**DIGITAL NURTURE 3.0**

**SERVICENOW WEEK 4**

**SERVICENOW SCRIPTING**

**NAME:** Sreevatsan T

**REGISTRATION NUMBER:** RA2111003011142

ServiceNow scripting is an essential part of customizing and extending the platform to meet specific business requirements. The ServiceNow platform uses JavaScript for both client-side and server-side scripting, allowing developers to automate processes, create workflows, and enhance the user experience. Here's a detailed overview of ServiceNow scripting, covering different areas such as client-side, server-side, and platform scripting.

1. Client-Side Scripting

Client-side scripts are executed in the user's browser and primarily manage the user interface (UI). They are useful for validating data, manipulating forms, and improving the overall user experience.

a. Types of Client-Side Scripts:

* Client Scripts: These run in the user's browser and respond to events on forms. They can control form behavior and validate input fields.
  + Types of Client Scripts:
    - onLoad: Runs when a form is loaded.
    - onChange: Executes when a specific field value changes.
    - onSubmit: Runs before form submission to validate the data.
    - onCellEdit: Runs when a field in a list is edited.

Example:

javascript

Copy code

function onLoad() {

// Hide the priority field when the form loads

g\_form.setVisible('priority', false);

}

* UI Policies: Used to dynamically change the behavior of fields on forms. UI policies control field visibility, mandatory settings, and read-only states.
  + UI Policy Actions: Define how a field behaves (read-only, mandatory, hidden) based on conditions.

Example:

javascript

Copy code

// Make a field mandatory when the incident's priority is "High"

function onCondition() {

g\_form.setMandatory('impact', true);

}

* UI Actions: Add buttons, links, and context menu items to forms or lists that allow users to perform specific actions.
  + UI actions can be scripted to perform custom operations when clicked.

Example:

javascript

Copy code

// Script to add a custom button that resolves an incident

if (current.state != 'Resolved') {

action.setRedirectURL(current);

current.state = 'Resolved';

current.update();

}

* GlideForm (g\_form) API: Provides functions for interacting with forms in the browser. It allows the manipulation of fields, UI elements, and controls.
  + Common methods:
    - g\_form.setValue(): Set the value of a field.
    - g\_form.getValue(): Get the value of a field.
    - g\_form.addErrorMessage(): Show an error message on the form.
    - g\_form.hideField(): Hide a specific field.

Example:

javascript

Copy code

g\_form.setValue('priority', '2'); // Set priority field value to "2"

* GlideUser (g\_user) API: Provides methods to retrieve user information and roles. It is useful when you want to perform actions based on the user’s role or details.

Example:

javascript

Copy code

if (g\_user.hasRole('admin')) {

g\_form.setVisible('confidential\_field', true); // Show field to admin users

}

2. Server-Side Scripting

Server-side scripts run on the ServiceNow server. They are responsible for handling back-end operations, data manipulation, and business logic. Server-side scripting allows you to create and update records, perform data validation, and automate workflows.

a. Types of Server-Side Scripts:

* Business Rules: Automate tasks when records are inserted, updated, deleted, or queried. Business rules can control how data is saved or processed before or after changes to a record.
  + Types of Business Rules:
    - before: Executes before a record is saved to the database.
    - after: Executes after a record is saved.
    - async: Executes asynchronously after a record is saved.
    - display: Executes when a record is queried from the database.

Example (before business rule):

javascript

Copy code

if (current.priority == 1) {

current.impact = 1; // Set impact to high if priority is high

}

* Script Includes: Reusable server-side scripts that can be called from other server-side scripts or client scripts. Script includes allow for better organization and code reusability.
  + Client-callable Script Includes can also be used by client-side scripts.

Example:

javascript

Copy code

var IncidentUtils = Class.create();

IncidentUtils.prototype = {

initialize: function() {},

getHighPriorityIncidents: function() {

var incidentGR = new GlideRecord('incident');

incidentGR.addQuery('priority', 1);

incidentGR.query();

return incidentGR;

},

type: 'IncidentUtils'

};

* GlideRecord API: This is the most important API for server-side scripting, as it allows developers to query, insert, update, and delete records in ServiceNow tables.
  + Common methods:
    - addQuery(): Adds conditions to a query.
    - query(): Executes the query to fetch records.
    - next(): Iterates through the returned records.
    - insert(): Inserts a new record.
    - update(): Updates an existing record.

Example:

javascript

Copy code

var gr = new GlideRecord('incident');

gr.addQuery('priority', 1);

gr.query();

while (gr.next()) {

gs.print(gr.number); // Print the number of each high-priority incident

}

* GlideSystem (gs) API: Provides methods for interacting with the system, logging messages, adding error messages, and managing system events.
  + Common methods:
    - gs.addErrorMessage(): Adds an error message for display to the user.
    - gs.log(): Logs messages to the system log.
    - gs.getUserID(): Gets the current user's Sys ID.

Example:

javascript

Copy code

gs.log('A business rule was triggered.');

* Scheduled Jobs: Scripts that run on a scheduled basis (daily, weekly, or custom intervals). These are used for regular maintenance tasks like sending notifications or performing data cleanup.
* Event Scripts: ServiceNow allows you to trigger events that can run scripts in response to certain actions (such as a change to a record). These are often used to automate workflows.

3. Platform Scripting

Platform scripting in ServiceNow refers to scripting that interacts with other areas of the platform, such as workflows, integrations, and REST APIs.

a. Workflow Scripts

Workflow scripts run as part of workflows and automate complex processes by integrating various activities (like approvals, task assignments, or custom scripts).

* Workflow Script Execution: Workflow scripts can run during transitions between workflow stages or as part of workflow activities.

Example:

javascript

Copy code

// Script to automatically assign a task to a specific group

current.assignment\_group = 'Hardware Support';

current.update();

b. Flow Designer Scripting

Flow Designer provides a low-code environment to create workflows using Flows and Actions. Script-based actions can be included in flows to extend their capabilities.

* Action Scripts: Flow Designer allows custom scripts to be used within flow actions to perform more advanced logic.

Example:

javascript

Copy code

// Custom Flow Action script

(function execute(inputs, outputs) {

var gr = new GlideRecord('incident');

gr.get(inputs.incident\_id);

gr.priority = 1;

gr.update();

})(inputs, outputs);

c. REST and SOAP API Integrations

ServiceNow supports REST and SOAP APIs, allowing integration with external systems. Developers can create custom REST messages and scripts to send or receive data between ServiceNow and other platforms.

* REST API Example:

javascript

Copy code

var r = new sn\_ws.RESTMessageV2();

r.setEndpoint('https://example.com/api/data');

r.setHttpMethod('GET');

var response = r.execute();

var responseBody = response.getBody();

d. Email Scripts

ServiceNow can send outbound emails with dynamic content using email notification scripts. You can embed scripts in the body of email notifications to include specific data from records.

Example:

javascript

Copy code

// Script to include the incident number in an email notification

var incidentNumber = current.number;

template.print('Incident Number: ' + incidentNumber);

4. Best Practices for ServiceNow Scripting

* Use GlideRecord responsibly: Avoid querying large datasets in loops as it may degrade system performance.
* Leverage Script Includes: Use script includes for reusable server-side logic to maintain code modularity and avoid duplication.
* Client-Server Interaction: Minimize client-server roundtrips for better performance by using GlideAjax for server-side operations invoked from client scripts.
* Optimize Business Rules: Keep business rules efficient by running them only when necessary and using asynchronous rules when appropriate.
* Secure your scripts: Use proper security checks and Access Control Rules (ACLs) to ensure that users don’t have unauthorized access to sensitive data.