ServiceNow Scripting: This is the development of customized scripts that automate some processes and enhance their functionalities, as well as help in customizing the ServiceNow platform. Scripts are needed to extend some of ServiceNow's functionalities over the base configurations. Major kinds of scripting include Business Rules, Client Scripts, Script Includes, and Glide Ajax. Scripting allows developers to apply dynamic interactions and automate workflows and interfaces with other systems that will give users a customized experience as well as the administrator.

Scripting in ServiceNow: ServiceNow scripting is the core of ServiceNow that can enable users to create their custom functionalities and automate tasks. It applies the use of JavaScript to write code run on the server-side scripting or in the user's browser client-side scripting. The server-side scripting encompasses Business Rules, Script Includes, and Scheduled Jobs, while the client-side scripting encompasses Client Scripts and UI Policies. Scripting plays an important role in Customizing workflows, integrating with external systems, and high business logic.

ServiceNow Scripting: ServiceNow Scripting is a deep exploration of scriptwriting techniques and best practices. It typically includes the following:

Introduction to ServiceNow Scripting: Basics of Scripting, An overview of scripting, such as the ServiceNow scripting environment and tools.

Business Rules: Introduction to Business Rules, which allows the user to write, manage, and maintain Business Rules to automate back-end processes

Writing Client Scripts: Handling client-side logic and interaction with users by writing Client Scripts.

Script Includes: Creating reusable server-side scripts and functions

UI Policies and Data Policies: Configuring UI Policies and Data Policies for controlling form behavior and data validation

GlideAjax: Usage of GlideAjax for communication between client and server-side scripts

Integration and REST APIs: Integration with external systems using REST APIs and other integration methods

Debugging and Testing: Techniques to debug and test scripts for reliable execution and performance.

How ServiceNow works: The ServiceNow system provides an integrated ITSM platform with one data model that harmonizes both business and IT operations. The cloud architecture on which it is built provides the option for multi-tenancy, whereby the same instance can be utilized by several organizations, all with different data isolated for them. Key features include A user-friendly interface with very intuitive navigation; a comprehensive set of

modules for numerous IT and business functions and native automation and integration capabilities. The ServiceNow platform also allows for custom application development and more extensive reporting and analytics functionality.

- Configuring the Platform: Configuration in ServiceNow describes the personalization of core settings on the platform for organizational processes and requirements. This includes
- Defining User Roles: Creation of roles and permissions in the management of access controls to various parts of the system.
- Module Configurations: Customizing the available modules or setting up new ones according to the specific business needs, such as workflow and business rule configurations
- Data Schema Settings: Table and field management and proper relationships among them to accommodate the data model for the desired functionalities
- Automated Processes: Rules, notifications, and scripts setup to automate the workflow and reduce manual efforts
- Integration: ServiceNow integration with other systems via APIs and integration connectors to have uninterrupted flow and interoperability of data.

Personalization on the Platform: Personalization allows users to tailor their ServiceNow experience to better meet their needs and preferences. Examples include:

- Customization Forms: This could be layout changes, adding or removing fields, or field types so that they collect the right information most efficiently.
- Modifying Lists: Personalized views can be achieved by selecting columns, filters, and sort options so that data is presented to the user for easy consumption.
- Dashboards Development: Creation and setup of dashboards, ready with widgets and reports, a graphical summary of the most important metrics and related data points
- Preference Setting: User preference, default views, and preferred notification to improve user experience
- Custom Apps and Modules Development: Development of custom applications and modules of your choice through ServiceNow's development tools, in response to unique business needs.

Incident Module: A must-have module for managing disruptions to IT services, including

- Incident Logging: Description of the incident, urgency, and effect
- Categorization and Prioritization: Appending categories and priorities to incidents for easier handling and to get timely solutions.
- Assignment: The incidents are distributed to related teams or individuals on skills and who can be found online
- Resolution and Closure: The resolution of incidents is through diagnosis and fixing of the root cause followed by later closure with feedback to the user.
- Incident Management: Monitoring the progress of incidents and compiling reports to Analyze trends and performance.

Problem Module: Problem management is oriented toward resolving the root cause of incidents so that they do not recur. It involves

- Problem Identification: Identifying recurring or regular patterns or incidents that suggest the existence of a hidden problem
- Diagnosis/Analysis: Finding the root cause of problems through detailed analysis and troubleshooting
- Solution Development: Providing possible solutions or workarounds to implement in overcoming the identified problems.
- Documentation**:** Documentation of problems or errors together with solutions in a knowledge base to assist the easier resolution of future incidents.
- Review and Closure: Evaluation of the effectiveness of the solution and formal closure of problems solved.

Change Module: The change Management module provides control over IT system changes. They include:

- Change Requests: Constructing and documenting a change request.
- Assessment and Impact Analysis: Checks on the potential impacts and risks concerning the change in process.
- Approval Workflow: Obtaining approvals from stakeholders beforehand, before a change is put in.
- Implementation: Enforce the change as planned and in line with procedure
- Post-Implementation Review: Evaluates the impact of a change, whether it was successful or not, and documents lessons Learned.

Lists: Lists in ServiceNow are a view to display data in the table format and can be used to do efficient data manipulation. Some other functionalities in the lists are:

- Column Customization: Users have the privilege to select the column names to be presented, also their placement according to user convenience.
- Sorting and filtering: Use of sort and filter options to display a specific sub-relation of the data
- Batch Actions: Updating or deleting numerous records in one go.
- Saved Views: Creating views saved for multiple uses to facilitate workflow simplification
- Export Options: Export list data to CSV or Excel formats to allow external analysis.

Forms: Forms are used for detailed information capture in ServiceNow. The major features include:

- Field Configuration: Defining the types of fields (e.g. text, choice, date) and their. layout on the form.
- Section Organization: Grouping related fields into sections to enhance form usability. and readability.
- Business Rules: Defining rules to handle automatic actions triggered by form data, such as fields. validations or dynamic content changes.

- User Interaction: Interfaces for entering, editing, and viewing details about a given record.
- Personalization: Revolving forms to suit certain business requirements and to enhance the efficiency of the data entry process.

Client-Side Objects

1. GlideForm (g_form):

purpose: Manages form fields, and controls client-side interactions.

Methods and Properties:

- g_form.getValue('field_name'): Get the value of a field.
- g_form.setValue('field_name', 'value'): Set the value of a field.
- g_form.showFieldMsg('field_name', 'message', 'info'): Show a message beside a field.
- g_form.addOption('field_name', 'value', 'label'): Add an option to a choice field.

Example:

```
// Set the value of a field

g_form.setValue('short_description', 'New incident created');

// Show a message on the form

g_form.showFieldMsg('short_description', 'Please provide a detailed description', 'warning');
```

2. GlideUser (g_user):

Purpose: Allows access to data about the currently authenticated user.

Methods and properties:

- g_user.userName :Gets the current user's name.
- g_user.email :Gets the current user's email.

EXAMPLE

```
// Get the current user's name

var username = g_user.userName;

console.log(' Logged in user: ' + username);
```

3. GlideDialogWindow:

Opens and manages modeless dialogs on the client

Methods and properties:

- g_dialog.open('dialog_name'): Opens a dialog window.
- g_dialog.close(): Closes the currently open dialog.

Example:

```
// Open a custom dialog

var dialog = new GlideDialogWindow('my_dialog');

dialog.render();
```

Server-Side Objects

1. **GlideRecord:**

• **Purpose:** Talk to ServiceNow database to do CRUD operations on

records.

• **Methods and Properties:**

- new GlideRecord('table_name'): Creates a new GlideRecord object for a
- specific table.
- gr.query(): Executes the query in order to get records.
- gr.next(): Moves to the next record in the result set.
- gr.getValue('field_name'): Gets a field's value.
- gr.setValue('field_name', 'value'): Sets a value for a field.
- gr.update(): Saves changes to the record.
- gr.deleteRecord(): Deletes the record.

Example:

```
// Query the incident table and update a record

var gr = new GlideRecord('incident');

gr.addQuery('number', 'INC0010001');

gr.query();

if (gr.next()) {

 gr.setValue('short_description', 'Updated description');

 gr.update();

}
```

2. GlideSystem (gs):

Purpose: Utility methods and information about the system environment.

Methods and Properties:

- gs. info('message'): Logs an informational message to the system log.
- gs.getUserID(): Gets the ID of the currently logged-in user.
- gs.getUserName(): Gets the currently logged-in username.

Example:

```
// Logs a message to the system log

gs.info('This is a log message.');

// Gets the ID of the current user

var userId = gs.getUserID();
```

3. GlideDateTime:

Purpose: Utilizes date and time functionality in a server-side script.

Methods and Properties:

- new GlideDateTime(): Creates a new GlideDateTime instance with the date and time set to the computer's current ones.
- getValue(): Returns the date and time value as a string.
- setValue(): Sets the date and time value.

• Example:

```
// Create a GlideDateTime object and log the current date and time

var gdt = new GlideDateTime();

gs.info('Current date and time: ' + gdt.getValue());
```