ServiceNow

WEEK – 4

Koushik Mondra

KL University

2110030040cse@gmail.com

➔ In ServiceNow, scripting is a critical aspect for automating processes, customizing the platform, and extending its functionality. ServiceNow supports JavaScript as its primary scripting language, and there are two main types of scripting in the platform: client-side scripting and server-side scripting.

➔ In general terms what exactly is client side and server side

Client-Side:
Client-side refers to operations that happen on the user's device, typically in a web browser. It includes everything that the user interacts with directly, such as forms, buttons, and web pages. Client-side operations involve displaying data, validating user input, and managing UI elements.

Server-Side:
Server-side refers to operations that happen on the server, which hosts the application or website. The server handles things like processing user input, storing and retrieving data from databases, and executing business logic. Server-side scripting runs on the web server, and the results are sent back to the client (browser) after processing.

➔ Example: In an online shopping cart, when a user adds an item, **client-side scripting** (like JavaScript) updates the cart instantly, displaying the new item and total price without reloading the page. Once the user places the order, **server-side scripting** processes the payment, updates inventory and saves the order to the database. After processing, the server sends a confirmation page back to the user.

Server-Side and Client-Side Scripting in ServiceNow:

➔ **Client-Side Scripting**: This refers to scripts that run in the user's browser. It is primarily used to control and enhance the user experience on the client side. These scripts can interact with forms, validate user inputs, and dynamically adjust form elements (e.g., hiding/showing fields, making fields mandatory, or auto-populating values) before data is sent to the server.

➔ These are used to execute JavaScript logic on forms or lists. They are triggered by form actions (like loading or saving the form). Client scripts allow you to interact with form fields, validate inputs, show or hide fields, or make fields read-only.

Types of client scripts include:

onLoad: Executes when a form is loaded.
onChange: Executes when a field's value changes.
onSubmit: Executes when the form is submitted.
onCellEdit: Executes when a list cell is edited.

➔ **UI Policies**: These are used to dynamically change the behavior and appearance of form fields based on conditions. UI Policies can make fields mandatory, visible, or read-only.

➔ **UI Actions**: These are used to add buttons, links, and context menus on forms or lists that trigger actions like saving a record, canceling an action, or running custom scripts.

Real time example of client-side scripting:

➔ Form Validation in a Web Application

When a user fills out a registration form on a website, client-side scripting (using JavaScript) checks if the entered data is valid before the form is submitted to the server. For instance, it ensures that the email format is correct and that mandatory fields like "Username" and "Password" are filled in. If there's an error, the user receives immediate feedback without reloading the page.

➔ **Server-Side Scripting**: These scripts run on the server, and they handle tasks such as data validation, manipulation of records, and processing after a form is submitted. Server-side scripts are crucial for interacting with the database, applying business rules, and executing workflows or scheduled tasks.

➔ Server-side scripts run on the server and are used for manipulating records, processing data, and interacting with external systems.

➔ Key Types of server-side scripts:

Business Rules: These are scripts that execute whenever records are inserted, updated, deleted, or queried from the database. Business rules can run before or after the database operation. They are used to enforce business logic, validate data, or initiate workflows.

Script Includes: These are reusable server-side scripts that can be called from other scripts like Business Rules, Workflows, or UI Actions. They help modularize and reuse code.

Scheduled Jobs: These scripts run on a predefined schedule to perform tasks like generating reports, updating records, or cleaning up data.

Glide System (g_scripting): This is the set of APIs provided by ServiceNow to interact with the platform's data and services. GlideRecord is used to query and manipulate records in ServiceNow tables. GlideDateTime used to handle date and time fields.

➔ Real-Time Example of Server-Side Scripting in ServiceNow:

Example: Automatically Assigning a Support Ticket Based on Category

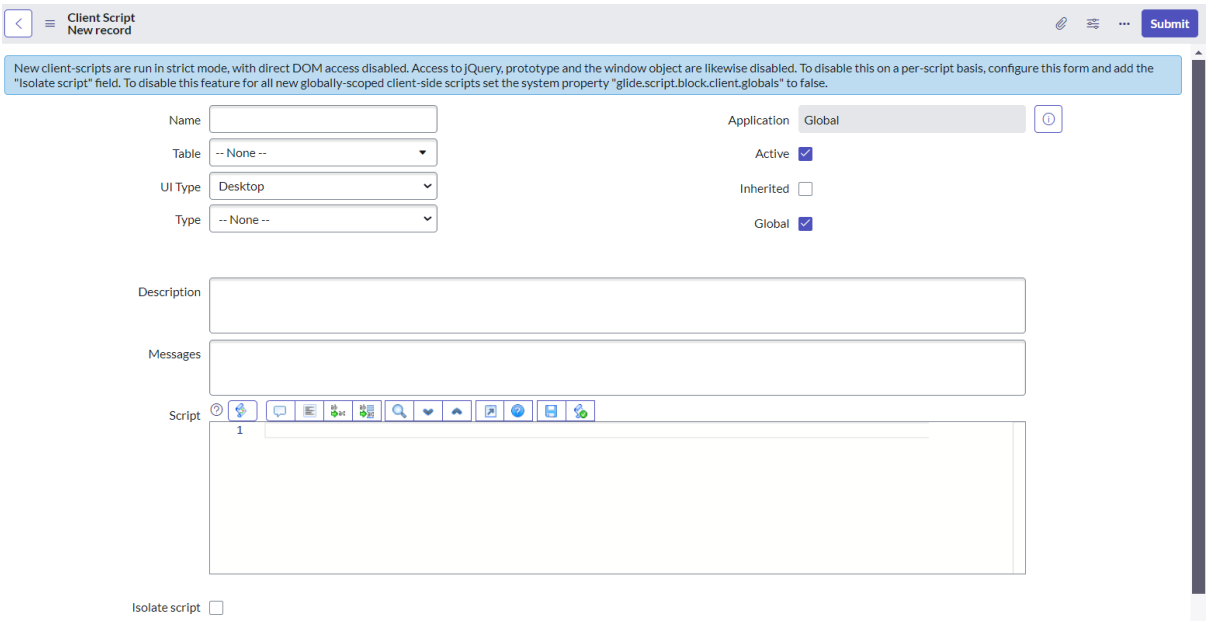
In ServiceNow, when a new **Incident** is created, you may want to automatically assign it to the correct team based on the selected category (e.g., "Network", "Hardware"). This can be done using a **Business Rule** (a server-side script).

Creating a Client Side Script

➔ Client Scripts are used to run JavaScript on the client-side, typically in the browser, to control form behaviour. Creating new scripts in ServiceNow involves several steps, depending on what type of script you want to create.

➔ Steps:

1. Navigate to System Definition > Client Scripts
2. Click New to create a new Client Script
3. Fill the Form



➔ The above image shows the form of client script record. Which includes Name, Table, Type, Script and after filling click on save.

➔ **Example**: A script to make a field mandatory when a specific condition is met.

```
function onChange(control, oldValue, newValue, isLoading) {
  if (isLoading || newValue == '') {
    return;
  }
  if (newValue == 'Urgent') {
    g_form.setMandatory('short_description', true);
  }
}
```

Creating a Server Side Script

➔ Creating a Business Rule: **Business Rules** are server-side scripts that run when records are inserted, updated, deleted, or queried.

➔ Steps:

1. Navigate to System Definition > Business Rules
2. Click New to create a new Business Rule
3. Fill the Form

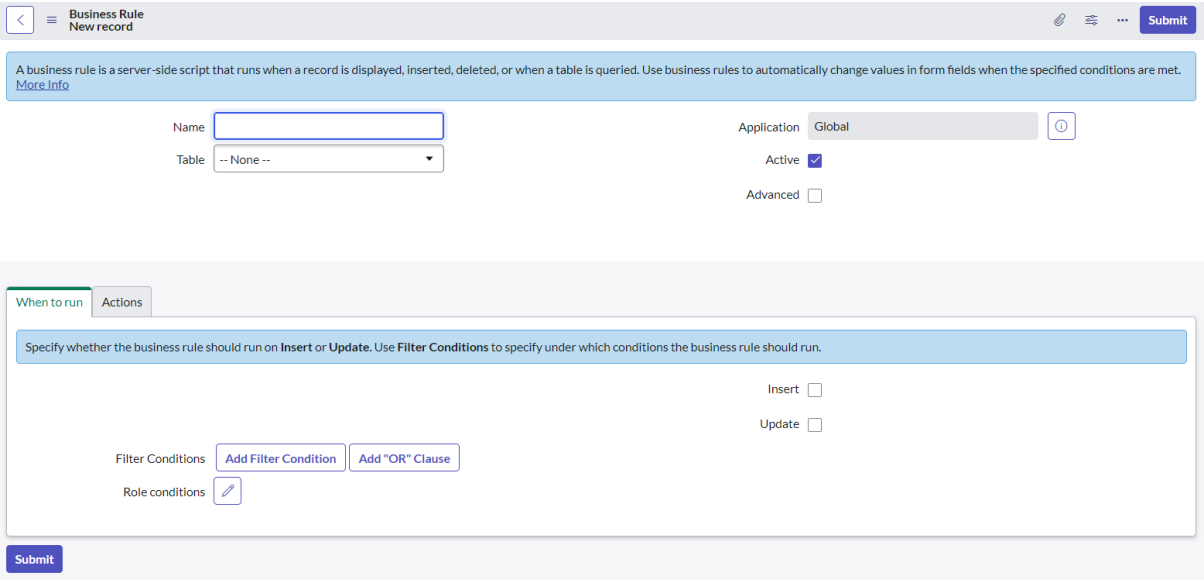
Name: Enter a descriptive name for the rule.

Table: Select the table where the rule will apply (e.g., Incident).

When to Run: Choose the trigger (before, after, or async).

Conditions: Define any conditions under which the rule should execute.

Script: Write your server-side JavaScript code in the Script field.



➔ A script to automatically assign a ticket to a group based on category.

```
(function executeRule(current, previous /*null when async*/) {
  if (current.category == 'Network') {
    current.assignment_group = 'Network Support';
  } else if (current.category == 'Hardware') {
    current.assignment_group = 'Hardware Support';
  }
})(current, previous);
```

➔ Above are 2 best examples of client-side and server-side scripting in ServiceNow.

Summary:

Client-side scripting enhances the user interface and controls the user experience by dynamically updating form fields, validating inputs, and improving interactivity.

Server-side scripting focuses on data processing, enforcing business rules, and automating workflows once the data is submitted.