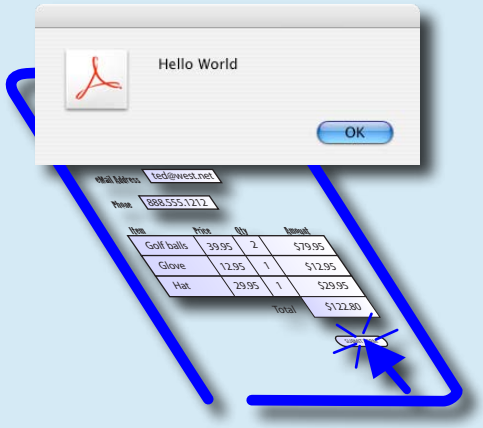
Writing JavaScripts



Indicates that the script or task cannot be executed in Adobe Reader

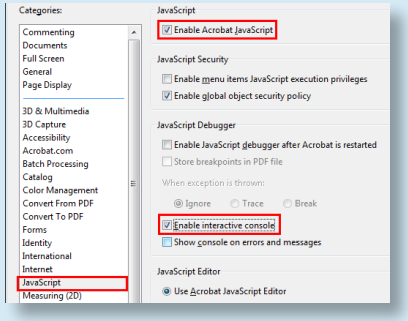
46

Using the JavaScript Debugger

The JavaScript Debugger is very helpful when writing JavaScripts to report bugs and test lines of code.

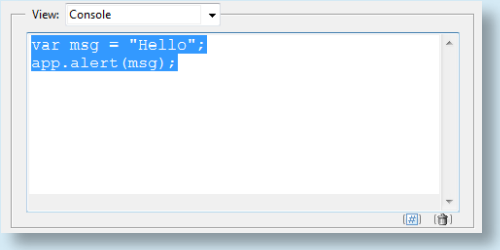
To enable and use the JavaScript debugger, do the following.

- Open the Preferences dialog box. Press CTRL/Command + K to open the Preferences.
- Click JavaScript in the left pane.
- In the right pane check the *Enable Acrobat JavaScript* check box.
- Check the *Enable interactive console* check box.
- Click OK, to return to the Document pane.



To Use the JavaScript Debugger:

- Press CTRL/Command + J to open the JavaScript Debugger.
- Type a script in the Console window.
- To run a script, select the parts you want to test and press the Num Pad Enter key.



Setting Custom Zoom Levels

When you make adjustments for the Initial View, you choose zoom views from a set of preset magnifications.

If you want to use a zoom view other than a preset, you need to write a JavaScript to set a custom view.

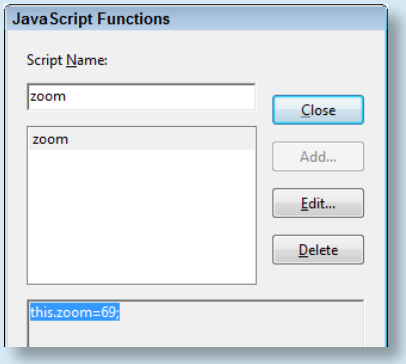
Here's how you do it:

- Choose Advanced | Document Processing | Document JavaScripts.
- Type a name in the Script Name field box and click Add.
- Delete the default text and type:

```
this.zoom = 69;
```

  The above script sets the zoom view to 69% when the file is opened.

- Click OK and click Close in the JavaScript Functions dialog box and save the file.



48

Navigating Views

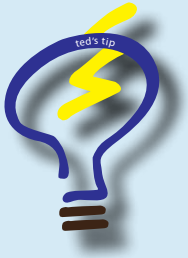
On a button action do the following:

- Create a Button field.
- Click the Actions tab.
- Choose Run a JavaScript from the Select Action drop down menu.
- Type one of the following scripts in the JavaScript Editor.

  To go to the next page: `this.pageNum++;`

  To go to the previous page: `this.pageNum--;`

  To go to a specific page: `this.pageNum = 0;`
- Click OK in the JavaScript Editor.
- Click Close in the Button Properties dialog box.



Keep in mind that JavaScript is zero based. Therefore if you want to write a script that opens page 7 in a document, your script needs to be written as:

```
this.pageNum = 6;
```

Creating an Application Alert

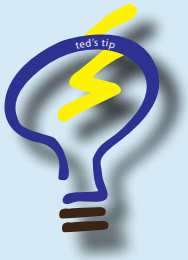
Application alerts can help you save some space on a form. Rather than adding instructions on the form itself, you can create one or more application alert dialog boxes to inform users procedures for filling in a form.

To create an application alert, follow these steps:

- Choose Advanced | Document Processing | Document JavaScripts.
- Type a name in the Script Name field box and click Add.
- Delete the default text and type (without the line numbers):

```
1. var msg = "Please fill in all form fields";
2. app.alert(msg);
```

  Whatever text you add between the quotes in line 1 will appear in the dialog box.
- Click OK, click Close in the JavaScript Functions dialog box, and save the file. Upon saving the file, the dialog box should appear and give you a preview of the application alert.



The above script opens the alert dialog box each time the file is opened because the script is added as a Document JavaScript.

The same script could also be added to a Page Action (open the Properties dialog on a Page in the Pages Panel and click Action and add the script in the same manner), a bookmark, a button, or a Document Action.

50 Dismissing Application Alerts

At times you'll want to alert a user when a document is first opened. However, if a user opens your form several times and continually sees the same dialog box appear each time the file opens, it can become quite annoying.

To provide a user an option to dismiss the application alert, use the example code below on a Document Level JavaScript (see Tip #48 for adding a Document Level JavaScript):

```
1.  if(typeof(global.bShowWarning1) ==
    "undefined")
2.  {
3.  global.bShowWarning1 = true;
4.  global.bPersistWarn1 = false;
5.  }
6.  if(global.bShowWarning1)
7.  {
8.   var oCk = {bAfterValue:false};
9.  app.alert({cMsg:"Instructions: Fill
    out this form carefully and click
    the Submit Form button when fin-
    ished.", nIcon:1,oCheckbox:oCk,
    cTitle:"Reminder"});
10. global.bShowWarning1 = !oCk.bAfterValue;
11. }
```

This button has sample code: Click Me

50 Adding Line Breaks to Alerts

You might want to format the text a little in an alert dialog box to emphasize a line of text. By adding a few control switches you can easily add line breaks to messages in application alert dialog boxes.

Here's a sample script that adds a line break.

```
1.  var msg = "Here's a script with a line breaks.\n\nVis-
    it my blog at\n\nhttp://www.acrobatusers/blogs/
    tedpadova"
2.  app.alert(msg);
```

In line one the \n\n coding creates the line break.

Click Me

51 Invoking Actions on Field Entry

Actions such as application alerts, calculations, and a variety of other action types can be invoked when a user tabs into a field.

To add a JavaScript when a user tabs into a field, do the following:

- Open the field properties on the field where you want the action to be invoked.
- Click the Actions tab.
- Open the Mouse Trigger drop down menu and choose On Focus. This option sets the action to when a user tabs into the field.
- Choose Run a JavaScript from the Select Action drop down menu and click Add.
- Type your script in the JavaScript Editor.

54

Assessing Viewer Versions

Some features in Acrobat viewers 6, 7, 8, 9 and above may not be available to users of earlier versions of Acrobat. To help a user understand if a feature won't work in an earlier version, you can have Acrobat assess the viewer version and report in an application alert dialog box a message if it detects a version lower than one you want to specify in the code.

To write a script that assesses viewer versions, add the following script to a Document Level JavaScript:

```
1. //is the viewer version less than 9.0
2. if(app.viewerVersion < 9.0)
3. {
4. var msg = "Some features won't work in
   Acrobat viewer versions less than version 9.0.";
5. app.alert(msg);
6. }
```

Line 1 in the above code is a comment line. When you see a line preceded by // remember that the line has no effect on executing the script.

It's a good idea to add comments so that when you want to later review some of your code it will be much clearer to you.

Many of the scripts in this document won't have comments due to space limitations.

Click Me

Assessing Viewer Types

Another assessment function available to you in Acrobat is assessing the viewer type. This code should be added to forms that have certain actions that can't be executed in Adobe Reader.

To write a script that assesses viewer types, add the following script to a Document Level JavaScript:

```
1. //is Adobe Reader as opposed to Adobe Acrobat
2. if(app.viewerType == "Reader")
3. {
4. var msg = "Filling out this form requires
   either Adobe Acrobat Pro or Adobe Acrobat
   Standard";
5. app.alert(msg);
6. }
```

If you want to assess both viewerVersion and viewerType, you can add both scripts to a Document Level JavaScript.

Simply add the above script following line 6 in the viewerVersion script.

Click Me

56

Assessing Document Information

You can use JavaScript to report back to you certain information about a document such as Title, Author, Subject, Keywords and more.

To write a script that reports the PDF author on a button action, do the following:

```
1. var f = this.getField("textField");
2. f.value = ("Author is: " + this.author);
```

Line 1 identifies the text field where the information will be reported. The script is written on a button and when clicked the author of the PDF is shown in the text field.

Coding for other information that can be reported in the JavaScript Console (see Tip #45 for more on using the JavaScript Console) or within a text field includes:

```
this.title; // title field
this.subject; // subject field
this.keywords; // keywords field
this.producer; // PDF producer application
this.creator; // authoring document
this.creationDate; // date PDF was created
this.modDate; // date PDF was modified
this.pageNum; // page number of current page
this.numPages; // number of pages in the PDF
this.filesize; // the file size of the PDF
this.path; // directory path for file location
this.numTemplates; // # of templates in the PDF
this.numFields; // number of fields in the file
```

Auto-tabbing Fields

You can set up fields with auto-tabs so that when a user completes filling in a field, the cursor jumps to the next field. You might use this feature if you have a SS number or phone number where the numbers are completed in separate fields.

To set up an auto tab, do the following:

- Write a Document Level JavaScript and enter the following code:

```
1. function goNext(item, event, cName)
2. {
3.     AFNumber_Keystroke(0, 0, 0, 0, "",true);
4.     if (event.rc &&
5.     AFMergeChange(event).length ==event.target.
       charLimit)
6.     item.getField(cName).setFocus();
7. }
```

- Create three fields (tel.1, tel.2, tel.2) and set the limit of characters in the Options tab to 3 for the first field, 4 for the second, and 3 for the third. I'll use a Social Security number as an example.
- Open the Format tab in the first field properties and click the Custom Keystroke Script radio button and click Edit.
- In the JavaScript Editor, add the following code.

```
1. goNext(this, event, "tel.2");
```

In the above script, tel.2 is the next field that follows the field where the script is written.

- Repeat the same steps for the second field and jump to the next field (tel.3).

59 Printing Pages Via JavaScript

Adding buttons to print pages is particularly useful when your forms are displayed in Full Screen mode.

To write a script that prints the current page in view, use the following script on a button field:

```
1. this.print (false, this.pageNum,
   this.pageNum, {bShinkToFit: true});
```

The false item in the above script prevents the Print dialog box from opening and the page is sent directly to the printer. The {bShrinkToFit: true} item shrinks the page to fit within the paper size if the PDF page is larger than the output page size.

To print all pages in a document, use:

```
1. this.print(true);
```

60 Printing from the JavaScript Console

If you happen to be viewing a file in Full Screen mode and you want to print a page or the entire document while remaining in Full Screen mode, you can use the JavaScript Console.

To print a document while in Full Screen mode, do the following:

- Press Command/CTRL + J to open the JavaScript Editor.
- In the JavaScript Console type the following code to print the entire document.

```
1. this.print(true);
```

- Press the Num Pad Enter key on your keyboard.

Eliminating Fields From Print

You may want to print a document without the data in certain fields to be printed on the output. Use the Form Field drop down menu in the General tab in the Document Properties and choose Visible but doesn't print and the field won't print on the output.

If you have some conditional items where if X occurs, then you don't want certain fields to print, you can use a JavaScript to control whether fields print or not.

To do so, use the following example:

- On a button field create a JavaScript and enter the following code:

```
1. var f = this.getField(myFfield");
   // myField is the field name
2. f.print = false;
```

Printing with Annotations

If you want to print a file with the annotations added to the output, use the following JavaScript on a button field:

- Create a button field and add a JavaScript using the following code:

```
1. this.print (true, {bAnnotations:
   true, bShrinkToFit: true, bReverse: true});
```

- The above code is used to print annotations and the bShrinkToFit toggle is on as well as printing in reverse order. To eliminate any of the items, delete those not desired from the code.

62

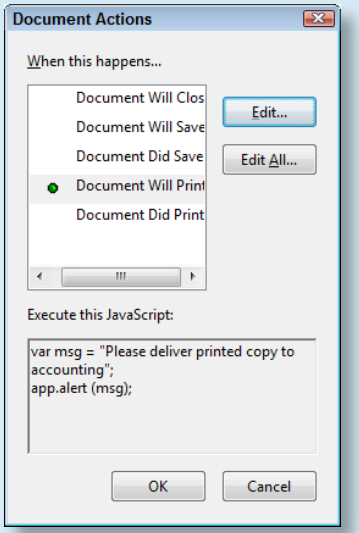
Creating Document Actions

You can execute JavaScripts during certain document actions such as saving a file, printing a file, or closing a file. You may want to add an application alert when a file prints to provide the user instructions on what to do with the form after printing it.

To write a script on a Document Action, do the following:

- Choose Advanced | Document Processing | Set Document Actions.
- Choose an action in the scrollable list (for example Document Will Print).
- Click Edit to open the JavaScript Editor.
- Add an application alert in the JavaScript Editor such as the example shown:

```
1. var msg = "Please deliver printed copy to
   accounting";
2. app.alert(msg);
```



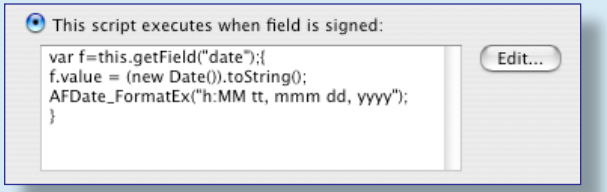
Date Stamping a Document

You may want to date stamp a form at the time a user signs the form or submits the form.

You can add a JavaScript on a signature field or a button. Use the following example to add a date stamp when a user signs a form:

- Create a Signature field on a form and click the Signed tab in the Signature Properties dialog box.
- Click the *This field executes when the field is signed* radio button and click the Edit button.
- Type the following script in the JavaScript Editor:

```
1. var f=this.getField("date");{
   f.value = (new Date()).toString();
   AFDate_FormatEx("h:MM tt, mmm dd, yyyy");
   }
```



date in the script above is the name of the field where the date will be added.

Click Me

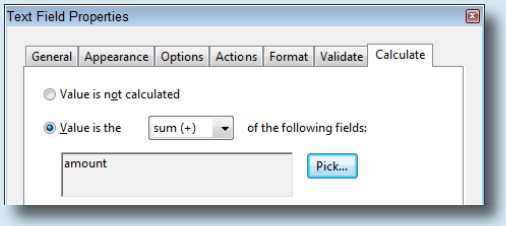
64

Summing Rows and Columns

The easiest way to sum a column or row is to not write a JavaScript and use the built-in formula for summing data that Acrobat provides you. To use the built-in formula for summing a series of data fields, do the following:

- Create a column (or row) of fields formatted as numbers.
- Be certain to name the fields with hierarchical names such as amount.0, amount.1, amount.2, etc.
- Double-click the field that records the data sum with the Select Object tool to open the field properties dialog box.
- Click the Calculate tab.
- Choose *sum(+)* from the *Value is the* drop down menu.
- Click the Pick button.
- Check the box where you see the parent name in the Field Selection dialog box.
- Click OK.

When you return to the Calculate tab in the Text Field Properties, you should see the parent name shown in the field box below the *Value is the* drop down menu.



Using Simplified Field Notation

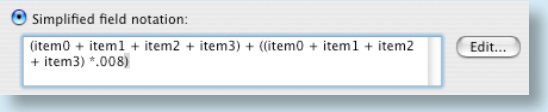
Acrobat offers only a limited number of built-in formulas. If you want math functions such as division and subtraction, you need to use either JavaScript or Simplified Field Notation (SFN).

Simplified Field Notation is a much easier scripting language based on syntax similar to formulas written in spreadsheets. To calculate a sum and add a sales tax using SFN, do the following:

- Open a text field properties for a total field and click Calculate.
- Click the *Simplified field notation* radio button and click the Edit button.
- Type the following script in the JavaScript Editor:

```
1. (item0 + item1 + item2 + item3) + ((item0 +
   item1 + item2 + item3) *.00725)
```

- The above script sums data in fields item0, item1, and item2 and adds a sales tax of 7.25%. Note that the fields all have separate names and do not use parent/child names.



You cannot use hierarchical names with Simplified Field Notation.

66 Summing Data with JavaScript

You may want to sum data in a column or row and add a sales tax to the sum, a shipping charge, apply a discount, or some other math function as part of the field calculation.

If you use the built-in formula described in Tip #63 you can't add a second calculation. If you use a Simplified Field Notation script, you need to name each field separately in the formula —something that could be very tedious with long columns or rows.

In order to extend the formula to calculate a total and apply the total to another calculation, your best option is to use JavaScript.

- Open the Calculate tab on a field to hold the data calculation and click Edit for Custom Calculation Script.
- In the JavaScript Editor, type the following code:

```
1. var f = this.getField("amount");
2. var a = f.getArray();
3. var sum = 0;
4.    for (i =0; i < a.length; i++)
5.    sum += a[i].value;
6. event.value = sum + (sum * .08);
```

Line 1 sets up an array with a field having a parent name of *amount*. The form contains several fields named *amount.0*, *amount.1*, *amount.2*, etc. The loop is executed and picks up the value for each field and adds it to a variable called sum.

The last line adds the sum to the sum * a sales tax of 8%.

67 Calculating a Sales Tax

Sales tax calculations can be handled easily with built-in formulas, Simplified Field Notation, and JavaScript when you have separate fields for a subtotal and a tax field.

Using JavaScript the following script would be used to calculate a sales tax:

- Open the Calculate tab on a field to hold the data calculation and click Edit for Custom Calculation Script.
- Type the following script in the JavaScript Editor:

```
1. var f = this.getField("subtotal");
2. event.value = Math.round(f.value * 7.25) / 100;
```

- The above script assigns a variable to a field called *subTotal* where the sum of data in a row or column is calculated. The second line of code calculates a sales tax of 7.25%.

Calculating a Shipping Charge

A shipping charge might be calculated based on the total cost of items purchased. The following formula calculates a shipping charge of 12% of the total purchase amount.

- Open the Calculate tab on a field to hold the data calculation and click Edit for Custom Calculation Script.
- Type the following script in the JavaScript Editor:

```
1. var g = this.getField("subtotal");
2. event.value = Math.max(2, Math.floor(g.value /
   12));
```

- The above script assigns a variable to a field called *subTotal* where the sum of data in a row or column is calculated. The second line of code calculates a shipping charge of 12%.

70

Importing Images

You may want to add a photo or logo to a form that permits the user to import an image on a button action.

Here's how to do it:

- Create a button field and choose Icon only in the Options tab.
- Click the Actions tab and add a JavaScript.
- In the JavaScript Editor, type the following code:

```
1. event.target.buttonImportIcon();
```

When the form filler clicks the button, the Select Icon dialog opens.

Showing/Hiding Fields

Acrobat provides you an action to show and hide fields. If you have many fields to show/hide, you need to check each item in the Show/Hide Field dialog box. An easier way to handle the task is to use JavaScript.

- Create a button field and click the Actions tab and add a JavaScript.
- In the JavaScript Editor, type the following code:

```
1. var f = this.getField("myField");//parent name
2. f.hidden=true;//hides all fields with the same
   parent name
```

To show hidden fields use the following script in line 2:

```
2. f.hidden=false
```

For an example of extensive use of showing and hiding fields, see the companion file in this portfolio.

Click Me

Resetting a Form

Acrobat has a built-in function for resetting a form. This action works fine when you want to reset all fields or all fields with a few exceptions on a form.

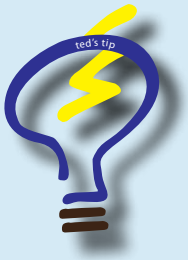
You may have different sections of a form that you don't want to reset such as the identifying information or fields that contain fixed data that you use for calculations.

Using JavaScript the following script would be used to reset form fields having the same parent name:

- Create a button and add a JavaScript.
- Type the following script in the JavaScript Editor:

```
1. var resetFields = this.getField("customerData");
2. this.resetForm(resetFields);
```

- The above script resets the fields with the parent name of customerData.



If you want to reset all fields on a form, use the following script:

```
1. this.resetForm();
```

This kind of script might be added as a Document Level JavaScript or Page Action script along with an Application Response dialog box that prompts the user to reset a form.

If a user partially completes an order form, the user would say no to resetting a form. If the user starts anew, then perhaps all the data might be reset on the form.

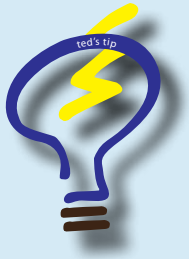
Setting Fields to Read Only

Fields that are set to Read Only in the General tab of the field properties dialog box, prevent data entry or changing the field data by an end user. If the form is secured with Acrobat Security, the user won't be able to edit the fields and change the data.

You might set fields to Read Only that have fixed price values and calculations.

To set a field for Read Only, do the following:

- Open the properties dialog box on any field.
- Click the General tab.
- Check the box for Read Only.



You might start with fields that are marked for reading and writing and later, after the user executes certain actions, you might want to lock the fields to prevent data entry. To handle this task add a JavaScript in the Calculate tab where you want the field marked as Read Only.

Here's a sample:

```
var f = this.getField("item")
f.readonly=true;
```

All fields with the same parent name are marked as Read Only.

To disable Read Only, use the following in a script like the above:

```
f.readonly=false;
```

Deleting Fields

Deleting fields can provide another level of security on a form. When a user submits a form or prints a form, you can delete all fields with no data. Something like a travel reimbursement form might be one example of a form where deleting fields could be used.

If there are no fields to add data, the form can't be changed by another user.

Using JavaScript the following script would be used to delete a series of fields containing no data:

- Add a JavaScript to a Signature Field, a Document Action, or a Submit button.
- Type the following script in the JavaScript Editor:

```
for (var i = 0; i < 5;i++)
    {var f = this.getField("amount."+i);
    if (f.value == "")
    removeField("amount."+i);
}
```

- The above script deletes all fields with the parent name of *amount* that have no data.

Deleting Zeros from Calculation Fields

If you have a table and the last column or row of data is a calculated field (something like price * qty), the default result is 0.00 until data are added for the fields within the calculation.

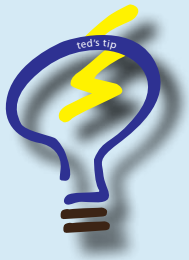
You can eliminate default zeros in calculated fields with JavaScript.

Example script:

```
1. var f = this.getField("price.0");
2. var g = this.getField("qty.0");
3. if (g.value !=0)
4.     event.value = f.value * g.value;
5. else
6.     event.value ="";
```

In lines 1 and 2 the variables are assigned for a price and quantity field. The result of the calculation is an amount determined by multiplying the price * the quantity.

Line 3 checks the qty field and if it's not zero, the calculation is performed. If the result is zero, line 6 is executed where no value is entered in the field resulting in eliminating default zeros.



Note that you cannot eliminate zeros using the built-in formulas or Simplified Field Notation. You must use JavaScript in order to eliminate the zeros.

74 Adding URLs to Text

Text added to a background design that contains a URL or email address is automatically recognized in Acrobat viewers 7 and above. There's no need to add links or buttons to navigate to a URL or open your default email client on a button click.

However, if you ask a user to add a URL to a text field, the text field isn't recognized as a URL link. Using JavaScript, you can create the link by doing the following:

- Create a Text field in Form Edit mode.
- Open the Actions tab and choose Mouse Up for the Mouse Trigger. This trigger will execute the action when a user clicks on the text field.
- Add a JavaScript in the Action tab and type the following in the JavaScript Editor:

```
1. if ( !/^\s*$/.test(event.target.value) )
   app.launchURL(event.target.value);
```

- The next time a user clicks in the text field, the URL opens in the user's default Web browser.

75 URL Links in New Windows

By default, a URL link opens in the same window if the browser is open and displays a web page. Using JavaScript you can force a URL link to open in a second window:

- On a link button or any field containing a URL link use the following script:

```
1. app.launchURL("http://www.mycompany.com/file.
   pdf", true);
```

The true statement in the above code triggers opening the link in a new browser window.

Adding Annotations Using JavaScript

77.

You might have a need for a user to add a comment note, add a Stamp comment, or a Text Box comment based on a conditional response.

Perhaps clicking a check box where the item is Other among a list of other options is one example where you might use this feature. If the check box is clicked, a Text Box is added to the document page or some such other type of condition.

Note that the file must be enabled with Adobe Reader usage rights for this to work in Adobe Reader.

- Add a JavaScript to a button, a check box, or radio button and click the Actions tab.
- Type the following in the JavaScript Editor:

```
1. var annot = this.addAnnot({
2.     page: this.pageNum,
3.     type: "FreeText",
4.     rect: [396,298,629,406],
5.     strokeColor: color.white,
6.     fillColor: ["RGB", .85, .85, .85],
7. }
```

Line 3 identifies the type of annotation to be added to the page. In this example FreeText is a Text Box comment.

Line 4 specifies the Text Box rectangle coordinates. You need to precisely determine where the coordinates are for the annotation you want to add using the script (See Tip #78 for how to determine x,y coordinates). Lines 5 and 6 set attributes for the border color and the fill color of the Text Box. Click Me

Adding Fields Using JavaScript

You can add fields using JavaScript. You might add a field for user input based on a user response.

This task cannot be performed in Adobe Reader when forms are created in Acrobat. However, you can create dynamic forms in Adobe LiveCycle Designer that enable Adobe Reader users to spawn fields on a page.

Example script:

```
1. var f = this.addField("newField", "text",
2.     page = this.pageNum,[100,100,200,200])
3.     f.textSize = 9;
4.     f.textfont = font.Helvtica;
5.     f.textColor = color.white;
6.     f.fillColor = new Array("RGB", .9, .9, .9);
7.     f.alignment = "left";
8.     f.multiline = true;
9.     f.readonly = false;
```

Line 2 identifies the x,y coordinates for the field rectangle. Before writing the script, be certain to identify the exact coordinates where you want the field to appear.

See Tip #78 for an easy method for identifying x,y coordinates.

Click Me

Determining x,y Coordinates

Tips #76 and #77 use x,y coordinates to determine the bounding box for annotations and fields. Before you start writing a script you need to know the coordinates for where you want the object to appear on the page.

To easily determine the coordinates you'll use in your JavaScript code, do the following:

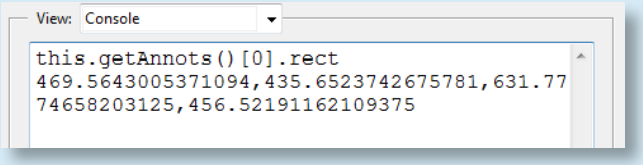
- Open the Comment and Markup toolbar.
- Click the Rectangle Comment tool.
- Draw a Rectangle comment in the area of your form where you want to create an annotation or field.
- Open the JavaScript Console (Command/CTRL + J).
- Enter the following code in the Console.

```
1. this.getAnnots()[0].rect;
```

- Press the Num Pad Enter key on your keyboard.

  The JavaScript Console reports back to you the coordinates.

- Ignore all digits after the decimal point and use the whole numbers for the x/y coordinates in your script.



79

Changing Text Colors

By default, Acrobat permits you to identify negative values with a red color in the Format tab. If you want to emphasize a value that's not a negative value or use another color other than red, you can do so using JavaScript.

Example script in the Format tab in a text field:

```
1. var f = this.getField("fieldName");
2.    if (event.value >50)
3.    f.textColor = color.red;
4. else
5.    f.textColor = color.black
```

The above script sets the text color to red if the value in a field named *fieldName* is greater than 50.

80

Using Custom Colors

Acrobat provides you with some default colors that you can assign to text and objects. If you want to create a custom color, look up a color in a program like Adobe Photoshop and record the RGB (or CMYK) values. The colors in Photoshop use whole numbers to express color.

In Acrobat you define a color by using something like ["RGB", .503, .982, .495]

To convert the color values from Photoshop to Acrobat, take a Photoshop color and divide by 255. Therefore a color like a red value of 149 in Photoshop would be a red value of .584 in Acrobat. The syntax for changing a color in a field would be something like:

```
1. event.target.textColor = ["RGB", .503, .982,
   .495];
```

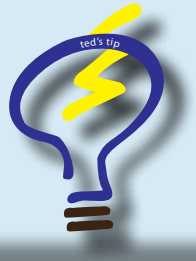
82 Spawning Pages From Templates

Spawning pages from templates provides you with options for many uses. PDF files can be made more dynamic by adding content on-the-fly through new pages added to a document.

- Add a page to a document that you want to use for a template and navigate to that page.
- Choose Advanced | Document Processing | Page Templates in Viewer mode. (Note that you cannot access the Page Templates command in Form Edit mode).
- Type a name in the Page Templates dialog and click Add. To hide the template from view, click the Eye icon.
- Add a JavaScript to a button field and enter the following code in the JavaScript Editor.

```
1. var a = this.getTemplate("myTemplate");
2. a.spawn ({
3.     nPage:this.numPages,
4.     bRename:true,
5.     bOverlay:false
6. })
```

Be certain that you use the name you assigned to the template in Line 1. When you click the button, a new page is spawned from the template.



By default the bOverlay switch is off meaning new pages are added to the document. If you want to overlay the new page on an existing page (like adding a watermark), set the toggle to true as shown below:

```
4. bOverlay:true,
```

Click Me

82 Changing OCG States

Optional Content Groups are layers in a PDF document. You have available an action to Set Layer Visibility, but there are conditions where the action won't work and you need to use a JavaScript to change layer visibility.

To examine scripts that change layer visibility open the file in this portfolio: ***housePlansForm.pdf***. Look over the JavaScripts in this document to discover how to change layer visibility using JavaScript.

83 Limiting Character Strings

You can limit the number of characters added to a text field with a JavaScript. To do so, use the example below:

- Open the Text field properties on a field.
- Click the Format tab.
- Click the Custom Keystroke Script radio button.
- Click Edit and add the following script in the JavaScript Editor.

```
1. var f = this.getField("myField");
2. if (event.value.length > 10)
3.     {
4.         var msg = "Your entry is too long
           (limit 10 chars).";
5.         app.alert(msg);
6.     }
```

Sample field:

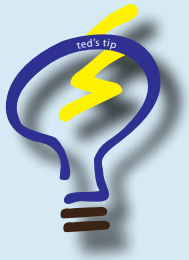
Popup Menus for URL Navigation

Popup menus provide you with navigation options that can be used when viewing PDF files in Full Screen views. The following popup menu item opens different URL in your default Web browser.

- Add the following script to a button:

```
1. // Declare pop-up properties as an array.
2. var aParams = [
3.     {cName: "Acrobat User Community", cReturn:
       "www.acrobatusers.com"},
4.     {cName: "-"},
5.     {cName: "Acrobat User Blogs", cReturn:
       "http://www.acrobatusers.com/blogs"},
6.     {cName: "Ted's Blog Site", cReturn:
       "http://www.acrobatusers.com/blogs/tedpadova"}
7. ];
8. var cChoice = app.popUpMenuEx.apply( app, aParams
   );
9. if ( cChoice != null ) app.launchURL(cChoice);
```

The above popup menu works much better than bookmarks if you intend to display your form in Full Screen mode.



You can add an unobtrusive popup menu to a form that provides URLs where users can explore more product information, contact info, company brochures, etc. One button can open many links while not disturbing the form design.

Click Me

85

Popup Menus for Page Views

The app.popUpMenuEx object is the preferred way to write a script for application popup menus. However, you can get by with a much easier script that works well in all Acrobat viewers.

The following example uses a simple method for navigating pages in PDF.

- Navigate to a view you want for a page link to a popup menu item.
- Choose View | Navigation Panels | Destinations.
- Click the Create new destination tool in the Destinations panel.
- Repeat the steps for creating destinations to all views you want to link to in a popup menu.
- Create a button field and add a JavaScript with the following code:

```
1. var cChoice = app.popUpMenu
2.     (["First Page", "Home"],
3.     ["Second Page", "Samples"],
4.     ["Third Page", "JavaScripts"],
5.     ["Fourth Page","URL Links"],
6.     ["Fifth Page","Sample Forms"]);
7. this.gotoNamedDest(cChoice);
```

The first item in steps 2-6 is a parent name that shows a child name when the menu is opened. The child names are the same exact names as the destination names and a click on the name opens the destination view.

Popup Menus for Opening Files

To create a popup menu that opens files, be certain to place the files in the same folder as the document containing the popup menu.

Add the following JavaScript to a button:

```
1. var cChoice = app.popUpMenu
2.     (["Human Resource Forms", "hrForm.pdf",
       "vacationLeave.pdf"],
3.     ["Sales Forms", "salesOrder.pdf", "customerID.
       pdf", "purchaseForm.pdf"],
4.     ["Administration Forms", "creditApplication
       Form.pdf", "creditApprovalLetter.pdf"]);
5. this.slave = app.openDoc((cChoice), this);
```

Each of the items (2-4) has multiple submenu items that open the respective PDF documents.

Don't forget the semicolon (;) at the end of line 4. Each of the other lines of code (2-3) end with a comma.

Emailing a PDF

From a button or a JavaScript assigned to another action you can attach the open PDF document to an email message.

On a button field, add the following script:

```
1. // use mailDoc in this routine
2. this.mailDoc(true, "yourname@address.com", "cc
   goes here", "bcc goes here","Subject goes here");
```

88

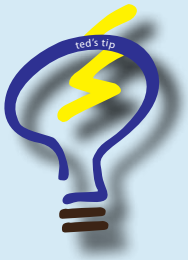
Emailing Form Data

From a button or a JavaScript assigned to another action you can attach just the form data from the open document to an email message.

On a button field, add the following script:

```
1. // use mailForm in this routine
2. // gets the user address from a field
3. var f = this.getField("useraddress");
4. this.mailForm(true, f.value, "", "","Subject");
```

The above script attaches an FDF form data file to an email message. Line 3 picks up an email address from the form that the PDF author added to the form and uses that address for the To field in a new email message.



Note that if you want to reference fields on a page that have been filled in by you and you don't want them to be visible to the end user, you can fill in a field and hide the field.

Open the General tab in the field properties dialog box and choose Hidden from the Form Field drop down menu.

Creating Fixed Response Options

You may have a form where you want a user to make a selection from a Combo Box and based on that selection, offer choices respective to the first choice in a second Combo Box.

Something like choosing a shirt in one column and have sizes appearing in a second column might be one example of such a case. If you have shirts, pants, hats, and other articles of clothing , the choices in the second column will change based on the first column choice.

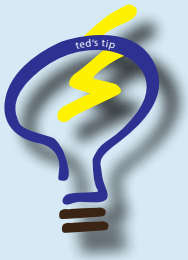
For the script (opposite page) to execute properly, you need to create a table with at least two columns (or rows). The script is added to the fields in the first column (or row) in the Format tab, then duplicated for all subsequent columns/rows.

Do the following to set up the table:

- Create a table with 5 columns as shown below.
- Open the text field properties in the first field in column one and click the Format tab.
- Add a Custom Keystroke Script and type the code at right in the JavaScript Editor.

Item	*Type*	*Price*	*Qty*	*Amount*
Golf Balls	Pinnicle	$19.95	2	$39.90
Golf Shoes	Size 9.5	$89.95	1	$89.95
Golf Hats	Navy	$19.95	1	$19.95
			Total	$149.80

```
1.  if (!event.willCommit) {
2.  var a = this.getField("type.0");
3.  a.clearItems();
4.  switch (event.change) {
5.      case "Golf Balls":a.setItems([“”,
        "Titlest", "Pinnicle", "MaxFli"]);
6.  break;
7.      case "Golf Hats":a.setItems([“”, "Navy",
        "White", "Green"]);
8.  break;
9.      case "Golf Shirts":a.setItems([“”, "Small",
        "Medium", "Large", "Extra Large"]);
10. break;
11.     case "Golf Shoes":a.setItems([“”, "Size 9",
        "Size 9.5", "Size 10","Size 10.5",
        "Size 11", "Size 11.5", "Size 12"]);
12. break;
13.     case "Golf Tees":a.setItems([“”, "25",
        "50", "100"]);
14. break;
15. }
```



If you want a fixed price for each item selected for Item and Type, add a JavaScript in the Calculate tab for the Price field.

Open a price field and you can view the JavaScript added here to create the fixed prices.

Creating App Response Dialog Boxes

An application response dialog box typically presents the user with a question and solicits an answer. Based on the response certain actions can be invoked. You might want to ask a user how many products to purchase. The user responds and you can then take the data from the dialog box and add it to a field object.

To perform such a task, do the following:

- Create a text field.
- Create a button field and add the JavaScript below:

```
var t = this.getField("qty");
// the target field
var cResponse = app.response({
    cQuestion: "How many do you want?",
    cTitle: "Quantity to purchase"});
        if ( cResponse == null){
        // if Cancel is selected
app.alert("Cancel order for this item?");
    cResponse = 0;
        }
else
app.alert("You ordered, \""+cResponse+"\"...
 items",2);//confirmation dialog
{
t.value = cResponse; // places the data from the
dialog to the target field
}
```

Click Me

91.

Adding Data to Secondary Forms

If you have a product brochure of a large file size, you might want to add application response dialog boxes to populate a secondary form or copy data from a form page on a product brochure to another document you use as an order form.

On a button field, add the following script:

- Create a form and add the following Document Level JavaScript:

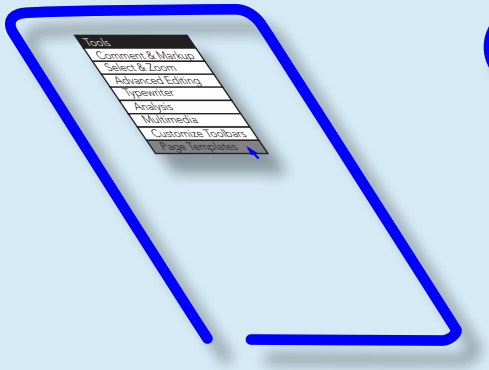
```
this.disclosed = true;
```

- On a button field in another document add the following script:

```
this.slave = app.openDoc("myForm.pdf",this);
this.bringToFront();
for (var i = 0; i <4; i++) {
    var quantity = this.getField("qty."+i);
    var result = this.slave.getField("oQty."+i);
        result.value = quantity.value;
    var amount = this.getField("total." +i);
    var cResult = this.slave.getField("oAmount."+i);
        cResult.value = amount.value;
}
slave.bringToFront();
```

If you want Adobe Reader users to fill in a form apart from a brochure, add the form to a PDF Portfolio. Add a submit form button and enable the form with Reader usage rights. When the Reader user submits a form in a PDF Portfolio, only the form is sent back to you.

Adding Menu Items



Scripts cannot be written in Adobe Reader, but the menus created in Acrobat do appear in Adobe Reader.

About Adding Menu Commands

You can add custom menu commands in Acrobat and the same commands you create in Acrobat are also usable in Adobe Reader.

To create a custom menu command you need to add a JavaScript to your JavaScripts folder.

On Windows the directory path for the JavaScripts folder is: Program Files/Adobe/Acrobat 9.0/Acrobat/JavaScripts

On the Macintosh the path is: Users:User Log-on/ Library/ Acrobat User Data/90_x86/JavaScripts

Copy the file attachments in this document to the JavaScripts folder on your computer.

Save your text files with a .js extension.

Add a menu command that counts the total fields on a from. This can help compare forms, diagnose forms, and provide information on forms that have been secured with Acrobat Security.

When you select the menu item, a dialog box opens reporting the find.

- Write a JavaScript with a text editor using the following code:

```
1. function totalFields(){
2. app.alert("Total Fields: " + this.numFields);
3. }
4. app.addMenuItem ({cName: "Number of Fields",
   cParent: "Tools", cExec: "totalFields()"
5. });
```

The above script adds the menu command to the Tools menu (line 4).

93

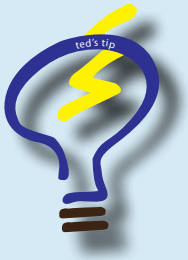
Counting Page Templates

Add a menu command in the Tools menu that counts Page Templates. Counting Page Templates can be helpful in diagnosing forms and provide information about forms that have been secured with Acrobat Security.

- Write a JavaScript with a text editor using the following **code:**

```
1. function totalTemplates(){
2.     app.alert("Total Page Templates: " + this.
       numTemplates);
3.     }
4.     app.addMenuItem ({cName: "Page Templates",
   cParent: "Tools", cExec: "totalTemplates()"
5. });
```

The above script adds the menu command to the Tools menu (line 4).



You can add both menu commands in Tips #92 and #93 in the same script. Copy lines 1-3 in one script and paste it after line 3 in the first script. Then paste lines 4-5 at the end of the first script.

Check out the file link in this portfolio and copy the tools.js content to your JavaScript folder and you won't need to write the scripts.

95

Listing Menu Items

It you want to access a menu command using JavaScript or add a menu item, you need to know the menu items names used by JavaScript.

To view a list of menu item names, do the following:

- Open the JavaScript Console (Command/CTRL + J).
- Type the following code in the Console:

```
1. app.listMenuItems();
```

- Press the Num Pad Enter key and all the menu item names are shown in the JavaScript Console.

To display toolbar buttons, type the following in the JS Console:

```
app.listToolbarButtons();
```

Adding a URL to a Menu

You can add menu commands that open URLs in your default Web browser.

Here's the JavaScript code for accessing the FedEx/Kinkos web site to order prints. Replace the URL in this script with your favorite print service.

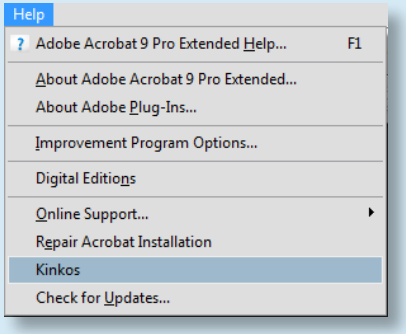
- Write a JavaScript with a text editor using the following code:

```
1. app.addMenuItem({
2. cName: "FedExKinkos",
3. cParent: "Help",
4. cExec: "app.launchURL('https://
   partner.fedexkinkos.com/partnerInit.do');",
5. nPos: 16
6. });
```

Line 5 in the above code positions the command in the Help menu. You may want to change the position by typing a different value.

When the menu command is selected your default Web browser opens and takes you to the target URL.

Copy the code from the kinkos.js file in this portfolio.



Add a File | New Menu Command

Acrobat provides you an option to create new blank pages. However, the new pages by default have a text block that can be annoying when you want to use blank pages for temporary documents to hold form fields.

With a little JavaScript you can create the missing File | New command that Acrobat has avoided through 9 generations. The following code creates the command with several different custom page size options:

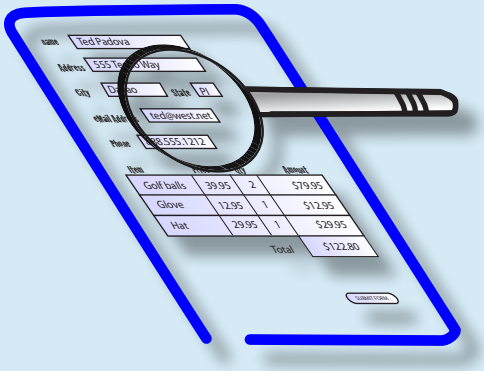
- Type the following code in a text editor and save the file to your JavaScript folder with a .js extension:

```
1.  trustedNewDoc = app.trustedFunction( function
    (nWidth, nHeight)
2.  {
3.  app.beginPriv();
4.  switch( arguments.length ) {
5.  case 2:
6.  app.newDoc( nWidth, nHeight );
7.  break;
8.  case 1:
9.  app.newDoc( nWidth );
10. break;
11. default:
12. app.newDoc();
13. }
14. app.endPriv();
15. })
16. app.addSubMenu({ cName: "New", cParent: "File",
    nPos: 0 })
```

The **newPage.js** file is contained in this portfolio.

```
17. app.addMenuItem({ cName: "Letter Portrait", cPar-
    ent: "New", cExec: "trustedNewDoc();"});
18. app.addMenuItem({ cName: "Letter Landscape", cPar-
    ent: "New", cExec: "trustedNewDoc(792,612);"});
19. app.addMenuItem({ cName: "A4", cParent: "New",
    cExec: "trustedNewDoc(420,595)"});
20. app.addMenuItem({ cName: "Slide (10x7.5)", cParent:
    "New", cExec: "trustedNewDoc(720,540);"});
21. app.addMenuItem({ cName: "Screen (640x480)", cPar-
    ent: "New", cExec: "trustedNewDoc(640,480);"});
22. app.addMenuItem({ cName: "Note (4x3)", cParent:
    "New", cExec: "trustedNewDoc(288,216);"});
23. app.addMenuItem({ cName: "Custom...", cParent:
    "New", cExec:
24. "var nWidth = app.response({ cQuestion:'Enter Width
    in Points',\
25. cTitle: 'Custom Page Size'});"
    +"if (nWidth == null) nWidth = 612;"
    +"var nHeight = app.response({ cQuestion:'Enter
    Height in Points',\
26. cTitle: 'Custom Page Size'});"
    +"if (nHeight == null) nHeight =
    792;"+"trustedNewDoc(nWidth, nHeight) "});
```

working with data



Data can be collected from forms completed by Adobe Reader users, but distributing forms and aggregating data requires Acrobat.

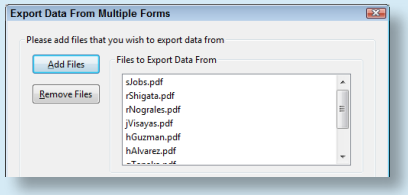
98

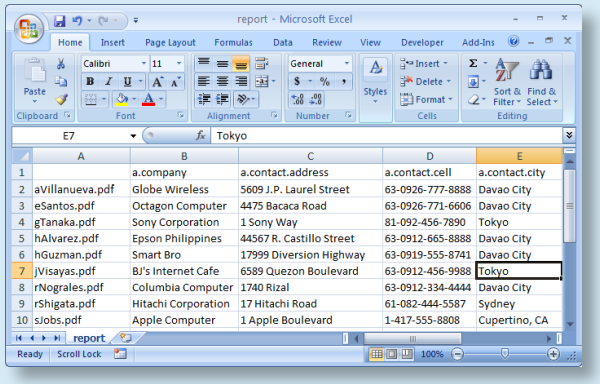
Merging data into spreadsheets

You can easily aggregate data into spreadsheets from Acrobat. If you have forms that were not distributed and designed for retrieving via email or Acrobat.com, you can export the form data to a .csv file that can be opened in Microsoft Excel.

Here's how to do it.

- Choose Forms | Manage Form Data | Merge Data Files into Spreadsheet.
- Click Add Files in the Export Data From Multiple Forms wizard.
- Click Export.
- Click View File and the file opens in Microsoft Excel.





	A	B	C	D	E
1		a.company	a.contact.address	a.contact.cell	a.contact.city
2	aVillanueva.pdf	Globe Wireless	5609 J.P. Laurel Street	63-0926-777-8888	Davao City
3	eSantos.pdf	Octagon Computer	4475 Bacaca Road	63-0926-771-6606	Davao City
4	gTanaka.pdf	Sony Corporation	1 Sony Way	81-092-456-7890	Tokyo
5	hAlvarez.pdf	Epson Philippines	44567 R. Castillo Street	63-0912-665-8888	Davao City
6	hGuzman.pdf	Smart Bro	17999 Diversion Highway	63-0919-555-8741	Davao City
7	jVisayas.pdf	BJ's Internet Cafe	6589 Quezon Boulevard	63-0912-456-9988	Tokyo
8	rNograles.pdf	Columbia Computer	1740 Rizal	63-0912-334-4444	Davao City
9	rShigata.pdf	Hitachi Corporation	17 Hitachi Road	61-082-444-5587	Sydney
10	sJobs.pdf	Apple Computer	1 Apple Boulevard	1-417-555-8808	Cupertino, CA

ted's tip

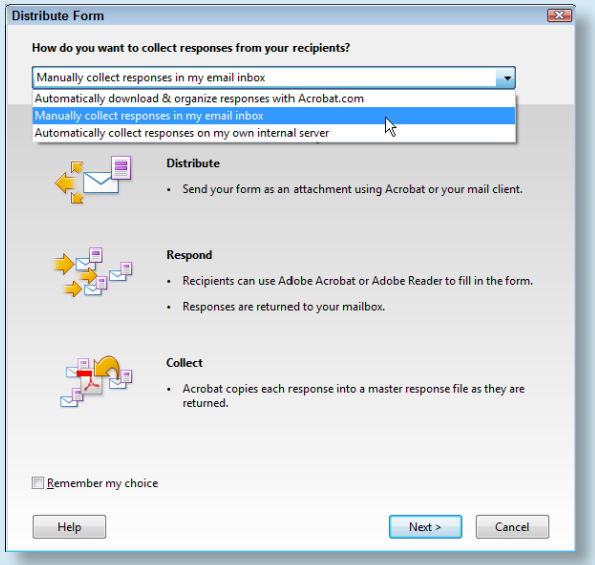
You can only export data into spreadsheets from Viewer mode. Aggregating data requires that all forms have identical field names.

Distributing Forms

You can distribute forms in either Form Edit mode or Viewer mode and Acrobat 9 offers new distribution options.

Do the following to distribute a form.

- Choose Forms | Distribute Form in either Form Edit mode or Viewer mode or click the Distribute Form button in Form Edit mode.
- Choose a collection method in the Distribute Form wizard. Your choices include:
  - Distribute and collect forms on Acrobat.com.
  - Distribute and collect forms via email.
  - Distribute and collect forms on a web server.
- Click Next and you walk through steps to distribute the form.



Collecting Data Via Email

You can distribute forms for email collection. When form fillers send a form back to you it comes as an email attachment. When you double click the attachment, your form is automatically added to a PDF Portfolio.

- When you distribute a form, choose the option: *Manually collect responses in my email box* in the Distribute Form wizard.
- Click Next in the Distribute Form wizard and make a choice for: *email now via Acrobat* or *save the file and email it later*.
- Click Next in the Distribute Form wizard and choose a location for a data file to be saved on your hard drive.
- Click Next.
- Click Finished. The Form Tracker opens and your form is reported to have been successfully distributed. When recipients email completed forms back to you, the dataset file is used to aggregate the forms in a PDF Portfolio.

When prompted to save you dataset file, click the Browse button and locate a folder on your hard drive where you want the file saved. Be certain not move the file until you have collected data from all recipients.

Collecting Data Via Acrobat.com

Acrobat 9 introduces a new support service called Acrobat.com. Using Acrobat.com you can share files, distribute forms, track reviews and forms, and hold web conference sessions.

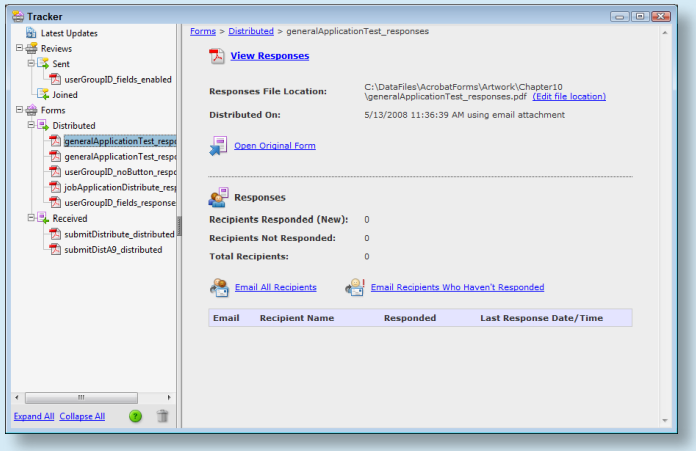
Acrobat.com makes the process of distributing forms and collecting data a much easier task for managing forms than earlier versions of Acrobat.

To distribute a form on Acrobat.com, do the following:

- In the Distribute Form wizard choose: *Automatically download & organize responses with Acrobat.com.*
- Click Next.
- Log on to Acrobat.com.
- If you don't have an Adobe ID click Create Adobe ID. If you do have an ID, click Sign In.
- Your default web browser is launched and takes you to Acrobat. com. In the Acrobat.com interface you have options for tracking forms and managing form data.

Tracking Forms

The Tracker in Acrobat is used to track reviews and distributed forms. After distributing a form you use the Tracker to manage your distributed documents.

To open the Tracker, do the following:

- Choose Forms | Track Forms in either Form Edit mode or Viewer mode.
- The Tracker provides several options that include:
  - View Responses. Opens a PDF Portfolio where responses can be viewed.
  - View Responses. Opens a PDF Portfolio where responses can be viewed and the data sorted and analyzed.
  - Email All Recipients. Click Email all Recipients to send a message to all those invited to fill out the form.
  - Email Recipients who Haven't Responded. Click and you can remind those who haven't responded to get on the stick and complete the form.