

Acrobat Forms Tips and Techniques

3 Formatting and Calculations

We now have an invoice that can be filled in, but it is rather dumb. A user must perform his own calculations to figure out totals and shipping. Fortunately, the latest version of the forms plug-in includes a JavaScript interpreter that will allow you to rectify this dismal situation. In particular, the format and calculations tabs are at your disposal.

Formatting

Acrobat forms allows you to specify a number of pre-canned formats for fields including numbers, dates, percentages, times, and special formats like zip codes. It even allows you to create custom formats using completely arbitrary JavaScripts.

1. Open this [sample file](#) (FrmTnT03.pdf) or use the sample file from the previous tips that you've already worked on. Make sure you've got the form tool selected.
2. Select all of the fields in the rightmost column and hit enter (this is the same as a double click and brings up the properties pane). These fields are calculated fields and it is useful to distinguish them from non-calculated fields. Change the text color to black and check the read-only check box (as these fields are calculated the user doesn't have to interact with them).
3. Select the hand tool and verify that your fields are now non-interactive to the form user. Switch back to the form tool and double-click on the **Price.0** field and switch to the *Format* tab. Note that there are quite a few pre-canned formats that you can use. Select the *Number* format and specify 2 numbers after the decimal point and a dollar currency symbol.
4. Modify the other price fields to match this format. Repeat the formatting with the **Total**, **SalesTax**, **Shipping** and **GrandTotal** fields.

Simple Calculations

Calculations usually involve some scripting unless the form is extremely simple. The *Calculate* tab offers a simple formula builder for summing, multiplying, and obtaining minimum, maximum, and average values. Of course, a custom calculation script can solve any calculation, with a little bit of extra work.

1. Use the sample file from the previous step. With the forms tool, select the **Total.0** field and bring up the properties dialog. Switch to the *Calculate* tab.
2. This is a simple calculation where we want **Total.0** to be equal to **Price.0** multiplied by **Quantity.0**. Click on the *Value Is The...* radio button and select the *product* operator. Now select the *Pick* button.
3. This brings up a list of all fields in the document. Choose the **Price.0** field and hit the *Add* button. Choose the **Total.0** field and hit the *Add* button. Now close the field picker dialog. Notice that the list of fields in the simple calculation section contains both field names. If you already know the field names you can simply type them in here rather than going through the field picker.
4. Select the hand tool and enter a price and quantity into the appropriate fields. You should see your calculation take place! Switch back to the form tool and modify the rest of the **Total** fields so that they are live too.
5. This invoice is kind of strange in that it doesn't include a **SubTotal** field. This would be useful for later calculations so let's create one. It doesn't matter where or how big it is. Mark it as being hidden (see the *Appearance* tab). You can use hidden fields to hold temporary calculations for convenience or debugging purposes.
6. Switch to the *Calculate* tab. The **SubTotal** field is merely the sum of all the total fields. Again, this is a simple calculation, so choose the *Value Is The...* radio button and select the *sum* operator. You can either enter each of the fields separately (e.g. **Total.0**, **Total.1**, etc.) or you can simply enter **Total**. The program is smart enough to figure out that a plain **Total** means the sum of all fields that look like **Total.***.
7. You might want to make the **SubTotal** field visible to check that your calculations are working correctly.

8. Finally, the **GrandTotal** field is the sum of the **SubTotal**, **SalesTax**, and **Shipping** fields. Use the simple calculation capabilities of Forms to calculate the value of this field.

Calculation Order

Because you have defined many fields with calculations, there may be dependencies between some fields that require that they be calculated in a certain order. For example, the **SubTotal** field must be calculated after all of the **Total** fields are calculated otherwise it would get an erroneous result. The program allows you to modify the calculation order of fields so that your calculations are performed correctly.

1. Choose the **Tools⇒JavaScript⇒Calculation Order...** menu item. It brings up a list of all fields that have calculations associated with them.
2. You'll want to make sure that the **Total** fields occur before the **SubTotal** field in the list. Use the *Up* and *Down* buttons to re-order the fields.

Complex Calculations

The easy parts of the invoice are now complete but the **SalesTax** and **Shipping** fields remain. These involve some custom JavaScript programming.

1. Select the **SalesTax** field using the form tool and bring up the properties dialog. Switch to the *Calculate* tab and select the *Custom Calculation Script* radio button. The *Edit...* button should now be highlighted.
2. Click on the *Edit...* button to bring up the JavaScript programming window. This window allows you to type in an arbitrary JavaScript program and will pre-check it for syntax errors and other problems automatically. You may want to review the Acrobat Forms JavaScript Object Model document (found in your Acrobat Help directory) after this tutorial for more in-depth information about JavaScript and how to manipulate form fields.
3. Sales tax should be equal to the value of the **SubTotal** field multiplied by 8.25% (or whatever is appropriate for your state). We first need to get the value of the subtotal field into JavaScript. The basic idea is that you want to bind a JavaScript variable to an Acrobat Form field. You do this via the `getField()` method of the document object, passing it in the name of the field like this:

```
var f = this.getField("SubTotal");
```

Our variable *f* now points to the field **SubTotal**. As a side note, the *this* object almost always points to the current document.

4. Next add the following line of code:

```
event.value = f.value * 0.0825;
```

The *event* object is the target of the current calculation event, in this case, it's the **SalesTax** field. We set the value of the event to be the value of the variable *f* (the **SubTotal** field) multiplied by the sales tax.

5. Click on the OK button, hopefully you didn't have any typos in your script. If so, the program will beep and position the cursor at the offending line. Compare the above two statements against your script if necessary.
6. Click on the *Shipping* field with the forms tool and bring up the properties dialog for this field. Select the *Calculation* tab, custom format, and click *Edit...*
7. Read the shipping explanation at the bottom of the invoice. Using the remnants of your high school word problem skills this roughly translates into: $\text{Shipping} = \max(\text{SubTotal} / 50, 5)$. This can be transmogrified into JavaScript as follows:

```
var f = this.getField("SubTotal");  
event.value = Math.max(5, Math.floor(f.value / 50));
```

Ouch! In order to use complex mathematical operations like *max* and *floor* you need to access the *Math* object and its methods and properties. Peeling apart the second line of the script, we want to make sure that we don't charge pennies for shipping so we truncate any numbers after the decimal point (using *floor*) after we've divided the sub-total by 50. Finally, we want a minimum of 5 dollars which means that we have to choose the maximum of either 5 or the calculated shipping.

8. Switch back to the hand tool and try out your invoice, it should now calculate totals, tax, shipping, and the grand total perfectly!

Automatic Values

The invoice works well now but there are a number of touches we can add to polish it off. Firstly, it would be useful to have a current date field that automatically fills in the invoice date.

1. Select the forms tool and create a text field in the upper right hand corner of the invoice. Call this field **Today**. Select the *Format* tab and format the field as a *Date* with the short month, day, and long year option (e.g. Jan 3, 1998). Make sure the field is read-only as it will be a calculated field.
2. Select the *Calculate* tab, the *Custom Calculation* radio button and click on the *Edit...* button.
3. We want the value of **Today** to be today's date. Enter the following script:

```
event.value = util.printd("mm/dd/yy", new Date());
```

This script sets the calculation event's value (that is, the value of **Today**). The *new Date()* expression creates a new date object initializing it to the current date and time. The utility object is used to format the date into something legible (numeric month/date/year) and assigns this value to **Today**. Easy! You can use this type of calculation script to automatically calculate the current time, page number (*this.pageNum*), document modification date (*this.modDate*), etc. See the *Acrobat Forms JavaScript Object Model* and in particular the document object properties for ideas.

4. Select the hand tool and modify one of the fields in the document, notice how our invoice date is automatically calculated when the user modifies the document!
5. You may want the date to be automatically be calculated whenever the user opens the document (as opposed to when he changes it). In that case, you would create a document level script (that is, one that is executed when the document first opens) to calculate the date. Go to the **Tools⇒JavaScript⇒Document Scripts...** menu item. This brings up a list of scripts attached to this document.
6. Type in a new name: Initialize, and click on the *Add...* button. This brings up our familiar JavaScript editing window. Delete the automatically generated text and add the following script:

```
var f = this.getField("Today");  
f.value = util.printd("mm/dd/yy", new Date());
```

You can use document level scripts to define functions to be used by many fields. If you find yourself typing the same script over and over again, you might want to write a function that does the bulk of the processing and invoke that function instead. For example, if you have a set of statistical functions or complex date calculations, you might want to push these to the document level.

Wrapping Up

The invoice is pretty much complete but it would be handy to have a button to clear the invoice and a button to print the invoice.

1. Select the forms tool and create a button field and call it **Reset**. Go to the *Appearance* tab and give it a thin beveled border and choose some complementary border, background, and text colors.
2. Go to the *Options* tab and give it a *Text Only* layout with a caption of "Reset".
3. Go to the *Actions* tab and add a *Mouse Up* action. Choose *Reset Form* from the list and click okay.
4. Select the forms tool and create a button field and call it **Print**. Go to the *Options* tab and give it a *Text Only* layout with a caption of "Print".
5. Go to the *Actions* tab and add a *Mouse Up* action. Choose *Execute Menu Item*, click on the *Edit...* button and then choose the **File⇒Print** menu item.

All done! Try out your form to make sure it works okay and debug any problems that may have slipped in inadvertently. You can look at the finished form we've included [here](#) (FrmTnt04.pdf) for ideas.

Happy form authoring.