

# Client Script

Client scripts allow the system to run JavaScript on the client (web browser) when client-based events occur, such as when a form loads, after form submission, or when a field changes value.

Use client scripts to configure forms, form fields, and field values while the user is using the form.

## Client scripts can:

- make fields hidden or visible
- make fields read only or Editable
- make fields optional or mandatory based on the user's role
- set the value in one field based on the value of other fields
- modify the options in a choice list based on a user's role
- display messages based on a value in a field

## Client Script will execute in four ways:

**onLoad()** — runs when the system first renders the form and before users can enter data. Typically, **onLoad()** client scripts perform client-side-manipulation of the current form or set default record values. (Runs when a form is loaded)

**onSubmit()** — runs when a form is submitted. Typically, **onSubmit()** scripts validate things on the form and ensure that the submission makes sense. An **onSubmit()** client script can cancel form submission by returning a value of false. (Runs when a form is submitted)

**onChange()** — runs when a particular field value changes on the form. The **onChange()** client script must specify these parameters. (Runs when a particular field value changes)

- **control**: the DHTML widget whose value changed.  
**Note:** **control** is not accessible in mobile and service portal.
- **oldValue**: the value the widget had when the record was loaded.
- **newValue**: the value the widget has after the change.
- **isLoading**: identifies whether the change occurs as part of a form load.
- **isTemplate**: identifies whether the change occurs as part of a template load.

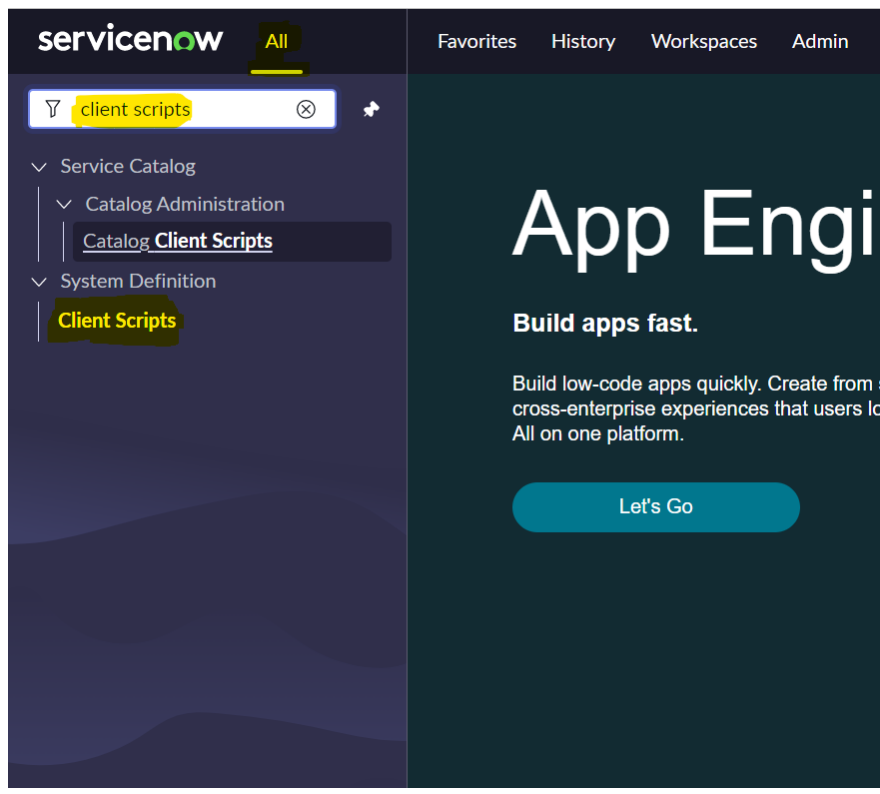
**onCellEdit()** — runs when the list editor changes a cell value. The **onCellEdit()** client script must specify these parameters. (Runs when a cell on a list changes)

- **sysIDs**: an array of the **sys\_ids** for all items being edited.
- **table**: the table of the items being edited.
- **oldValues**: the old values of the cells being edited.
- **newValue**: the new value for the cells being edited.
- **callback**: a callback that continues the execution of any other related cell edit scripts.  
If **true** is passed as a parameter, the other scripts are executed or the change is committed if

there are no more scripts. If **false** is passed as a parameter, any further scripts are not executed and the change is not committed.

## How to create client script:

- 1) In the left navigation under System Definition, we will find the client script



2) Click on new button to create Client script:

Client Script New record

Name: **Popup Example** *Client script name*

Table: **Vulnerability Management [u\_v...** *in which table client script will execute*

UI Type: **Desktop** *in which type of device the client script will execute*

Type: **Desktop**  
**Mobile / Service Portal**  
**All** *in which type of device the client script will execute*

Application: **Global**

Active: ☒

Inherited: ☐

Global: ☒ *Client script will applicable for all views or particular views*

Description:

Messages:

Script: *Here we will write our script*

-> Actions gets triggered based on these 4 types of events:

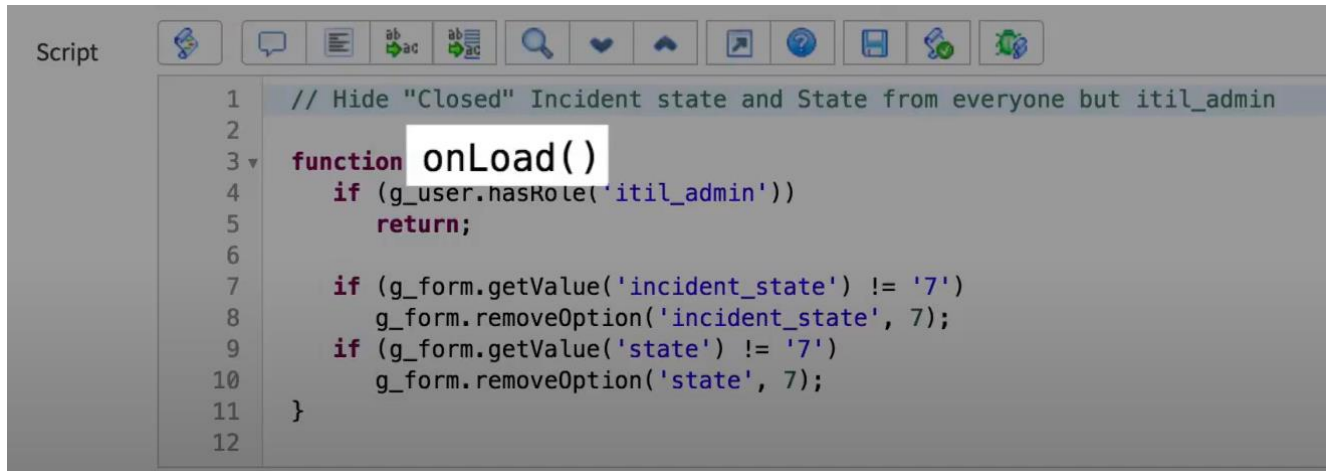
UI Type: **Desktop**

Type: **-- None --**  
**-- None --**  
**onCellEdit**  
**onChange**  
**onLoad**  
**onSubmit**

Description:

## OnLoad():

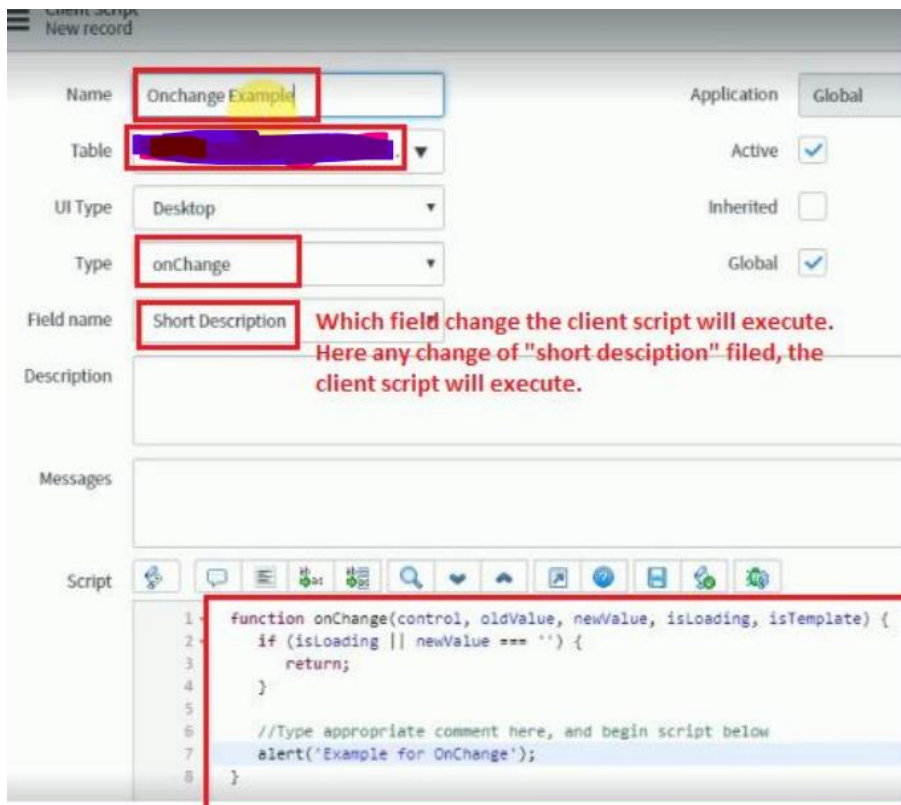
If we select the onload then when the page loads, the client script will run.



```
1 // Hide "Closed" Incident state and State from everyone but itil_admin
2
3 function onLoad()
4     if (g_user.hasRole('itil_admin'))
5         return;
6
7     if (g_form.getValue('incident_state') != '7')
8         g_form.removeOption('incident_state', 7);
9     if (g_form.getValue('state') != '7')
10        g_form.removeOption('state', 7);
11 }
12
```

## OnChange():

onChange() client script will get executed when user change any value in an existing form.



Name: Onchange Example

Table: [Redacted]

UI Type: Desktop

Type: onChange

Field name: Short Description

Which field change the client script will execute.  
Here any change of "short description" filed, the client script will execute.

Script

```
1 function onChange(control, oldValue, newValue, isLoading, isTemplate) {
2     if (isLoading || newValue === '') {
3         return;
4     }
5
6     //Type appropriate comment here, and begin script below
7     alert('Example for OnChange');
8 }
```

## OnSubmit():

onSubmit() client script will get executed when user submits the form.

The screenshot shows the ServiceNow form editor interface. At the top, 'UI Type' is set to 'Desktop' and 'Type' is set to 'onSubmit'. The 'Inherited' checkbox is unchecked, and the 'Global' checkbox is checked. Below these are fields for 'Description' and 'Messages'. The 'Messages' field contains the text: 'Cannot open a priority 1 incident on a low impact event'. The 'Script' section is active, showing a client script function:

```
1 function onSubmit() {  
2     var priority = g_form.getValue('priority');  
3     var impact = g_form.getValue('impact');  
4     if (priority == 1 && impact > 2) {  
5         alert(getMessage('Cannot open a priority 1 incident on a low impact event'));  
6         return false;  
7     }  
8 }  
9
```

## onCellEdit():

onCellEdit() is used in list view. Cell means a particular row inside a column (Like a excel cell). For list view in ServiceNow we will get the value as a tabular form so here one value is like cell value. So here if we want to modify any value and trigger a client script then we can use onCellEdit

The screenshot shows the ServiceNow form editor interface for a field named 'Coalesce'. The 'Field name' dropdown is set to 'Coalesce'. Below are fields for 'Description', 'Messages', and 'Script'. The 'Script' section is active, showing a client script function:

```
1 function onCellEdit(sysIDs, table, oldValues, newValue, callback)  
2     var saveAndClose = true;  
3  
4     g_form.addInfoMessage('Coalesce settings changed. After all coalesce fields are configured, use the  
<b>Index Coalesce Fields</b> related link to check if a new database index is required.');
```