Enriching SAP MII (SAP Manufacturing Integration and Intelligence) UIs Using Dojo

Applies to:

SAP MII. For more information, visit the Manufacturing homepage.

Summary

Very often we face a challenge to design MII UIs with different HTML controls, maintain appropriate layout and for these we need to write lot of HTML and Javascript code. In this article I have tried to show how we can use Dojo toolkit to minimize our effort in writing Javascript and HTMLs to deliver powerful, good-looking, professional-quality MII Application Uis.

Author: Avijit Dhar

Company: IBM India Pvt. Ltd.

Created on: 12 October 2009

Author Bio

Avijit Dhar is currently working in IBM India as an SAP MII Consultant and also involved in the SOA based Composite Application development.

Table of Contents

Introduction

Very often we face a challenge to design MII UIs with different HTML controls, maintain appropriate layout and for these we need to write a lot of HTML and Javascript code. As we always expect that our MII UIs will be more usable. This might manifest itself in a variety of ways. The UI should be faster. It should be better looking. It should be easier to operate by the user. It should help the user enter the required information properly, and the page should be easier to navigate and to achieve all these we expect to write a lot less code than if we were developing the functionality by writing it all ourselves. Less code means less opportunity for error.

How do we make these gains in usability? Fortunately, we have an answer to this and that is Dojo. Now delivering powerful, good-looking, professional-quality web-based applications is far easier than ever before. You are no longer on your own, required to write every last bit of code. It's thanks to things like Dojo that I can say that.

Dojo is a set of tools that helps you build better browser-based applications. Dojo is built mostly using client-side JavaScript and today it is revolutionizing web development, and gaining momentum fast. It packs the standard JavaScript library we have always wanted, the collection of standard UI Widgets which can easily replace the customized HTML controls and CSS layout implemented again and again in different projects. Performance can be improved either by making things run faster or by making things appear to run faster. Data validation can be improved by bringing the validation of data closer to the entry of data. Can Dojo really deliver? Let's find out.

Plan for Enhancing a MII UI Input Form

We begin by selecting a page from an MII application that will be the target for our Dojo enhancements (see Figure 1.1). This page comes from an MII Application and allows a user to enter Goods Issue Details at Shop Floor.

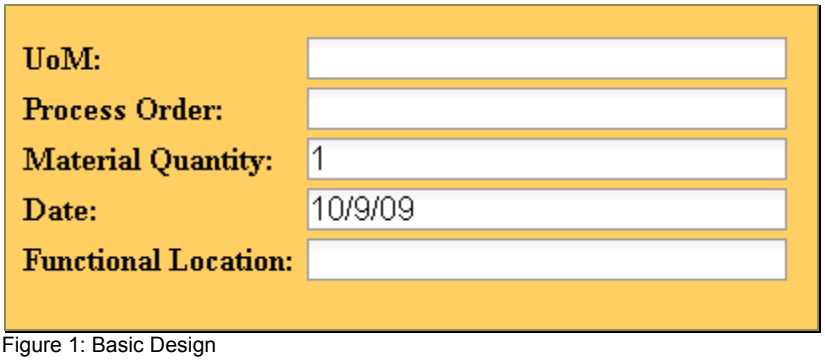


Figure 1: Basic Design

This page has a very basic design—we start with this minimal design to keep the examples as simple as possible. Let's walk through each of the fields on this form and discuss the usability problems. A discussion of how Dojo can solve these problems then follows.

- The first data entry field is used to hold the **UoM** (Unit of Measurement). UoM is a very important info for any Goods Issue so —the field must be required. Are there any other validations? Not only do we want to get the data, but also we'd like it to be in a consistent format. Possibly the data should be stored in all capital letters or the data is not in all capitals – let's choose the latter. And as the **UoM** is written in short form (for example. may be due to lack of space) so if we can display a message next to the field to instruct the user on what kind of data can be entered into the field. The prompt message displays while the field is in focus.
- Next we have a numeric field call Process Order an important input data and require numeric entry only. Non numeric should not be allowed. We can also set an additional check like the entry should be a 12 digit numeric value.
- For the Material quantity field let's set validations like entered value should be numeric, negative quantity should not be allowed; value should be within a range etc.

- Date field always requires a Date Picker for which we need to write a hell lot of Javascript, also require extra validations like Date Format.
- For a field like Functional Location we need to set a data format as well as user entered value should have Hyphens (-).

After studying the above input form we can see that almost all the fields require some input validations, restrictions etc. Now it is a decision point how easily we implement all these validations? We can use JavaScript. Given the power of JavaScript, the sky is the limit in terms of types of validations we can perform. We can trigger a JavaScript function to run after the user enters a field, and that function can check to see if data is entered, check for a minimum or maximum length, or even perform sophisticated pattern matching using regular expressions.

Problem solved, correct? Not quite. The problem with depending on JavaScript as our validation technique is that we have to write lots of code to implement the checks. JavaScript code is required to perform the validation. Other JavaScript code tells the validation when to run. And even more JavaScript code is needed to display the error messages back to the user. Code, code, and more code. Suddenly, this approach doesn't seem as desirable anymore.

But this is where Dojo can come to the rescue. In this part of the tutorial, we explore how Dojo can help us in providing basic client-side validations, change UI Layout very easily. In other words, we'll be able to turn on validation by using simple HTML markup, but we'll let Dojo provide the complex JavaScript code automatically. Let's get started.

Including Dojo in the Form

Validate the UoM Field

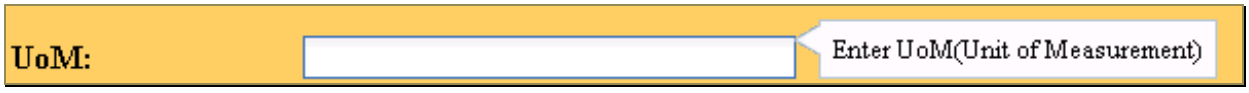
Let's look at the "UoM" field first. As discussed in the previous section we will some validation for this field. We turn on validation by using special attribute values in the HTML markup for these fields. The following code will add validation very easily.

```
<label for="UoM">UoM:</label>
<input type="text" id="UoM" size="20"
   dojoType="dijit.form.ValidationTextBox"
   required="true"
regExp="[\w]+"
   propercase="true"
   promptMessage="Enter UoM(Unit of Measurement)"
   invalidMessage= "Invalid UoM. Re-enter"
   trim="true/>
```

To summarize what has happened: All we've done is add some new attributes to the <input> tag for the field. Each of the new attributes affects the validation in some way. Notice the following line of code from the preceding example:

```
dojoType="dijit.form.ValidationTextBox"
```

This attribute is not a standard HTML <input> tag attribute. Remember the code we needed to include the parser? It is shown here: *dojo.require("dojo.parser");* The parser reads through the HTML and looks for any tag that contains dojoType as one of its attributes. Then the magic happens. The parser replaces the element with the **Dojo widget** specified by dojoType. In this case, the widget dijit.form.ValidationTextBox is substituted for the Document Object Model (DOM) element created from the <input> tag. Notice that the name of the widget specified as the value for the dojoType attribute is the same as the argument for the dojo.require call. Now let's run the page and see how it behaves. Because this is the first field on the page, the field gets focus, and the cursor immediately is placed on the input area for the UoM field.



Notice that we get a message box that says **"Enter UoM(Unit of Measurement)"** Dojo calls this a Tool Tip, and it has dynamic behavior. It is only displayed when the field has focus (the cursor is in the field), and once the field loses focus, the message disappears.

Try entering different values in the field and press <tab> to leave the field. For example, enter "kG" and watch it be transformed into "Kg" with leading and trailing spaces removed and the first letter of the name capitalized. In case if you leave the field blank it will show a warning alert beside the field input(shown in Fig 3) box as the Dojo widget attribute called required="true" attribute setting tells Dojo that this field must be entered. The alert shown above will remain active until user enters appropriate and properly formatted value.

The regular expression syntax *regExp="[\w]+"* removes white spaces entered as wrong input. The regular expression also validates any white space even in the text input and displays the **invalidMessage** string.

The setting for propercase tells Dojo to make sure that the first letter is capitalized and subsequent letters are in lowercase.

Validate the Process Order Field

The Process Order Field accepts numeric input. So we need to use the following code snippet to create a numeric input field.

```
<input id="ProcessOrder" type="text"
        dojoType="dijit.form.NumberTextBox"
        name= "processOrder"
        promptMessage= "Enter a Process Order"
        required= "true"
        invalidMessage= "Please enter a valid numeric Process Order" />
```

In this case we have used `dojoType="dijit.form.NumberTextBox".` This is also set as a required field and if user enters non numeric value the field will display the invalidMessage as an alert.

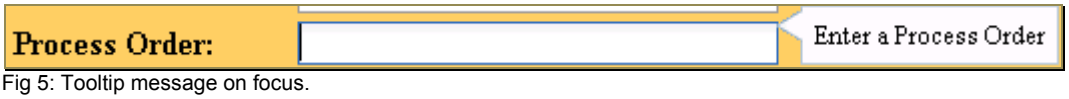


Fig 5: Tooltip message on focus.

Fig 6: alert message on invalid entry

Validate the Quantity Order Field.

The Quantity field also accepts numeric input similar to Process Order and we will also see how can add an extra set of validation as discussed previously.

```
<input id="MaterialQuantity" type="text"
        dojoType="dijit.form.NumberTextBox"
        name= "elevation"
        value="1"
        constraints="{min:1,max:20000,places:0}"
        promptMessage= "Enter a valid Quantity between 1 and 20000"
        required= "true"
        invalidMessage= "Invalid Quantity. Re-enter" />
```

This is the object that contains properties used to validate the data for any widget. Here using constraint we have set the min and max range of Quantity that user can enter. If user tries to enter beyond range the invalid message will be displayed.

Easier Date Entry

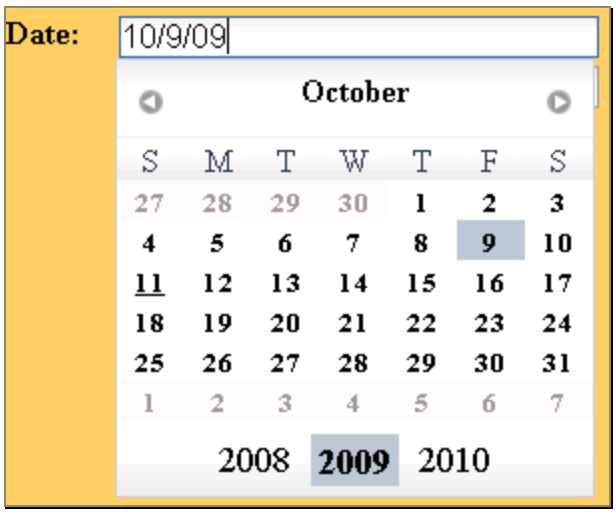
Custom validation routines for validating dates and times are another implementation detail that just about any web developer who has been around a while has had to produce at some point or another. Although dates and times have well-defined formats that are quite universal, the generic HTML INPUT element offers no support, and the load is pushed off to JavaScript for validation and custom formatting. Fortunately, Dijit makes picking dates and times just as easy as it should be. With Dijit, you use **dijit.form.DateTextBox** to turn any textbox into a widget with a calendar. First, add it to the header as shown here:

```
dojo.require("dijit.form.DateTextBox" );
```

Then, add it to the textbox:

```
<input type="text" name="Date" value="2009-10-09"
       dojoType="dijit.form.DateTextBox"
       required="true" />
```

That's it! No additional effort is required. The DateTextBox in Figure automatically pops up a beautiful little calendar for picking a date when the cursor enters the INPUT element,



Validate the Functional Location Field

As discussed we added some extra level of validation for this field. The input data format should be like ***(NNNN-NNN-NN-NN)***. If we can come up with a regular expression to test this format, then we can implement it very easily. On gaining the focus the field displays a prompt alert with the appropriate message.


```
     <input type="text" id="FunctionalLocation" size="30"
dojoType="dijit.form.ValidationTextBox"
trim="true"
required="true"
regExp="\d{4}[\-]\d{3}[\-]\d{2}[\-]\d{2}"
promptMessage="Enter Functional Location (NNNN-NNN-NN-NN) following this format"
invalidMessage="Invalid zip code (NNNN-NNN-NN-NN)."
     />
```

Wrapping It Up

Well, that was a rush! Here's the form source code we've built bit by bit. You can save this code as an .irpt file and make it run very easily after preparing the dojo environment.

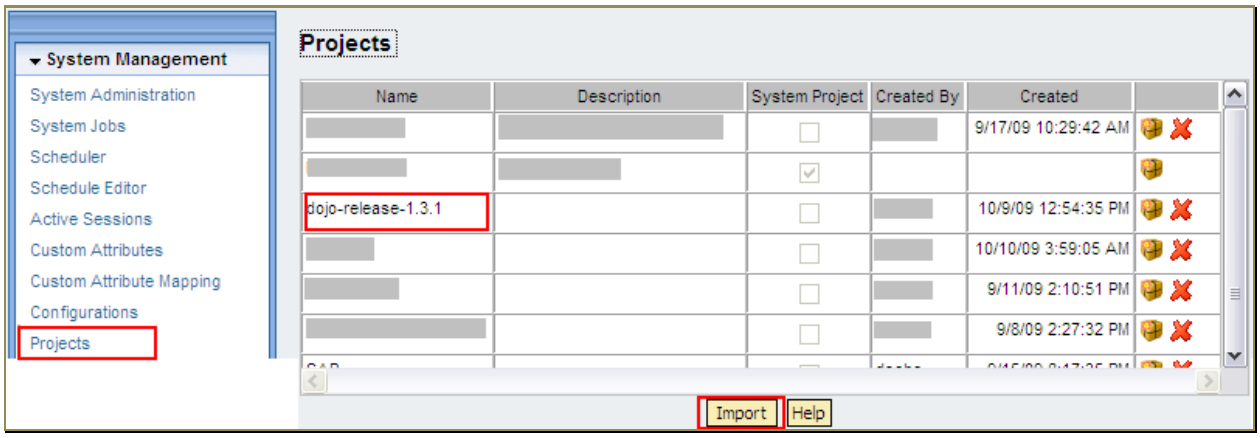
```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
           "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title> Demo Input Form</title>
    <style type="text/css">
        @import "http://o.aolcdn.com/dojo/1.0/dijit/themes/tundra/tundra.css";
        @import "http://o.aolcdn.com/dojo/1.0/dojo/resources/dojo.css"
    </style>
    <script type="text/javascript" src="http://o.aolcdn.com/dojo/1.0/dojo/dojo.xd.js"
        djConfig="parseOnLoad: true"></script>
    <script type="text/javascript">
       dojo.require("dojo.parser");
       dojo.require("dijit.form.Form");
       dojo.require("dijit.form.NumberTextBox");
       dojo.require("dijit.form.ValidationTextBox");
       dojo.require("dijit.form.DateTextBox");
    </script>
</head>
<body class="tundra" bgcolor="#FFCC60">
<div dojoType="dijit.form.Form" id="testForm" jsId=" testForm "
encType="multipart/form-data" action="" method="">
<table>
<tr>
<td><label for="UoM">UoM:</label></td>
<td><input type="text" id="UoM" size="20"
   dojoType="dijit.form.ValidationTextBox"
   required="true"
   regExp="[\w]+"
   propercase="true"
   promptMessage="Enter UoM(Unit of Measurement)"
               invalidMessage= "Invalid UoM. Re-enter"
```

```
   trim="true"/>
</td>
<tr><td><label for="ProcessOrder">Process Order:</label></td>
<td>
        <input id="ProcessOrder" type="text"
                dojoType="dijit.form.NumberTextBox"
                name= "elevation"
                promptMessage= "Enter a Process Order"
                required= "true"
                invalidMessage= "Please enter a valid numeric Process Order" />
</td></tr>
<tr><td><label for="MaterialQuantity">Material Quantity:</label></td>
<td>
        <input id="MaterialQuantity" type="text"
                dojoType="dijit.form.NumberTextBox"
                name= "elevation"
                value="1"
                constraints="{min:-1,max:20000,places:0}"
                promptMessage= "Enter a valid Quantity between 1 and +20000"
                required= "true"
                invalidMessage= "Invalid Quantity. Re-enter" />
</td></tr>
<tr><td><label for="Date">Date:</label></td>
<td>
        <input type="text" name="Date" value="2009-10-09"
                dojoType="dijit.form.DateTextBox"
                required="true" />
</td></tr>
<tr><td><label for="FunctionalLocation">Functional Location:</label></td>
<td>
<input type="text" id="FunctionalLocation" size="30"
dojoType="dijit.form.ValidationTextBox"
trim="true"
required="true"
regExp="\d{4}[\-]\d{3}[\-]\d{2}[\-]\d{2}"
promptMessage="Enter Functional Location (NNNN-NNN-NN-NN) following this format"
invalidMessage="Invalid zip code (NNNN-NNN-NN-NN)."
/>
</td></tr>
</table>
</div>
</body></html>
```

This form is easier to navigate, is easier for adding data, and is patient but firm about accepting good data. Yet it takes only a few lines of JavaScript and some extra HTML attributes. Dijit is a very powerful thing indeed!

Getting Dojo

As Dojo is a client-side JavaScript toolkit, and its heart lies in some well tuned JavaScript scripts. Technically, Dojo doesn't need a web server. You can install Dojo into any directory, build Dojo-based web applications, and load them all. Downloading the latest official Dojo release is by far the most traditional way to prepare for development. You can find official releases of the toolkit at http://dojotoolkit.org/downloads. Once you have downloaded Dojo, you might initially be surprised that it's all not in one JavaScript file. A quick look at what unpacks reveals that the code base is broken into the same architectural components — Base (dojo/dojo.js), Core (dojo), Dijit (dijit), Dojox(dojox), and Util (util). To use dojo for developing MII UIs you need to import the downloaded dojo toolkit .zip (**dojo-release-1.3.1.zip**) as a project in MII Server. Go to Project Management perspective and import.



Summary

The primary purpose in developing Dojo was to address the inadequacies inherent in JavaScript programming, in other words, to make JavaScript programming easier. Using Dojo we can minimize costs on the programming budget. In MII developers need to put a lot of extra effort and time to write javascript related to UI validation, layout change, achieving different functionality etc. Dojo can be used by anyone who develops Web pages and already writing complex JavaScript. They are already encountering all the problems that Dojo was meant to solve. By adopting Dojo, they can increase their productivity and reduce their level of frustration. And the good news is that you can adopt Dojo as quickly or as slowly as they wish. Dojo won't step on any of your existing code because of its use of a separate namespace. So you can begin adding Dojo to existing pages for new features and Dojo makes it easy to get started.

Related Content

To download and know more in detail about dojo please use the following links.

http://dojotoolkit.org/downloadsReference 2

http://dojocampus.org/

http://www.dojotoolkit.org/book/dojo-book-1-0

For more information, visit the Manufacturing homepage.

Disclaimer and Liability Notice

This document may discuss sample coding or other information that does not include SAP official interfaces and therefore is not supported by SAP. Changes made based on this information are not supported and can be overwritten during an upgrade.

SAP will not be held liable for any damages caused by using or misusing the information, code or methods suggested in this document, and anyone using these methods does so at his/her own risk.

SAP offers no guarantees and assumes no responsibility or liability of any type with respect to the content of this technical article or code sample, including any liability resulting from incompatibility between the content within this document and the materials and services offered by SAP. You agree that you will not hold, or seek to hold, SAP responsible or liable with respect to the content of this document.