Why and when to use JavaScript in CRM

The following outlines scenarios where you should use JavaScript in Microsoft Dynamics CRM.

Form Scripts

When using JavaScript, you want to have actions performed in form scripts. This is essentially form programming. Because the results of this style are instantaneous, it is the most preferred method and the most used. It is also highly adaptable to interact with various customer scenarios. Form scripts are used most for the following tasks:

- Data Validation
- Automation
- Process enhancement and enforcement

You'll want to configure commands for the controls you add to the ribbon command. There are three items that the ribbon command delineates:

- Enabling rules
- Displaying rules
- Actions

When to Use Form Programming

Microsoft Dynamics CRM provides many options to control business processes. Form programming is one option. The primary advantage of form programming is that it is immediate. It does not require data to be submitted to the server and, therefore, provides the best performance for many scenarios. Because it allows for human interaction, it is also the most flexible option.

Tasks frequently performed by using form programming include the following:

- Data validation
- Automation
- Process enhancement and enforcement

Factors to Consider When You Use Form Programming

The logic applied in form programming can only be executed in the browser of someone interacting within the Microsoft Dynamics CRM application entity form. Because of this, you should not rely solely on form programming to initiate or enforce processes related to your data. Records can be created or updated directly through the Web service APIs or through workflows outside the context of the form. Form programming should complement business logic applied by using plugins and workflows so that all records created or updated in any manner will comply with the same processes.

While form programming provides capabilities to hide form fields so that users may not be able to view or update some entity data, it does not represent a complete solution for enforcing security requirements. A user can see entity data that is not visible on the form by using a variety of methods, such as **Advanced Find** or sometimes by just printing a form.

Form Events Reference

All Client-Side code is initiated by events. In Microsoft Dynamics CRM you will associate a specific function within a Script library to be executed when specific events occur.

All form events have a user interface you use to specify one or more event handers. Each event handler specifies a single function within a Script library and any parameters that can be passed to the function.

On Load Event

The **Onload** event occurs after the form has loaded. It cannot prevent the window from loading. Use the **OnLoad** event to apply logic about how the form should be displayed, to set properties on fields, and interact with other page elements.

OnSave Event

The **OnSave** event occurs when a user presses certain buttons on the form. The event always occurs, even when the data in the form has not changed. To determine which button was clicked to perform the save, use the getSaveMode method.

Field On Change Event

The OnChange event usually occurs when the data in a form field has changed and focus is lost. Data in the field is validated before and after the OnChange event. All fields support the OnChange event.

Form Scripting Quick Reference

Quick reference of frequently used form scripting methods based on tasks you perform with them.

The Xrm.Page object provides a hierarchy of objects that can be used to interact with Microsoft Dynamics CRM 2011 forms.

Important

Microsoft Dynamics CRM 4.0 used the crmForm object to provide access to form fields. The crmForm is deprecated in Microsoft Dynamics CRM 2011. Scripts using crmForm will continue to work in Microsoft Dynamics CRM 2011 to support backward compatibility, but certain Microsoft Dynamics CRM 2011 features, such as the ability to navigate to different forms for an entity, the ability to access a field displayed more than one time on a form, or the ability to hide form elements cannot be achieved using crmForm.

Xrm.Page

The Xrm.Page object serves as a namespace object to consolidate three properties on the form:

- **Xrm.Page.context Xrm.Page.context** provides methods to retrieve information specific to an organization, a user, or parameters that were passed to the form in a query string.
- **Xrm.Page.data.entity Xrm.Page.data** provides an entity object that provides collections and methods to manage data within the entity form.
- **Xrm.Page.ui Xrm.Page.ui** provides collections and methods to manage the user interface of the form.

Two shortcut methods are provided to get direct access to commonly used controls.

- **Xrm.Page.getAttribute** Provides direct access to the Xrm.Page.data.entity.attributes.get method.
- **Xrm.Page.getControl** Provides direct access to the Xrm.Page.ui.controls.get method.

Use the Xrm.Page Object Model

Microsoft Dynamics CRM 2011 introduces a new object model for form programming. This new object model provides the following additional capabilities:

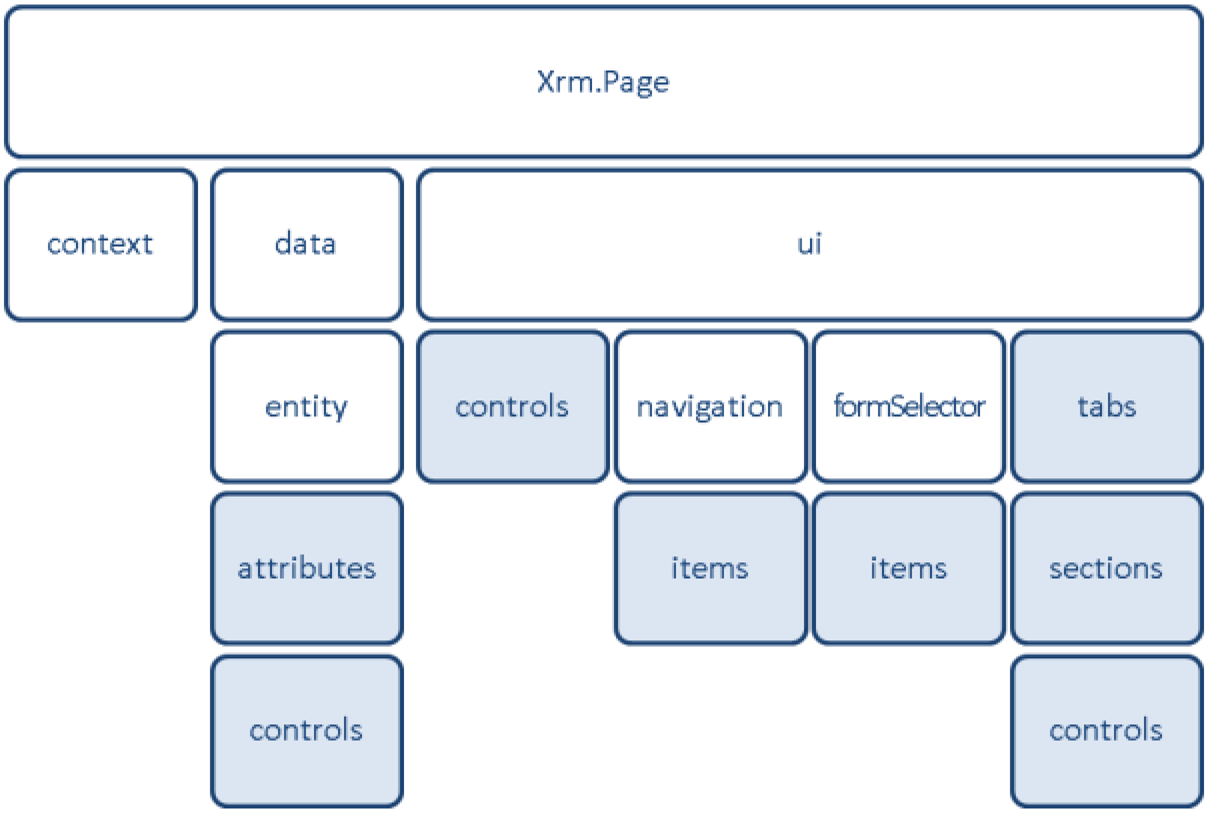
- Show and hide user interface elements.
- Support multiple controls per attribute.
- Support multiple forms per entity.
- Manipulate form navigation items.

Xrm.Page Object Hierarchy

Xrm.Page provides a namespace container for three objects described in the following table:

Object	Description
context	Provides methods to retrieve information specific to an organization, a user, or parameters that were passed to the form in a query string.
data	Provides access to the entity data. The data object does not have any methods; it provides namespace separation to the entity data to provide for expansion in future versions of Microsoft Dynamics CRM.
ui	Contains methods to retrieve information about the user interface, in addition to collections for several sub components of the form.

Microsoft Dynamics CRM uses the XRM Page Hierarchy JavaScript model as displayed below.



Prior to the introduction of this model in Dynamics CRM 2011, Microsoft Dynamics CRM 4.0 used the crmForm object to provide access to form fields. The crmForm object is deprecated in Microsoft Dynamics CRM 2011. Scripts that use the crmForm object will continue to work in Microsoft Dynamics CRM 2011 to support backward compatibility, but the new capabilities in Microsoft Dynamics CRM cannot be achieved by using the crmForm object. Please refer to the section of the book on JavaScript best practices for more information on this topic.

The Xrm.Page object serves as a namespace object to consolidate three objects on the form:

- **Xrm.Page.context:** Xrm.Page.context provides methods to retrieve information specific to an organization, a user, or parameters that were passed to the form in a query string.
- **Xrm.Page.data.entity:** Xrm.Page.data provides an entity object that provides collections and methods to manage data within the entity form.
- **Xrm.Page.ui:** Xrm.Page.ui provides collections and methods to manage the user interface of the form.

Collections

Below are the Xrm.Page object model collections.

- **Attributes:** The Xrm.Page.data.entity.attributes Collection provides access to each entity attribute that is available on the form. Only those attributes that correspond to fields added to the form are available.
- **Controls:** Three objects contain a controls collection:
  - **ui.controls:** The Xrm.Page.ui.controls Collection provides access to each control present on the form.
  - **attribute.controls:** Because an attribute may have more than one control on the form, this collection provides access to each of them. This collection will contain only one item unless multiple controls for the attribute are added to the form.
  - **section.controls:** This collection only contains the controls found in the section.
- **Navigation.items:** The Xrm.Page.ui.navigation.items Collection provides access to navigation items that are visible on the left side of the form.
- **FormSelector.items:** When multiple forms are provided for an entity, you can associate each form with security roles. When the security roles associated with a user enable them to see more than one form, the Xrm.Page.ui.formSelector.Xrm.Page.ui.formSelector.items Collection provides access to each form definition available to that user.
- **Tabs:** You can organize each form by using one or more tabs. The Xrm.Page.ui.tabs Collection provides access to each of these tabs.
- **Sections:** You can organize each form tab by using one or more sections. The tab Xrm.Page.ui tab.sections Collection provides access to each of these sections.

Each object possesses several methods to retrieve data, get or set object properties, or perform actions. Generally speaking for majority of use cases a subset of methods utilizing these model hierarchy is utilized. Given below are examples of some common methods that utilize the methods in this model.

Xrm.Page.context

Xrm.Page.context provides methods to retrieve information specific to an organization, a user, or parameters that were passed to the form in a query string. Some of the common methods used are:

- **getOrgUniqueName:** Returns the unique text value of the organizations name.
  - var OrganizationUniqueName = context.getContext().getOrgUniqueName();
- **getClientUrl:** Returns the base server URL. When a user is working offline with Microsoft Dynamics CRM for Microsoft Office Outlook, the URL is to the local Microsoft Dynamics CRM Web services.
- **getUserId:** Returns the GUID value of the SystemUser.id value for the current user.
  - var userGUID = context.getContext().getUserId();
- **getUserRoles:** Returns an array of strings representing the GUID values of each of the security roles that the user is associated with.
  - var currentUserRoles = Xrm.Page.context.getUserRoles();

Xrm.Page.data.entity

The Xrm.Page.data.entity Attribute Methods are frequently used to get and set propertied of various attributes on the form. Few examples of the methods commonly used are:

- Get the value from a CRM field
  - var varMyValue = Xrm.Page.getAttribute(“CRMFieldSchemaName”).getValue() ;
- Set the value of a CRM field
  - Xrm.Page.getAttribute(“po_CRMFieldSchemaName”).setValue(‘My New Value’);
- Get the selected value of an optionset
  - Xrm.Page.getAttribute(“CRMFieldSchemaName”).getSelectedOption().text;
- Set the requirement level
  - Xrm.Page.getAttribute(“CRMFieldSchemaName”).setRequiredLevel(“none”);
  - Xrm.Page.getAttribute(“CRMFieldSchemaName”).setRequiredLevel(“required”);
  - Xrm.Page.getAttribute(“CRMFieldSchemaName”).setRequiredLevel(“recommended”);

Xrm.Page.ui

Xrm.Page.ui contains methods to retrieve information about the user interface as well as collections for several subcomponents of the form. Xrm.Page.ui provides access to the following collections:

- **Xrm.Page.ui.controls Collection:** A collection of all the controls on the page. Given below are examples of common methods used in this collction:
  - Xrm.Page.ui.controls.get("po_assignedtoteam").setVisible(true); // Sets the field “po_assignedtoteam” to visible on the form
  - Xrm.Page.ui.controls.get("po_assignedtoteam").setDisabled(true); // Disables the field “po_assignedtoteam” on the form
- **Xrm.Page.ui.navigation.items Collection:** Xrm.Page.ui.navigation.items does not contain any methods. Provides access to navigation items through the items collection
- **Xrm.Page.ui.formSelector:** A common method used in this collection is the method to get the type of form currently in use as the example below illustrates:
  ```
  function checkFormType() {
  var formType = Xrm.Page.ui.getFormType();
  if (formType == 2) {
  // Code to execute some logic
  }
  }
  ```
- **Xrm.Page.ui.tabs Collection:** A collection of all the tabs on the page. Some of most common examples are to use methods in this collection to hide/unhide tabs and sections within a tab. Here are few examples:

- Xrm.Page.ui.tabs.get("Financial Services").setVisible(false); // Hides the "Financial Service" tab
- Xrm.Page.ui.tabs.get("Territory-Info").sections.get("TerritoryChangeType").setVisible(true); // Sets the section "TerritoryChangeType" to visible

Jscript Reference for Microsoft Dynamics CRM 2011

- **Xrm.Page.data.entity.attributes** – field access and manipulation on the form
- **Xrm.Page.ui.controls** – working with ui controls on the form
- **Xrm.Page.ui.navigation.items** – working with navigation items on the form
- **Xrm.Utility** – a set of useful helper functions

Here are some examples of the most common I have come across:

How to get text field value:

```
var value = Xrm.Page.data.entity.attributes.get("fieldname").getValue();
```

How to get the ID of a lookup field:

```
var id = Xrm.Page.data.entity.attributes.get("fieldname").getValue()[0].id;
```

How to get the ID of the current record:

```
var id = Xrm.Page.data.entity.getId();
```

How to get the Text of a lookup field:

```
var text = Xrm.Page.data.entity.attributes.get("fieldname").getValue()[0].name;
```

How to set the value of a lookup field:

```
function PopulateLookup(fieldname, id, name, type) {
    var lookupData = new Array();
    lookupData[0] = new Object();
    lookupData[0].id = id;
    lookupData[0].name = name;
    lookupData[0].entityType = type;
    Xrm.Page.getAttribute(fieldname).setValue(lookupData);
}
```

How to set value of a Text field:

```
Xrm.Page.data.entity.attributes.get("fieldname").setValue("my value goes here!");
```

Notes: as with all drop-downs, each option has a value and display text and you should also check for null

How to get option set value:

```
var opField = Xrm.Page.data.entity.attributes.get("fieldname");
var value = opField.getValue();
```

How to get option set text:

```
var opField = Xrm.Page.data.entity.attributes.get("fieldname");
var text = opField.getText();
```

How to select option in Option Set field:

```
Xrm.Page.data.entity.attributes.get("fieldname").setValue(2);
```

How to get values from a Data field:

```
var dateField = Xrm.Page.data.entity.attributes.get("fieldname").getValue();
if (dateField != null) {
        var day = dateField.getDate();
        var month = (dateField.getMonth() + 1); // increment by 1 as Jan starts at 0
        var year = dateField.getFullYear();
}
```

How to set a Date field to Todays date:

```
Xrm.Page.data.entity.attributes.get("fieldname").setValue(new Date());
```

How to Hide and Show a field:

```
var field = Xrm.Page.ui.controls.get("fieldname");
field.setVisible(false);  // hide field
field.setVisible(true);   // show field
```

How to Disabled and Enabled a field:

```
var field = Xrm.Page.ui.controls.get("fieldname");
field.setDisabled(true);   // disable field
field.setDisabled(false); // enable field
```

How to Hide and Show a Navigation Item:

```
var navitem = Xrm.Page.ui.navigation.items.get("navitemname");
navitem.setVisible(false);  // hide
navitem.setVisible(true);   // show
```

How to Hide and Show a Tab:

```
var tab = Xrm.Page.ui.tabs.get("tabname");
tab.setVisible(false);  // hide
tab.setVisible(true);   // show
```

How to Save, Save and Close, and Close a Form:

```
Xrm.Page.data.entity.save();                        // Save
Xrm.Page.data.entity.save("saveandclose");      // Save and Close
Xrm.Page.ui.close();                                  // Close
```

How to Identify Form Mode:

```
var formMode = Xrm.Page.ui.getFormType();
switch(formMode)
{
case 1: // Create
break;
case 2: // Update
break;
case 3: // Read Only
break;
case 4: // Disabled
break;
}
```

How to get the ID of the current user:

```
var id = Xrm.Page.context.getUserId();
```

How to get all the security roles for the current user:

```
var roles = Xrm.Page.context.getUserRoles();
```

How to open an existing CRM record in a new window:

```
Xrm.Utility.openEntityForm("account", guid);
```

Note: Just replace the first parameter with the entity logical name and the second parameter with the record guid.

How to open a new CRM record window:

```
Xrm.Utility.openEntityForm("account");
```

How to refresh a Sub-Grid:

```
Xrm.Page.ui.controls.get("gridname").refresh();
```

How to set Requirement Level of field using JavaScript:

Xrm.Page.getAttribute("fieldname").setRequiredLevel("none"); // No Constraint
Xrm.Page.getAttribute("fieldname").setRequiredLevel("required"); // Business Required
Xrm.Page.getAttribute("fieldname").setRequiredLevel("recommended"); // Business Recommended

How to get field Requirement Level:

Xrm.Page.getAttribute("fieldname").getRequiredLevel();

How to add an OnChange event handler to a field:

Xrm.Page.getAttribute("fieldname").addOnChange(functionName);

How to identify if a field value has changed:

Xrm.Page.getAttribute("fieldname").getIsDirty();
Note: this returns a Boolean value true/false