AcroTeX.Net

AcroTeX PDF Blog

Processing Acrobat Forms using JavaScript

External Processing of a Field

Part 7: Choice Fields, the Combo Box

D. P. Story

ttp://www.acrotex.net
ublished: December 14, 2008

Table of Contents

10. Choice Fields: The Combo Box

The *combo box* is very much like the list box (AcroTeX Blog #17), both are lists of text items. The combo box features a drop-down list of items, hence, takes up less real estate than the list box, and, unlike the list box, the combo box can be configured to allow the user to enter custom text. Spell checking can be also be performed when the combo box allows for user text; however, the combo box does not allow for multiple selection.

Below is an example of each, the combo box on the right is *editable*, meaning it allows for the user to enter text.

Enter some text in the combo box on the right. The custom text remains at the top-most most level and is never part of the list, but is retained as the value of the combo box, if the document is saved.

Custom text is *always the current item* in the list, and is discarded if it is not; that is, if you pull the list down and select an item from the list, the custom text is discarded and cannot be recovered unless the user types it in again, making it the current item.

Because the list box and combo box are very similar when the combo box is *not editable*, in this article, we'll concentrate on the editable case. The reader is referred to AcroTeX Blog #17 on list boxes to review the techniques of getting, setting, inserting, and deleting items in a list (or combo) box.

10.1. Getting and Setting the Value of a List Box

To review from AcroTeX Blog #17, each item in a combo box has an export value and an appearance (face, user) value.

- The *export value* of an item in a combo box is not seen by the user. It is the value, or values, of the selected item(s) that are submitted to a server-side script, for example.
- The *appearance (face, user) value* of an item is the text string you see in the list. This is also referred to in various parts of *JavaScript for Acrobat API Reference* as the *user value*.

If the combo box is editable, the export value and the appearance (user) value are the same for text entered by the user.

In addition to these two, there is an *index value* that can be used to get and set the items in a list. The export value and face value of a list item are internally stored in an array (the *options*

array). The index value corresponds to the items position in the array. An index value of 0 is the first item in the list, an index value of 1 is the second item in the list, and so on.

10.2. Getting the Number of Items in a Combo Box

The number of items in a combo box can be obtained from the `Field.numItems` property. For example, the number of items in the combo box above is obtained by pushing button below.

```
var f=this.getField("myCombo");
app.alert("There are "+f.numItems
            +" items in this combo list.");
```

Note that there are 6 items in the list if there is no custom text, and 6 items in the list if the user, that's you, has entered custom text. My comment earlier that the custom text is never part of the list is borne out by the `Field.numItems` property.

10.3. Getting and Setting

We can extract information from a combo box using various Field properties and methods:

- `Field.value`: get and set by export value;
- `Field.currentValueIndices`: get and set by index value;
- `Field.getItemAt`: get either export value or appearance value by index value.

The techniques of using these three properties and methods are the same–with one exceptions discussed below—-as illustrated in the section Getting and Setting in AcroTEX Blog #17. We refer the reader to that section.

The exception just referred to concern an editable combo box. Most notably, when the user enters text into an editable combo box, In the case the combo box is editable, `Field.currentValueIndices` gets a value of -1. This is a signal that the current value of the combo box is a value entered by the user. A script should test for a value of -1, then use `Field.value` to get the user value. `Field.value` can also be used to set a custom value as well.

Here's an example that is a variation on the Example 9.4 of AcroTEX Blog #17.

Example 10.1. Get the current value of an editable combo box. Select a item in the combo box, or enter your own value. Press the Get button to execute the script for processing this combo box.

The code for the push button follows.

```
1  var f = this.getField("myCombo10-1");
2  if ( f.currentValueIndices == -1 )
3      var strMsg = "You entered your own text of \""+f.value
4          +"\", which has an export value of \""+f.value+"\"."
5  else
6      var strMsg = "You selected an item from the list. "
7          +"The face value is \""
8          +f.getItemAt(f.currentValueIndices,false)
9          +"\" and its export value is \""
10         +f.getItemAt(f.currentValueIndices,true)
11         +"\". Nuff said!"
12 app.alert(strMsg);
```

Comments. We test for user input in line (2), if yes, we build a custom message based on `f.value` (the *`Field`*`.value` property), and if no (6)–(11), we build a message based in the `f.currentValueIndices`, using `f.getItemAt` to extract the face value and export value of the item selected. □

10.4. Creating, Inserting, Deleting, and Clearing a List

For combo boxes, use the same techniques as illustrated for list boxes in Section 9.4, in the section with the same title, Creating, Inserting, Deleting, and Clearing a List.

10.5. Changing the Appearance and the Properties of a List Box

As we have done in the previous blogs, we'll survey the General, Appearance, and Option tabs, saving the other tabs (Actions, Format,Validate, and Calculate) for another day.

10.6. The General Tab

Below, see Figure 1, page 6, is a screen shot of the General tab of the List Box Properties, parallel to the dialog box, are the JavaScript properties that correspond to the elements on the General tab of the List Box Properties dialog box. The properties of the General tab, are the same as those of the other fields.

We have pretty well illustrated these properties in the previous blogs, AcroTEX PDF Blogs #13, #14, #15, and #15, so no examples will be presented here. Move on!

10.7. The Appearance Tab

Let's do the same now for the appearance tab, see Figure 2 on page 7.

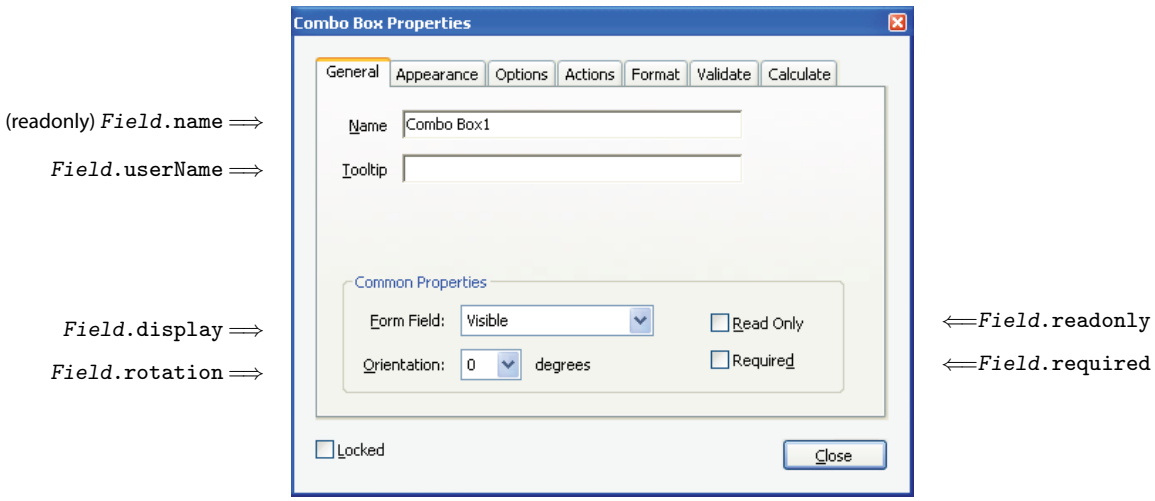

Figure 1: The General Tab of Combo Box Properties

As with the General tab, use these properties, listed in Figure 2, by first acquiring the Field object of the target field, and assigning values to the JavaScript properties. In general, some properties and methods are only available in Acrobat, and not available on Adobe Reader. See the *JavaScript for Acrobat API Reference* for full documentation, pay attention to the quick keys that accompany the properties and methods in the *JavaScript for Acrobat API Reference*.

Border Color: Determines the color of the border; `Field.strokeColor` is its counter-part in JavaScript.

Fill Color: The background color of the combo box.

Font Size: The size of the text to appear in the list. Setting the `Field.textSize=0` is equivalent to selecting Auto from the drop-down list in the user interface.

Font: The font to be used to render the text items in the list.

Thickness: Use `Field.lineWidth` to set the width of the border surrounding the bounding rectangle of the combo box. Possible values are 0 (no boundary), 1 (Thin), 2 (Medium), 3 (Thick).

Line Styles: How the border is rendered, the user interface offers the choices: Solid, Dashed, Beveled, Inset, and Underlined. The corresponding values for `Field.borderStyle` are `border.s`, `border.d`, `border.b`, `border.i`, and `border.u`.

Executing `Field.borderStyle=border.s` sets the Line Style to Solid.

Text Color: This property determines the color of the the text. To set the color of the text to red, execute `Field.textColor=color.red`.

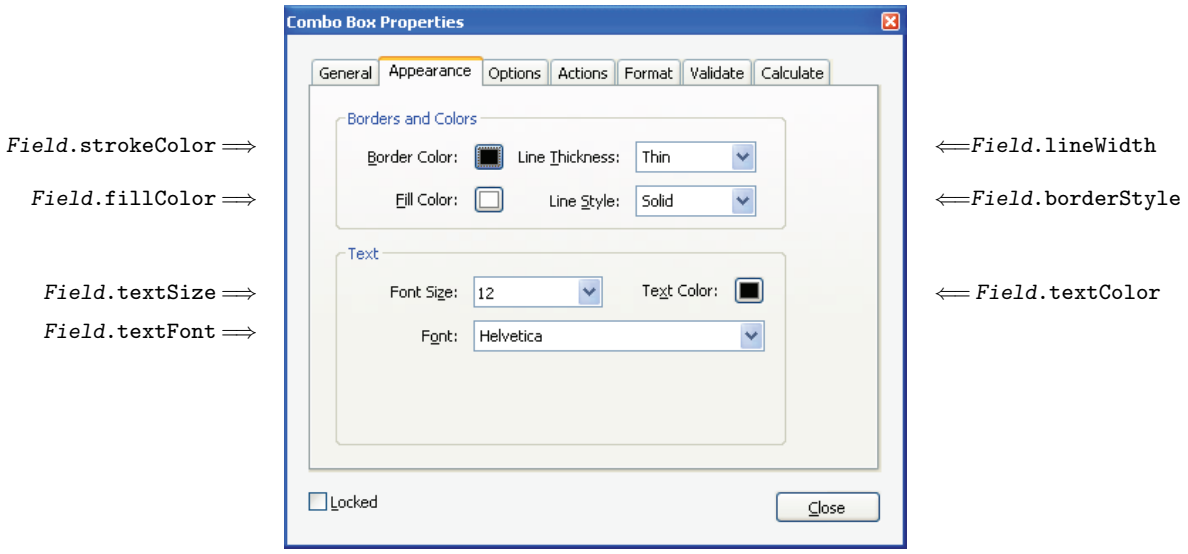

Figure 2: The Appearance Tab of Combo Box Properties

We have pretty well illustrated these properties in the previous blogs, AcroTEX PDF Blogs #13, #14, #15, and #16, so no examples will be presented here.

10.8. The Options Tab

In Figure 3, page 8, we set up the correspondence between each user interface element and its JavaScript counterpart.

Item/Export Value/Item List: These user interface elements can be filled in by using either *Field*.getItemAt, for one item at a time, or *Field*.setItems, for all items at one time. See the examples in the section Getting and Setting of AcroTEX Blog #17.

Sort Items: There is no JavaScript property for checking this box; however, if you want the items in a list sorted, you can sort them using JavaScript. The following script should do the job.

```
var f=this.getField("comboName");
aSorted=new Array();
for (var i=0; i<f.numItems; i++)
    aSorted.push(f.getItemAt(i,false),f.getItemAt(i,true))
aSorted=aSorted.sort();
f.setItems(aSorted);
```

The above code works, but I was a little nervous about it. Originally, I thought that I had to have a sorting function, like the following one.

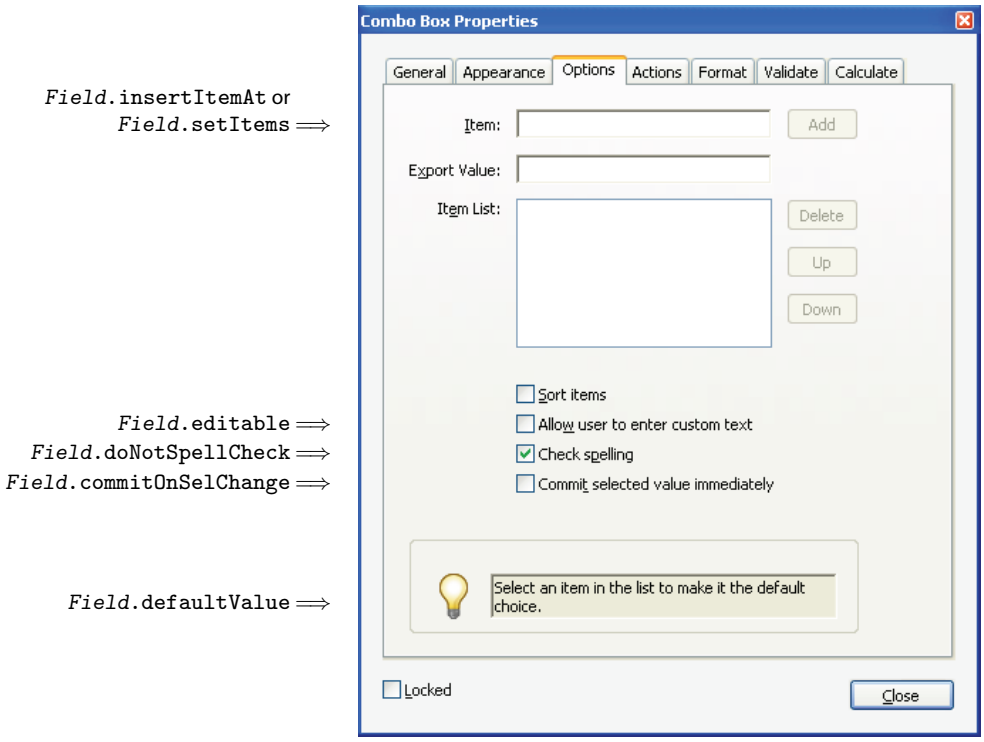

Figure 3: The Options Tab of Combo Box Properties

```
var sortIt = function(a,b){
   if (a[0] < b[0]) return -1;
   if (a[0] > b[0]) return 1;
   if (a[0] == b[0]) return 0;
}
aSorted=aSorted.sort(sortIt);
```

Insert these lines in place of lines (5).

Allow user to enter custom text: Set `Field.editable=true` to allow the user to enter text.

Check spelling: The JavaScript version of this check box is given in the negative. Set `Field.doNotSpellCheck=true` if you *do not want spell checking*. Normally, spell checking is done by default.

Commit selected value immediately: If `Field.commitOnSelChange=true` commits the selection immediately, even though the field has not lost focus. Losing focus is the usual way of committing the choice. The following two list boxes demonstrate the difference between the two modes.

Below is an example that appeared in the Blog 17, but modified for combo boxes.

Example 10.2. This example writes to the JavaScript Debugger Console window. Open the console window, if you are using the puny Adobe Reader, click on console window.

For Selecting Items in the list.

- In the combo box, select one, then another item in the list box (no multiple selections allowed), nothing is written to the window. Now click you mouse outside the list box, the data is committed, and a message is written to the window.
- Now click on the push button, the caption should say **commOnSelChange**. Repeat the exercise above. As you make selections within the list box, they are reported to the console window. You don't have to commit the selection by clicking outside the combo box.

For Enter Custom Text in the Combo Box. There is no difference between modes, select immediately or not. The combo box waits until the data is committed, either by pressing the Enter key, or tabbing out of the combo box, or clicking the mouse outside the combo box. After the data is committed, the value is reported to the console window.

□

Select an item in the list to make it the default choice. Setting `Field.defaultValue` to the *export value* of one of the items makes that item the default item. The default value is the item that is highlighted when the combo box is reset, perhaps using `this.resetForm()`. Acrobat is needed to save any changes and to make them permanent. See AcroTEX PDF Blog 17 for an example of the use of `Field.defaultValue`.

Well, that's pretty much it for now, I simply must get back to my retirement. 𝔇𝔖