

SAS Macro Technique for Embedding and Using Metadata in Web Pages

Paul Gilbert, Troy A. Ruth, Gregory T. Weber
DataCeutics, Inc., Pottstown, PA

ABSTRACT

This paper will present a technique to develop dynamic user interfaces within a SAS/Intrnet-based clinical reporting application. Our goal was to develop a sophisticated interface while keeping to a single user input screen. In a typical SAS/Intrnet application, user input on one screen is used to dynamically generate further selections on a subsequent screen. For simple applications this can be effective, but can become cumbersome when the application requires multiple screens. Even just two screens can make tasks such as storing and recalling a user's previous selections difficult, as well as make the interface less intuitive. We were able to accomplish our goal by loading the required data into a single page as metadata, making it easily accessible to drive the user selections. SAS macros were used to generate the required metadata and embedded JavaScript. In this paper, we will discuss and share the method that we used.

INTRODUCTION

The technique presented in this paper is a small part of a clinical reporting application; however, it is a major factor in determining how the user interacts with the system. The goal was to make the user interface as simple and intuitive as possible. In this case, the specific goal was to present all user input required to produce a report on a single web page. The support of this requirement called for the inclusion of client-side scripting to manage the dynamic update of the user selections. For this application, we chose to use JavaScript for our client-side scripting needs. Another major requirement of the system was that it be easily extensible, with the ability to quickly develop and introduce new user input screens to meet ongoing clinical reporting requirements. To this end, a SAS macro was developed to generate the required metadata and embedded JavaScript. A library of JavaScript functions was built to work with the metadata generated by the macro. In this paper, we will review the SAS macro and the use of the accompanying JavaScript functions.

PROBLEM

The inclusion of multi-level drilldown report selections in the clinical reporting user interface can be implemented by presenting the most general category selection on a screen and then using user input to dynamically generate more specific category selections on subsequent screens. As the drilldown relationships become more complex, the user interface suffers, as the presentation becomes less intuitive and the task of storing and recalling a user's selections across pages becomes difficult.

SOLUTION

Our solution is to present drilldown selections on a single screen and use client-side scripting to dynamically display the dependent selection menus as the general selections are made. In this way, related selections may remain together and there is no need to store and retrieve the general to specific selections across pages. To illustrate the use of the technique, we created a simple set of multi-level selections using the hierarchy of Compound, Indication and Study. *Table 1* below shows the contrived data for imaginary "Compound 1".

TABLE 1 – Metadata Used

Compound		Indication		Study	
Display	Value	Display	Value	Display	Value
Compound 1	1	Indication 1	1	Study 1	1
				Study 2	2
				Study 3	3
		Indication 2	2	Study 1	4
				Study 2	5
				Study 3	6
				Study 4	7
		Indication 3	3	Study 5	8
				Study 1	9

In order to support the single-screen drilldown selections and dynamically generate the refined selections, the web document must be loaded with the above metadata.

1. LOADING THE METADATA

When the user input screen is first created, the metadata is loaded into the web document, including all the data required for all possible user selections. Though just one of many possible ways to accomplish this task, we chose to use JavaScript arrays as the data structure for our metadata. The example below demonstrates the JavaScript used to generate the metadata arrays. Only the metadata for Compound 1 from *Table 1* is shown, although multiple compounds can be supported simultaneously in the same structure. The string "|," is the delimiter separating the elements to be split into the JavaScript array.

```
var indication_lookup = "1|,|1|,|1|,|..." .split("|,");
var indication_value = "1|,|2|,|3|,|..." .split("|,");
var indication_display = "Indication 1|,|Indication 2|,|Indication 3|,|..." .split("|,");

var study_lookup = "1|,|1|,|1|,|2|,|2|,|2|,|2|,|2|,|3|,|3|,|..." .split("|,");
var study_value = "1|,|2|,|3|,|4|,|5|,|6|,|7|,|8|,|9|,|10|,|..." .split("|,");
var study_display = "Study 1|,|Study 2|,|Study 3|,|Study 1|,|Study 2|,|Study 3|,|Study 4|,|Study 5|,|Study 1|,|Study 2|,|..." .split("|,");
```

The compound level data is not generated as metadata in the above arrays, because compound is the starting point of our general to specific selections, so the compound data is defined directly in the top level selection control. This will be apparent later when we look at the HTML source.

The metadata consists of three components: a "look-up" array, a "value" array, and a "display" array. The "look-up" array is used to determine which items are related to the prior selections in the general to specific selection hierarchy. For example, the selection of Compound 1 will return the value 1. This value is compared against the elements in the indication_lookup array. For each element in the look-up array that matches the selected value, an option will be added to the next hierarchical selection control. The corresponding element in the "value" array is used to set the value property of the added option and the same element in the "display" array is used to supply the display text for the new option.

A SAS macro was created to generate the JavaScript array metadata within the page. The SAS macro serves both to allow many different sources of data to be consistently represented as metadata and to insulate SAS programmers who may not be familiar with JavaScript from the specifics of JavaScript syntax. The source code is shown below.

```
%macro generate_metadata(inData=,
                        metaLabel=,
                        metaLookup=,
                        metaValue=,
                        metaDisplay=,
                        metaDelim=%str(|,))

proc sort data=&inData(keep=&metaLookup &metaValue &metaDisplay) nodupkey
          out=generate_metadata_work;
          by &metaLookup &metaValue &metaDisplay;
run;

data _null_;
  set generate_metadata_work end=last;
  file _webout;

  length lookup value display $32767;
  retain lookup value display;

  lookup=trim(lookup) || trim(&metaLookup);
  value=trim(value) || trim(&metaValue);
  display=trim(display) || trim(&metaDisplay);

  if last then do;
    put "var &metaLabel_Lookup = " ' "' lookup ' "' .split(" ' " &metaDelim ' "' );";
    put "var &metaLabel_Value = " ' "' value ' "' .split(" ' " &metaDelim ' "' );";
    put "var &metaLabel_Display = " ' "' display ' "' .split(" ' " &metaDelim ' "' );";
    put "var &metaLabel = [&metaLabel_Lookup, &metaLabel_Value, &metaLabel_Display];";
  end;
  else do;
    lookup=trim(lookup) || trim(&metaDelim);
    value=trim(value) || trim(&metaDelim);
    display=trim(value) || trim(&metaDelim);
  end;
end;
```

```

    end;
run;

proc datasets;
    delete generate_metadata_work;
quit;
%mend generate_metadata;

```

2. ACCESSING THE METADATA

As general to specific selections are made, additional JavaScript is executed to evaluate the metadata and generate the applicable options into the next selection control in the selection hierarchy. The source code of this JavaScript is shown below.

```

function generateSelections(source, destination, arrDisplay, arrValues, arrLookup)
{
    /* Create object references to the source and destination selection controls */
    var oSource=document.all.item(source);
    var oDestination=document.all.item(destination);

    /* Update the destination selection box */
    /* Clear all of the entries in the current selection box */
    oDestination.options.length = 0;

    /* Populate the destination options */
    /* Loop through the metadata arrays, */
    /* Insert the available items as selections if the current source selection matches the
Lookup value */
    for (i=0; i<arrValues.length; i++)
    {
        if (oSource.value==arrLookup[i])
        {
            /* Create a new option object and append it to the destination selection box */
            var oNewOption = document.createElement("option");
            oDestination.options.add(oNewOption);

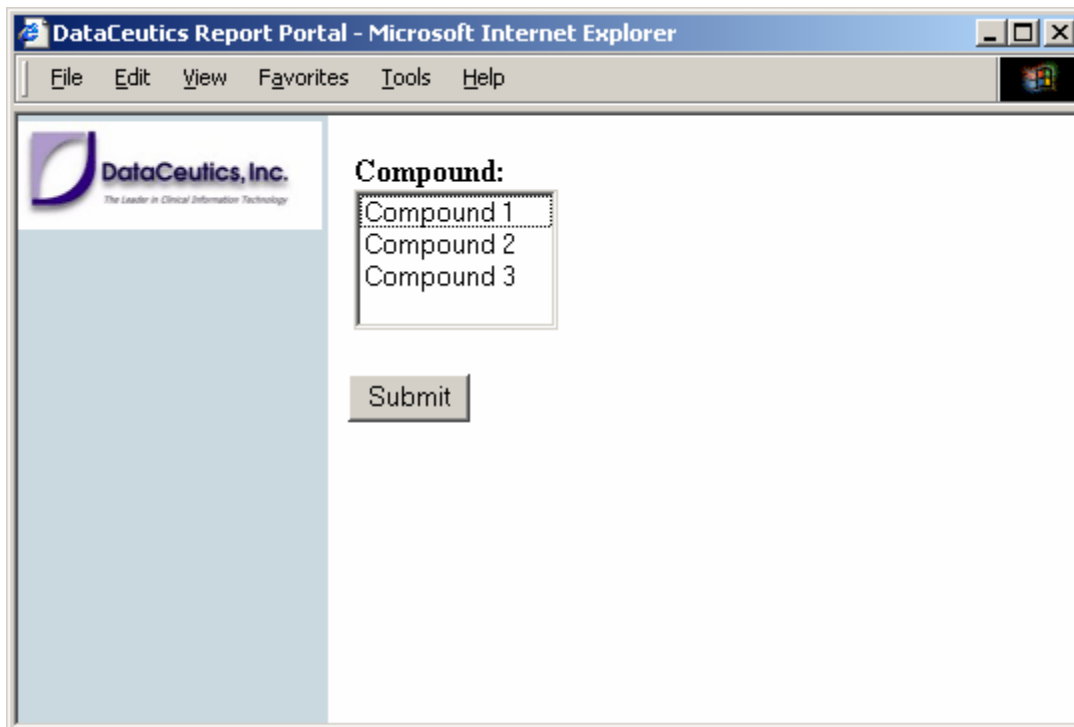
            /* Set the Display text and value of the new option */
            oNewOption.innerHTML = arrDisplay[i];
            oNewOption.value = arrValues[i];
        }
    }
}

```

The above function is intended to be called from the onChange event of selection controls. The most general selection control is completely defined within the web document, as the selections at this level do not depend on the choices of the user. However, as a selection is made at this level, the onChange event is initiated and the above function is executed to dynamically produce the options in the next selection control.

FUNCTIONAL OVERVIEW

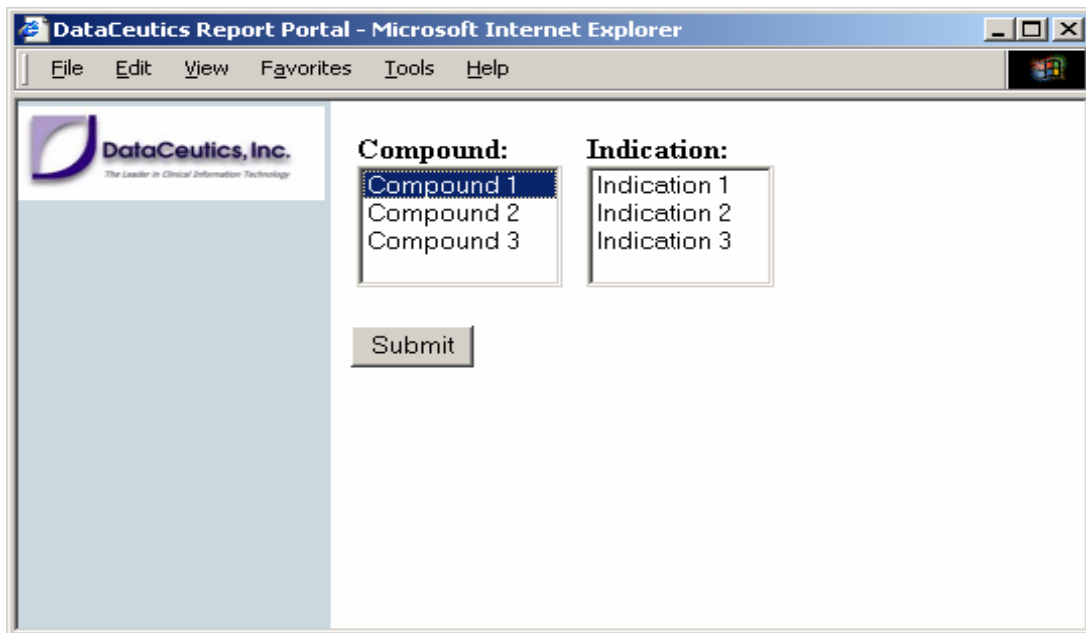
The screenshots below depict a sequence of user selections and the resulting selection objects being populated. Following the screenshots will be the HTML and embedded JavaScript code used to generate the selection controls and link to the next selection in the hierarchy using a call to the generateSelections() JavaScript function.



```
<B>Compound:</B><BR>
<SELECT size=4 name="compoundSelect"

onChange="generateSelections('compoundSelect','indicationSelect',indication_Display,
indication_Value, indication_Lookup);">
  <OPTION value=1>Compound 1</OPTION>
  <OPTION value=2>Compound 2</OPTION>
  <OPTION value=3>Compound 3</OPTION>
</SELECT>
```

As alluded to above, notice that the options for Compound 1 (value=1), Compound 2, and Compound 3 are defined directly in the HTML. As the options for the indication selection control are dependent upon the compound selection, there are no options pre-defined in HTML for indication, as shown below. In the above simplified example, the user will select a compound. This will trigger a lookup to metadata stored within the web page and generate the next selection box, showing the available indications for the selected compound.

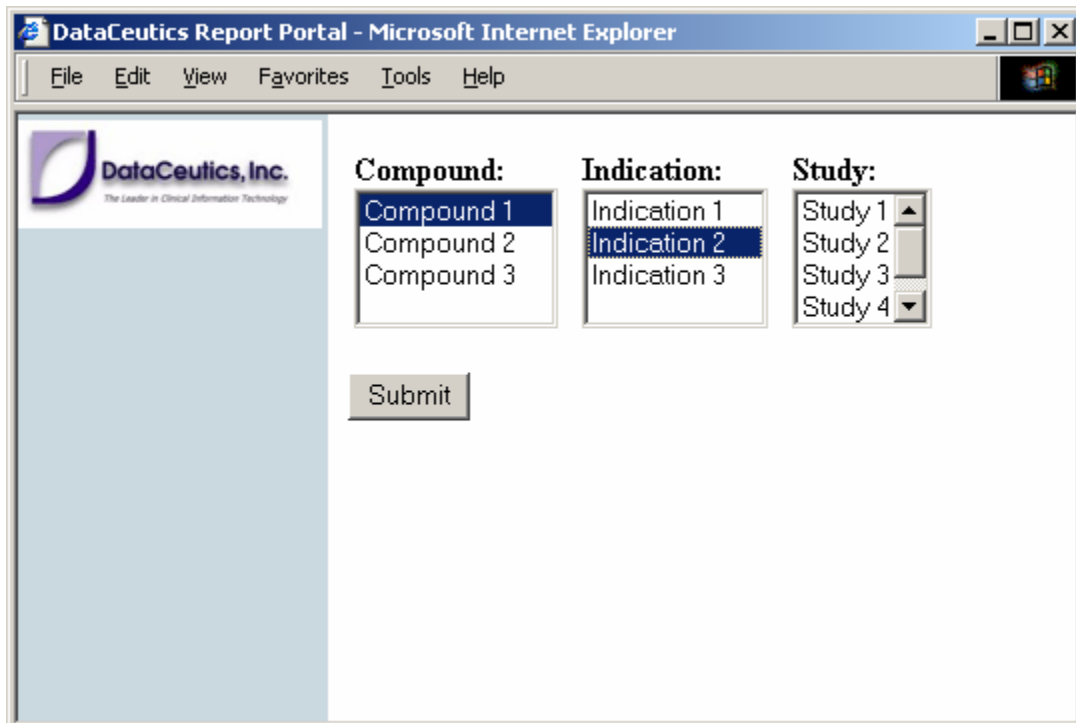


```

<B>Indication:</B><BR>
<SELECT size=4 name="indicationSelect"
    onChange="generateSelections('indicationSelect','studySelect',study_Display,
study_Value, study_Lookup);
    document.getElementById('study').style.display='';">
</SELECT>

```

The screen below shows the result of selecting Indication 2. Only the studies associated with Indication 2 are presented to the user.



```

<B>Study:</B><BR>
<SELECT size=4 name="studySelect">
</SELECT>

```

Since study is the most specific selection category in our example, notice that the HTML to produce the study selection control does not include a call to the generateSelections() JavaScript function.

This example can be made more robust through the introduction of techniques such as using JavaScript to display and hide the hierarchical selections, until their direct ancestors have been selected and support for multiple selections at one or more selection levels. These and other techniques are beyond the scope of this paper.

CONCLUSION

The goal was to make the user interface as simple and intuitive as possible. There are several solutions that we could have used to generate and store the single screen user interfaces. The technique employed was chosen because of its adaptability to the various usage scenarios, as well as its encapsulation in SAS macros which can be called knowledgeably by SAS programmers. The library of SAS macros and JavaScript functions that were developed greatly facilitates the rapid creation of report interface screens. Using this technique we were able to present all user input required to produce a report on a single web page.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors at:

Gregory T. Weber - weberg@dataceutics.com

Troy A Ruth - ruth@dataceutics.com

Paul Gilbert - gilbertp@dataceutics.com

DataCeutics, Inc. (<http://www.dataceutics.com>) is a SAS Alliance Partner and a CDISC member.



| SAS Alliance

SAS and all other SAS Institute, Inc. product or service names are registered trademarks or trademarks of SAS Institute, Inc. in the USA and other countries. © indicates USA registration.