Snippet	Comments
<tbody data-bind="foreach: oproducts">	Loop through oproducts collection.
<span data-bind="text: ProductId.Name"></span><a href="#" data-bind="click: $root.lookupProduct">Lookup</a>	Using a “text” binding makes the product field is read only. The link opens up the standard product look up.
<span data-bind="text: UoMId.Name"></span><a href="#" data-bind="click: $root.lookupUOM ">Lookup</a>	Unit of measure appears in the list because it is a required field for the opportunity product.
<select data-bind="options: $root.availableSchedule, optionsText: 'ProjectedScheduleText', value: new_projectedschedule.Value, optionsValue: 'ProjectedScheduleValue', optionsCaption: 'Choose...'"></select>	The “options” binding is exclusive to the html ‘select’ tag. “availableSchedule” contains a collection of all options relating to this field’s option set values. The “value” property indicates what the currently selected value should be. This is where the current record’s selection is bound to the control. The “optionsValue” indicates what field in the view model to bind to.
<input data-bind="value: Quantity"/>	The “value” bind property makes this field editable.
<a href="#" data-bind="click: $root.removeOppProd">Delete</a>	The “Delete” button is bound to the view model’s “removeOppProd” method.
<button data-bind="click: addOppProd">Add Opportunity Product</button>	The “Add” button is bound to the view model’s “addOppProd” method.
<button data-bind="click: save">Save</button>	The “Save” button is bound to view model’s “save” method.
var OpportunityProductModel = function () {	Is the start of the page view model definition

var pageViewModel = new OpportunityProductModel();
 	
ko.applyBindings(pageViewModel);
	Activates knockout and associates the page view model OpportunityProductModel

self.RetrieveOptionsetData = function () {	This is the main entry point function, mainly because I have to guarantee that the option set meta-data is retrieved before any binding to the HTML happens. Originally, I was performing this step independent of the retrieving of existing opportunity products. Sometime, the select tag did not properly bind. When it did not work, the select options would populate properly. However, the currently selected value would not properly be assigned. Consequently, this function is also responsible for retrieving the opportunity product data.

function OpportunityIdConstructor() {
  this.Id = "";
  this.LogicalName = "opportunity";
  this.Name = "";
  this.__metadata = new __metadataConstructor2(null, "Microsoft.Crm.Sdk.Data.Services.EntityReference");
    };
	These constructors are necessary in order to set up the data for subsequent updating back into CRM. To understand more clearly, use the developer tools (IE F12) to analyze the JSON structure.