Reverse if false ☑

Tables that extend the specified **Table** inherit this UI Policy

Inherit ☐

# Business Rules

- 2,000+ Business Rules exist baseline.
- Run whenever records are accessed.
- Business rules run whenever they trigger desired filter & role conditions.
- Executes on the server side, doesn't monitor forms & lists.
- Advanced tab is used for Scripting.
- If the active option is selected then only the business rule comes into action.
- The following are the options when the Advanced option is selected:
  - When
  - Order
  - Delete
  - Query



dev272866.service-now.com/now/nav/ui/classic/params/target/sys_script.do%3Fsys_id%3D6f0e6317532700109686ddeeff7b1250%26sysparm_record_target%3Dsys_script%26sysparm_rec...

Business Rule
Display: Fulfiller UI and Queues

This is a **high priority** file type, and if altered it might generate a skipped record during the next upgrade. For more information, see here

A business rule is a server-side script that runs when a record is displayed, inserted, deleted, or when a table is queried. Use business rules to automatically change values in form fields when the specified conditions are met. More Info

Name: Display: Fulfiller UI and Queues

Table: Chat Setup [sys_cs_live_agent_setup]

Application: Global

Active ☑

Advanced ☑

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

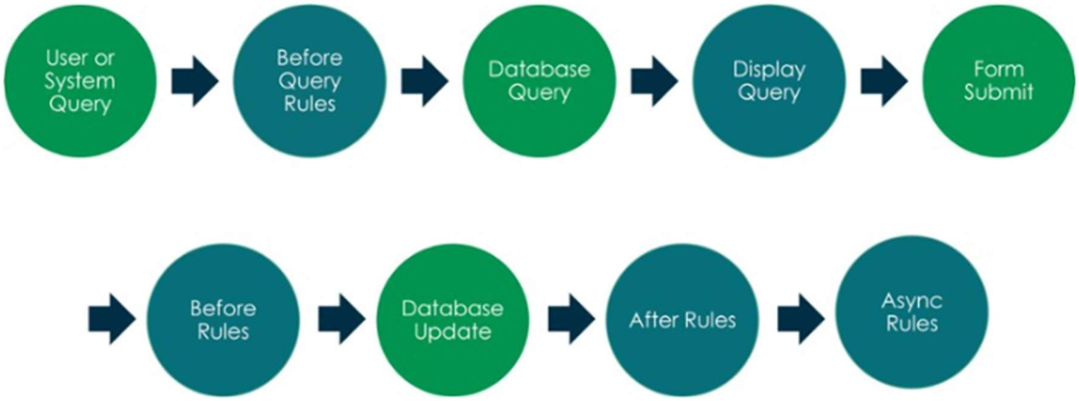
When: display

Order: 100

Filter Conditions

Role conditions

## When Business Rules Execute



# Debugging

### Glide System Logging Methods

- The below are the debugging information to the system log.

  - gs.info() – Used for messaging

  - gs.error() – Used for critical errors

  - gs.warn() – When issue occurs

  - gs.log() – Used in global scope

- Sends debugging information to the top of the form.

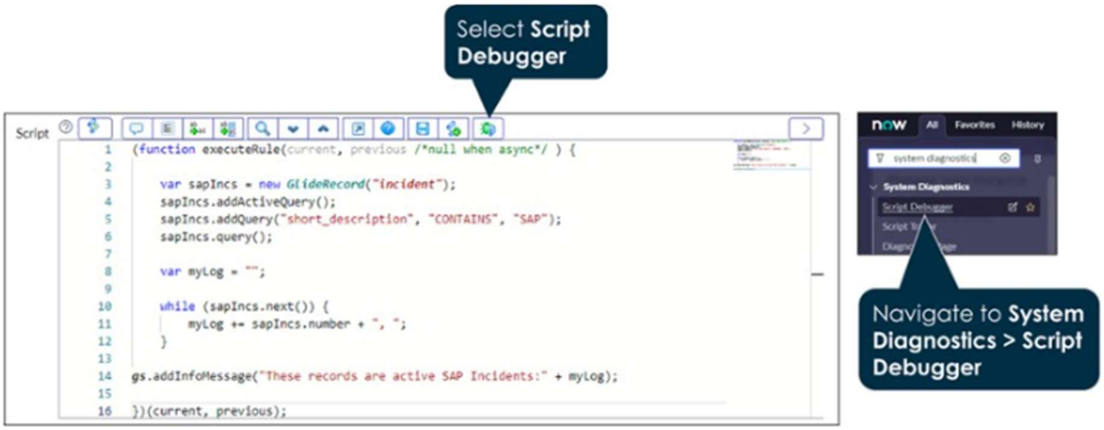
  - gs.addInfoMessage()

  - gs.addErrorMessage()

  - gs.methodName(<message>);

# Script Debugger

- Used for debugging server-side JavaScript

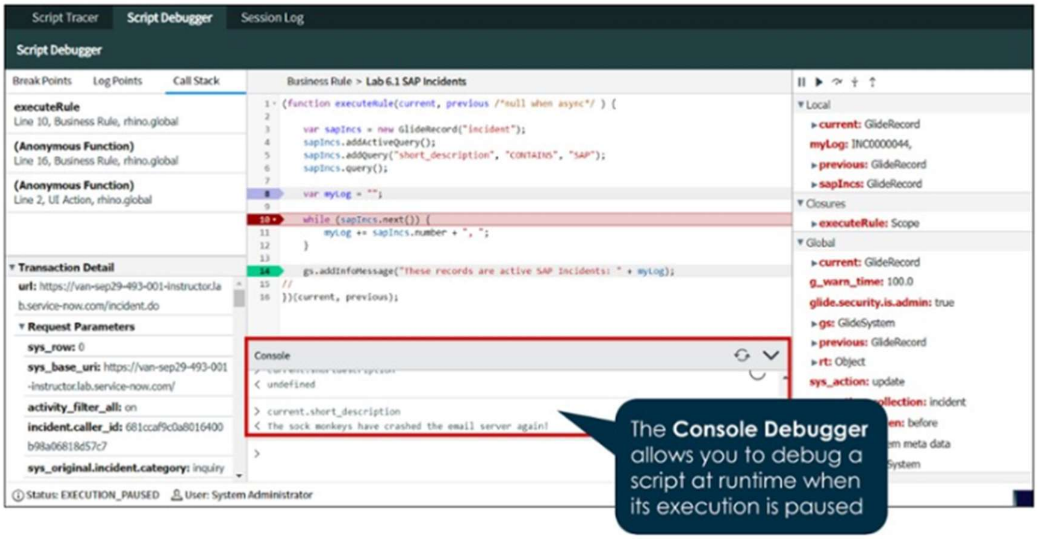


Select **Script Debugger**

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

    var sapIncs = new GlideRecord("incident");
    sapIncs.addActiveQuery();
    sapIncs.addQuery("short_description", "CONTAINS", "SAP");
    sapIncs.query();

    var myLog = "";

    while (sapIncs.next()) {
        myLog += sapIncs.number + ", ";
    }

gs.addInfoMessage("These records are active SAP Incidents:" + myLog);

})(current, previous);
```

Navigate to **System Diagnostics > Script Debugger**

- The ability to step through the script line-by-line.
- **Log Points:** Inserting log statements into the code without changing the code.
- **Break Point:** That pauses the execution of a program at a specific line of code.
- **Transaction Details:** Inspect Network, URL, HTTP Request.
- **Script Tracer:** Used for debugging server-side JavaScript.
- **Call Stack:** It pinpoints exactly which script needs to be replaced.



**executeRule**
Line 10, Business Rule, rhino.global

**(Anonymous Function)**
Line 16, Business Rule, rhino.global

**(Anonymous Function)**
Line 2, UI Action, rhino.global

**Transaction Detail**
**url:** https://van-sep29-493-001-instructor.lab.service-now.com/incident.do
**Request Parameters**
**sys_row:** 0
**sys_base_uri:** https://van-sep29-493-001-instructor.lab.service-now.com/
**activity_filter_all:** on
**incident.caller_id:** 681ccaf9c0a8016400b98a06818d57c7
**sys_original.incident.category:** inquiry

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

    var sapIncs = new GlideRecord("incident");
    sapIncs.addActiveQuery();
    sapIncs.addQuery("short_description", "CONTAINS", "SAP");
    sapIncs.query();

    var myLog = "";

    while (sapIncs.next()) {
        myLog += sapIncs.number + ", ";
    }

    gs.addInfoMessage("These records are active SAP Incidents: " + myLog);
//
})(current, previous);
```

```
< undefined

> current.short_description
< The sock monkeys have crashed the email server again!
>
```

Local
**current:** GlideRecord
**myLog:** INC0000044,
**previous:** GlideRecord
**sapIncs:** GlideRecord
Closures
**executeRule:** Scope
Global
**current:** GlideRecord
**g_warn_time:** 100.0
**glide.security.is.admin:** true
**gs:** GlideSystem
**previous:** GlideRecord
**rt:** Object
**sys_action:** update
**...llection:** incident
**...en:** before
...m meta data
...System

Status: EXECUTION_PAUSED User: System Administrator

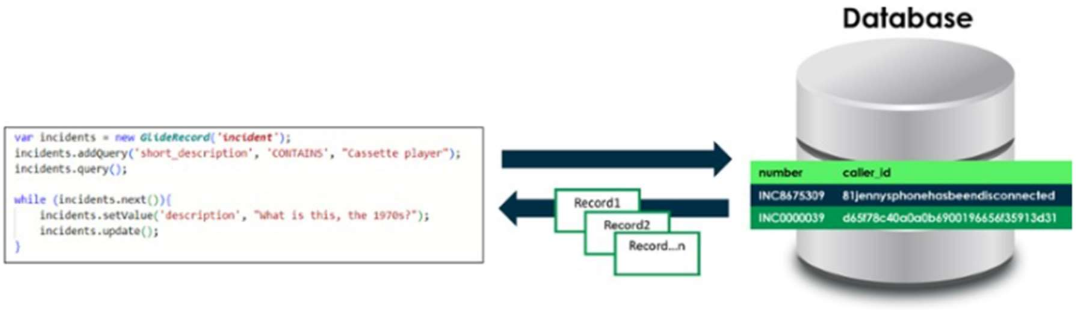
The **Console Debugger** allows you to debug a script at runtime when its execution is paused

# Glide System

- The glide system & scoped glide system APIs provide methods to access system-level information.
- Glide System Methods fall into 3 Categories
  - USER
  - General
  - Date/Time
- Scoped Glide System Methods fall into 3 different categories:
  - USER
  - General
  - Logging Details
- **Applications of Glide System**
  - Database Interaction
  - To facilitate aggregate operations like sum, multiplication, average on records & etc.
  - Date & Time Handling
  - Communication between client scripts & server-side scripts.
- **Applications of Glide Query**
  - Field Casting
  - Parsing Encoded Queries
  - Better Join Support
  - Allow opting out of field/choice checking
  - Scoped table permission checking

# Glide Record

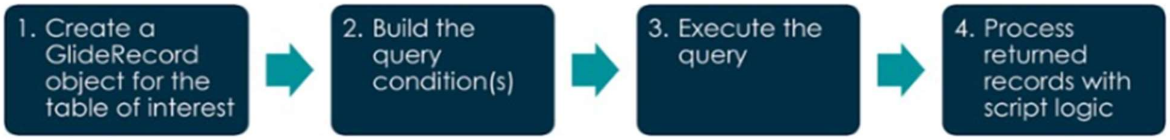
- Glide Record basically interacts with the database for inserting, updating, deleting the records.
- Glide Record queries can be created for any table.



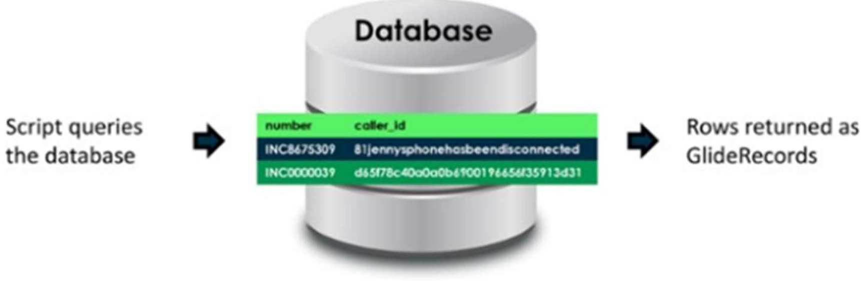
```
var incidents = new GlideRecord('incident');
incidents.addQuery('short_description', 'CONTAINS', "Cassette player");
incidents.query();

while (incidents.next()){
    incidents.setValue('description', "What is this, the 1970s?");
    incidents.update();
}
```

| number | caller_id |
|---|---|
| INC8675309 | 81jennysphonehasbeendisconnected |
| INC0000039 | d65f78c40a0a0b6900196656f35913d31 |

- The query() method executes a select statement against a database table.
- Glide records are the rows returned from the database.
- Glide record used for database operations instead of writing SQL Queries.
- It executes on the server side.
- **The procedure for creating a GlideRecord Query is below.**



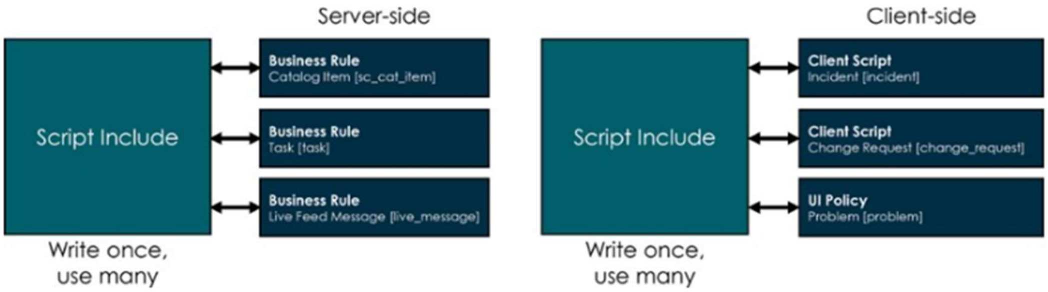
1. Create a GlideRecord object for the table of interest
2. Build the query condition(s)
3. Execute the query
4. Process returned records with script logic



| number | caller_id |
|---|---|
| INC8675309 | 81jennysphonehasbeendisconnected |
| INC0000039 | d65f78c40a0a0b6900196656f35913d31 |

# Script Includes

- Must be called to run
- Stores reusable JavaScript for execution on the server.
- Can be client-callable.



- There are 1,100 Script Includes in the baseline.
- There are three types of Script Includes
  - Store One Classless Function: Stores one reusable function.
  - Define a new class: Collection of functions.
  - Extend an existing: Adds functionality to the existing function.

- Code Reusability: It can be used for multiple scripts.
- Makes our code base cleaner & stronger.
- Performance Optimization & Easy Maintainability.

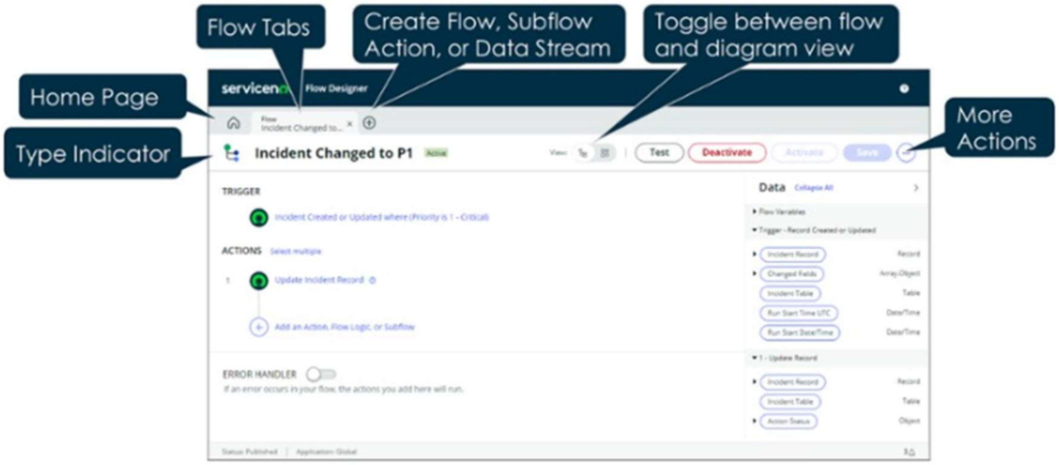
  Realtime Example:-

  Amazon Business Logistics such as

  - Distance of delivery
  - Weight of the parcel
  - Delivery Speed.
  - Calculating Complex business logics
  - lknoj

# Flow Designer

- We use flow designer to develop new flows or when there is a need for a major redesign to an existing workflow developed in workflow editor.
- It designs the environment that creates flows.



Flow Tabs

Create Flow, Subflow Action, or Data Stream

Toggle between flow and diagram view

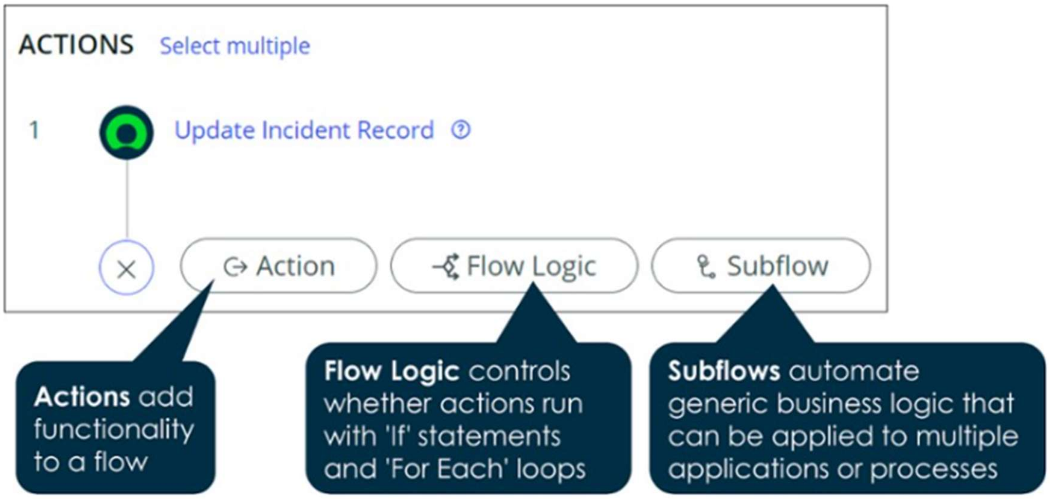
Home Page

Type Indicator

More Actions

- Flows consists of a trigger and at least one action.
- Triggers start a flow.
- Actions automate processes or record operations.
- Subflows are a sequence of reusable actions.

- Actions automate the process.
- Flow designer is controlled by flow-designer role.



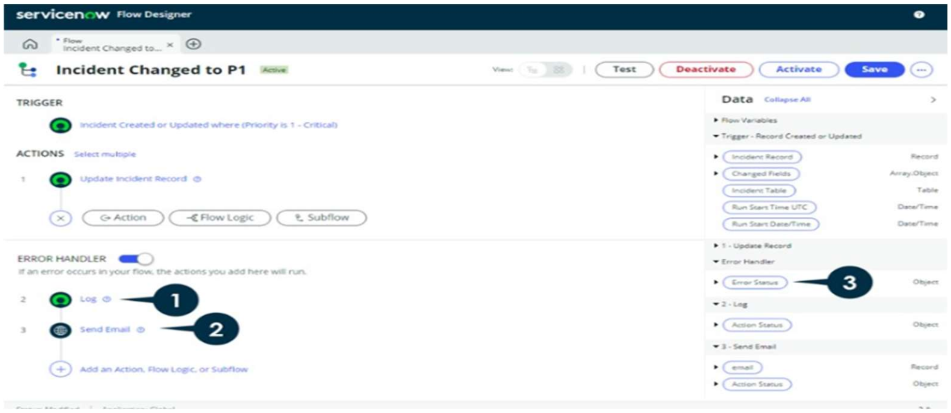
**Actions** add functionality to a flow

**Flow Logic** controls whether actions run with 'If' statements and 'For Each' loops

**Subflows** automate generic business logic that can be applied to multiple applications or processes

# Workflow Editor

- **It uses workflow editor to manage existing workflows that are still functioning as expected.**
- **The flow diagram enables to add & edit flow designer components.**
- **The following trigger types allow flow digramming**
    - **Record Trigger**
    - **Date Trigger**
    - **Service Catlog**
- **Flow Data- Additional actions – Additional Pills.**
- **Flow Logic – A trigger data pill to check the trigger incident priority.**
- **Complex data- It displays a parent, child, and sibling hierarchy.**
- **Error handler – It identifies errors in the flow & runs a sequence of actions and subflows & identifies the issues.**
- **Actions & Subflows provide reusable codes that are made into capsules.**
- **Flows – They Automate the process.**
- **Subflows – They break down complex problems into smaller chunks.**
- **Actions – These are the individual steps within a flow or subflow.**
-



Incident Changed to P1 Active

TRIGGER

Incident Created or Updated where (Priority is 1 - Critical)

ACTIONS Select multiple

1 Update Incident Record

ERROR HANDLER

If an error occurs in your flow, the actions you add here will run.

2 Log

3 Send Email

Add an Action, Flow Logic, or Subflow