


Field And Form Rules - Supported JavaScript Functions

We have added **jQuery** and **\$CS** in the global scope, allowing the user to write code by using jQuery (v 1.8.3) and the \$CS library.


 You cannot use \$ to access elements that use **jQuery**.

JavaScript Functions in the \$CS Library

Requirement	Function	Examples	Remarks
Get Field Values	<code>\$CS.getValue(field)</code>	<ul style="list-style-type: none"> <code>var statusId=\$CS.getValue("STATUS");</code> <code>var subject= \$CS.getValue("SUBJECT");</code> <code>var created_date=\$CS.getValue("CREATEDDATE"); //This will return a JavaScript Date Object</code> <code>var additional_hardware=\$CS.getValue("RES_3_QUS_3"); //For checkbox type resource fields returns an array of selected resources</code> 	To retain the existing checked resources, store them in an array and add resources to it. Then, set the resource field by using this updated.
Get Requester Field Values	<code>\$CS.getValue(field.attribute)</code>	<ul style="list-style-type: none"> <code>var email_id=\$CS.getValue("REQUESTER.EMAILID");</code> <code>var department=\$CS.getValue("REQUESTER.DEPARTMENT");</code> <code>var job_title=\$CS.getValue("REQUESTER.JOBTITLE");</code> <code>var mobile_number=\$CS.getValue("REQUESTER.MOBILE");</code> <code>var phone = \$CS.getValue("REQUESTER.CONTACTNUMBER");</code> <code>var requester=\$CS.getValue("REQUESTER");</code> <code>var employeeID=\$CS.getValue("REQUESTER.EMPLOYEEID");</code> <code>var site=\$CS.getValue("REQUESTER.SITE");</code> <code>var is_vipuser=\$CS.getValue("REQUESTER.ISVIPUSER");</code> <code>var user_name=\$CS.getValue("REQUESTER.USERNAME");</code> <code>var userID=\$CS.getValue("REQUESTER.USERID");</code> <code>var login_name=\$CS.getValue("REQUESTER.LOGINNAME");</code> <code>var domain=\$CS.getValue("REQUESTER.DOMAINNAME");</code> 	
Set a Field Value	<code>\$CS.setValue(field,value)</code>	<ul style="list-style-type: none"> <code>\$CS.setValue("STATUS","1"); \$CS.setValue("SUBJECT","test request");</code> <code>\$CS.setValue("UDF_DATE1", new Date());</code> <code>\$CS.setValue("RES_3_QUS_3", ["CD RW", "External Harddisk", "Optical Mouse"]); //Will replace the existing resources with the provided input.</code> 	
Get Text	<code>\$CS.getText(field)</code>	<code>var status=\$CS.getText("STATUS"); var impactdetail=\$CS.getText("IMPACTDETAILS");</code>	
Set Text	<code>\$CS.setText(field,text)</code>	<code>\$CS.setText("STATUS", "Open");</code>	
Add Options	<code>\$CS.addOptions(field,options)</code>	<code>\$CS.addOptions("STATUS",["Open","Closed"]);</code>	Ensure that they are Pick Type fields and an array.
Remove Options	<code>\$CS.removeOptions(field,options)</code>	<code>\$CS.removeOptions("STATUS",["Open","Closed"]);</code>	Same as above.
Remove all options	<code>CS.removeAllOptions(fields)</code>	<code>\$CS.removeAllOptions(["STATUS"]); // If all options of status field is removed, new request creation will not work. Since every request should have a status. \$CS.removeAllOptions(["STATUS","PRIORITY"]);</code>	Same as above.
Enable Fields	<code>\$CS.enableField(fields)</code>	<code>\$CS.enableField(["LEVEL","PRIORITY","URGENCY"]);</code>	

Disable Fields	<code>\$CS.disableField(fields)</code>	<code>\$CS.disableField(["LEVEL", "PRIORITY", "URGENCY"]);</code>	
Hide Fields	<code>\$CS.hideField(fields)</code>	<code>\$CS.hideField(["LEVEL", "PRIORITY", "URGENCY"]);</code>	
Show Fields	<code>\$CS.showField(fields)</code>	<code>\$CS.showField(["LEVEL", "PRIORITY", "URGENCY"]);</code>	
Mandate Fields	<code>\$CS.mandateField(fields)</code>	<code>\$CS.mandateField(["LEVEL", "PRIORITY", "URGENCY"]);</code>	
Make Fields Optional	<code>\$CS.nonMandateField(fields)</code>	<code>\$CS.nonMandateField(["LEVEL", "PRIORITY", "URGENCY"]);</code>	
Stop Form Submission	<code>\$CS.stopFormSubmission()</code>	<code>var status=\$CS.getText("STATUS"); if(status=="Closed"){ \$CS.stopFormSubmission(); }</code>	
Find Login Status of Requester	<code>\$CS.isRequester()</code>		
Check for Technician Login	<code>\$CS.isTechnician()</code>		
Find User Role	<code>\$CS.hasRole(role)</code>	<code>\$CS.hasRole("SDAdmin");</code>	
Find Logged-in User ID	<code>\$CS.getLoggedInUserName()</code>	<code>var userName= \$CS.getLoggedInUserName();</code>	
Set Field Dependency Object	<code>\$CS.setFieldDependency(dependencyObject)</code>	<pre>var dependencyObj= { 'FIELDS': ['Country', 'City', 'Support Rep'], 'VALUES': { 'India': { 'Mumbai': ['Ali Hassan', 'Neha Agarwal'], 'Chennai': ['Guru Prasath', 'Ramesh Kumar'] }, 'America': { 'California': ['Donald Miller', 'Lisa Turner'], 'Chicago': ['Margaret Taylor', 'Ronald Lewis'] }, 'Russia': {} } }; \$CS.setFieldDependency(dependencyObj);</pre>	<p>Here, the same function works for both two-field and three-field dependencies. This method is used to create dependency among additional fields only. The format for dependency object should be as follows: The order of dependent fields should be provided in FIELDS array from right to left i.e. fields in FIELDS array at second position should be dependent on the field in the first position. In addition, options in VALUES should be in the array format for the right most field, whereas options in VALUES should be in the JSON Object format for all other fields. Provide correct casing for field's label and option's label. Otherwise, the dependency will not work properly. Please refer above example for dependency object format.</p>

Set Tasks	<code>\$CS.setTasks(tasksArray)</code>	<code>\$CS.setTasks(["templateTask1","templateTask2","templateTask3"]);</code>	
Unset Tasks	<code>\$CS.unSetTasks(tasksArray)</code>	<code>\$CS.unSetTasks(["templateTask1","templateTask2","templateTask3"]);</code>	
Add Description	<code>\$CS.setDescription(description)</code>	<code>\$CS.setDescription("Application crashes / hangs frequently in user environment causing instability for the system.");</code>	
Get Description	<code>\$CS.getDescription()</code>	<code>var descriptionContent=\$CS.getDescription();</code>	
Disable Options	<code>\$CS.disableOptions(fieldId,options)</code>	<code>\$CS.disableOptions("STATUS",["Open","Closed"]);</code>	
Enable Options	<code>\$CS.enableOptions(fieldId,options)</code>	<code>\$CS.enableOptions("STATUS",["Open","Closed"]);</code>	
Check Field Visibility	<code>\$CS.isVisible(field)</code>	<code>var isPriorityFieldVisiable=\$CS.isVisible("PRIORITY");</code>	
Check Resource Edit Status	<code>\$CS.isEnabled(field)</code>	<code>\$CS.isEnabled("PRIORITY");</code>	
Check Mandatory Status	<code>\$CS.isMandated(field).</code>	<code>\$CS.isMandated("PRIORITY");</code>	
	<code>\$CS.getLoggedInUserEmailId()</code>	<code>var userEmail= \$CS.getLoggedInUserEmailId();</code>	
Find Logged-in User domain name	<code>\$CS.getLoggedInUserDomainName()</code>	<code>var userDomainName= \$CS.getLoggedInUserDomainName();</code>	
Get server time	<code>\$CS.getServerTime()</code>	<code>var ServerTime=\$CS.getServerTime();</code>	

 Errors during script execution will be displayed in the console.

Module/Section	Trigger Event	Use Case	Script
Custom Events in Field and Form Rules			
Asset Details		Hide the Vendor field in the Asset Details page.	<pre>\$CS.findElement("asset_detailview").on("page:load",()=>{ //Code that should run after asset page is loaded \$CS.hideElement("vendor"); }) tab_</pre>
Change Roles	Triggers when the Change Roles page is loaded.		<pre>\$CS.executeEvent("change_roles_container","page:load",()=>{ \$CS.hideElement("change_manager"); });</pre>
Change Notes	Triggers when the Change Notes section is loaded in stages.	Hide the Delete button in the Notes page.	<pre>\$CS.executeEvent("change_notes_container").on("page:load",()=>{ //code that should run after change notes rendered });</pre>
Change Work Log	Triggers after the Work Logs List View page is loaded.	Hide the Delete button in the Work Logs page.	<pre>\$CS.executeEvent("worklog_listview","page:load",()=>{//code })</pre>
Task List View	Triggers after the Tasks List View page is loaded.	Hide the New button in the Tasks List View page.	<pre>\$CS.findElement("task_listview").on("page:load",()=>{//code})</pre>
Tab Change Custom events			
Tab Change Events	Triggers when any stage tab is changed.		<pre>console.log("stage changed " + data); let active_tab = data; \$CS.findElement("stages-tabs-panel").off("tab_change").on("tab_change", function (e, d) { active_tab = window.location.hash && window.location.hash.split("/") [0].substring(1); e.stopPropagation(); \$CS.findElement(active_tab + "-tabs-panel").on("tab_change", function (e, d) { console.log("sub tab changed " + d); }) console.log(`stage tab changed \${d}`); }) })</pre>
		Hide any sub tab in Changes.	<pre>//change Module \$CS.executeEvent("change_stage", "tab_change", function (e, data) { \$CS.hideElement("approvals-tab") })</pre>
		Hide any sub tab in Releases.	<pre>//release \$CS.executeEvent("release_stage", "tab_change", function (e, data) { \$CS.hideElement("approvals-tab") })</pre>
listen:click events			
Service Catalog	Triggers when any service category is clicked.	listen:click Events	<pre>\$CS.findElement("service_category","starts_with").on("listen:click",function(event){console.log("clicked");})</pre>

Requests	Triggers when the Reply all button is clicked in the Request Details page.		<code>\$CS.findElement("reply_btn").on("listen:click",()=>{//code});</code>
	Triggers when the Recommend button is clicked in the Request Details page.		<code>\$CS.findElement("recommend_btn").on("listen:click",()=>{//code});</code>
	Triggers when the Reply to button is clicked in the Request Details page.		<code>\$CS.findElement("reply_to_btn").on("listen:click",()=>{//code});</code>
	Triggers when the Forward button is clicked in the Request Details page.		<code>\$CS.findElement("forward_btn").on("listen:click",()=>{});</code>
	Triggers when the Resend button is clicked in the Request Details page.		<code>\$CS.findElement("resend_btn").on("listen:click",()=>{});//No I18N</code>
Outlook page event			
Outlook page	Triggers only on the Outlook page. Form section meta is passed in this event, which contains Outlook data.		<code>\$CS.findElement("outlook_page").off("outlook_request_form").on("outlook_request_form",function(evt,data,meta){ console.log(data);\$CS.setText("SUBJECT",\$CS.getText("SUBJECT")+data.user_email); });</code>

Field and Form Rules Functions

Function	Remarks	Parameters	Script
readAndPopulateData	Reads a CSV file and populates values from CSV to single line fields as options. readAndPopulateData(path, map, options)	Path parameter is CSV file path. The CSV file has to be placed inside the custom folder. Map parameter has CSV and fields map. The first key is considered as primary field, which should be unique. Options parameter Dependency order in which fields should be populated. Primarykey field and form rules key should be given for this object. onchange function is triggered when any field configured in Map is Change. autopopulate parameter is used when primary key is changed If the autopopulate is set to True the options with respect to Map will be provided	<code>\$CS.readAndPopulateData("Catalog.csv",{ "WorkOrder_Fields_UDF_CHAR3":"Incident Type", "CATEGORY": "Service Category", "SUBCATEGORY": "Sub Category", "ITEM": "Sub Sub Category", }, { dependency:["WorkOrder_Fields_UDF_CHAR3","CATEGORY","SUBCATEGORY"], default_value:"Unable to launch MS Excel", primarykey:"WorkOrder_Fields_UDF_CHAR3", onChange:(a,b)=>{ //code } })</code>
referField	Populate options from API to single line fields. Function (fieldId, entityName, options)	fieldId - The FAFR key should be given entityName - The name of the entity should given. Example: ("udf_pick_119", "requests", etc) options object type select2 - Options with url should be given.	<code>\$CS.referField("WorkOrder_Fields_UDF_CHAR1","udf_pick_119",{url:"api/v3/requests/udf_pick_119"})</code>

Function Name	Remarks	Parameters		Examples
		Name/Type	Description	
GetText	Returns the text value of a field.	field/String	data-cs-field for the field must be provided.	<code>\$CS.getText("change_requester","change_view") \$CS.getText("stage")</code>
		form/String	The getText is used from PageScript form. Parameter is mandatory to get value from a form.	
getValue	Returns ID to make API calls.	field/String	data-cs-field for the field has to be provided or FieldName suggested in FAFR must be used.	<code>\$CS.getValue("stage") \$CS.getValue("change_requester","change_view")</code>
		form/String	The getText is used from PageScript form. Parameter is mandatory to get value from a form	
setValue	Used to update the formValue with the ID passed in the method.	field/String	data-cs-field for the field has to be provided or FieldName suggested in FAFR must be used.	<code>\$CS.setText("title","sds","change_edit")</code>
		value/String or Array	ID has to be passed.	
		forUnset/String		
setText	Used to set value to the form field based on the input text parameter.	fieldValue/String	data-cs-field for the field has to be provided or FieldName suggested in FAFR must be used. text Value that has to be set for the respective field.	
hideElements	Hides the fields with respect to matched element data-cs-field>data-name>data-id>id. If the element matches any of these attributes, the fields will be hidden.	field/String or Array	data-cs-field for the element.	<code>\$CS.hideElement("request_type")</code>
showElement	showElement method shows hidden fields with respect to matched element data-cs-field>data-name>data-id>id. If element matches any of these attributes, the hidden fields will be shown.	field/String or Array	data-cs-field for the element.	<code>\$CS.showElement("request_type")</code>
addElement	Appends an element before or after the given element.	selector/String	jQuery selector has to be given.	
		element/String	An html string has to be given.	
ajax	ajax method is used to make a get call for a url and return the response in a synchronous manner.	position/String	jQuery function name ex ("after","before","html")	<code>\$CS.ajax("/api/v3/requests");</code>
		URL/String	The url for which get call has to be made.	
addButton	The addButton function is used to add button in the given position.	selector/String	jQuery selector has to be given.	<code>\$CS.addButton("reply_btn","samplebtn",()=>{})</code>
		name/String	The Name of the button.	
		callback/function	Callback function is triggered when the button is clicked.	
		options/object	<div>class css class for the Button</div> <div>position css position example before.after</div>	

executeEvent	The executeEvent method is used to attach a dynamic event on the window even if the element does not present.	elementId/String	The data-cs-field for the element that dynamic event to be triggered has to be given.																													
		eventType/String	The dynamic event name should be given.																													
		callback/function	Callback function triggered when the dynamic event is triggered.																													
addTab	Adds a tab in the details page of request, change, and release modules.	content/String	The content to be shown when tab is clicked. Note: url can be content type but the type parameter has to be given as url.		\$CS.addTab("hello world","html","sampleTab","history-tab")																											
		type/String	HTML or URL should be provided according to the use case.																													
		name/function	Callback function is triggered when the dynamic event is triggered.																													
		selector/String	data-cs-field of tab after the custom tab has to be rendered should be given.																													
		callback/function	Callback is called after the tab is clicked.																													
addWidget	Adds a widget on the selected element.html. Accepts url type widgets.	content/String	The content to be shown when tab is clicked. Note: URL can be content type, but the type parameter has to be given as url.		\$CS.addWidget("hello world","html","resourceWidget","#tab-content",{position:"html"})																											
		type/String	HTML or URL should be provided according to the use case.																													
		name/function	Callback function triggered when the dynamic event is triggered.																													
		selector/String	jQuery selector																													
		options/object	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>header</td><td>boolean</td><td>mentions if header has to be present</td></tr><tr><td>position</td><td>string</td><td>css position ex before,after, and prepend</td></tr></table>	Parameter	Type	Description	header	boolean	mentions if header has to be present	position	string	css position ex before,after, and prepend																				
		Parameter	Type	Description																												
header	boolean	mentions if header has to be present																														
position	string	css position ex before,after, and prepend																														
isAttachmentEmpty	The isAttachmentEmpty method is used to return regardless of whether the attachment field in the form is empty or not.	boolean		\$CS.isAttachmentEmpty();																												
collapseTab	The collapseTab method is used to collapse a zcollapsible panel. The data-cs-field for the zcollapsible panel has to be provided as argument.			\$CS.collapseTab("change_description")																												
expandTab	The collapseTab method is used to expand a zcollapsible panel. The data-cs-field for the zcollapsible panel has to be provided as argument.			\$CS.expandTab("change_description");																												
disableField	The disableField method is used to disable form fields. FieldIds must be passed in arguments to disable fields. To disable multiple fields, the Field ID can be passed as array.			\$CS.disableField("STATUS"); \$CS.disableField(["STATUS","PRIORITY"])																												
enableField	The enableField method is used to enable the disabled form field. FieldIds must be passed in arguments to enable fields. To enable multiple fields, the Field ID can be passed as array.			\$CS.enableField("STATUS"); \$CS.enableField(["STATUS","PRIORITY"])																												
hideSection	The hideSection method is used to hide a section in form. data-section must be passed as argument.			\$CS.hideSection("Requester Details")																												
showSection	The showSection method is used to show hidden sections. data-section must be passed as argument.			\$CS.showSection("Requester Details")																												
disableSection	The disableSection method is used to disable a section. data-section for the section must be provided as argument.			\$CS.disableSection("Requester Details")																												
addMoreResource	The addMoreResource method is used to append the resource fields data in tabular format in description. Use Case: Consider a scenario where a travel request form is created. The user needs to select hotels for a tour because there might be more than one hotel stays in a single tour. For this business requirement, we can populate the value of the given fields in the editable table, and the same table data will be copied to the description.	resource/String	Id of the section the button should be added	<pre>var fields = [{ display_name: "Item", fafrKey: "ITEM" }, { display_name: "Subcategory", fafrKey: "SUBCATEGORY" }, { display_name: "Category", fafrKey: "CATEGORY" }, { display_name: "Vendor_Name", fafrKey: "WorkOrder_Fields_UDF_CHAR11" }, { display_name: "Source", fafrKey: "WorkOrder_Fields_UDF_CHAR2" }, { display_name: "Destination", fafrKey: "WorkOrder_Fields_UDF_CHAR3" },]; var options = { section: "WorkOrder_Fields_UDF_CHAR1", individualValidation: false, resetVal: false, table: fields, addButton: function () { var self = this; var element = jQuery(<div class='form-footer' style='border: 1px solid #cccccc'><input type='button' id='addButtonCUB' value='' + "Add" + " class='btn btn-primary"></div>); element.on("click", function () { self.constructRow(); }); \$CS.findElement("#addButtonCUB").length == 0 && \$CS.findElement("submit").before(element); }, }; \$CS.addMoreResource("", options);</pre>																												
		options/object	<table><tr><th>Name</th><th>Type</th><th>Description</th></tr><tr><td>individual Validation</td><td>boolean</td><td>Individual validation needed</td></tr><tr><td>section</td><td>string</td><td>Resource section name</td></tr><tr><td>button</td><td>object</td><td>name and jQuery element can be passed</td></tr><tr><td>resetVal</td><td>boolean</td><td>Resets the field</td></tr><tr><td>addButton</td><td>function (optional)</td><td>add function is called initially to add button, which is used to add tabular data.</td></tr><tr><td>getVal</td><td>function (optional)</td><td>Data and key will be provided while the extracted data must be provided for getVal.</td></tr><tr><td>validate</td><td>function (optional)</td><td>Validation to be performed before adding tabular data</td></tr><tr><td>addRow</td><td>function (optional)</td><td>addRow in table html,index,data,type arguments are provided</td></tr><tr><td>constructRow</td><td>function (optional)</td><td>constructRow function is used to constructRow for a table</td></tr></table>	Name	Type	Description	individual Validation	boolean	Individual validation needed	section	string	Resource section name	button	object	name and jQuery element can be passed	resetVal	boolean	Resets the field	addButton	function (optional)	add function is called initially to add button, which is used to add tabular data.	getVal	function (optional)	Data and key will be provided while the extracted data must be provided for getVal.	validate	function (optional)	Validation to be performed before adding tabular data	addRow	function (optional)	addRow in table html,index,data,type arguments are provided	constructRow	function (optional)
Name	Type	Description																														
individual Validation	boolean	Individual validation needed																														
section	string	Resource section name																														
button	object	name and jQuery element can be passed																														
resetVal	boolean	Resets the field																														
addButton	function (optional)	add function is called initially to add button, which is used to add tabular data.																														
getVal	function (optional)	Data and key will be provided while the extracted data must be provided for getVal.																														
validate	function (optional)	Validation to be performed before adding tabular data																														
addRow	function (optional)	addRow in table html,index,data,type arguments are provided																														
constructRow	function (optional)	constructRow function is used to constructRow for a table																														
getApprovalStatus	The getApprovalStatus method returns current approval status. Note: Only applicable to the request page.			\$CS.getApprovalStatus()																												
Get Login Name getLoggedInUserLoginName	returns the LoggedIn userName.			\$CS.getLoggedInUserLoginName()																												
getDescription	returns the Description value as String.			\$CS.getDescription();																												

Field Dependency setFieldDependency	The setFieldDependency method is used to set field dependency. Example: category, subcategory, item.		This method is used to create dependency among additional fields only. The format for dependency object should be as follows: The order of dependent fields should be provided in FIELDS array from right to left: Fields in FIELDS array in the second position should be dependent on the field in the first position. Options in VALUES should be in the array format for the right most field, whereas options in VALUES should be in the [SON Object] format for all other fields. Provide the correct letter case for field label and option label. Otherwise, the dependency will not work properly.	<pre> var dependencyObj= { 'FIELDS' : ['Country','City','Support Rep'], 'VALUES':{ 'India':{ 'Mumbai': ['Ali Hassan','Neha Kumar'], 'Chennai': ['Guru Prasath','Ramesh Agarwal'], }, 'America':{ 'California': ['Donald Miller','Lisa Turner'], 'Russia':{ 'Chicago': ['Margaret Taylor','Ronald Lewis'] } }, 'China':{ }, 'England':{ }, }, }, }; \$CS.setFieldDependency(dependencyObj); </pre>
getApprovalStatusValue	The getApprovalStatusValue method can be used to get approval status ID.			\$CS.getApprovalStatusValue();
mandateField	The mandateField function is used to mandate a form field. Field name must be passed as argument. For multiple fields to be mandated arguments should be given as array.			<pre> \$CS.mandateField("DESCRIPTION") \$CS.mandateField(["DESCRIPTION","ATTACHMENT"]) </pre>
isMandated	The isMandated method is used to find whether a form field is mandated.			\$CS.isMandated("DESCRIPTION")
nonMandateField		The nonMandateField method is used to non mandate a form field. Field name must be passed as argument. For multiple fields to be mandated, arguments should be given as array.		<pre> \$CS.nonMandateField("DESCRIPTION") \$CS.nonMandateField(["DESCRIPTION","ATTACHMENT"]) </pre>
exportPdf	The exportPdf function is used to export any page as a PDF.	options/object	<pre> { url: //page url non Mandatory fileName: //FileName of pdf page_settings: { timeout : 100 // timeout in milliseconds } } </pre>	<pre> \$CS.exportPdf({ url: //WorkOrder.do?woMode=viewWO&woID=1", fileName: "request_1" }) </pre>
"field:change" event	Execute script actions during inline edit.		<pre> Syntax \$CS.findElement("s{fieldName}").on("field:change",()=>{ //Handling to be done on field change }) </pre>	<pre> \$CS.executeEvent("change_view","form:load",()=>{ console.log(\$CS.getText("change_type","change_view")); // category field is edited in view mode \$CS.findElement("category").off("field:change").on("field:change",()=>{ console.log(\$CS.getText("category","change_view")) }) }) </pre>