



AcroTeX.Net

AcroTeX PDF Blog

Processing Acrobat Forms using JavaScript

External Processing of a Field

Part 5: Button Fields, the Radio Button Field

D. P. Story

Table of Contents

8	Button Fields: The Radio Button Field	3
8.1	Getting and Setting the Value of a Radio Button Field	3
8.2	Checking and Clearing a Radio Button Field	4
8.3	Changing the Appearance and the Properties of the Radio Button Field	5
	• The General Tab	5
	• The Appearance Tab	5
	• The Options Tab	7

8. Button Fields: The Radio Button Field

A button field is an interactive control the user manipulates with the mouse. Button fields do not take keyboard input from the user. There are three types of buttons fields,

1. A *Push button* is a purely interactive control that has not value.
2. A *check box* toggles between two states, on and off.
3. *Radio button fields* contain a set of related buttons that can each be on or off. Normally, when one radio button is on, the others in the same field are off.

In this article, we shall concentrate all our efforts on the *the radio button field*.

8.1. Getting and Setting the Value of a Radio Button Field

A radio button field consists of a series of (usually two or more) widgets, each widget can be placed in an 'on' or 'off' state.¹ In that sense they are similar to a check box. In the user interface, you create a radio button field by creating a series of radio buttons all with the same name (hence they are widgets), but with different button values. The purpose of a radio button field is to allow the user to select one of several mutually exclusive alternatives, as the example that follows illustrates:

What is your age range? 0 years to less than 16 years
 16 years to less than 21 years
 21 years to less than 30 years
 30 years or greater

Normally, in a radio button field only one of the widgets is allowed to be in the 'on' state.

Each widget has, what the user-interface calls, a *button value*. These button values, if known, can be used to turn a particular radio button on or off. We can get and set the button value of a radio button field using our old friend *Field.value*, as the next example illustrates.

Example 8.1. Get and Set button values of the radio button field that appears above.

Get:

The button value tells us which of the buttons is pressed, hence we know the user's age range.

Change the state of the radio button field, and press the Get button again, for fun.

Set:

The field can be cleared by setting *Field.value* to a value other than one of the button values—which is not a recommended method—, or by resetting the field, like so, □

¹For a review of widgets, see aPDFBlog15.

As with the check box, it is easiest to set the radio button field using `Field.checkThisField`, but depending on the particular situation `Field.value` can do it as well. To get the state of the radio button field, using `Field.value` is best because its value be interpreted immediately; for example, in Example 8.1. a field value of "16to21-" tells us that the user is 16 years old but less than 21 years old.

8.2. Checking and Clearing a Radio Button Field

The `Field.checkThisButton` can be used to check or clear a particular button in a radio button field. The syntax for `checkThisButton` is

```
Field.checkThisBox({ nWidget: Integer, bCheckIt: Boolean });
```

For a radio button field, this method can be used to check a button, it cannot be used to uncheck a button. (To uncheck a button, you have to check another button from that same field.)

Example 8.2. Use `Field.checkThisBox` to check a radio button, and to fail to clear a button.

You'll notice that when you shift-click, the second button is not cleared. To clear the field, we can reset the radio button field, .

The script for the push buttons follows.

```
1 var f = this.getField("rb8-2");  
2 f.checkThisBox({nWidget: 1, bCheckIt: (!event.shift) });
```

Comments. (1) The `nWidget` parameter is a 0-based index; the widgets are indexed as they are created. The second widget was created after the first, so we give this parameter a value of 1 (0-based). The second parameter we get from whether the user, that's you, was pressing the shift key when you clicked on the button. □

The strategy of resetting the radio button field works if none of the buttons are on by default. If one is on by default, all buttons will be cleared, except for the one that is on by default, it will be set. For example, consider these two radio buttons, the first on is on by default. Press the reset button to see the results.

You can query individual radio buttons with `Field.isBoxChecked`. The methods takes single `nWidget` parameter

```
Field.isBoxChecked({ nWidget: Integer });
```

and returns `true`, if the widget whose indexed is passed as the `nWidget` value is checked, `false` otherwise.²

Example 8.3. Use `Field.checkThisBox` to check a radio button, and to fail to clear a button.

```
1 var f = this.getField("rb8-3");
2 var strMsg = "Report of the state of the radio buttons: ";
3 strMsg += "\r\u2022 Button 0 is "
4         + ((f.isBoxChecked(0)) ? "checked" : "clear");
5 strMsg += "\r\u2022 Button 1 is "
6         + ((f.isBoxChecked(1)) ? "checked" : "clear");
7 app.alert(strMsg);
```

Comments. After getting the `Field` object of the target radio button field, line (1), we build in lines (2)–(6) an alert box message based on the values of `f.isBoxChecked(0)` (for the first widget) and `f.isBoxChecked(1)` (for the second one). □

8.3. Changing the Appearance and the Properties of the Radio Button Field

Until such time arrives that we begin to examine the `Actions` tab—where JavaScript actions are defined—we shall concentrate on the appearance and other properties of the radio button field.

• The General Tab

Below, see [Figure 1](#) on page 6, is a screen shot of the `General` tab of the `Radio Button Properties`, parallel to the dialog box, are the JavaScript properties that correspond to the elements on the `General` tab of the `Radio Button 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](#), and [#15](#), so no examples will be presented here. More on!

• The Appearance Tab

Let's do the same now for the appearance tab, see [Figure 2](#) on page 7.

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

²Actually, the method returns 1 for `true` and 0 for `false`.

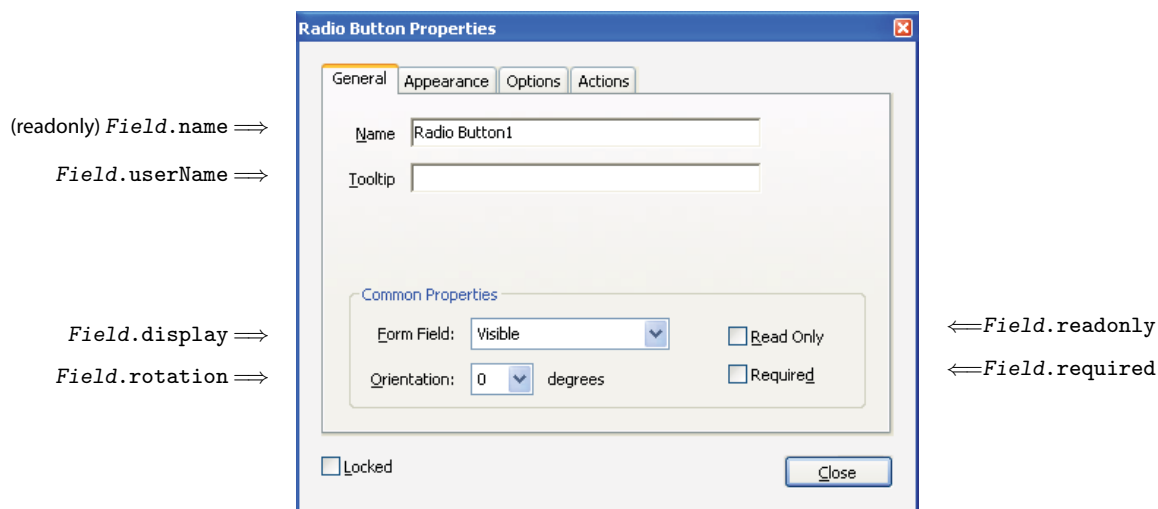


Figure 1: The General Tab of Radio Button Properties

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 radio button.

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

Font: The default value of the Font property in Figure 2 is Adobe Pi; consequently, the JavaScript property *Field.textFont* is readonly.

Thickness: Use *Field.lineWidth* to set the width of the border surrounding the bounding rectangle of the radio button. 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 “check mark”. To set the “check mark” to red, execute *Field.textColor=color.red*

We have pretty well illustrated these properties in the previous blogs, *AcroTeX PDF Blogs #13*,

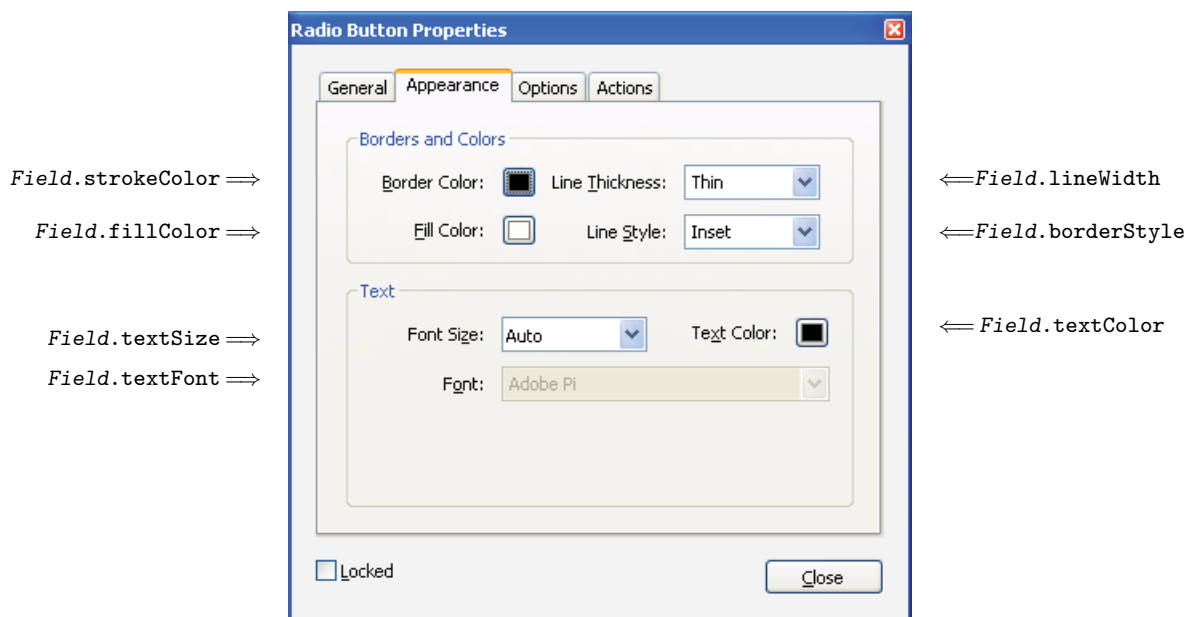


Figure 2: The Appearance Tab of Radio Button Properties

#14, and #15, so no examples will be presented here.

• The Options Tab

In [Figure 3](#) on page 8, we set up the correspondence between each user interface element and its JavaScript counterpart.

Button Style: What style of check to use, the user interface offers the following choices: Check, Circle, Cross, Diamond, Square, and Star. The corresponding values for *Field.style* are *style.ch*, *style.ci*, *style.cr*, *style.di*, *style.sq*, and *style.st*; for example, *Field.style=border.ci* sets the Button Style to Circle. The *Field.style* was illustrated in the discussion of the Options tab of the radio button, [AcroTeX Blog #15](#).

Button Value:³ This field is the user interface to setting the export value. To set this value programmatically, use *Field.exportValues*.

Button is checked by default: If this box is checked, the radio button will be checked (be given the 'on' appearance) when the for field is reset. Use *Field.defaultValue* to check or clear this box, and use *Field.defaultIsChecked* to test whether this box is checked. The *Field.defaultValue* was illustrated in the discussion of the Options tab of the check box,

³There must have been a change in name of the user-interface, this field used to be called Export Value, and still is for the check box.

`Field.style` ⇒
`Field.exportValues` ⇒
`Field.defaultValue` or
`Field.defaultIsChecked` ⇒
`Field.radiosInUnison` ⇒

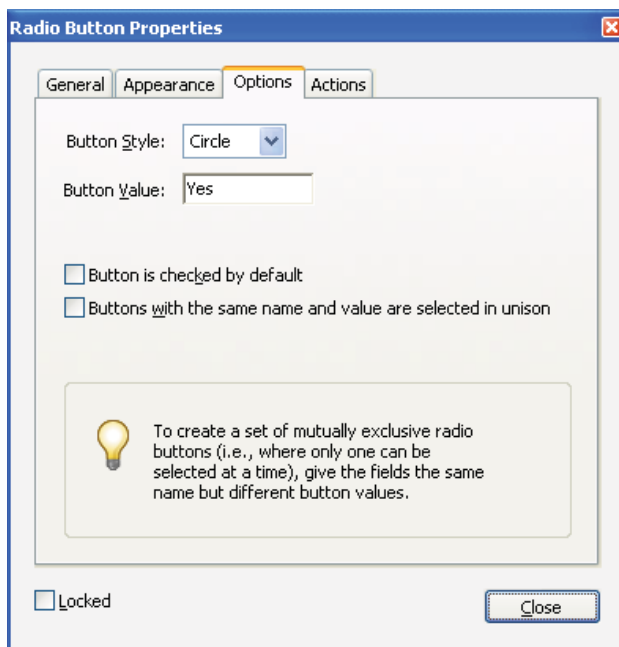


Figure 3: The Options Tab of Radio Button Properties

AcroTeX Blog #15.

Buttons with the same name and value are selected in unison Use the Field method `Field.radiosInUnison` to check this box, and to change the behavior of the radio field. Here is a characteristic of radio button fields that is not present in check boxes. Normally, we create a radio button field so that the name of the buttons are the same, but have different button values. Actually, even if they have the same button values, they'll still act like a radio button field should. The radio buttons below all have button value `AcroTeX_Rocks` (sorry). Click on them to get a feel for how AcroTeX Rocks.

After you have finished playing around with the buttons, click the button `AcroTeX_Rocks`. After clicking once, return to your favorite playground and click more radio buttons. Do you see the difference? Can't turn them off can you? Well we need at least one radio button with a different value

Repeat the experiment but with this set of radio buttons.

Some people didn't like the toggle in unison feature, this property was created to give the document author a choice of how the radio buttons behave.

One final remark before I have to snap off. When a field is copied so that there are several fields with the same name,⁴ the widgets appear in the Fields Navigation panel listed as `<terminal_field_name>#<index>`. For radio button fields this does not occur. By definition, two radio buttons with the same name are part of the same radio button field. For example, if an entire radio button field is copied and pasted from one part of the document to another, these new radio buttons are still considered part of the original field, rather than a duplicate of the field.

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

⁴These fields are then called widgets, in Acrobat jargon.