AEM Forms Designer

SmartDoc Technologies Courseware: www.aemforms.training

Last Updated: 8-1-2024

AEM Forms Designer

Before we Begin

AEM Forms Designer was previously called LiveCycle Designer and this training course supports all of the versions from LiveCycle Designer ES2 to the current version of AEM Forms Designer. Despite the name change, the program has stayed the same. The greatest changes to the program occurred between LiveCycle Designer ES2 and LiveCycle Designer ES4. To make things simple, we will refer to the various versions as simply Designer for the remainder of this class.

Environments:

You must have a copy of the AEM Forms Designer software to do the exercises in this class.

Prerequisites:

None.

www.aemforms.training

The support site (**www.aemforms.training**) is designed to support our students during and after a training session. Here is what you will find on the support site.

- The **Known Issues** section documents bugs and issues with various versions of AEM Forms.
- The **Sandbox** section lists AEM Forms Servers you can use for the hands-on exercises.
- The **Forum** section enables you to post, review, and answer questions about AEM Forms.

Limited Courseware License

This Student Training Manual and related COURSEWARE is property of SmartDoc Technologies LLC (SMARTDOC). SMARTDOC retains all right, title and interest *(including, without limitation, all patent, copyright, trademark, trade secret and other intellectual property rights)* in and to the COURSEWARE. SMARTDOC grants a limited and non-exclusive license to a STUDENT to use the COURSEWARE under one of the following conditions.

- STUDENT is an active SmartDoc subscriber. Once a STUDENT'S SmartDoc subscription has elapsed, STUDENT is no longer licensed to use this COURSEWARE.
- STUDENT is registered in a training class taught by SmartDoc Technologies.
- STUDENT has received a trial SmartDoc Subscription. Once the trial has elapsed, STUDENT is no longer licensed to use this COURSEWARE.

Students can use the COURSEWARE for their own study but cannot use these materials to teach training courses. Any use of the COURSEWARE by STUDENT for any purposes beyond self-study must be agreed to by SMARTDOC in writing prior to the usage and will require a $400 per day group license fee for each course. Each personalized SmartDoc Subscription and each limited COURSEWARE LICENSE must only be used by the STUDENT and is not transferrable. All rights not granted by SMARTDOC are reserved. STUDENT acknowledges that they have obtained only a limited license right to use the COURSEWARE.

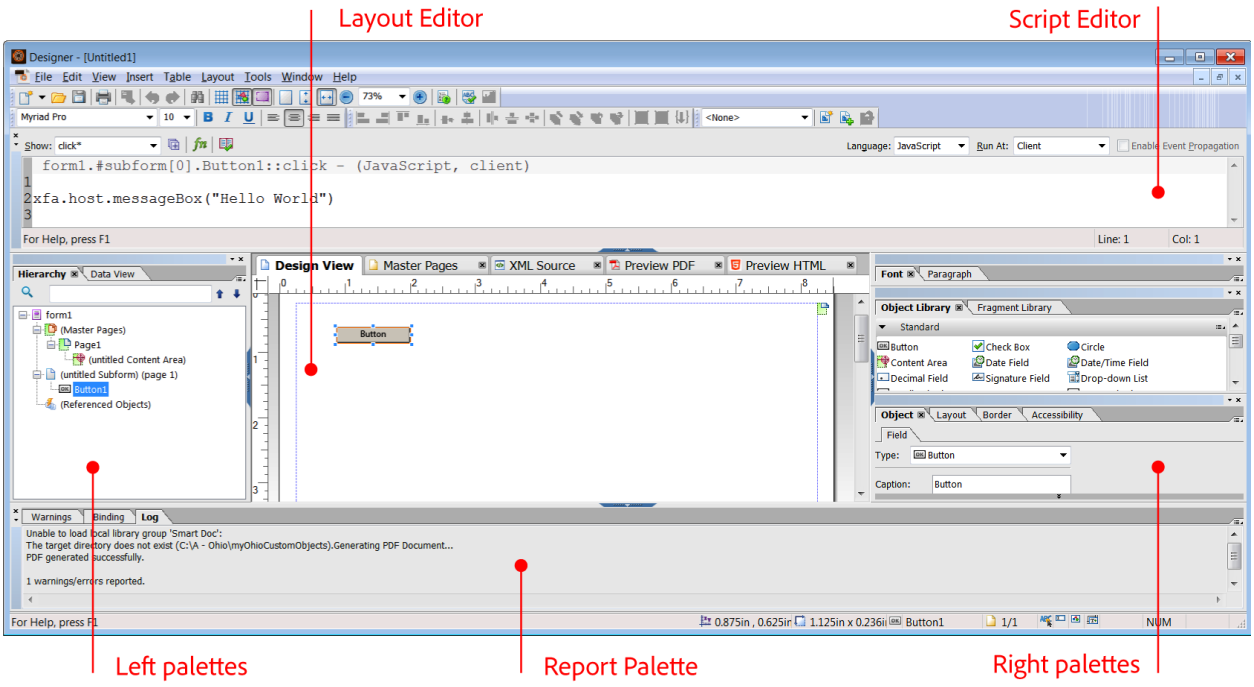
Introduction to Designer

Designer is a Microsoft Windows–based program that provides all the tools you need to create basic or sophisticated forms. You can create PDF or HTML forms with Designer. Designer is a multifaceted tool that supports the four most important requirements of smart forms:

- **Multiple format and device support:** With the new version of Designer, you can create one Designer file and render it in all the formats you need for your business. These formats include HTML, interactive and dynamic PDF forms, and read-only PDF documents. Your Designer forms and documents can be used on traditional PCs and Macintoshes and also on tablets and smartphones.
- **Data handling:** Designer makes it easy to get data structures into your forms and to output properly formatted data from your forms. Powerful tools are provided for data validation, data formatting, and data binding.
- **Scriptable interactivity:** Designer's scriptable interactivity goes way beyond what is available in Acroforms and other form technologies. Virtually every form object is scriptable at any step in your process. Designer has a sophisticated object model, which you can use to add powerful interactivity to your forms.
- **Graphic precision and fidelity:** The three preceding features would usually be enough for an interactive forms tool, but Designer also offers the graphic precision and quality of an Adobe solution. For the last 30 years, Adobe has been the unquestioned leader in computer graphics with industry leading products in graphic design, marketing, and digital publishing tools. You don't need to compromise on typography, layout, or graphic quality with Designer forms, and your one form can be used for all your business needs.

The Workspace

The workspace consists of many editors, views, and palettes. These palettes can float or be docked like they are on the right. You can configure your workspace in many different ways. This illustration shows a very common configuration of a Designer workspace with each of the major sections labelled.



Designer's interface showing each of the major sections.

The Layout Editor

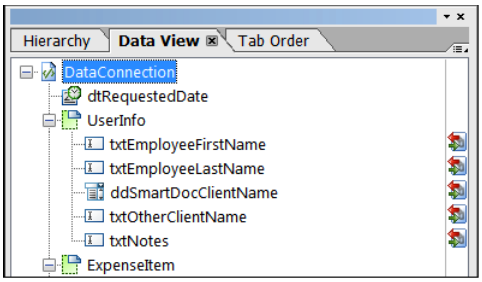
The Layout Editor is in the middle of the workspace, and it contains five main views: Design View, Master Pages, XML Source, Preview PDF, and Preview HTML. You'll see the Layout Editor when you have a document opened. The following is a brief introduction to each of the Layout Editor views:

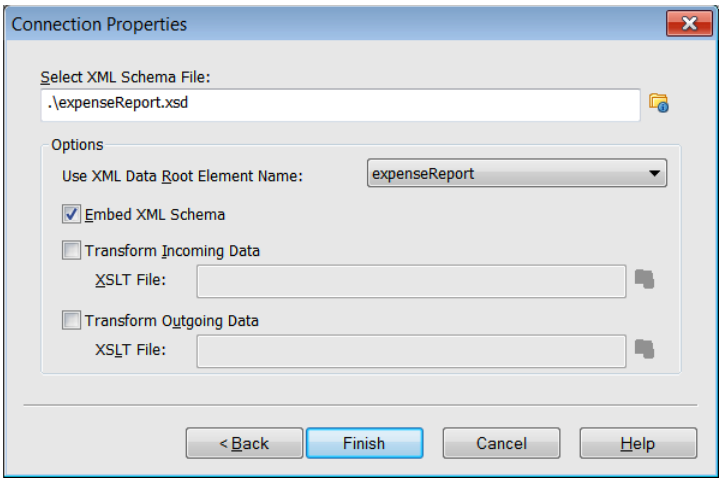
- **Design View:** Design View is the main view in the Layout Editor and the one that you'll use the most. This view displays a WYSIWYG view of your form and allows you to edit many of the visual properties of your form objects directly.
- **Master Pages:** The Master Pages tab shows an independent view of your master pages without any body page content. Master pages contain content areas that define the outer boundaries for body page objects.
- **XML Source:** If you don't see the XML Source tab, you can select it in the View menu. This is your form in XML format. The structure of Designer forms is all XML, and as you make additions and edits to your form in Design View, this XML is updated in the background. You can quickly locate the XML source code for any of your form objects by selecting the object in the Hierarchy palette while the XML Source view is open. You'll immediately see the XML Source view update, and your selected form object will be highlighted in the XML Source view.
- **Preview PDF:** You can preview your forms directly in Designer by switching to the Preview PDF tab. This tab uses your computer's default PDF viewer, so if you have Acrobat, you'll notice part of the familiar Acrobat interface wrapping around your form when you view it in this tab. You can see more of Acrobat by pressing Alt+F8.
- **Preview HTML:** If you don't see the Preview HTML tab, you can select it in the View menu. If it doesn't appear in your View menu, you're likely using a version of Designer prior to Designer ES4. You'll learn how to configure and use this view in, "HTML Forms."

The Left Palettes

The area on the left includes palettes you'll use to organize the structure of your form:

- **Hierarchy:** You'll spend a lot of time using this palette to view and organize the objects and structure of your form. When you select an object on your form, it's also selected in the Hierarchy palette, and vice versa. The Hierarchy palette displays your form's objects in a tree view, which is a great way to examine and edit your form's structure. Understanding this structure becomes particularly important when you start adding script to your form. The Hierarchy palette also displays referenced objects under the Referenced Objects node. A *referenced object* is an object that's added to a form only when it's required. Overflow leader and trailer subforms are examples of referenced objects. Whenever data flows across multiple pages or content areas, the overflow leader and trailer subforms are inserted into the form in the appropriate places.
- **PDF Structure:** This palette does not appear in the previous figure, but is available in Designer. It isn't usually relevant for the XML Forms Architecture (XFA) PDF forms you'll create with Designer. It's more relevant for Acroforms and PDF documents because it displays a hierarchical view of the accessibility tags in these types of PDFs. You'll use this feature in Designer if you're working with PDF forms with fixed pages and fixed background artwork. You add accessibility features to XFA PDFs with the Accessibility palette.
- **Data View:** When you open a form that's connected to a data source, the Data View palette shows you a graphical representation of your form's data elements. You can associate your forms with data sources through a process called data binding. You can bind your form objects to an XML Schema, a Sample XML Data file, an Adobe Data Model, a WSDL (Web Services Description Language) file, or an OLEDB (Object Linking and Embedding Database).
- **Tab Order:** The Tab Order palette shows all your form objects in a list that corresponds to the order that a user will see when tabbing through the form. You can change your form's tab order with this tool. The default tab order for Designer forms is left to right and top to bottom. If you don't see this palette, choose Window – Tab Order.



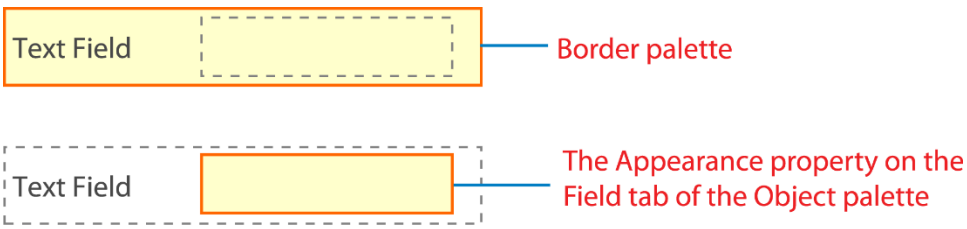


The Data View palette showing the schema binding on the SmartDoc Expense Report form.

The Right Palettes

The area on the right includes palettes you'll use to create and edit your form objects:

- **Font:** The Font palette enables you to edit the font family, size, and style of text in your objects. Many interactive form objects like Text Fields have both a caption and a value font property that can be set independently.
- **Paragraph:** The Paragraph palette enables you to edit the alignment, indents, and line spacing of your text.
- **Object Library:** The Object Library palette provides a collection of all available form objects. You can easily drag and drop form objects from this palette onto your form layout.
- **Fragment Library:** The Fragment Library palette provides a collection of your form fragments. Form fragments are reusable form parts that are saved as separate files. These fragments are ideal if you're creating or managing numerous forms that use the same form parts.
- **Object:** You use the Object palette to modify the properties of your form's interface objects. As you work with this palette, you'll learn that it's very useful and very adaptive. It enables you to edit and customize the most important aspects of your objects without having to provide any custom JavaScript. It's flexible because it reconfigures its options based on the object you select. For instance, if you select a master page object, you'll see the Master Page and Pagination tabs on the Object palette.
- **Layout:** The Layout palette works hand in hand with the Layout Editor. As you make changes to your object in the Layout Editor, they are reflected here, and vice versa.
- **Border:** Most of the information in the Border palette is self-explanatory. However, it's important to note that these border properties affect the entire object, not just the editable portion of the field. The entire area surrounding your field and caption will be outlined with a border that you set with this palette. If you want to edit the border around your field, use the Appearance property on the Field tab of the Object palette.
- **Accessibility:** You can use the Accessibility palette to set Screen Reader properties for your forms. You can set several different options for each field on your form; you'll learn more about this later.



The Border palette settings outline the entire object, but the Appearance property in the Object palette enables you to outline just the field.

The Script Editor

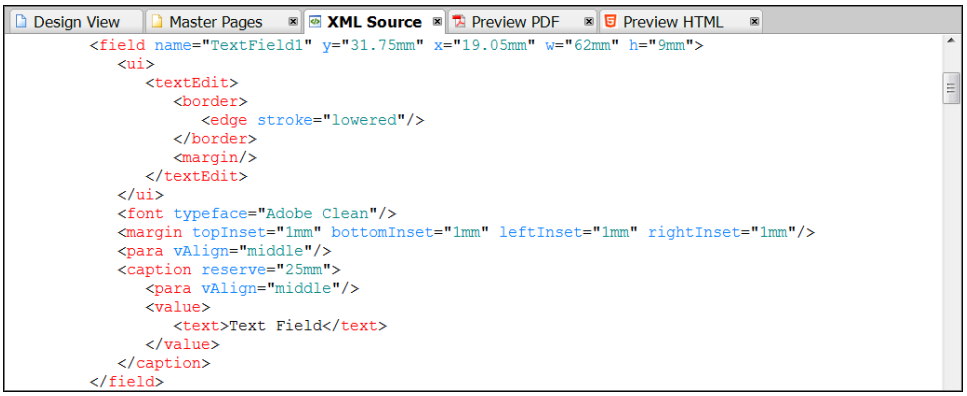
By default, the Script Editor is located directly above the Layout Editor and below the menu bar. You use this tool to create and edit scripts, and it's usually docked at the top of the Designer workspace. Scripts are an important part of smart forms because they enable you to customize the functionality of your forms at both the client and the server level. You'll learn more about this later in the manual.

The Report Palette

By default, the Report Palette is located directly below the Layout Editor, and it's usually docked at the bottom of the Designer workspace. This palette displays messages about your Designer file so you can correct issues as you work.

Adobe XFA and XDP

Designer creates forms based on the Adobe XML Forms Architecture (XFA). You will save these forms as XML Data Packets. (*.xdp). Designer forms can be rendered as many different types of PDFs and also as HTML forms. Designer enables this flexibility because it creates well-structured XFA to describe each aspect of your form. For instance, this is the well-structured XFA that describes a Text Field.



Saving

When you save your Designer file, you need to consider how it will be used. The following is a list of file format options available in Designer. You'll learn how and when to use these formats throughout the training.

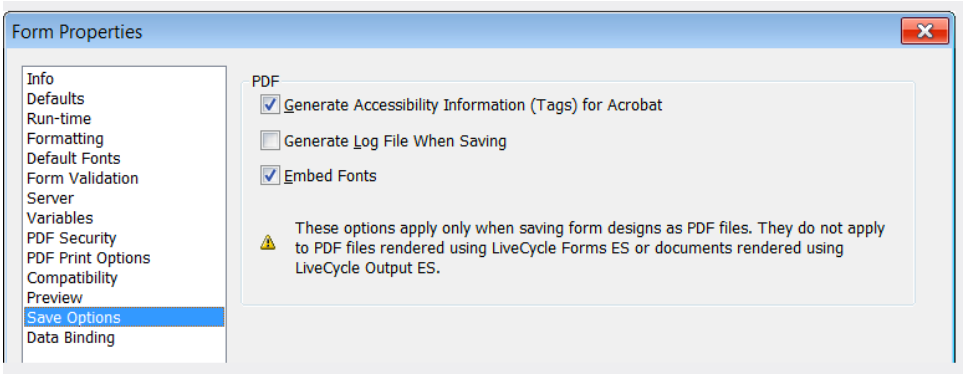
File Formats

– **Adobe Static PDF Form (*.pdf):** Saves forms as static PDFs based on the Acrobat and Adobe Reader target version specified in the form properties. Even if you specify a dynamic, flowable layout, this type of PDF form will still be static and will not re-render in response to user action. The fields on this form type can be interactive or non-interactive.

- **Adobe Dynamic XML Form (*.pdf):** Saves forms as dynamic PDFs based on the Acrobat and Adobe Reader target version in the form properties. In this case, a dynamic, flowable layout will re-render in response to user action. A user can add or remove table rows and even entire form sections with this type of PDF.
- **Adobe XML Form (*.xdp):** Saves forms in the native XML-based file format created by Designer. Use this option if you'll be using an AEM Forms Server to render PDF or HTML forms. The form design can contain dynamic and interactive elements.
- **Adobe Designer Template (*.tds):** Saves the basic structure for a form as a template. It can contain components and settings, such as fonts, page layout, formatting, and scripts. Use it as a starting point for a new form.
- **Adobe Designer Style Sheet (*.xfs):** Saves the internal style sheet in the form as an external style sheet. Style sheets can be used to provide consistent formatting to all your forms. Style sheets can control the look of caption and field value text, the appearance of object borders and background colors, as well as the size and style of Radio Buttons and Check Box objects.

Options for PDFs

When you create a PDF, Designer provide a few additional parameters. You can select these parameters in the Form Properties dialog. Select **File – Form Properties – Save Options**.

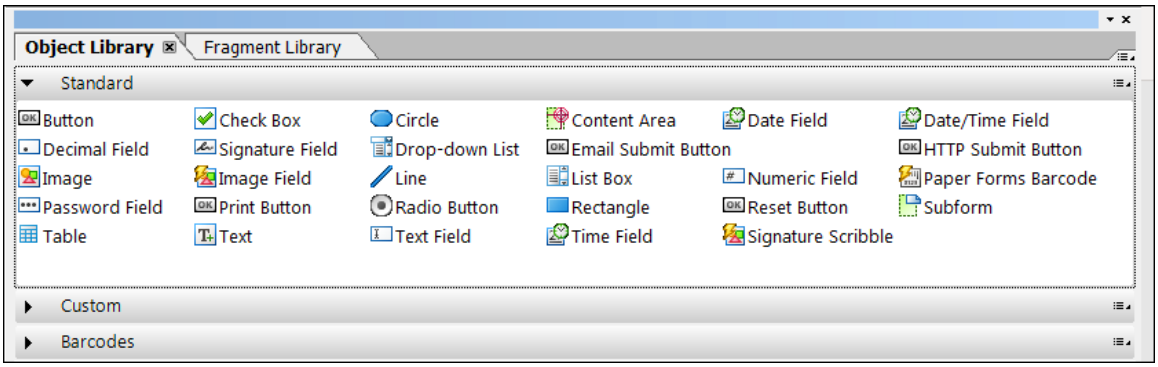


This is a summary of each option, and you will see more details in the training.

- **Generate Accessibility Information (Tags) for Acrobat**
  This option is necessary to make your PDF forms accessible for devices like screen readers.
- **Generate Log File When Saving**
  This option will generate a log file which you can view in your Report palette.
- **Embed Fonts**
  This option will put specific font information in your PDF file. On one hand this is beneficial because you can be assured that your fonts will render properly on a form-filler's machine. On the other hand, selecting this option makes your file much larger, which results in longer download times for your user. The performance advantage of smaller PDF forms is so substantial that you should think twice before choosing to embed fonts.

Standard Objects

Designer includes a library of standard form objects. These objects can be added to your form by dragging and dropping from the *Standard Object Library* to the *Layout Editor*.



Text objects and Text Field objects

These are often confused by new students. The *Text* object is simply static label text. You can use it for labels or for paragraphs of text. It is possible to make *Text* objects dynamic by introducing *floating fields*.

Text Field objects are interactive and dynamic. They provide users with a way to enter data on the form. *Text Field* objects support all the following features.

- Captions and interactive data entry fields
- Multiline text
- Display and validation patterns
- Rich Text

Numeric Field objects

Like *Text Field* objects, *Numeric Field* objects should be used when you need to collect or display data. However, unlike *Text Field* objects, *Numeric Field* objects should be used for numeric data like floats, integers, and currency values. The following are characteristics of this object type.

- Numeric fields only accept numeric input.
- Numeric fields support display and validation patterns.
- Numeric fields are *context-sensitive* to the Locale setting. A U.S. Locale setting will use a comma to denote thousands (1,000.) while a German Locale setting will use a decimal point to denote thousands (1.000).

Image objects and Image Field objects

These objects are like *Text* and *Text Field* objects because an *Image* object is also static, and an *Image Field* object is also interactive. If you need a static image on your form, use the *Image* object. If you need an image that the user can change at runtime, use the *Image Field* object.

Dropdown List objects

A *Dropdown List* is an ideal form object for an enumeration or an array. It displays a list of items and facilitates the user's selection of an item from the list. Consider these points about a *Dropdown List*.

- It accepts custom user entries.
- Although it initially displays only one item, once it is selected, it shows the entire list.

List Box objects

A *List Box* is like a *Dropdown List* because it can display a list of items and facilitate the user's selection of an item from the list. However, keep the following points in mind when considering the use of a *List Box*.

- It requires more form real estate than a *Dropdown List*.
- It does not accept custom entries from the user.

Button objects

Designer's standard *Button* object can be configured for any action your form may need. You can set the Control Type to Regular and write JavaScript on the click event to perform the functionality. Or you can set the Control Type to Submit to configure a form submission. Here is a short summary of Control Types for the *Button* object.

- Select **Regular** for the Control Type to run the JavaScript in the click event.
- Select **Execute** for the Control Type to execute a database query or call a web service.
- Select **Submit** for the Control Type to submit the form's data to a web service or other endpoint.

Specialty Button objects

In addition to the standard *Button* object, Designer also has these specialty Button objects.

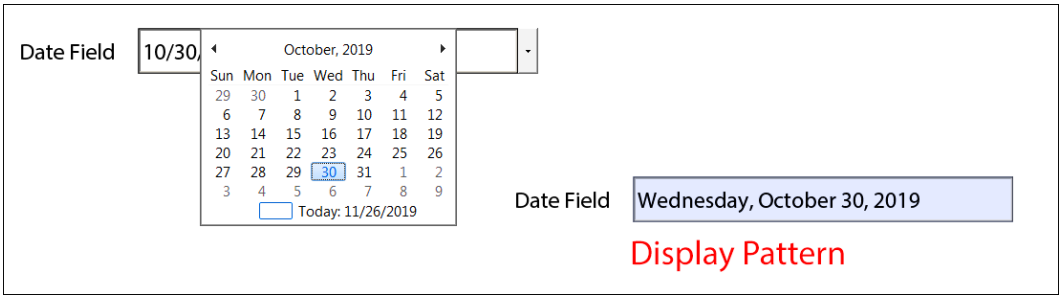
- The *Print Button* initializes the Print Dialog box.
- The *E-mail Submit Button* submits the form to a specified email address.
- The *HTTP Submit Button* creates an HTTP submission of the form to a specified URL.
- The *Reset Button* removes all the data input from the form.

Radio Button objects and Check Box objects

Radio Buttons objects are part of an exclusion group, but Check Box objects are not. Therefore, you will likely use Radio Button objects in cases of mutually-exclusive choices and Check Box objects in cases where one or more choices can be selected. You can set the style of either object to be a square or a circle. You also have a variety of check styles for each object.

Date Field objects

The *Date Field* object presents the user with a visual calendar dropdown to select the date. Once selected, the date is automatically formatted according to the Display Pattern property of the object.



Signature Field objects and Signature Scribble objects

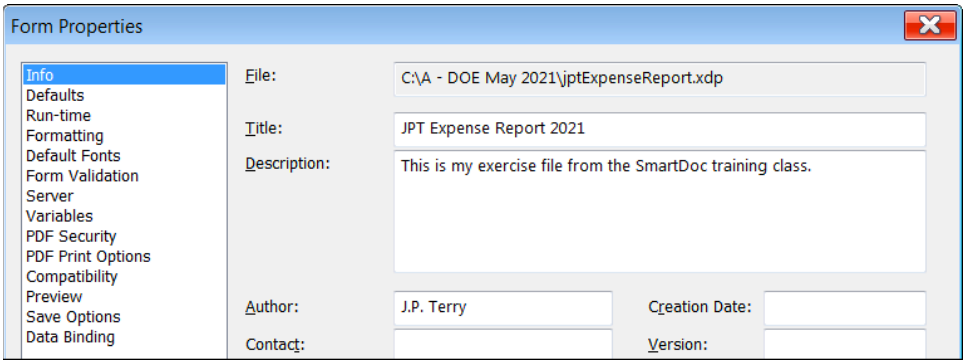
In short, the *Signature Field* object is used for PDF forms and the *Signature Scribble* object is used for HTML forms. The *Signature Field* object requires the end user to have a valid Adobe Acrobat digital signature to sign. The *Signature Scribble* object can be signed simply by scribbling a signature with a mouse or a stylus. If the end user has a touch device like an Apple iPad, they can sign the *Signature Scribble* object with their finger.

Exercises

Work with Objects

In this exercise you will be adding the different interactive elements to your form. You will be setting certain properties for each of the elements so that the element is relevant and works well in your form.

1. Open the Designer software if it is not already open.
2. Select **File – Open** and navigate to your Student Files.
3. Select *ExpenseReport2020_start.xdp* and click **Open**.
4. Select **File – Save As** and navigate to your working folder. Note: This can be a folder of your own choosing.
5. Enter **<yourname>ExpenseReport<year>.xdp** as the File name.
6. Click the *Save As type* dropdown and select **Adobe XML Form (*.xdp)**.
7. Click **Save**.
8. Select **File – Form Properties**.
9. Enter **<yourname> Expense Report <year>** for the *Title*.
10. Enter **This is my exercise file from the SmartDoc training class** for the *Description*.
11. Enter **<yourname>** for the Author.



Note: *This metadata will appear in AEM after the XDP file is uploaded to an AEM Server.*

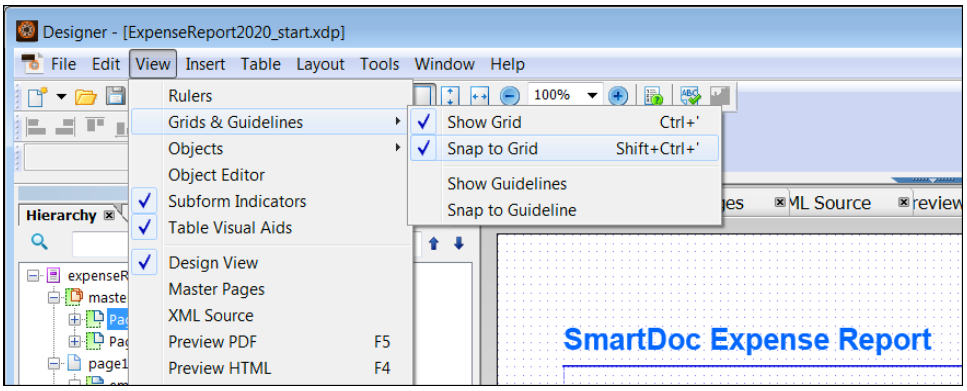
12. When you are done, click **OK**.
13. Expand your **Drawing Aids** palette. If you don't see your **Drawing Aids** palette, you can select **Window – Drawing Aids** to display the palette.

14. Notice that the X Interval is set to 16/in, and the Y Interval is also set to 16/in. This will produce a 1/16-inch grid.

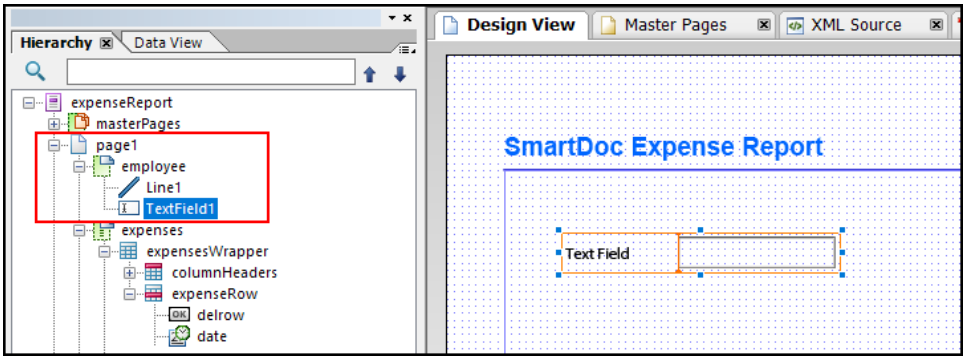
15. Close your **Drawing Aids** palette now by clicking the small X in the top-right corner of the palette.

16. Select **View – Grids & Guidelines – Show Grid** *(see illustration)*.

17. Select **View – Grids & Guidelines – Snap to Grid** *(see illustration)*.



18. Drag and drop a **Text Field** object from the *Standard* Library to the *employee* subform on page1 *(see illustration)*.



19. With the Text Field selected, enter **Email** for the *Caption* property. This property is located on the *Field* tab of the *Object* Palette.

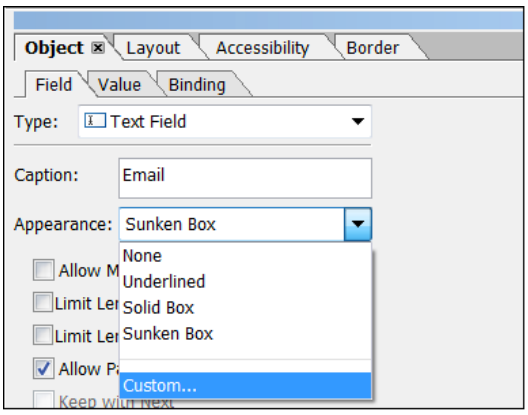
20. Select the *Binding* tab and enter **email** as the *Name* for the object.

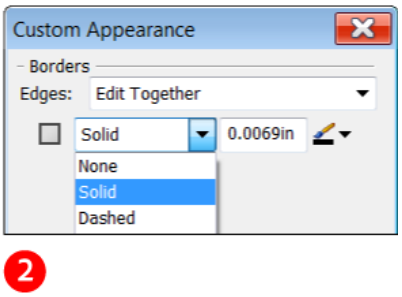
21. Click the *Field* tab.

22. Click the *Appearance* property dropdown and select **Custom** *(see illustration #1)*. The Custom Appearance dialog box will open.

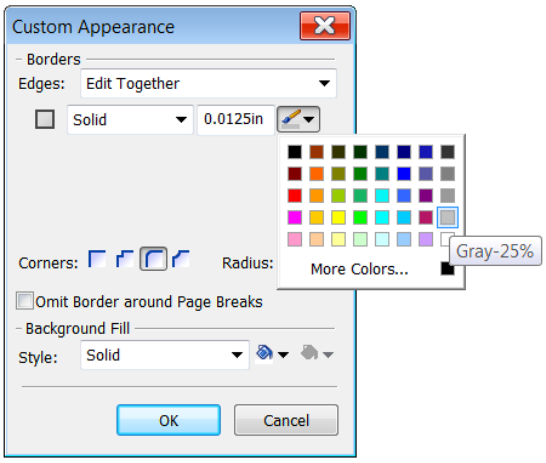
23. Click the *Edges* dropdown and select **Edit Together**. This should be the default.

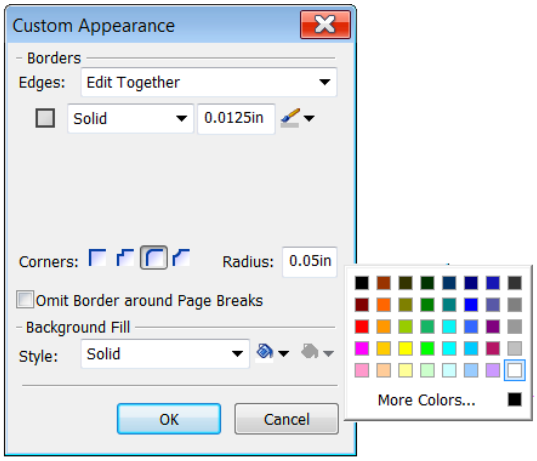
24. Click the next dropdown and select **Solid** for the Edges *(see illustration #2)*.



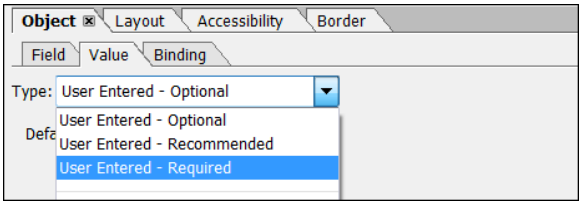


25. Enter **0.0125in** for the line size and select **Gray-25%** for the color *(see illustration below)*.
26. Select a round corner and enter **0.05in** for the Radius.
27. Click the *Background Fill Style* dropdown and select **Solid**.
28. Select **White** for the Fill Color *(see illustration below on the right)*.



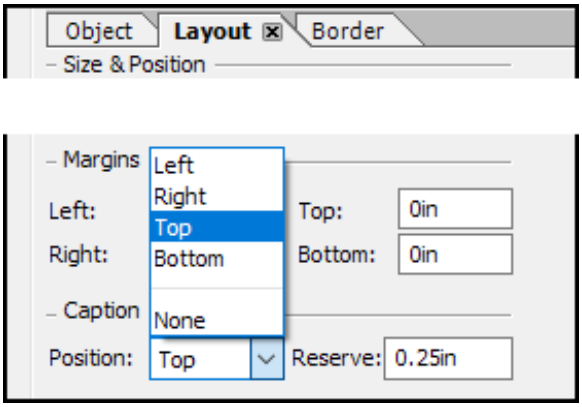


29. Click **OK** when you are done.
30. Select **Limit Length** in the *Field* tab of the *Object* palette.
31. Enter **255** for *Max Chars*. This should be the default.
32. Select the **Value** tab of the *Object* palette.
33. Click the Type dropdown and select **User Entered – Required**. This will make this field mandatory.



34. Select the *Layout* palette. If you don't see the *Layout* palette, select *Window – Layout*.

35. Select the *Caption Position* dropdown and change the value from *Left* to **Top** *(see illustration)*.



36. Enter **0.25in** for the *Caption Reserve*.

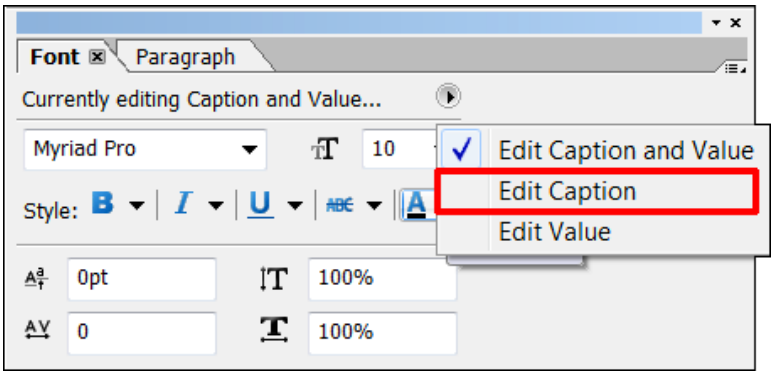
37. Enter **0in** for each of the four margins.

38. Enter **.5in** for *X* and **1.1875in** for *Y*.

39. Enter **2.4375in** for *Width* and **0.5625in** for *Height*.

40. Open the *Font* palette if it is not already opened.

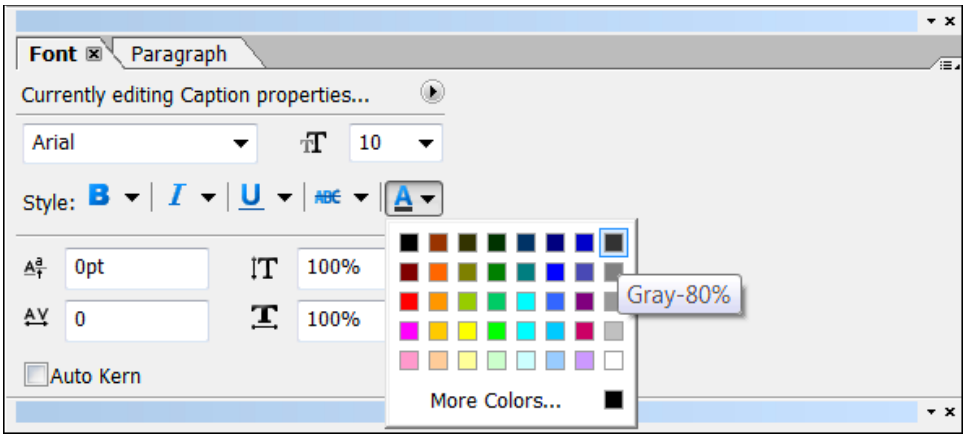
41. Click *Currently editing Caption and Value properties* in the Font Palette and select **Edit Caption**.



42. Click the font dropdown and select **Arial**.

43. Click the size dropdown and select **10pt**.

44. Click the color dropdown and select **Gray-80%** *(see illustration)*.

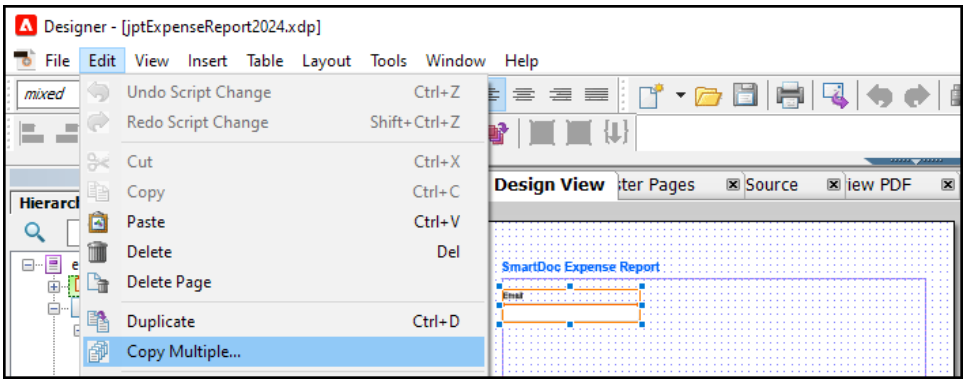


45. Click **Currently editing Caption properties** in the Font Palette and select **Edit Value**.

46. Click the font dropdown and select **Georgia**.

47. Click the size dropdown and select **10pt**.

48. Click the **Edit** menu and select **Copy Multiple** *(see illustration)*.

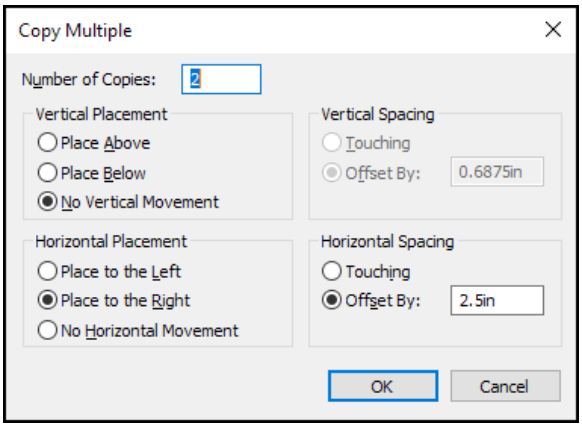


49. Enter **2** for *Number of Copies*.

50. Select **No Vertical Movement** *(see illustration)*.

51. Select **Place to the Right** *(see illustration)*.

52. Select **Offset By** and enter **2.5in** *(see illustration)*.



53. Click **OK** and 2 new fields will be created.

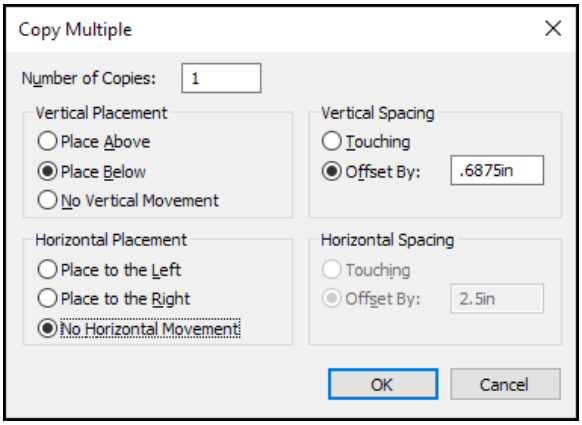
54. With all three fields selected, click **Edit – Copy Multiple** again.

55. Enter **1** for *Number of Copies*.

56. Select **Place Below** for *Vertical Placement*.

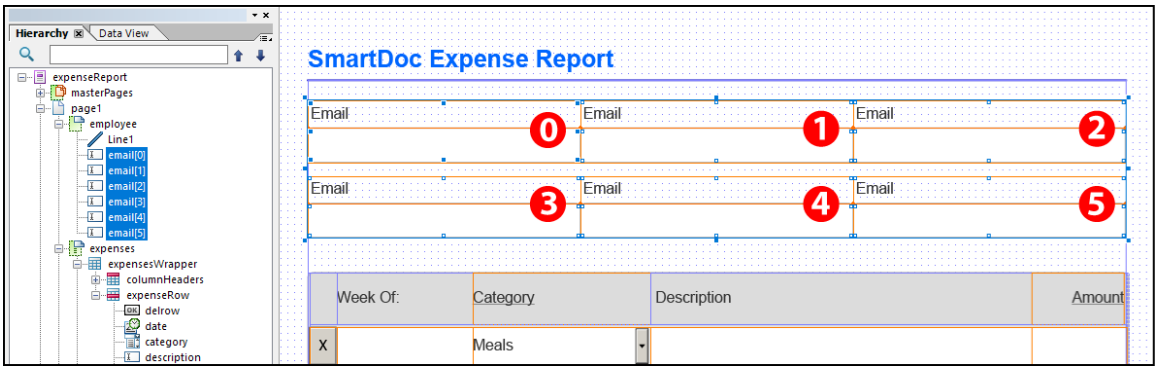
57. Select **Offset By** and enter **.6875in** for *Vertical Spacing*.

58. Select **No Horizontal Movement**. Your settings should look like this.



59. Click **OK**.

60. Move **email[5]** up in the *Hierarchy* palette so it becomes your second email field. This action should re-order your fields in the Hierarchy panel to match their order on the form *(see illustration)*.



61. Right-click the email[1] item in the *Hierarchy* palette and select **Rename Object**.

62. Enter **firstName** for the name. Notice that the remaining email fields have been renumbered in the hierarchy.

63. Select the *Field* tab of the *Object* palette and enter **First Name** for the caption.

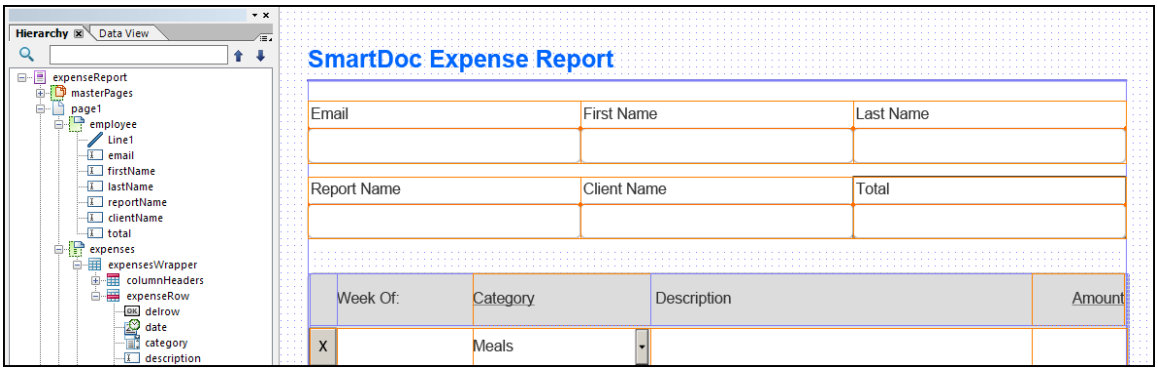
64. Right-click the email[1] item in the *Hierarchy* and select **Rename Object**.

65. Enter **lastName** for the name.

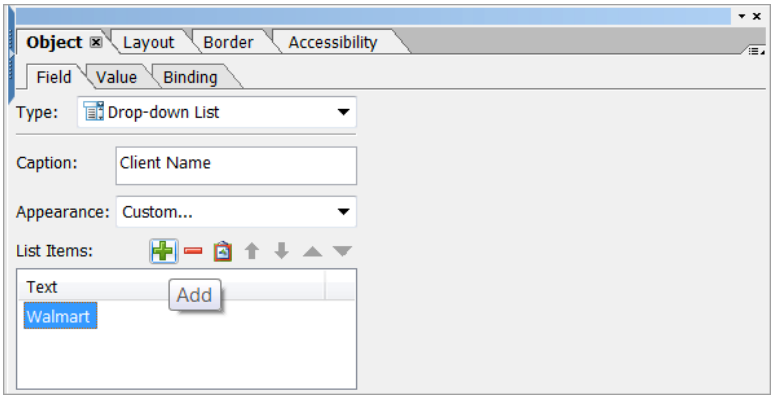
66. Select the *Field* tab of the *Object* palette and enter **Last Name** for the caption.

67. Right-click the email[1] item in the *Hierarchy* and select **Rename Object**.

68. Enter **reportName** for the name.
69. Select the *Field* tab of the *Object* palette and enter **Report Name** for the caption.
70. Right-click the email[1] item in the *Hierarchy* and select **Rename Object**.
71. Enter **clientName** for the name.
72. Select the *Field* tab of the *Object* palette and enter **Client Name** for the caption.
73. Right-click the email[1] item in the *Hierarchy* and select **Rename Object**.
74. Enter **total** for the name.
75. Select the *Field* tab of the *Object* palette and enter **Total** for the caption. Your form should now look like this.



76. Select **clientName** and change the *Type* to **Dropdown List**. The Type property can be found on the *Field* tab of the *Object* palette *(see illustration)*.
77. Click **Add** *(the plus icon)* on List Items *(see illustration)*.
78. Enter **Walmart** as a List Item in the new Dropdown List.



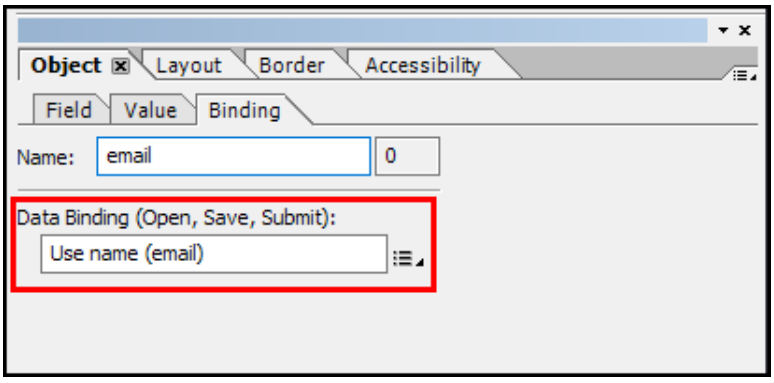
79. Click **Add** *(the plus icon)* again and add **Amazon, Target, Costco, Home Depot** as *List Items*. You will need to click **Add** each time to enter a new list item.
80. Select **total** and change the *Type* to **Numeric Field**. The *Type* property can be found on the *Field* tab of the *Object* palette.
81. Click the **Patterns** button on the *Field* tab of the *Object* palette.

82. Make sure the field has a useful *Display Pattern*. Here is one suggestion: **num.currency{}**

83. Click **OK**.

84. Select your Email field and open the *Binding* tab of the *Object* palette.

85. Notice the *Data binding* property is set to **Use name** *(see illustration)*.



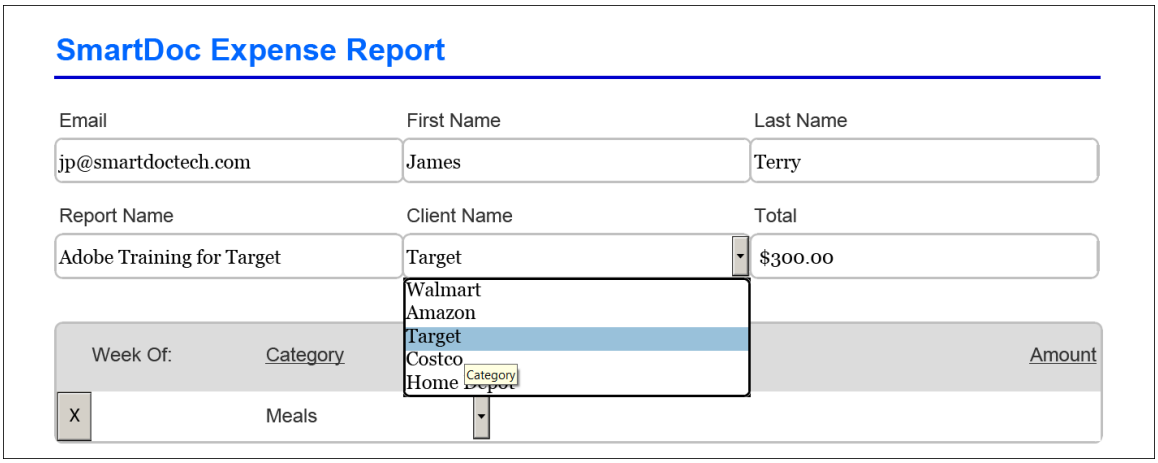
86. Also check the other fields to confirm their binding is set to **Use Name**.

87. Select **File – Save**.

88. Click **Preview PDF** to see your form in action.

89. Enter values for your fields.

You should see something that looks like this *(see illustration)*.

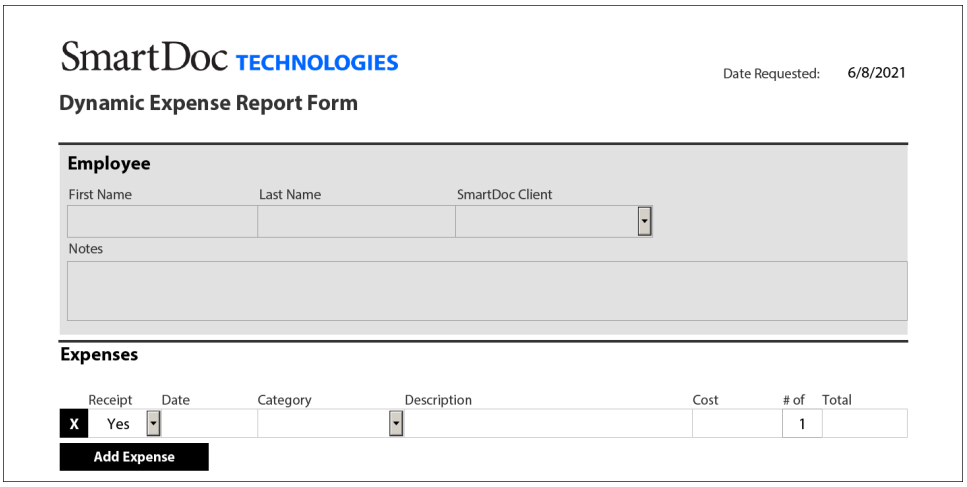


Dynamic Forms

Now that you know where everything is, let's review some important concepts about dynamic forms: master and body pages, subforms and flow, and tables. These are usually the concepts that trip people up when they start creating Designer forms. By mastering these ideas early, you'll avoid many pitfalls when designing and implementing your forms in the later exercises.

Examples

Dynamic Forms will grow or shrink based on data or user interaction. We will review a Dynamic Expense Report *(see illustration)* that will grow or shrink depending on the amount of expense items. Each new expense item will be represented by a new expense row. And we will review a Dynamic Insurance Form that will change options based on the type of policy a user selects.



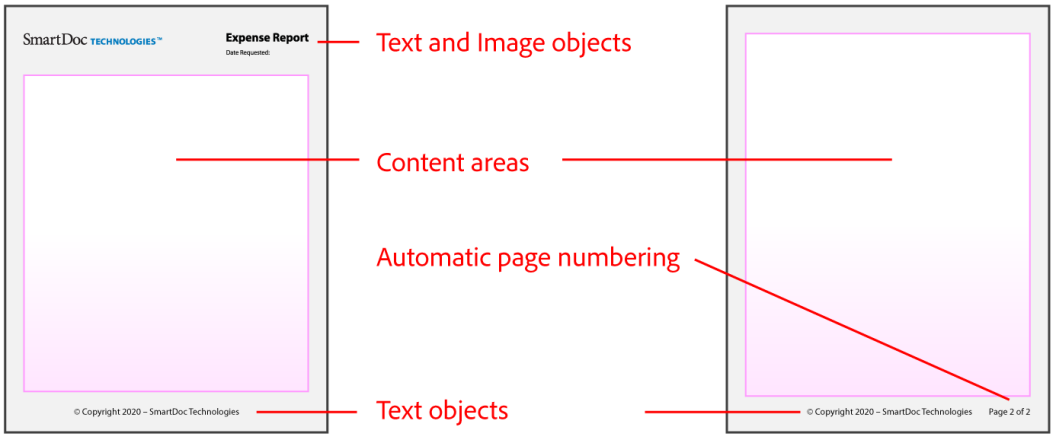
Master and Body Pages

Designer documents consist of master pages and body pages. Master pages define the layout and background elements of your form. You should put common page items like page numbers, repeating logos, and footer information on your master pages. Body pages should contain all the form objects that are unique to a particular page.

When you're working with multipage forms, it's quite possible that you'll need a different look and feel on different pages. For instance, an account opening form may require three pages of interactive objects and six pages of legal text in columns. In this case, it makes sense to define two different master pages: one for the body pages with the interactive objects and one for the body pages with the legal text.

Master pages

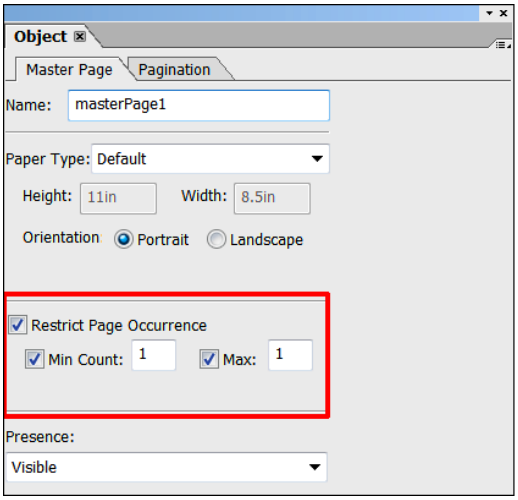
You edit your master pages in the Master Pages tab in the Layout Editor. If you don't see this tab, choose View - Master Pages. Every Designer file must have at least 1 master page. The SmartDoc Expense Report *(from the Student Files)* has two master pages. The first master page *(masterPage1 on the left)* contains the document's header on the top. The second master page *(masterPage2 on the right)* contains automatic page numbering on the bottom.

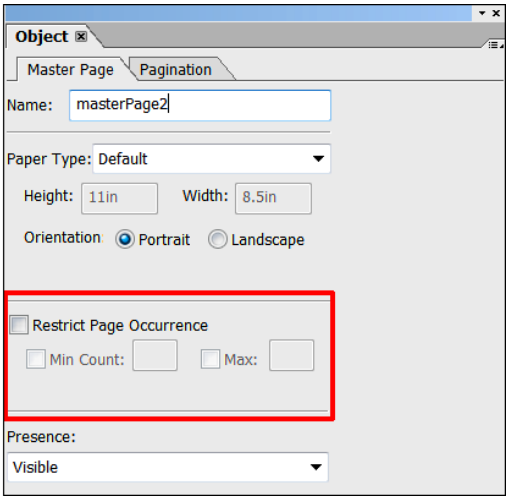


Both master pages contain Content Area objects which are the pink boxes. These objects define the outer bounds of the layout area for each associated body page. Content Area objects are particularly important for dynamic forms with flowing content. In the SmartDoc Expense Report form, the content area on masterPage2 is larger than the content area on masterPage1. This larger content area means the body pages associated with masterPage2 can display more information than the body page associated with masterPage1. Content Area objects are regular Designer objects and are found in the Standard Object Library palette.

The SmartDoc Expense Report is a dynamic form, and a dynamic form can grow to accommodate every data item. Whether it has 1 page or 1000 pages, it will always have one instance of masterPage1. This is achieved by setting the Restrict Page Occurrence property on masterPage1 as shown in the illustration on the left. So, the master page with the logo and header will only appear once and will always appear once, on the first page.

However, notice in the illustration on the right that masterPage2 doesn't have any page restrictions. Since it is the next master page in the hierarchy, it will be used for page 2 and for all subsequent pages in the dynamic form.



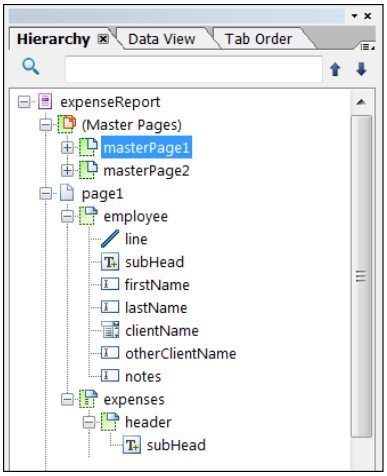


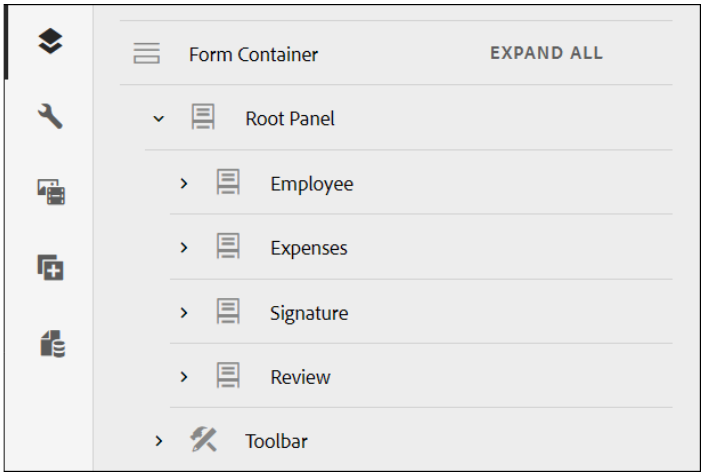
Body pages

Most of your form objects will be placed on body pages. Each body page will reference a master page. You can view your body page objects by switching from the Master Pages tab to the Design View tab in Designer. Body pages are subform objects in Designer and as such are treated just like any other subform.

Subforms and Flow

A *subform* is a container for 1 or more form objects. Subforms define the structure of your form, and you can nest a child subform inside a parent subform. In fact, there's no limit to the nesting you can do with subforms. Subforms in Designer *(left)* are like panels in AEM adaptive forms *(right)*.





The child objects of a subform can be positioned or flowed within the subform. You control this with the Content property of the subform. Subforms can be set to Positioned which will enable you to place your child objects on an X/Y grid relative to the subform. Subforms can also be set to Flowed and there are 3 different flow directions for the child objects: Top to Bottom, Western Text, and Right to Left. You will see each of these examples in the exercises.

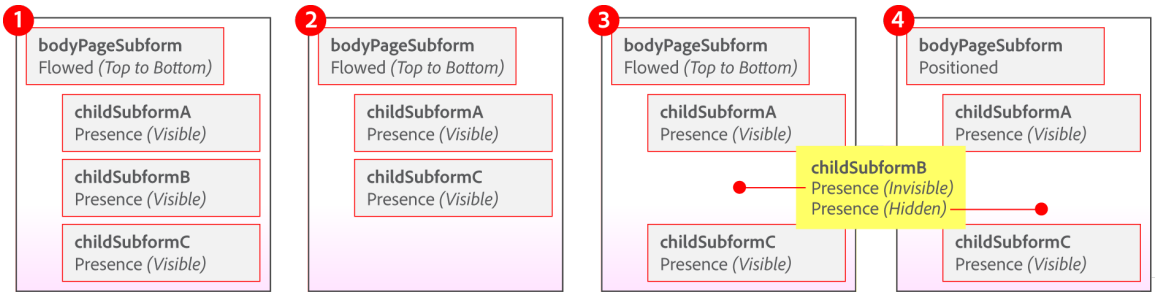
Subforms work in conjunction with the content areas that you learned about earlier. Content areas control where objects are located, and subforms control how objects are placed as the form is rendered. By grouping certain objects together in a subform, you can be assured that they'll be consistently positioned relative to each other as the form is rendered.

Depending on the incoming data or the user interaction, subforms can be repeated, expanded, or hidden. Repeating and expanding subforms are ideal for forms with repeating data like the SmartDoc Expense Report. Repeating subforms are child subforms that are placed inside an expanding parent subform. Because subforms are separate, independent sections of a form, they can be hidden or shown based on the needs of the user or other business logic.

Subforms can be challenging and complex, but they are what make dynamic documents dynamic. Without subforms and tables, which are just a type of subform, you wouldn't be able to create dynamic documents that grow and shrink based on data or user interaction.

Hidden and invisible subforms

You can hide and show subforms by setting the Presence property. The three most common values for the Presence property are *Visible*, *Hidden (Exclude from Layout)*, and *Invisible*. Depending on the option you choose for the Presence property and the Content property of the parent subform, different effects will result.



Consider the four scenarios illustrated here.

- In example #1, the parent is set to *Flowed* and all children are set to *Visible*.
- In example #2, the parent is also set to *Flowed* but childSubformB is set to *Hidden (Exclude from Layout)*. This will cause childSubformC and all subsequent subforms to automatically move up in the form to occupy the position previously held by childSubformB.
- In example #3, the parent is also set to *Flowed* but childSubformB is set to *Invisible*. It is true that childSubformB has disappeared but notice that childSubformC hasn't moved up like it did in the previous example. An object set to *Invisible* will still retain its place in the layout, it will simply not appear.
- In example #4, the parent is now set to *Positioned* so all child subforms will maintain their X and Y position relative to the parent. Even when childSubformB is set to *Hidden (Exclude from Layout)* childSubformC will not move up. A subform is only truly hidden when its parent is set to flowed.

Notice that childSubformB is set to Invisible in Example 3 and Hidden in Example 4, but the end result is the same. The gap in Example 4 is caused by the parent subform's Positioned setting.

Pagination and Subform Flow Is Different in HTML Forms

Pagination and subform flow work differently in the HTML renderings of your Designer files than they do in the PDF renderings. For instance, the repeating expense rows of the SmartDoc Expense Report will flow from page to page in a PDF rendering of the form. However, in the HTML rendering, these repeating rows will simply repeat on the same HTML page without flowing to a secondary page.

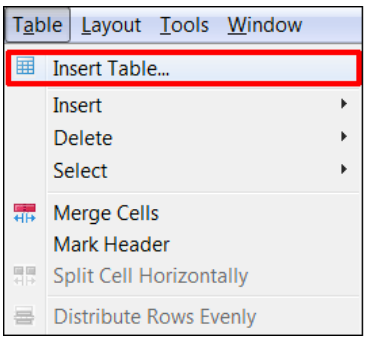
Tables

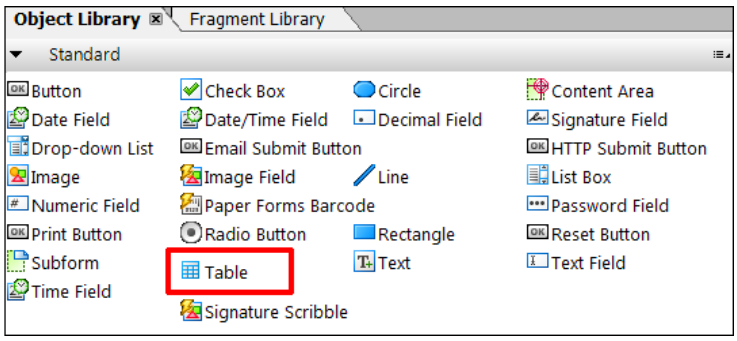
Tables in Designer are a lot like tables in a word processing program. You can use tables to organize data into a structured grid of related objects. But in Designer, tables are also complex container objects, so you can work with them just like you work with subforms.

The Table menu and Table object

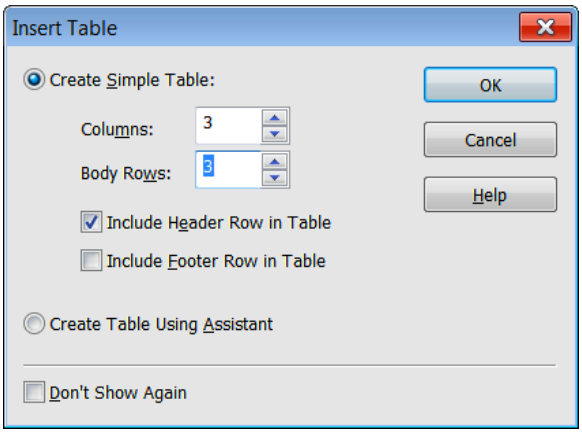
You can add a simple table to your form in either of the following ways.

- Click the *Table* menu and select *Insert Table (left)*
- Drag and drop a *Table object* from the Standard Object Library onto your form *(right)*





In both cases, Designer will launch the Insert Table dialog box *(see illustration)*.



The Insert Table dialog box enables you to create a simple table or to launch the Table Assistant.

The Table Assistant

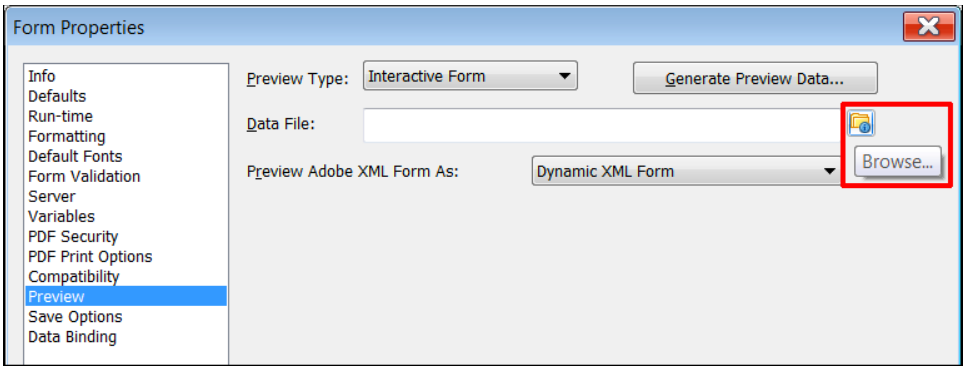
The Table Assistant is a multipage wizard tool, and it provides more features and additional information about tables. One of the additional tools in the Table Assistant is the Row Shading page, which enables you to set different colors for alternating rows. This is a useful design method to use on your forms. Changing the colors on alternating rows makes it easier for your users to read and comprehend table-based data.

Exercises

In these exercises, you will work with dynamic forms. Because these forms are dynamic, it's best to test them with sample data files to see how they will expand and contract. Ideally, you will test them with more than one data file to show different runtime possibilities.

The Expense Report with Dynamic Subforms

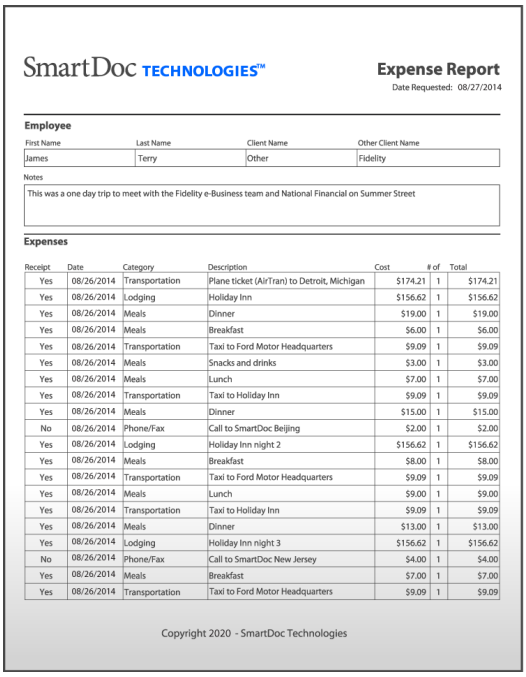
1. Select **File – Open** and navigate to your Student Files.
2. Select *expenseReportCompleted.xdp* and click **Open**.
3. Select **File – Form Properties**. Designer opens the Form Properties dialog box.
4. Select **Preview** and click the **Browse** icon *(see illustration)* to right of the *Data File* field.



5. Navigate to your Student Files.
6. Select *expenseDataShort.xml* and click **Open**.
7. Click **OK** to close the *Form Properties* dialog box.
8. Select **Preview PDF** to see the expense report fill with data. You should see 2 pages of expense report data.

	Receipt	Date	Category	Description	Cost	# of	Total
X	Yes	08/01/2009	Transportation	Train ticket to New York	$16.00	1	$16.00
X	No	08/01/2009	Meals	Breakfast with client	$10.00	1	$10.00
X	Yes	08/02/2009	Transportation	Taxi to office	$15.00	1	$15.00

9. Select **Design View** to close the PDF preview.
10. Select **File – Form Properties** again.
11. Select **Preview** and click the **Browse** icon to right of the *Data File* field.
12. Navigate to your Student Files.
13. Select *expenseDataLong.xml* and click **Open**.
14. Click **OK** to close the *Form Properties* dialog box.
15. Select **Preview PDF** to see the expense report fill with data. You will see more pages and more expense rows.



SmartDoc TECHNOLOGIES™

Expense Report

Date Requested: 08/27/2014

Employee

First Name	Last Name	Client Name	Other Client Name
James	Terry	Other	Fidelity

Notes

This was a one day trip to meet with the Fidelity e-Business team and National Financial on Summer Street

Expenses

Receipt	Date	Category	Description	Cost	# of	Total
Yes	08/26/2014	Transportation	Plane ticket (AirTran) to Detroit, Michigan	$174.21	1	$174.21
Yes	08/26/2014	Lodging	Holiday Inn	$156.62	1	$156.62
Yes	08/26/2014	Meals	Dinner	$19.00	1	$19.00
Yes	08/26/2014	Meals	Breakfast	$6.00	1	$6.00
Yes	08/26/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09
Yes	08/26/2014	Meals	Snacks and drinks	$3.00	1	$3.00
Yes	08/26/2014	Meals	Lunch	$7.00	1	$7.00
Yes	08/26/2014	Transportation	Taxi to Holiday Inn	$9.09	1	$9.09
Yes	08/26/2014	Meals	Dinner	$15.00	1	$15.00
No	08/26/2014	Phone/Fax	Call to SmartDoc Beijing	$2.00	1	$2.00
Yes	08/26/2014	Lodging	Holiday Inn night 2	$156.62	1	$156.62
Yes	08/26/2014	Meals	Breakfast	$8.00	1	$8.00
Yes	08/26/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09
Yes	08/26/2014	Meals	Lunch	$9.00	1	$9.00
Yes	08/26/2014	Transportation	Taxi to Holiday Inn	$9.09	1	$9.09
Yes	08/26/2014	Meals	Dinner	$13.00	1	$13.00
Yes	08/26/2014	Lodging	Holiday Inn night 3	$156.62	1	$156.62
No	08/26/2014	Phone/Fax	Call to SmartDoc New Jersey	$4.00	1	$4.00
Yes	08/26/2014	Meals	Breakfast	$7.00	1	$7.00
Yes	08/26/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09

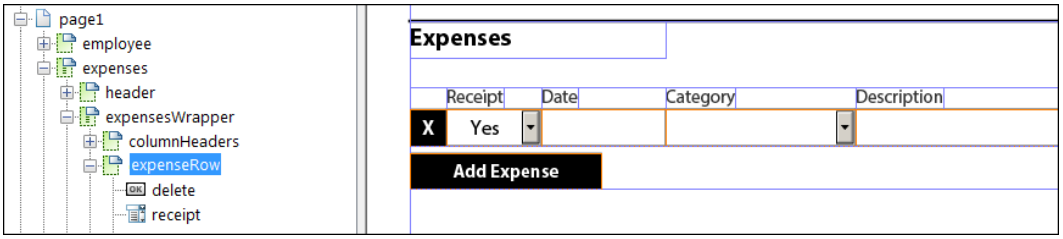
Copyright 2020 - SmartDoc Technologies

Receipt	Date	Category	Description	Cost	# of	Total
Yes	08/27/2014	Transportation	Plane ticket (AirTran) to Detroit, Michigan	$174.21	1	$174.21
Yes	08/27/2014	Lodging	Holiday Inn	$156.62	1	$156.62
Yes	08/27/2014	Meals	Dinner	$19.00	1	$19.00
Yes	08/27/2014	Meals	Breakfast	$6.00	1	$6.00
Yes	08/27/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09
Yes	08/27/2014	Meals	Snacks and drinks	$3.00	1	$3.00
Yes	08/27/2014	Meals	Lunch	$7.00	1	$7.00
Yes	08/27/2014	Transportation	Taxi to Holiday Inn	$9.09	1	$9.09
Yes	08/27/2014	Meals	Dinner	$15.00	1	$15.00
No	08/27/2014	Phone/Fax	Call to SmartDoc Beijing	$2.00	1	$2.00
Yes	08/27/2014	Lodging	Holiday Inn night 2	$156.62	1	$156.62
Yes	08/27/2014	Meals	Breakfast	$8.00	1	$8.00
Yes	08/27/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09
Yes	08/27/2014	Meals	Lunch	$9.00	1	$9.00
Yes	08/27/2014	Transportation	Taxi to Holiday Inn	$9.09	1	$9.09
Yes	08/27/2014	Meals	Dinner	$13.00	1	$13.00
Yes	08/27/2014	Lodging	Holiday Inn night 3	$156.62	1	$156.62
No	08/27/2014	Phone/Fax	Call to SmartDoc New Jersey	$4.00	1	$4.00
Yes	08/27/2014	Meals	Breakfast	$7.00	1	$7.00
Yes	08/27/2014	Transportation	Taxi to Ford Motor Headquarters	$9.09	1	$9.09

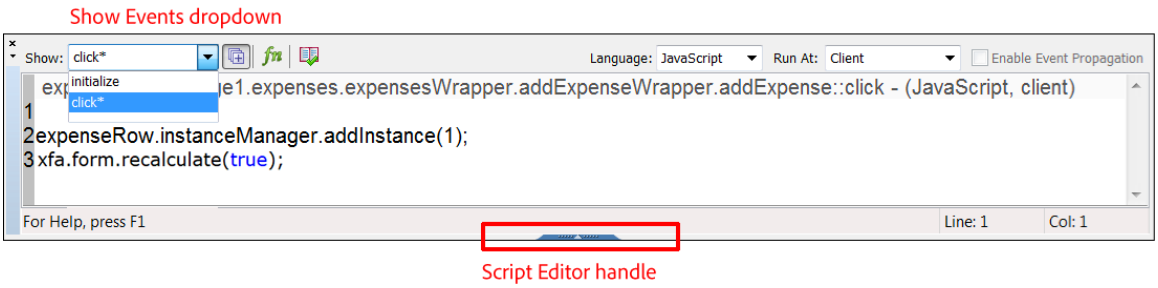
Sub Total: $751.00
Less Cash Advance: $.00
TOTAL $751.00

Copyright 2020 - SmartDoc Technologies Page 2 of 2

16. Select **Design View** to close the PDF preview.
17. Select the **Hierarchy** panel if it is not already selected.
18. Expand the hierarchy by selecting **page 1 – expenses – expensesWrapper – expenseRow** *(see illustration)*.



19. With **expenseRow** selected, select the **Binding** tab of the Object palette.
20. Notice the expenseRow has a data binding of **$.ExpenseItem[*]** and is set to repeat for each data item. This is why the form is dynamic. An *expenseRow* is created for each *ExpenseItem*.
21. Select **expensesWrapper** *(the parent of expenseRow)*.
22. Select the **Subform** tab of the *Object* palette.
23. Notice the expensesWrapper is set to *Flowed* with a Flow Direction of *Top to Bottom*. This will also be true for the parent subform *(expenses)* and the grand-parent subform *(page1)*. A body page is just another subform.
24. Select the **addExpense** button. The button's caption is *Add Expense*.
25. Expand the **Script Editor** by selecting the handle outlined in red *(see illustration)* and dragging the window downwards to make the editor larger.



26. Click the **Show Events** dropdown *(see illustration above)* and select the **click** event.

27. You will see the script that creates a new instance of the expenseRow.

expenseRow.instanceManager.addInstance(1);
xfa.form.recalculate(true);

The Dynamic Insurance Form

28. Select **File – Open** and navigate to your Student Files.

29. Select *dynamicInsuranceForm.xdp* and click *Open*.

30. Select **Preview PDF** to see the form in action.

31. Select the different types of insurance policies *(see illustration)*.

32. Notice the dynamic options will change each time you select a different insurance policy.

33. Click **Design View** to close the preview.

Subforms and Flow

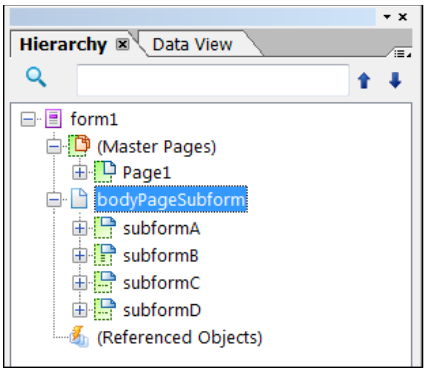
As with other Designer features, subforms are easy to understand if you look at an example.

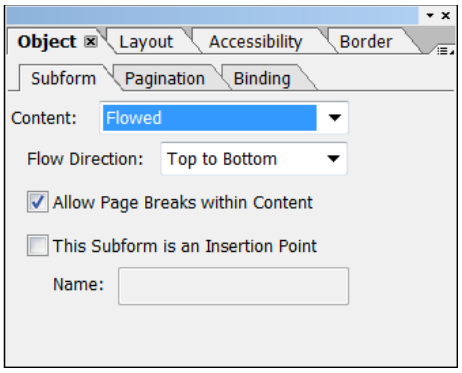
34. Select **File – Open** and navigate to your Student Files.

35. Select *basicSubform.xdp* and click **Open**.

36. With the **bodyPageSubform** selected *(left)*, click the **Subform** tab of the Object palette *(right)*.

37. Notice the Content property is set to **Flowed** and the Flow Direction is **Top to Bottom**. This property governs the 4 child subforms. Subforms A, B, C, and D will be Flowed, Top to Bottom.





38. Select **subformA** and notice its Content property is set to **Positioned**. The child objects of subformA will be positioned exactly where you place them in Design View. Each child object will maintain its X and Y position relative to subformA regardless of where subform A appears on the rendered form.

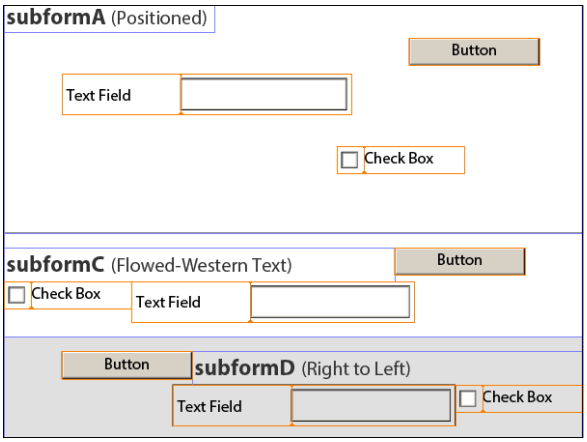
39. Select **subformB** and notice its Content property is set to **Flowed** and the Flow Direction is **Top to Bottom**, just like bodyPageSubform. The child objects will be flowed in the layout according to their position in the hierarchy.

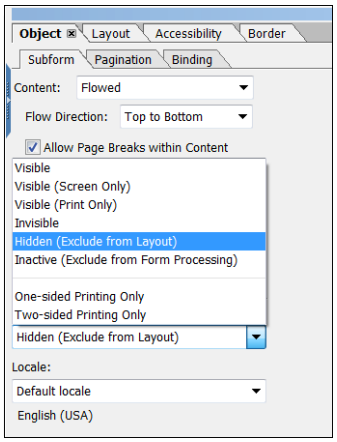
40. Select **subformC** and notice its Content property is set to **Flowed** and the Flow Direction is **Western Text**. The child objects will be placed left to right and wrapped onto the next line if more space is required.

41. Select **subformD** and notice its Content property is set to **Flowed** and the Flow Direction is **Right to Left**. The child objects will be placed right to left and wrapped onto the next line if more space is required.

42. Select **subformB** and notice its Presence property is set to **Visible**.

43. Click the Presence property dropdown and select **Hidden (Exclude from Layout)**.



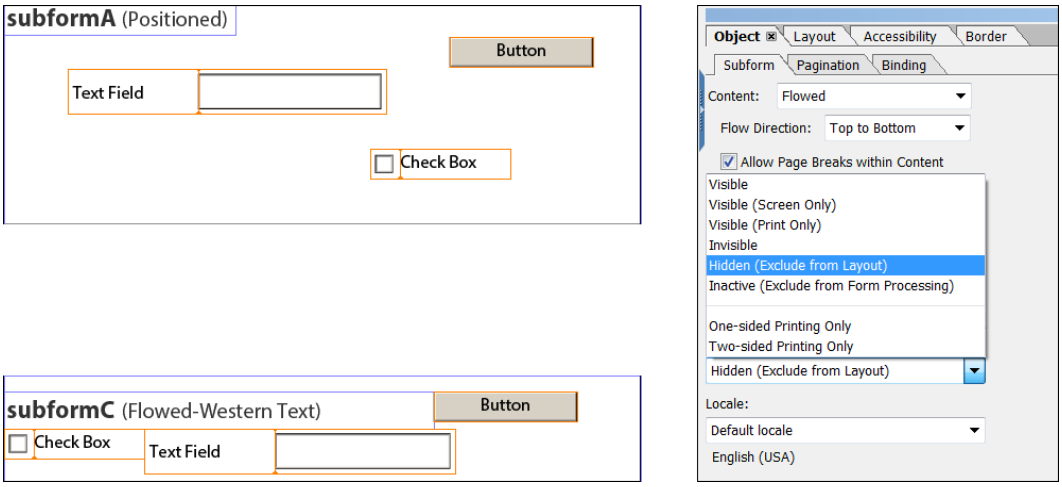


The form content below subformB will now move up because the parent *(bodyPageSubform)* is set to *Flowed* and the Flow Direction is *Top to Bottom*.

44. Select **subformB** again and change its Presence property to **Visible**.

45. Select **bodyPageSubform** and change its Content property is set to **Positioned**.

46. Select **subformB** and change its Presence property back to **Hidden (Exclude from Layout)**. This time, the form content below subformB does not move up because the parent *(bodyPageSubform)* is set to Positioned.

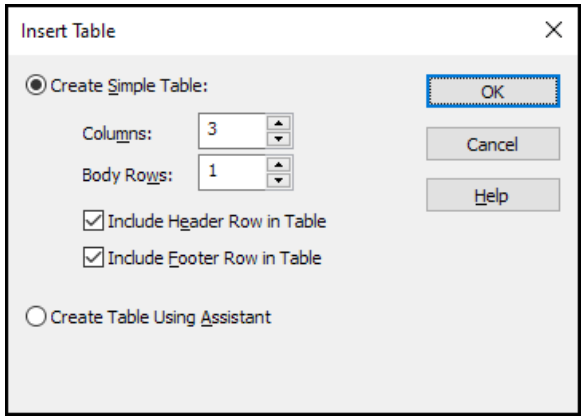


Note: *Remember, it is always the parent that controls the layout of the child objects.*

Create a Simple Table

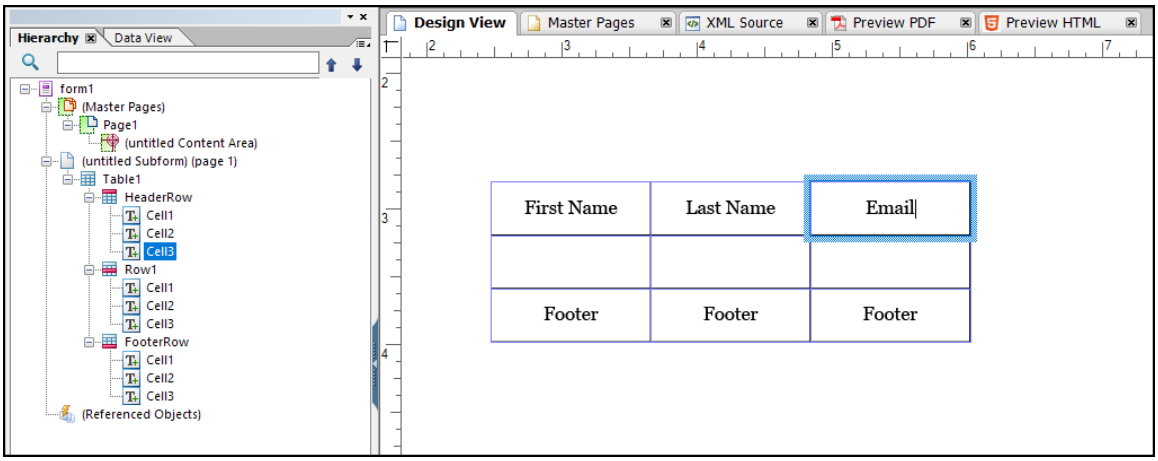
Follow these steps to create a simple table.

1. Select **File – New**. Designer launches the *New Form Assistant*.
2. Select **Use A Blank Form**. This is the default.
3. Click **Next** to continue.
4. Keep the defaults, and click **Finish** to create your new form.
5. Select the **Table** menu. The Table menu is on top, and it provides many tools for working with tables.
6. Select **Insert Table** and you will see the *Insert Table* dialog box.
7. Keep the default of *Create Simple Table* and select the **Include Header Row in Table** and the **Include Footer Row in Table** options *(see illustration)*.



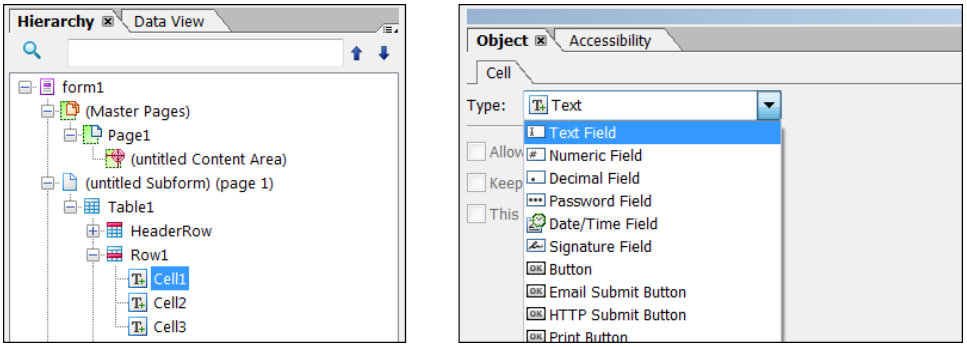
8. Click **OK**.
9. Designer will create a simple table in the Layout Editor *(see illustration)*.

10. Double-click on each Header and change their names so you have a column for **First Name**, **Last Name**, and **Email** *(see illustration)*.



11. Expand **Row1** in the Hierarchy palette *(see illustration on the left)* and you will notice that the form object in each cell is a Text object.

12. Select the **Cell1** text object and change its type to **Text Field**. You can do this in the *Type* property of the *Object* palette *(see illustration on the right)*.

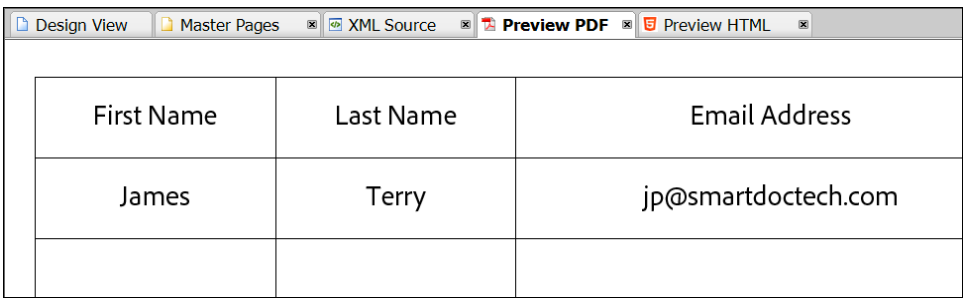


13. Select Cell2 and Cell3 and set their Type properties to Text Field so every cell in Row1 has a Text Field input.

14. Click **Cell3** and select the *Layout* palette.

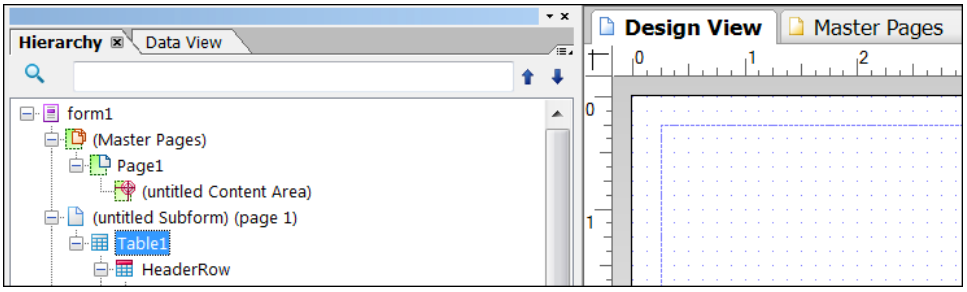
15. Enter **2.5** for the Width. This will enable you to enter longer email addresses.

16. Click **Preview PDF** and enter your name and email address into the first row.



17. Select **Design View** to close the PDF preview.

18. Right-click on the Table1 object in the Hierarchy palette *(see illustration)* and select Rename Object.

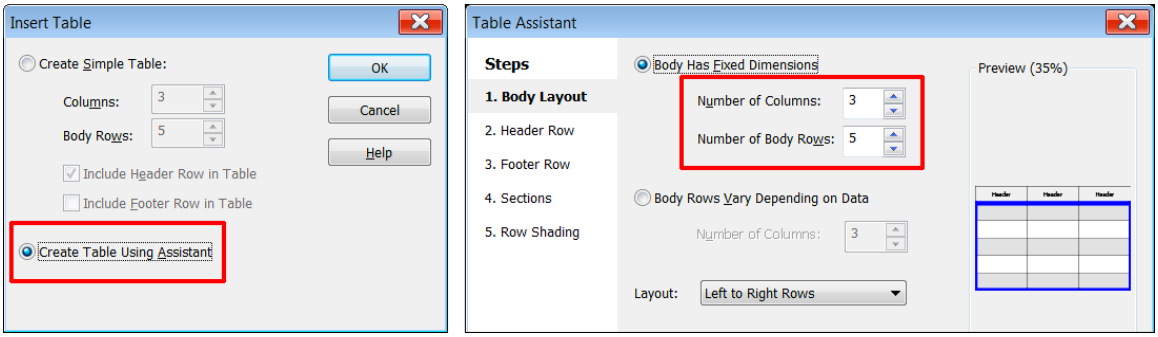


19. Enter **SimpleTable** for the name.

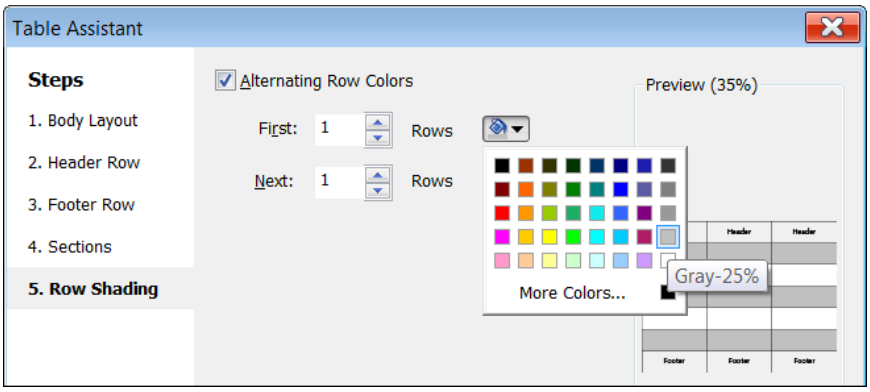
Use the Table Assistant

Follow these steps to create a more detailed table with the Table Assistant

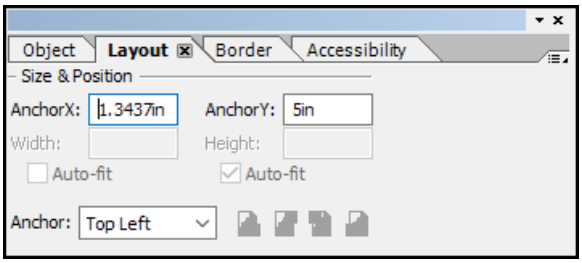
20. Click the **Table** menu and select **Insert Table**.
21. Select **Create Table Using Assistant** *(see illustration)* and click **OK**.
22. Select **3** Columns and **5** Body Rows *(see illustration)*.



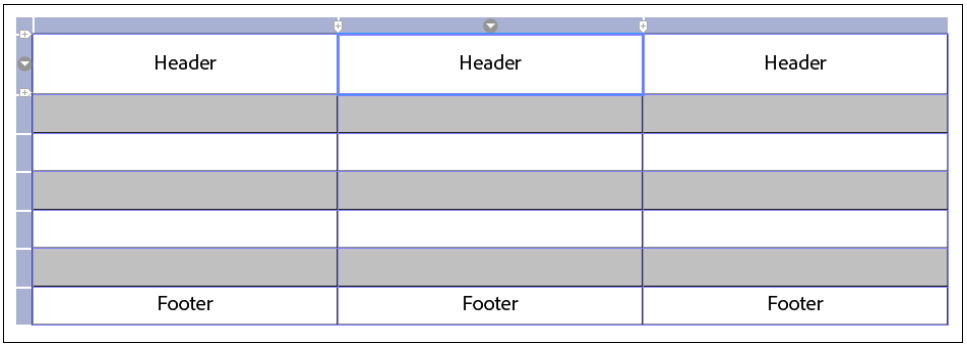
23. Click **Next**.
24. Select **Has Header Row** and click **Next**.
25. Select **Has Footer Row** and click **Next**.
26. Select **Has Body Rows And No Sections** and click **Next**.
27. Select **Alternating Row Colors** and select a **Gray-25%** for the First row.
28. Select **white** for the second row. This should be the default.



29. Click **Finish**.
30. Select the Layout palette if it is not already selected.
31. Enter an AnchorX and an AnchorY value so your new table does not overlap your SimpleTable.



32. You now have a table with alternating row colors *(see illustration)*.

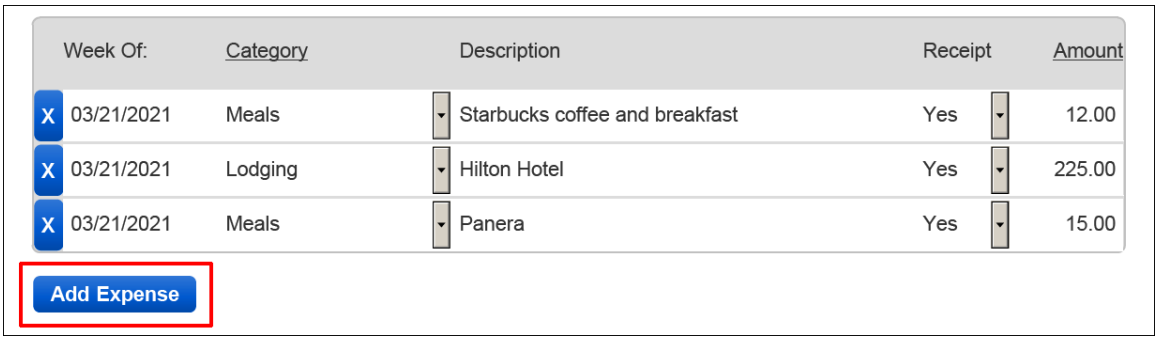


The Expense Report with a Dynamic Table

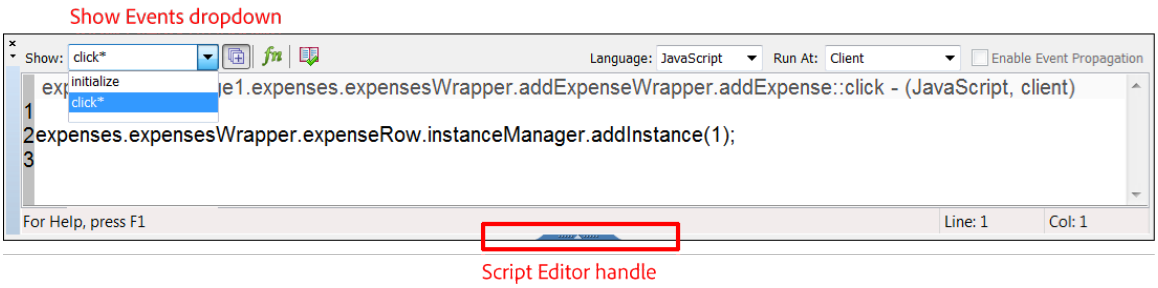
Follow these steps to see how a dynamic table is similar to the dynamic subforms in the SmartDoc Expense Report.

1. Select **File – Open** and navigate to your Student Files.
2. Select *ExpenseReport2020_finished.xdp* and click **Open**.
3. Select **File – Form Properties**.
4. Select **Preview** and click the **Browse** icon to right of the *Data File* field.
5. Navigate to your Student Files.

6. Select the **ExpenseReport2020_3records.xml** file and click **Open**.
7. Click **OK** to close the *Form Properties* dialog box.
8. Select **Preview PDF**.
9. Click the **Add an Expense** button and notice it has the same functionality as the other dynamic expense report.



10. Select **Design View** to close the PDF preview.
11. Expand the **expenses** subform in the Hierarchy palette.
12. Expand the **expensesWrapper** table so you can select the **expenseRow** table row.
13. With **expenseRow** selected, select the **Binding** tab of the Object palette.
14. Notice the expenseRow has a data binding of **$.ExpenseItem[*]** and is set to repeat for each data item. This is exactly the same as the previous example with the dynamic subforms.
15. Select the **addExpense** button. The button's caption is *Add Expense*.
16. Expand the **Script Editor** by selecting the handle outlined in red *(see illustration)* and dragging the window downwards to make the editor larger.



17. Click the **Show Events** dropdown *(see illustration above)* and select the **click** event.

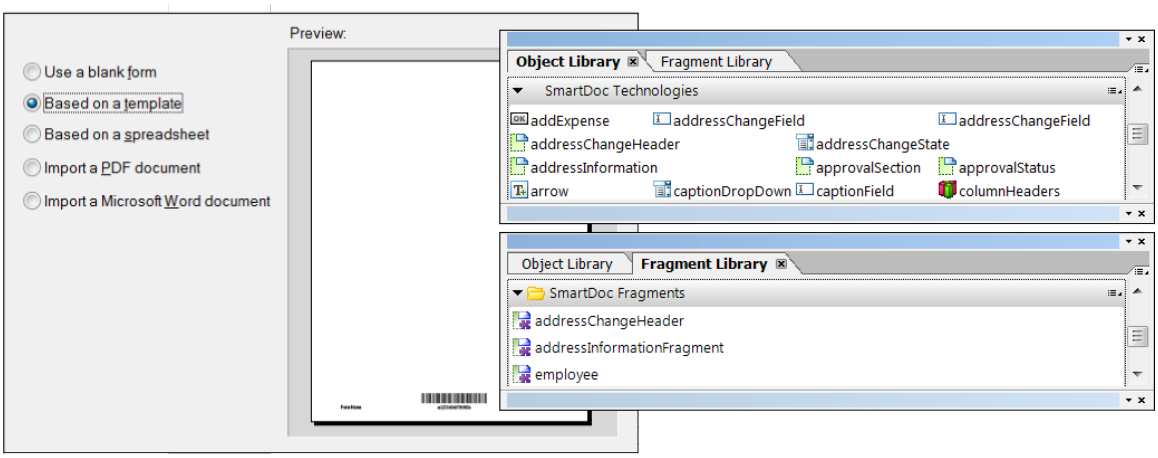
You will see the script that creates a new instance of the expenseRow.

expenses.expensesWrapper.expenseRow.instanceManager.addInstance(1);
xfa.form.recalculate(true);

> ***Note:*** *This script is slightly different than the other addInstance script. In this case, we need to go further up the hierarchy to the common expenses node in order to create a proper reference to the expenseRow table row. You will learn more about creating references in the Scripting section.*

Templates, Objects, and Fragments

It is a best practice to create your forms based on Designer's templates, custom objects, and fragments. Each of these enables you to create standardized components that you can use multiple times in your form system.

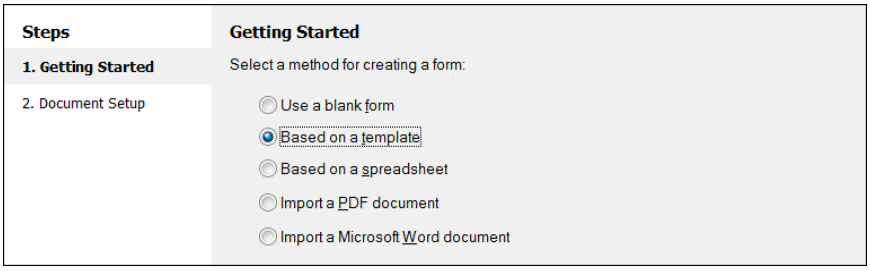


You will achieve form consistency through the use of reusable components. And you will achieve management efficiency because you can make updates once and have them automatically propagate through your form system.

Templates

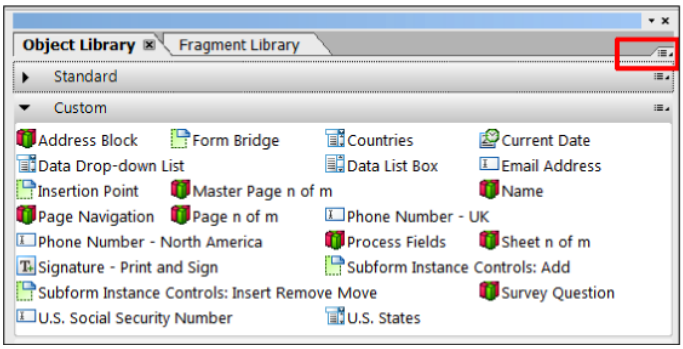
Designer enables you to create a new form based on a template. You can use the *out-of-the box* Designer templates or create your own custom templates. Templates enable you to set layout properties across many forms or documents. Templates can be used to standardize the following aspects of your forms.

- Layout properties, including page size, margins, headers, and footers
- Graphic design properties, including font, color, and style
- Common script objects and form variables

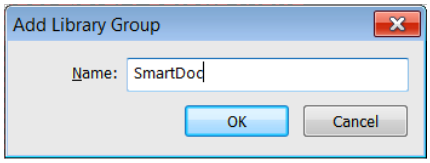


Custom Objects

In addition to custom templates, Designer also enables you to create custom objects for use in your forms. Designer supports the creation of custom objects and custom object libraries. You can create the perfect interface object with the font you want and functionality you need, and then save it for reuse on any form.



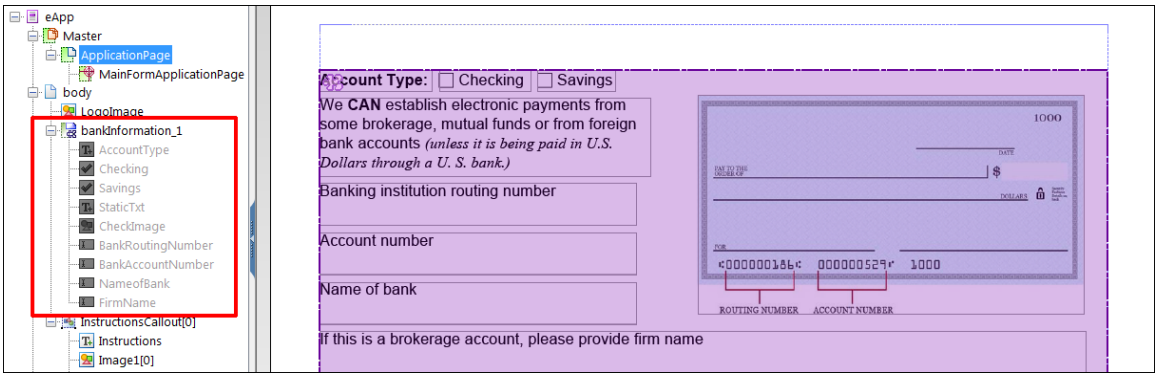
Object Library palette menu



You can create the perfect interface object with the font you want and functionality you need, and then save it for reuse on any form. A custom object can include design elements, data bindings, and JavaScript functionality. You can organize your objects in custom object libraries. You will create a SmartDoc object library in the exercises. The Object Library palette menu is represented by a very small dropdown icon in the upper-right of the Object Library palette, as shown here.

Form Fragments

You can take the standardization concept one step further with form fragments. Fragments enable you to create a section of a form and reuse it across many different forms. A form fragment will appear with a purple tint in Design View and the objects will be grayed-out in the Hierarchy palette *(see illustration)*.

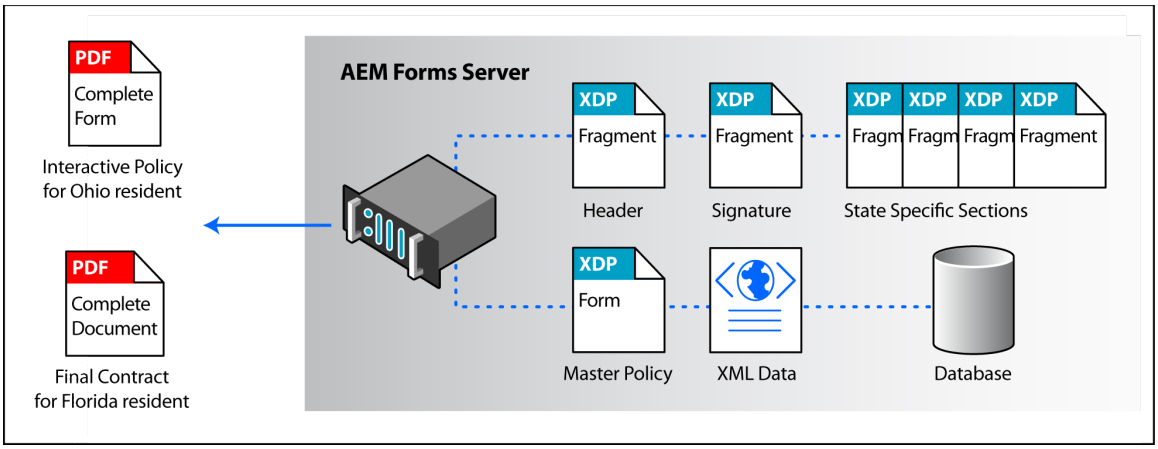


The following points are true about form fragments.

- All instances of a form fragment are exactly the same because each is only a reference to one master form fragment *(which is saved as a separate XDP file)*.
- When the form fragment is updated, all forms that use the fragment are updated automatically.
- Form fragments enable you to manage and maintain a library of forms more efficiently and effectively. You'll no longer need to execute and test the same change across many forms.
- If you do need to edit and change an instance of a form fragment, you can embed it in your form. However, this breaks the reference to the master so automatic updates no longer occur.

Form Fragments are resolved at runtime. This can happen in either of the following ways.

- An AEM Server can render a PDF form from XDP files.
- A Form Author can save a PDF form from Designer.



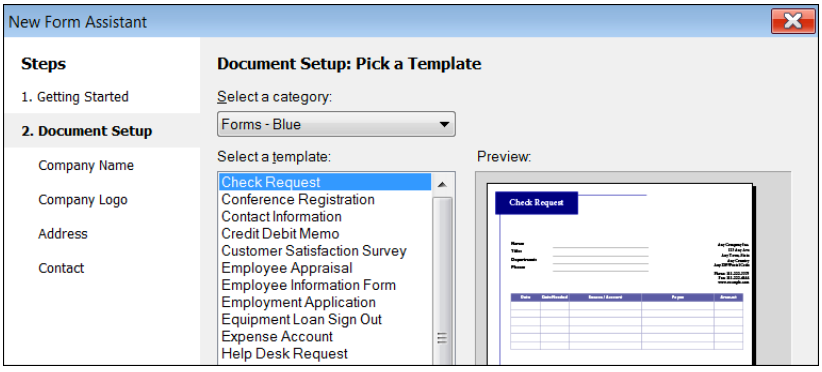
In both of these examples the PDF file will contain the fragment.

Exercises

Use Templates

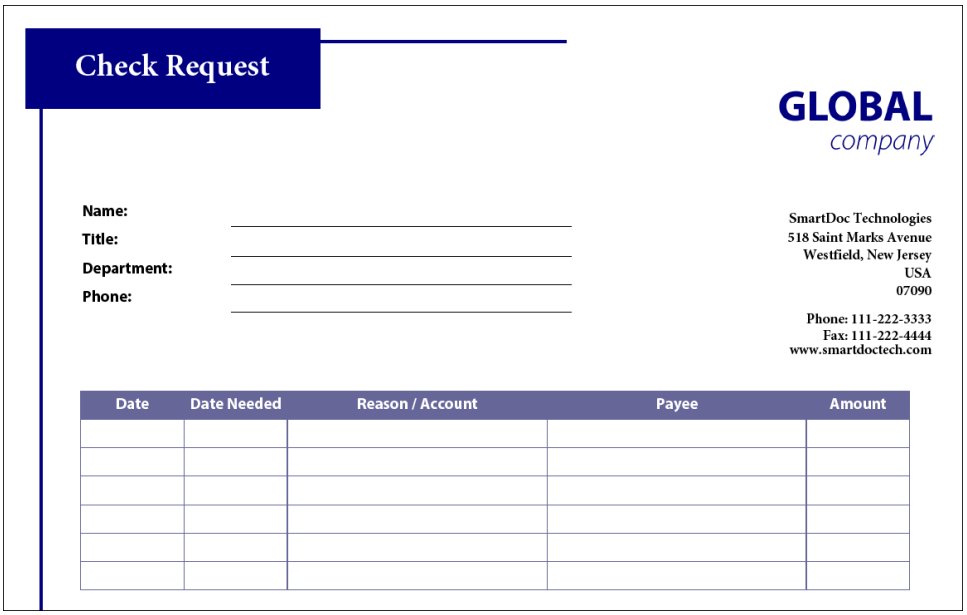
You can see templates in action by following these steps:

1. Open Designer if it is not already open.
2. Select **File – New** to launch the New Form Assistant.
3. Select **Based On A Template**.
4. Click **Next**. Review the many categories and templates included in the Designer program.
5. Click the *Select a category* dropdown and select **Forms – Blue** *(see illustration)*.
6. Select the **Check Request** template.



7. Click **Next**.
8. Enter your company name, and click **Next**.
9. Click **Browse** and navigate to your Student Files.
10. Select *globalLogo.png* and click **Open**.

11. Click **Next**.
12. Use the default address or enter your own, and click **Next**.
13. Use the default contact information or enter your own, and click **Finish**.
14. Click **Preview PDF**.
15. As you can see, your new form has been created and a great deal of work on the form is already completed.

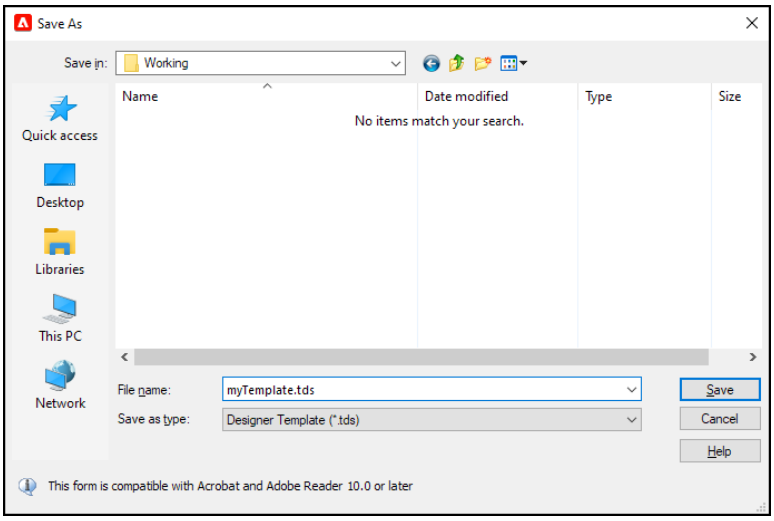


16. Select *Design View*.
17. Select File – Save As and navigate to your working folder. Note: This can be a folder of your own choosing.
18. Enter **formBasedOnTemplate.xdp** as the *File name*.
19. Click the *Save as type* dropdown and select **Adobe XML Form (*.xdp)**.
20. Click **Save**.

Create Custom Templates

In addition to using Designer's standard templates, you can create a template by saving your XDP as a TDS file.

1. Select **File – Open** and navigate to your Student Files.
2. Select *expenseReportTemplate.xdp* and click **Open**.
3. Select **File – Save As** and navigate to your working folder.
4. Click the *Save as type* dropdown list and select **Designer Template (*.tds)**.
5. Enter **myTemplate.tds** for the *File name*.
6. Your dialog box should now look like this *(see illustration)*.

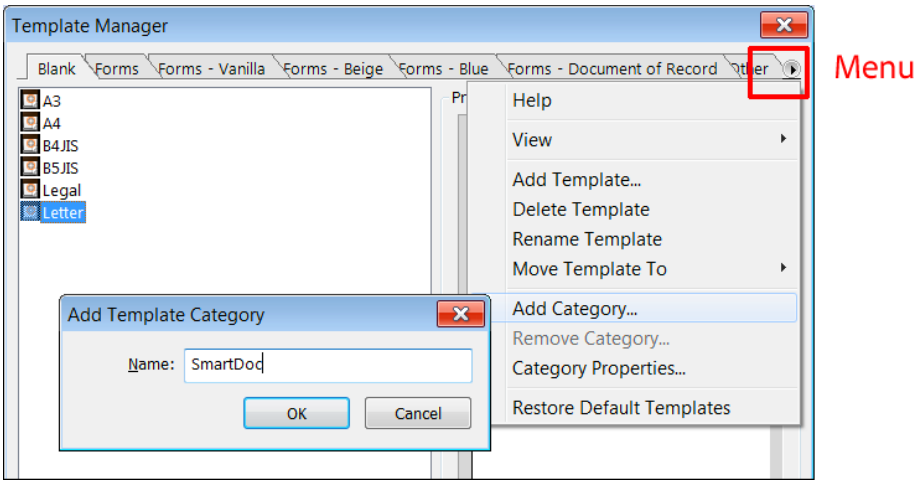


7. Click **Save**.

After you create your template, you can add it to Designer with the *Template Manager*.

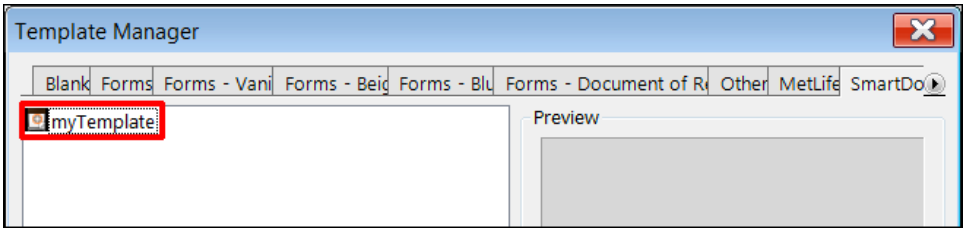
8. Select **Tools – Template Manager** to open the dialog box *(see illustration)*.
9. Open the **Menu** on the top right of the Template Manager *(see illustration)*.
10. Select **Add Category**.
11. Enter **SmartDoc** as the template category name, and click **OK**.

Your new SmartDoc category is added as a new tab in the Template Manager.



12. Select the **SmartDoc** tab.
13. Click **Menu** on the top right and select **Add Template**.
14. Navigate to your working folder.
15. Select your *myTemplate.tds* file and click **Open**.
16. Click **OK** to close the *Template Imported* pop-up.

17. You should see your *myTemplate* file in the SmartDoc tab of Template Manager.



18. Click **Close** on Template Manager.

Your new template is now active in Designer. You can use it to create new forms by following these steps.

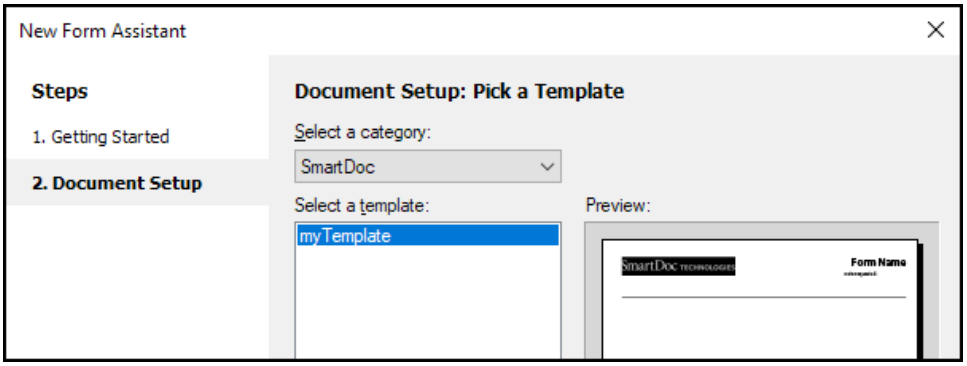
> ***Note:*** *How to set the default template in Designer.*
> *I do not recommend doing this because Designer's standard default template is usually ideal for blank forms. However, if you ever need to change the default template in Designer, you can select the template and click* ***Set Selected As Default****. This new template will now be your default template when you create a new blank form in the New Form Assistant.*

19. Select **File – New**.

20. Select **Based on a template** and click **Next**.

21. Click the *Select a category* dropdown and select the **SmartDoc** category.

22. Select your *myTemplate* template and click **Finish**.



23. Designer will create a form based on your template.

Create a Custom Object Library

Follow these steps to create a custom object library and add the SmartDoc objects to your library.

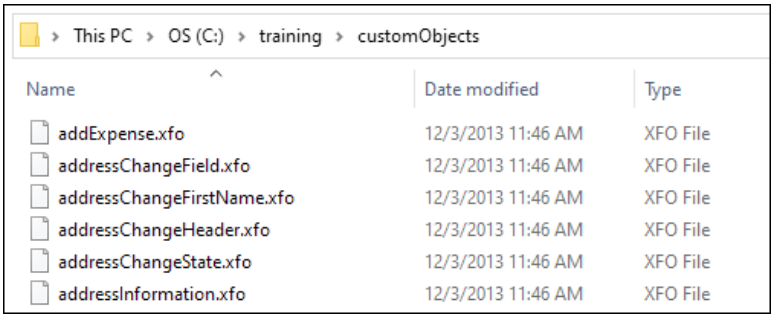
1. Open Windows File Explorer and create a folder in this location.

C:\training\customObjects

> ***Note:*** *You can have a different path if you are not able to create this particular path. You will just need to refer to your new path as you complete this exercise.*

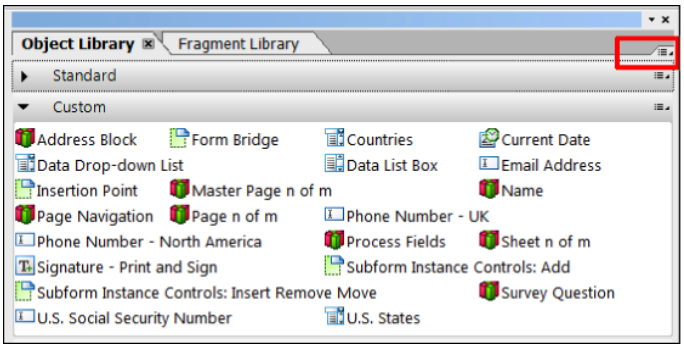
2. Open your Student Files and copy the *.xfo files from the Student Files\SmartDocObjects to the new customObjects folder.

3. Your folder should look like this when you are done.

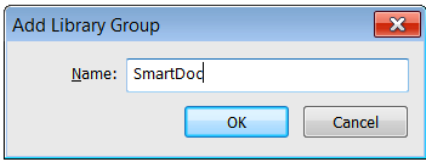


4. Go back to Designer.

5. Select the **Object Library palette menu** and choose **Add Group**. Designer launches the *Add Library Group* dialog box *(see illustration)*.



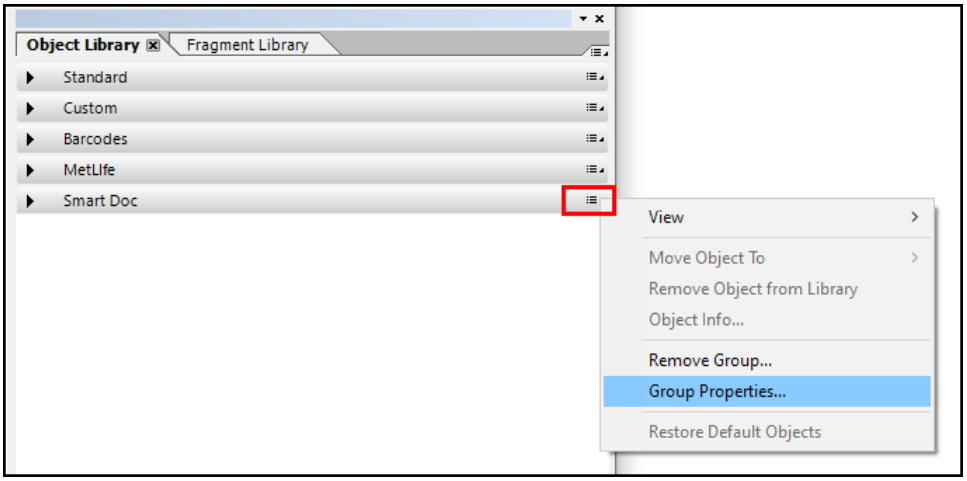
Object Library palette menu



6. Enter **SmartDoc** as the name for your new library.

7. Click **OK** to close the window.

8. Select the SmartDoc category menu and choose **Group Properties** *(see illustration)*.

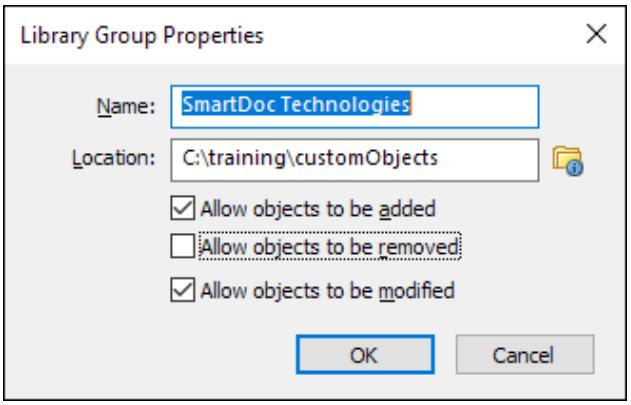


9. The Library Group Properties dialog box appears.

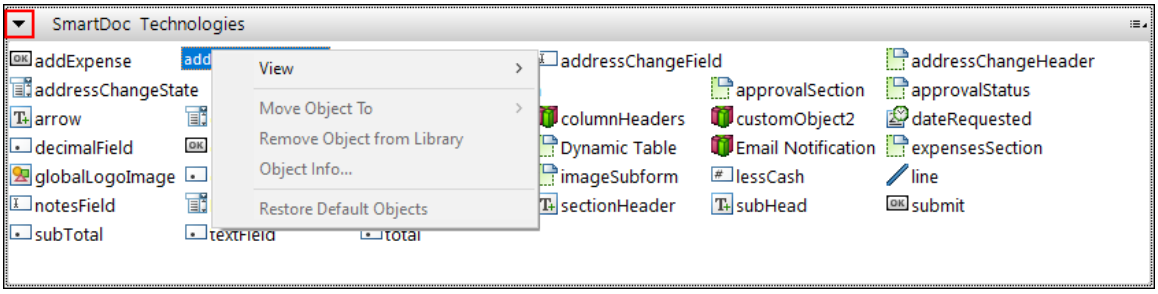
10. Click **Browse** *(the icon to the right of the Location field)* and navigate to your *customObjects* folder.

C:\training\customObjects

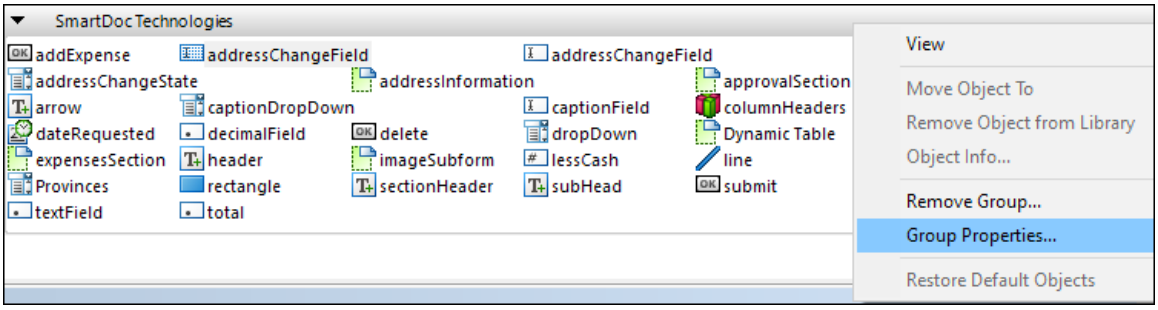
11. Select the **customObjects** folder and click **OK**.
12. Change the *Name* of the Library Group to **SmartDoc Technologies** *(see illustration)*.
13. Deselect **Allow objects to be removed**. You do not want to inadvertently remove an object.
14. Click **OK** when you are done.



15. Click the arrow icon *(down)* to show all of the SmartDoc objects *(see illustration)*.
16. You are now able to drag and drop custom objects from the SmartDoc Object Library to your form.
17. Right-click on an object and notice the *Remove Object from Library* is grayed-out *(see illustration)*.



18. Even though the *Remove Object from Library* is grayed-out, the Designer user can change the settings on their custom object library to enable this feature. Therefore, your custom objects are not really secure.
19. Click the dropdown menu on the *SmartDoc Technologies* library and select **Group Properties** *(see illustration)*.



20. Select *Allow objects to be removed*.

21. Click **OK**.

22. Right-click on the same object and notice the *Remove Object from Library* is now active.

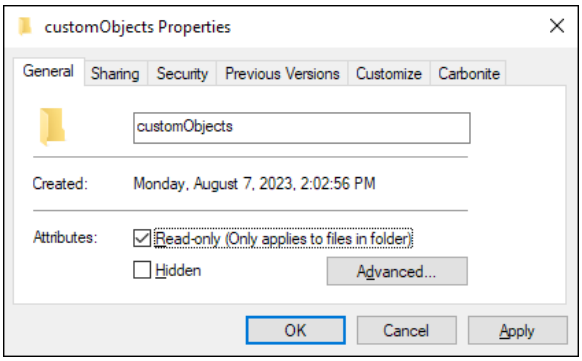
Secure Custom Objects in the File System

23. Open Windows File Explorer and navigate to your customObjects folder.

C:\training\customObjects

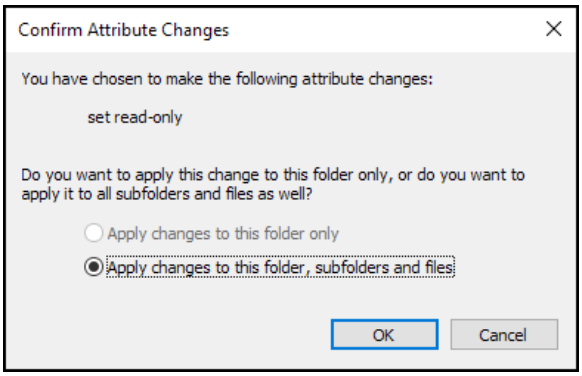
24. Right-click on your customObjects folder and select *Properties*.

25. Select *Read-only* and click *Apply*.



26. Select *Apply changes to this folder, subfolders and files.*

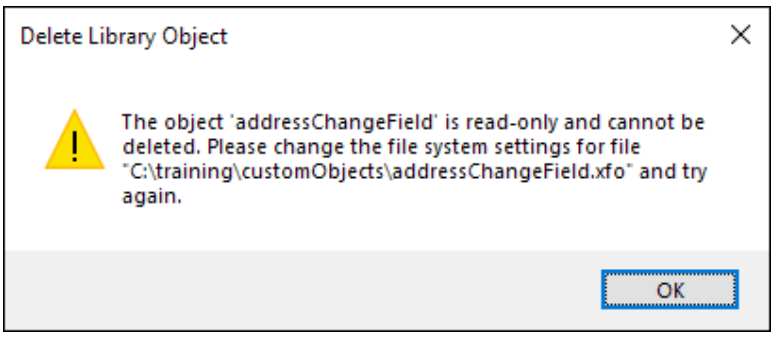
27. Click **OK** and **OK** again.



28. Go back to Designer.

29. Right-click on the same object and select *Remove Object from Library*.

30. You will see the *Delete Library Object* message *(see illustration)* showing that the object is now protected.



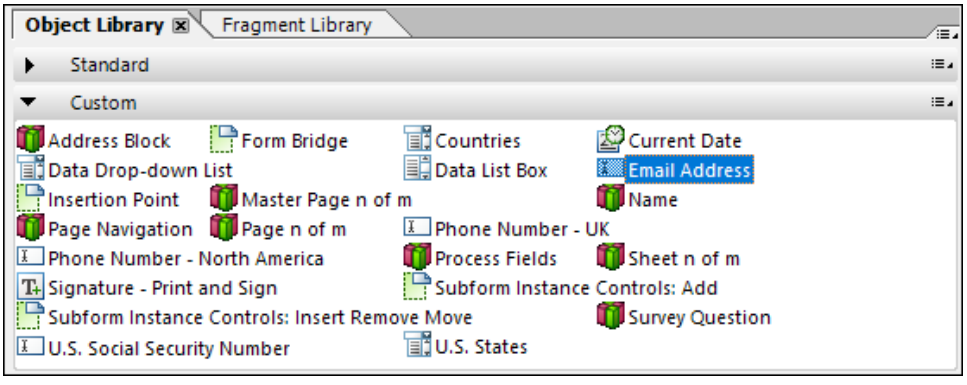
Create a Custom Object

In the same way that templates enable you to maintain consistent forms, custom objects enable you to maintain consistent form objects. In this section, you'll learn how to create your own custom object and add it to the SmartDoc Object Library. The following are some advantages of creating your own custom objects:

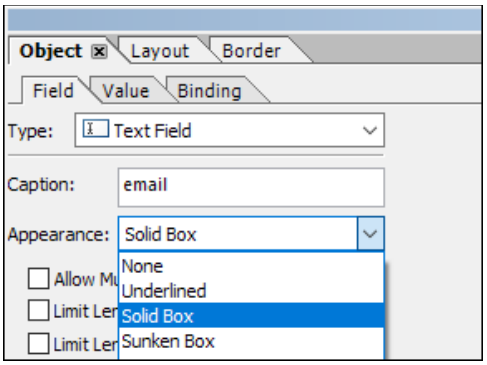
- Your custom objects can be made to match your company's graphic standards and data pattern requirements.
- You can group multiple form objects into one custom object. You'll see an example of this in the following exercise.
- Senior developers can add JavaScript to custom objects and share them with junior developers in your organization. Custom objects are saved as XFO files and can be shared on a network.

Follow these steps to create a custom object:

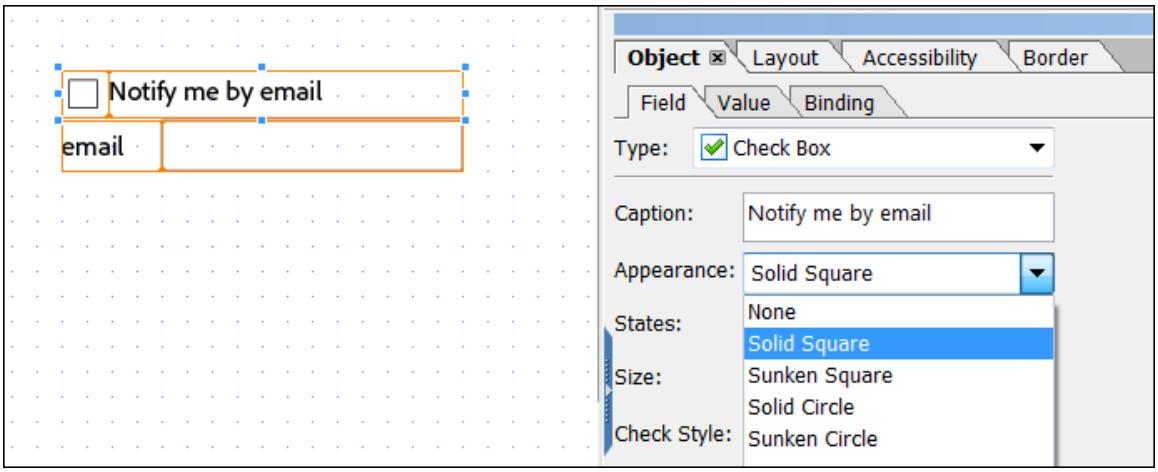
1. Select **File – New**. Designer launches the *New Form Assistant*.
2. Select **Use A Blank Form**. This is the default.
3. Click **Next** to continue.
4. Keep the defaults, and click **Finish** to create your new form.
5. Expand the *Custom Object Library*.
6. Drag and drop an **Email Address** field *(see illustration)* from the *Custom Object Library* to your form.



7. Select the *Field* tab of the *Object* palette *(see illustration)*.
8. Click the *Appearance* dropdown and select **Solid Box**.

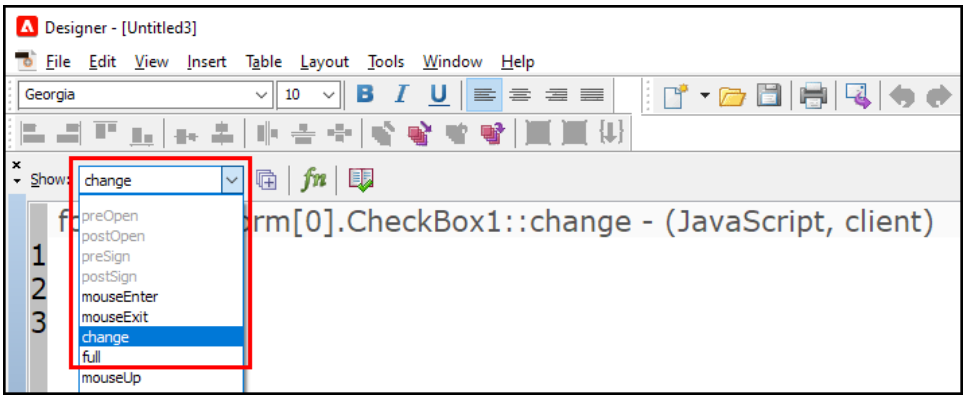


9. Collapse the *Custom Object Library* and expand the *Standard Object Library*.
10. Drag and drop a **Check Box** field from the *Standard Object Library* and place it above the Email Address field on your form.
11. Select the *Field* tab of the *Object* palette.
12. Click the *Appearance* dropdown and select **Solid Square**.
13. Change the Caption property to **Notify me by email**.



14. Select the **Layout palette** and change the width to **2** inches.
15. Expand the **Script Editor**, and select the **change** event *(see illustration)*.

Note: *The JavaScript.txt file in your Student Files contains JavaScript code and regular expressions.*



16. Enter this if statement into the change event.

```
if (this.rawValue == "1")
    {
    email.presence = "visible";
    }
else
    {
    email.presence = "invisible";
    }
```

17. Select the **Email Address** field, and select the validate event.

18. Change the regular expression on line 3 from the old to the new code below. Be sure to change only the regular expression within the parentheses.

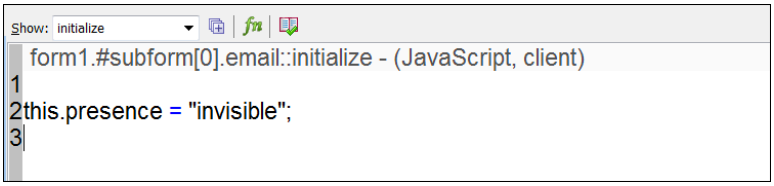
Old: "^[a-z0-9_\\-\\.]+\\@[a-z0-9_\\-\\.]+\\.[a-z]{2,3}$"

New: "^[A-Z0-9._%-]+@[A-Z0-9.-]+\.(com|org|net)$","i"

Note: *You can put scripts on multiple events of a form object.*

19. Select the **initialize** event *(see illustration)* of your Email Address field, and enter the following script:

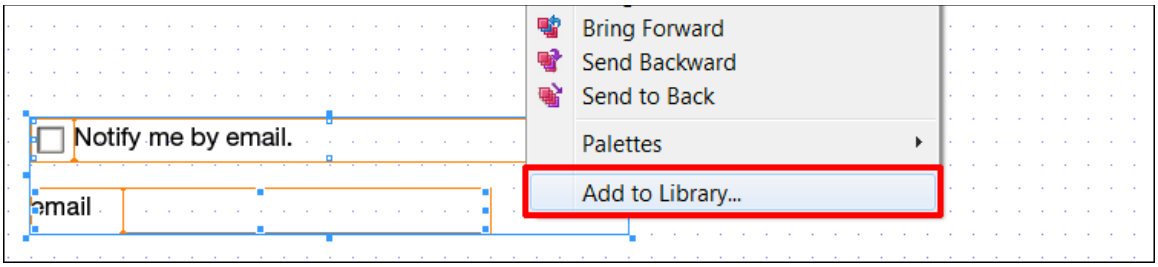
this.presence = "invisible";



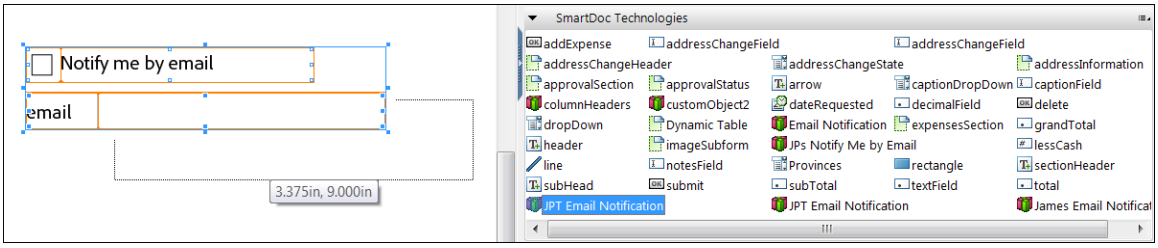
20. Close the *Script Editor* and expand your *SmartDoc Technologies Object Library*.

21. Select the **Email Address** field and the **Check Box** field together *(by holding down your Shift key)*.

22. Right-click on the objects and select **Add to Library…** *(see illustration)*.



23. Enter **<yourname> Email Notification** as the name of your new custom object.
24. Enter **Custom Components can contain JavaScript** for the *Description*.
25. Click the *Tab Group* dropdown and select **SmartDoc Technologies**.
26. Click **OK** to create your new custom object.
27. Select **File – New** to create a new form.
28. Click **Next** and **Finish** to create a new blank form.
29. Drag and drop the <yourname> Email Notification object from your SmartDoc Technologies Object Library to your form *(see illustration)*.



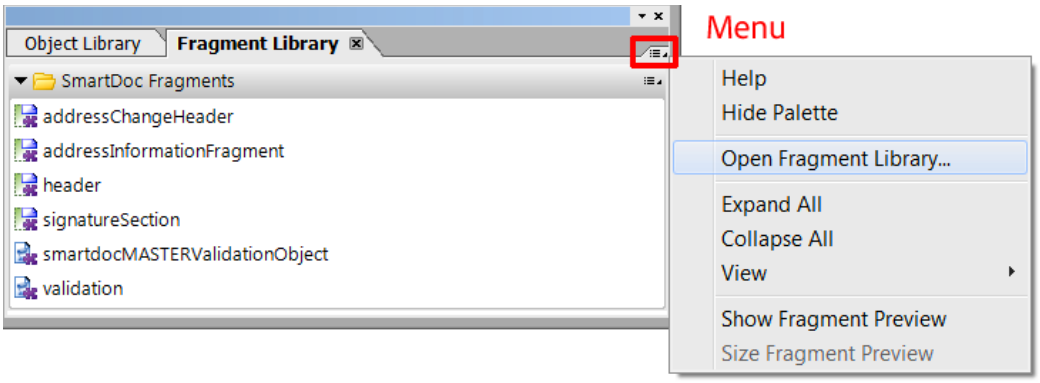
30. Select **Preview PDF** to see your object in action. Notice that the Email Address field is hidden when the form opens. This was accomplished by the script you added to the initialize event.
31. Click the **Notify Me By Email** check box. Each time a click causes the change event to fire, your script will either show or hide the Email Address field.

Your new object has all the same formatting and functionality of your custom object. However, your new object is a unique and independent object that isn't linked to your custom object. You can make changes to your new object that won't affect your custom object. Likewise, you can update your custom object, but any objects that were previously created from this object won't be updated.

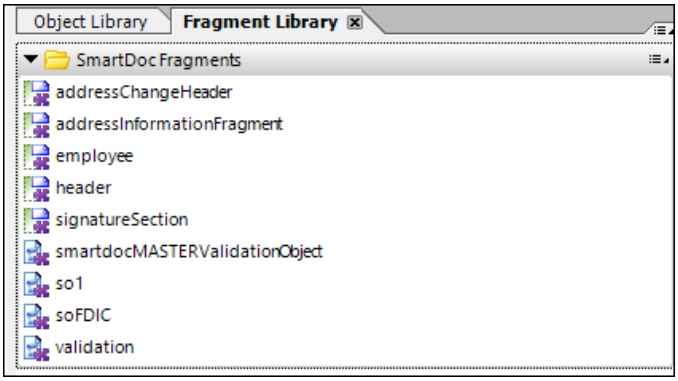
Create a Fragment Library

Designer doesn't include any standard fragments out-of-the-box, but it does include the Fragment Library panel, to which you can add your own custom fragments and that you can access by selecting Window – Fragment Library. Follow these steps to add fragments to your library:

1. Select the **Fragment Library** tab. It is next to the Object Library tab *(see illustration)*.
2. Select **Open Fragment Library** from the menu in the top right of the Fragment Library tab.



3. Navigate to your Student Files.
4. Select the **SmartDoc Fragments** folder and click **OK**.
5. You should see fragments load into your *Fragment Library* panel *(see illustration)*.

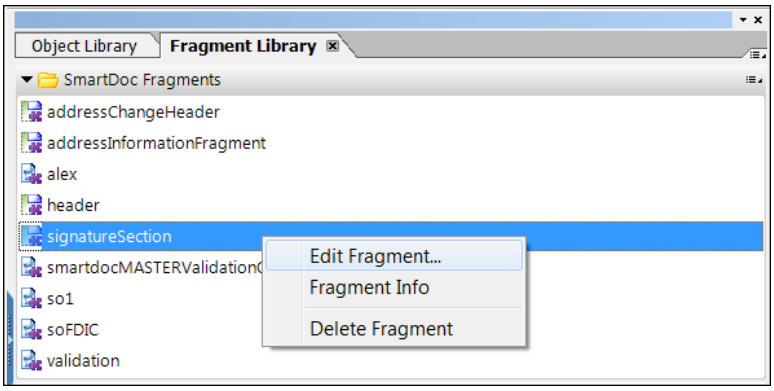


Use Fragments

Fragments make it easier to create a series of forms that all share a similar object. In this example, you will create three forms that all use the same signature section.

1. Select **File – New**. Designer launches the *New Form Assistant*.
2. Select **Use A Blank Form**. This is the default.
3. Click **Next** to continue.
4. Keep the defaults, and click **Finish** to create your new form.
5. Drag and drop the **signatureSection** fragment from the SmartDoc Fragments library onto your form. Notice that the fragment has a purple tint in Design View and the objects are grayed-out in the Hierarchy palette.
6. Save this file three times to your working folder with the following names:
   fragmentTest1.xdp
   fragmentTest2.xdp
   fragmentTest3.xdp.
7. Close all 3 of these files.
8. Locate the **signatureSection** fragment *(this is your master fragment)* in the Fragment Library *(see illustration)*.

9. Right-click on the **signatureSection** fragment and select **Edit Fragment**. This will open the fragment's source XDP file.

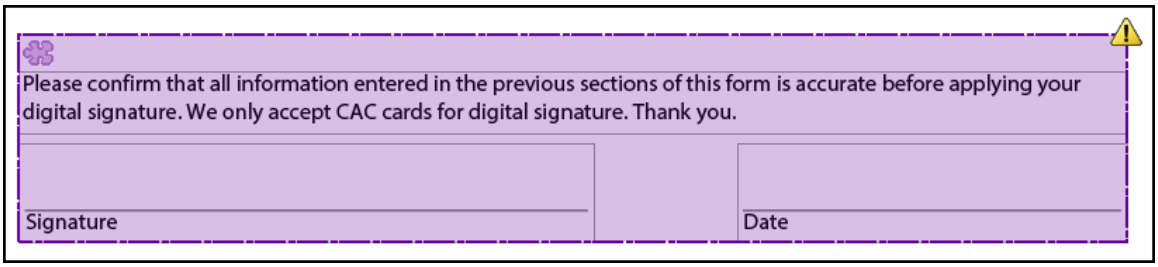


10. Select the **Instructions** text object and enter **<yourname>** at the end of the instructions.

    Note: *If the text already ends in Thank you, simply delete the Thank you to notice the change. The point is to make a change in the fragment to see how is populates in all the forms that reference the fragment.*

11. Select **File – Save**.

12. Select **File – Open** and open any of the three files you created previously. You'll see that each one of the signature sections has been automatically updated because each was a link to the master fragment (see Illustration).



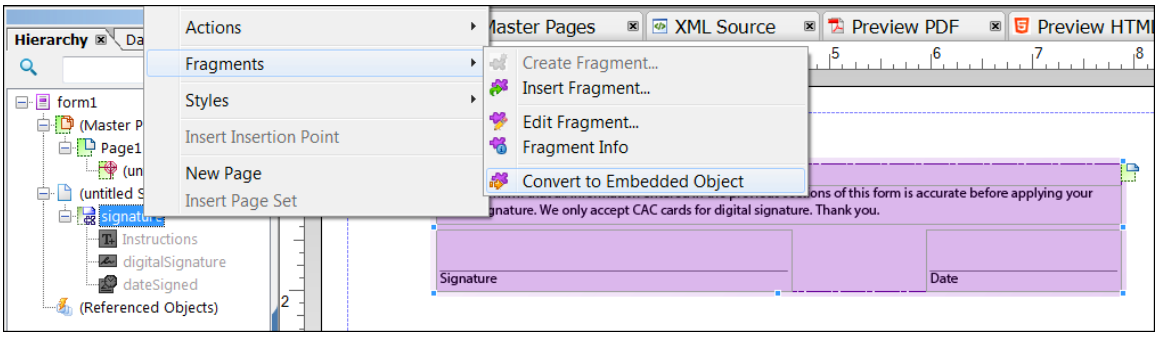
This technique has its advantages, but you can't edit an individual instance of a fragment.

Embed Fragments

If you want to edit a fragment instance, you need to embed it in your form, which will break the link to the master fragment.

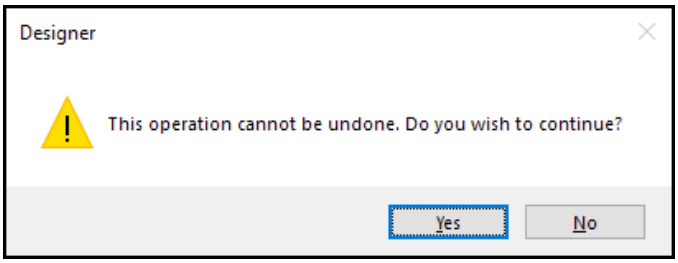
1. Open **fragmentTest3.xdp** in Designer.

2. Right-click the **signature** subform in the hierarchy and select **Fragments – Convert To Embedded Object**.

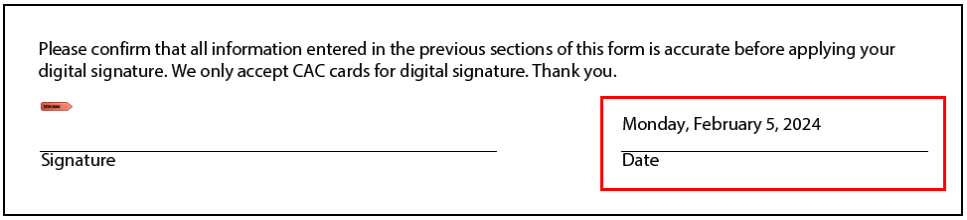


3. Click **Yes** on the Warning message.

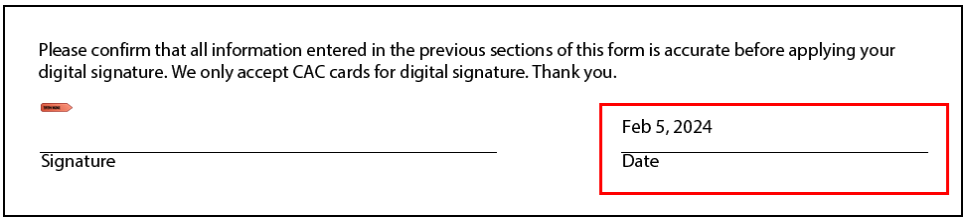
   Note: *The message informs you that embedding the fragment into the form removes all references to the external fragment file. Your objects are now local to your form, and you're able to edit them without affecting the fragment.*



4. Save and close **fragmentTest3.xdp**.
5. Locate the **signatureSection** fragment *(this is your master fragment)* in the Fragment Library.
6. Right-click and select **Edit Fragment**. This will open the fragment's source XDP file.
7. Select the **dateSigned** field.
8. Select the **Patterns** button on the Field tab of the Object palette.
9. Select **Sunday, April 1, 2007** in the Type list. This changes the display pattern to **date(**EEEE, MMMM D, YYYY**)**.
10. Click **Apply**, and then click **OK** to close the dialog box.
11. **Save** and close the file.
12. Select **File – Open** and open the three files you created previously.
13. Click **Preview PDF** to view your forms.
14. Click the Date field and select today's date in the calendar control.
15. The signature sections of the first two files have been updated, so you will see the new date pattern.



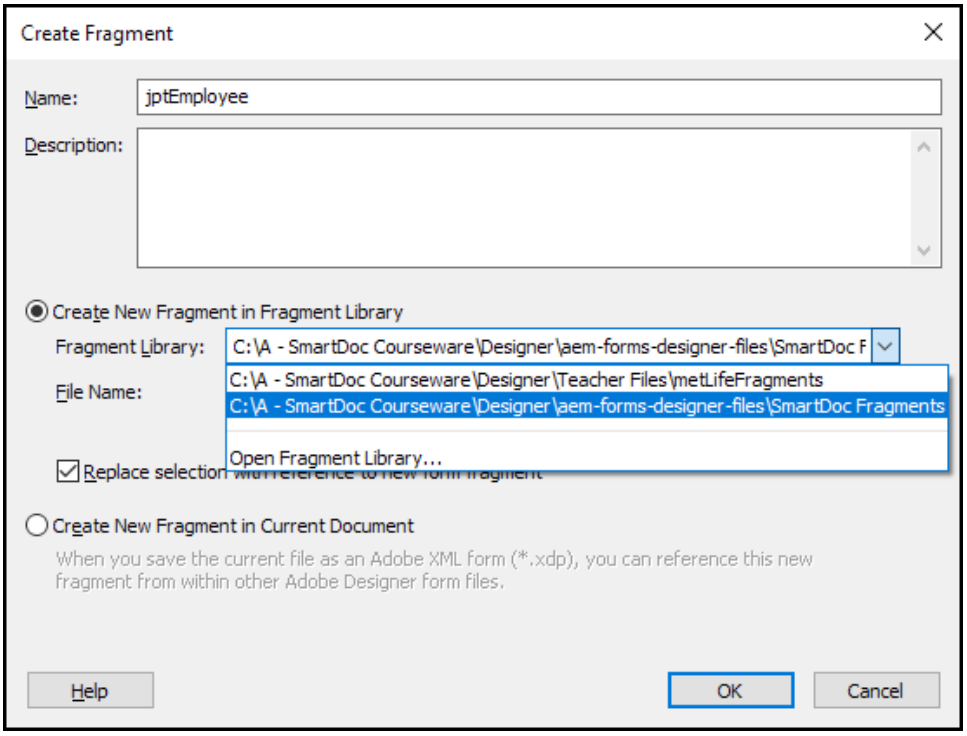
16. But the signature section in fragmentTest3.xdp hasn't been updated because the link to the master fragment was broken when you embedded the fragment. You will not see the update pattern in this form.



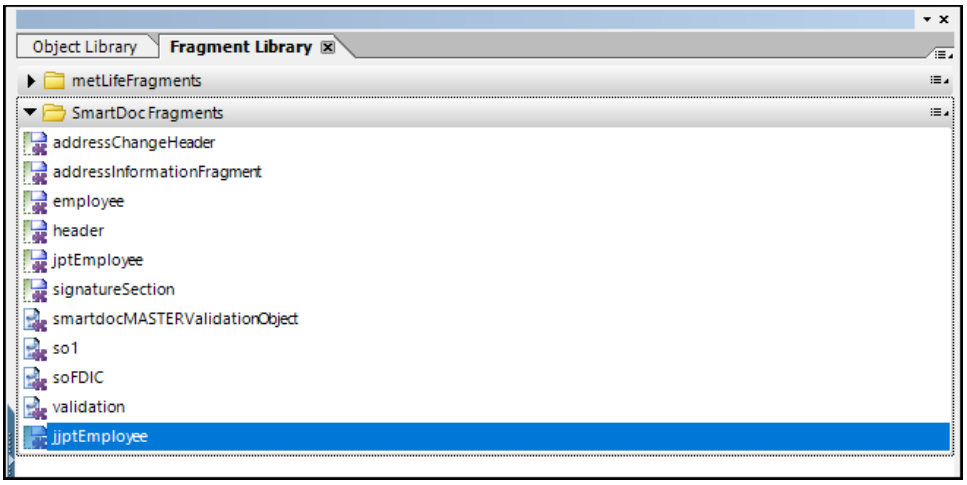
Create Fragments

Fragment references retain the graphic style and functionality of the source (or master) fragment. Creating a fragment is as easy as creating a custom object. Follow these steps to add a fragment to your SmartDoc library:

1. Select **File – Open** and navigate to your Student Files.
2. Select *expenseReportCompleted.xdp* and click **Open**.
3. Right-click the **employee** subform on page1.
4. Select **Fragments – Create Fragment**. The Create Fragment dialog box will open *(see illustration)*.
5. Enter **<yourname>Employee** for the *Name*.
6. Click the *Fragment Library* dropdown and make sure the path points to your *SmartDoc Fragments* library.

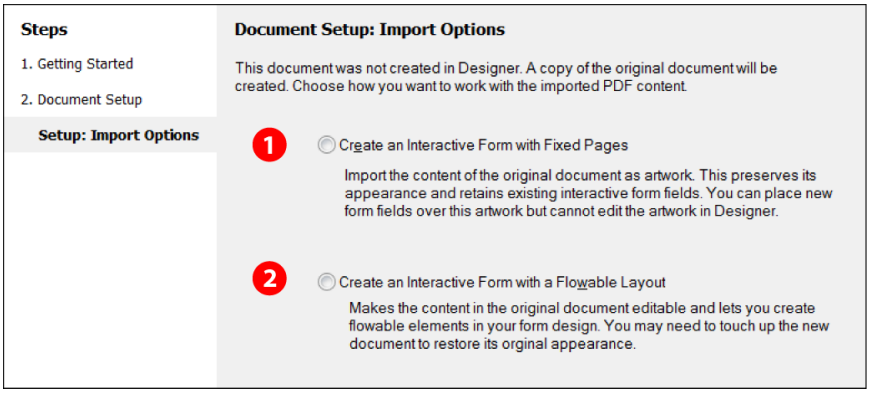


7. Click **OK**. This will add a new fragment to your library and create a reference to the fragment in your current form. This new fragment will have all the graphic styles and functionality of the employee subform.
8. You can see the new fragment in your SmartDoc Fragments library *(see illustration)*.


Object Library
Fragment Library
metLifeFragments
SmartDoc Fragments
addressChangeHeader
addressInformationFragment
employee
header
jptEmployee
signatureSection
smartdocMASTERValidationObject
so1
soFDIC
validation
jjptEmployee

Form Conversion

In some cases, you may want to convert existing forms into Designer. When you import a non-Designer file, you will be presented with the choice shown here. Since your decision determines what you can do with your Designer form, make sure you understand the difference between an *Acroform* (1), and an *XDP* form (2).



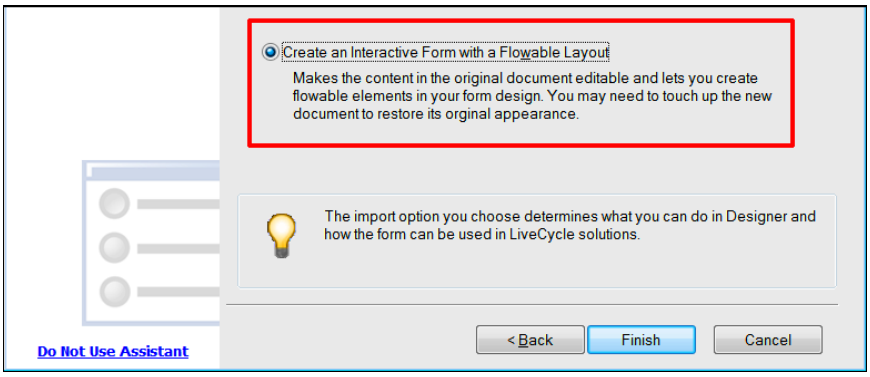
Acroforms

The first option, *Create an Interactive Form with Fixed Pages*, will produce an Acroform.

An Acroform's layout will look great, because the printstream PDF is used as background art. However, this is not a true XDP file, and it does not support many of the benefits of a Designer XDP file. The layout is static and not dynamic, and you cannot use all of Designer's interactive and dynamic form objects. It also does not support HTML rendering because the underlying structure is not completely XFA-based XML. Only use this option if you want a static PDF file with some interactive fields bolted on top.

XDPs

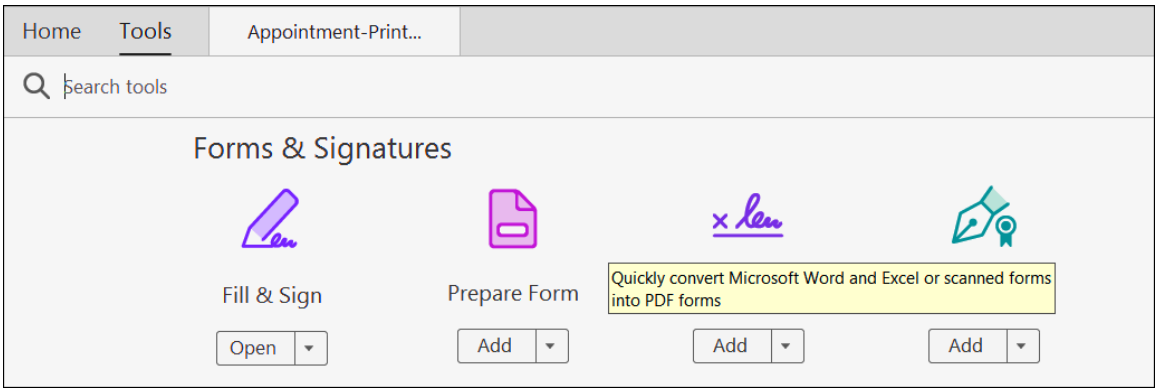
The second option, *Create an Interactive Form with a Flowable Layout*, will produce a true XDP file.



However, although this is a true XDP, the conversion often results in a very poorly structured XDP file. In most cases, you will be better off combining some parts of these files with your templates, custom objects, and fragments. This is a better approach to achieve proper form objects and a proper form structure. This option does support dynamic layouts and HTML rendering because it is a true XFA file.

Acrobat Professional

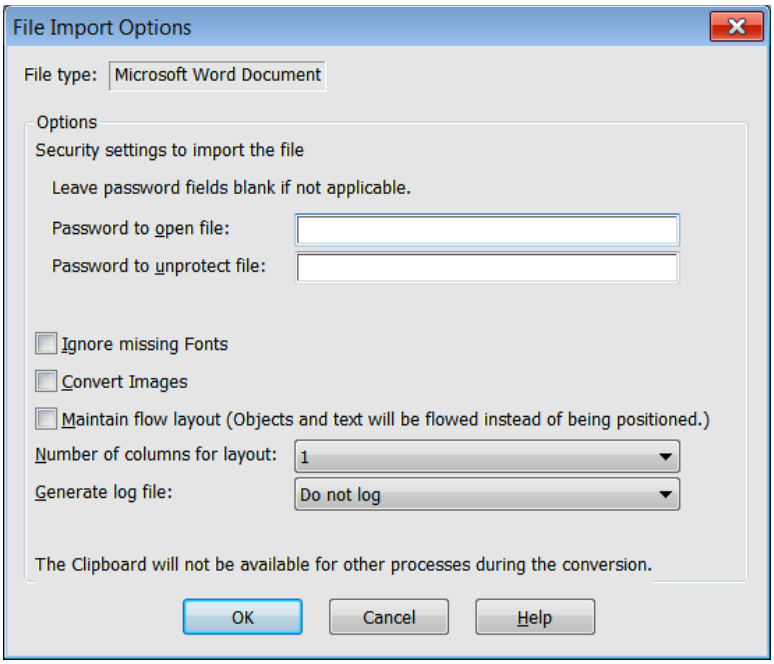
Acrobat Pro DC has a useful Prepare Form tool. It will scan various file formats and auto-detect form fields. The auto-detection is very good, and the field naming is very helpful. The result is an Acroform. If you have Acrobat Pro DC, you can experience the Prepare Form tool in the exercises.



Exercises

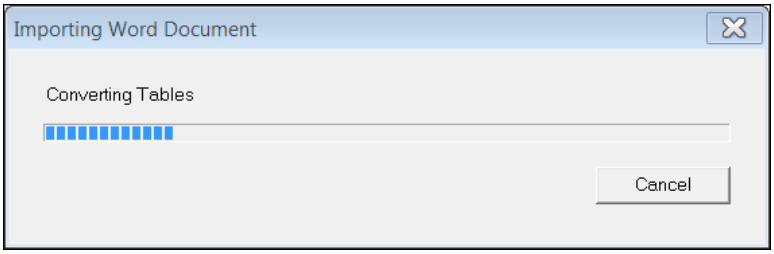
Convert a Microsoft Word File to XFA

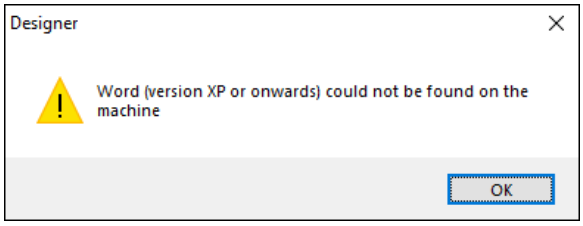
1. Select **File – New** and the *New Form Assistant* will open.
2. Select **Import a Microsoft Word** document and click **Next**.
3. Click **Browse** and navigate to your Student Files.
4. Select *Sample MS Word Form.docx* and click **Open**.
5. Click **Finish**.
6. Set these **File Import Options**. These should be the defaults.



7. Click **OK**.

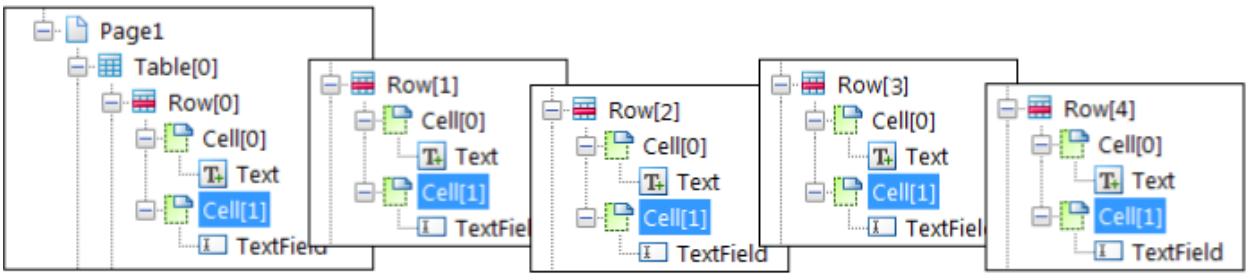
You should see the **Importing Word Document** dialog *(see illustration)*. The messages will update as the conversion works its way through each step.



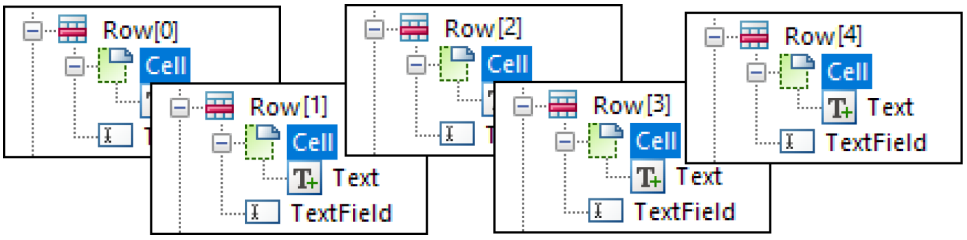
> ***Note:*** *If you receive this error:* ***Word (version XP or onwards) could not be found on this machine,*** *then you have two options. For this exercise, you can use the* ***ConvertedFromMSWordFile.xdp*** *from your Student Files. I converted this file from Microsoft Word on my machine.*
>
> 
>
>
> *If you want to solve this problem on your copy of AEM Forms Designer, you can download and run this Service & Cumulative Fix Pack:* ***Designer6.5.0_English_Cumulative_QF.msp***

8. Select **File – Save As** and navigate to your working folder.
9. Click the *Save as type* dropdown list and select **Adobe XML Form (*.xdp)**.
10. Enter **myConvertedWordFile.xdp** for the *File name*.

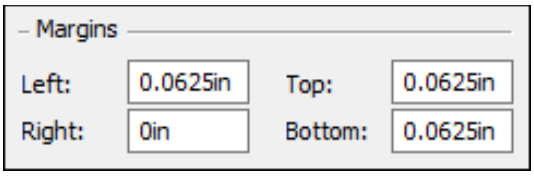
11. Click **Save**.
12. Select **Preview PDF**.
13. Enter your name into the Name field. Notice there is not enough room to fill in your entire name. This is because the fields do not expand like they do in Microsoft Word.
14. Select **Design View** to edit the form.
15. In the Hierarchy palette, expand **Table[0]** and each row in Table[0].
16. Hold down the Ctrl key and select (Ctrl + click) each of the 5 Cell[1] items from Row[0] to Row[4] in Table[0] *(see illustration)*.



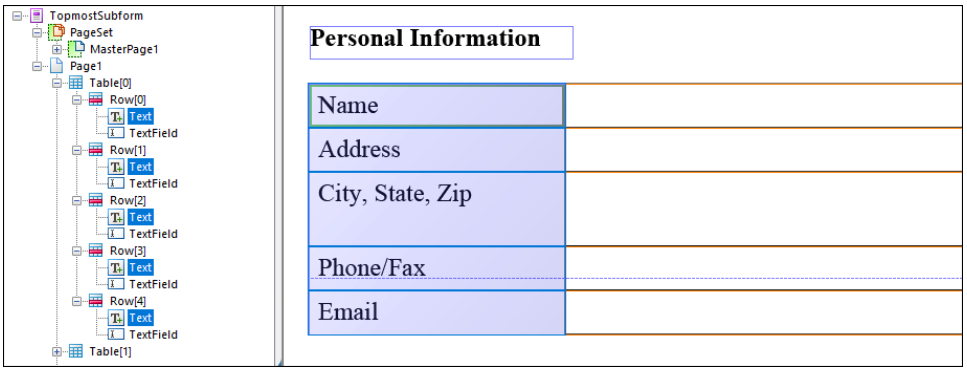
17. With all the Cell[1]s selected, go to the **Object – Cell** palette and change the *Type* from *Subform* to **Text Field**.
18. Click **OK** if the Warning window is prompted.
19. Keep all the **Text Fields** selected and select the Border palette.
20. Click the *Edges* dropdown and switch from *None* to *Solid*.
21. Click the *Background Fill Style* dropdown and switch from *Solid* to *None*.
22. Keep all the **Text Fields** selected and select the **Paragraph** palette.
23. Change the text to **Align Middle** and the *Left* indent to **.0625**.
24. Hold down the Ctrl key down and select (Ctrl + click) each of the 5 Cell items from Row[0] to Row[4] in Table[0] *(see illustration)*.



25. With the Cells selected, go to the **Object – Cell** palette and change the *Type* from Subform to **Text**.
26. With all the Text items selected, go to the **Layout** palette, and set the *Left, Top, and Bottom* margins to **.0625**.



27. The structure of the table is much simpler now *(see illustration)*.

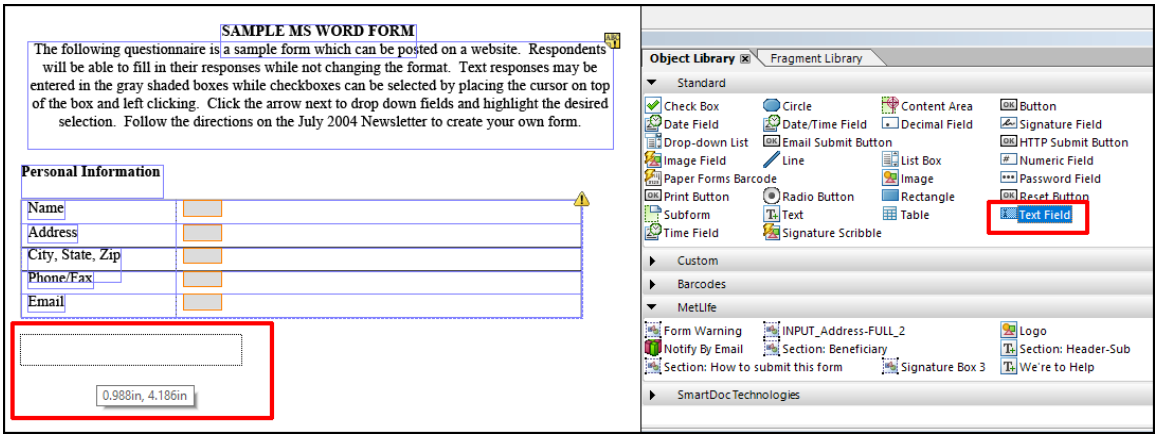


28. Select **Preview PDF**.

29. Enter your name into the Name field. Notice there is now enough room in the table to enter the data.

30. Select **Design View**.

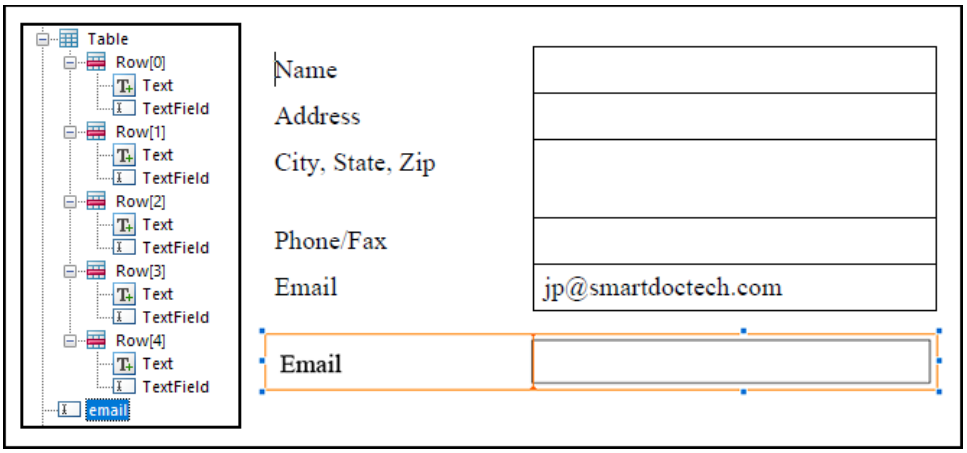
31. Drag and drop a **Text Field** object from the *Standard* Library to the area below the table *(see illustration)*.



32. With the Text Field selected, enter **Email** for the *Caption* property. This property is located on the *Field* tab of the *Object* Palette.

33. Select the *Binding* tab and enter **email** as the *Name* for the object.

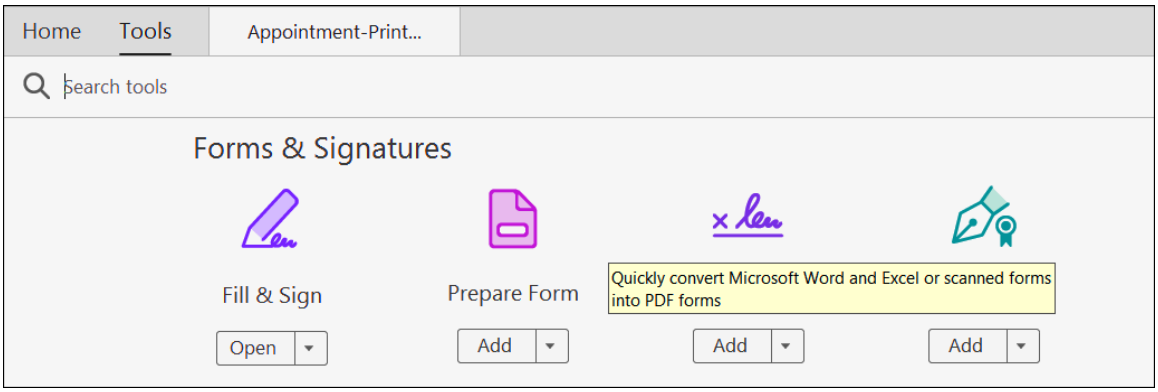
34. You can now stylize the email Text Field to match the converted text field in the table. When you are done, you will see the fields can be identical to the form filler *(see illustration)* but your new email Text Field will not have all of the unnecessary Table Structure of the converted form.



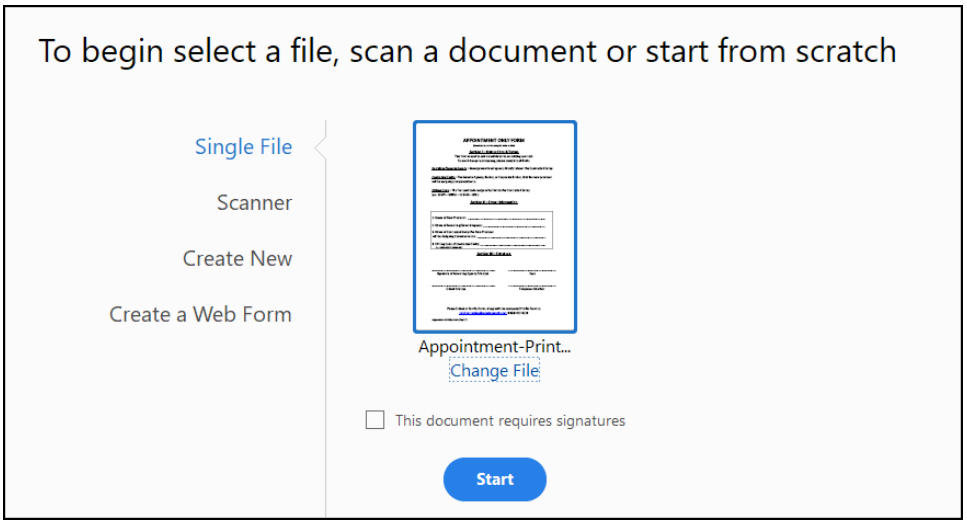
Use Acrobat's Prepare Form tool

Follow these steps to make an Acroform from a print-stream PDF file.

35. Open Adobe Acrobat DC.
36. Select **File – Open** and navigate to your Student Files.
37. Select *Appointment-Printstream.pdf* and click **Open**.
38. Select **Tools – Prepare Form**.

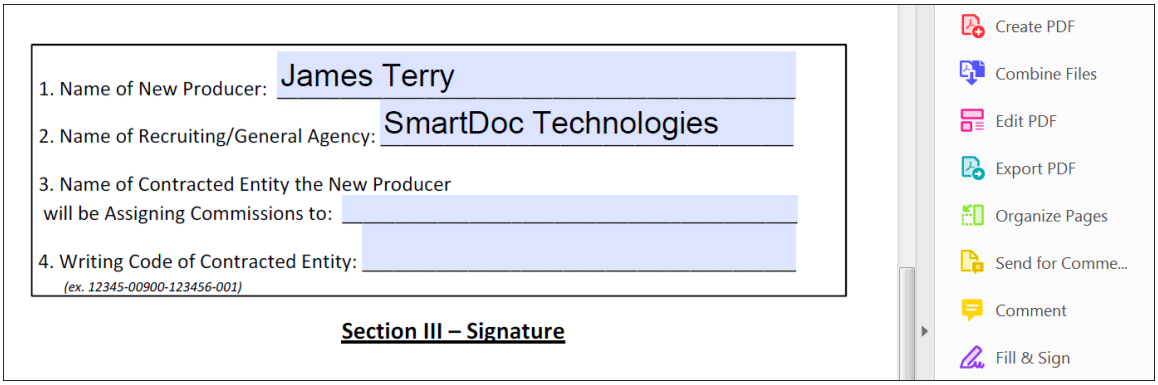


39. Click **Start** *(see illustration)*.

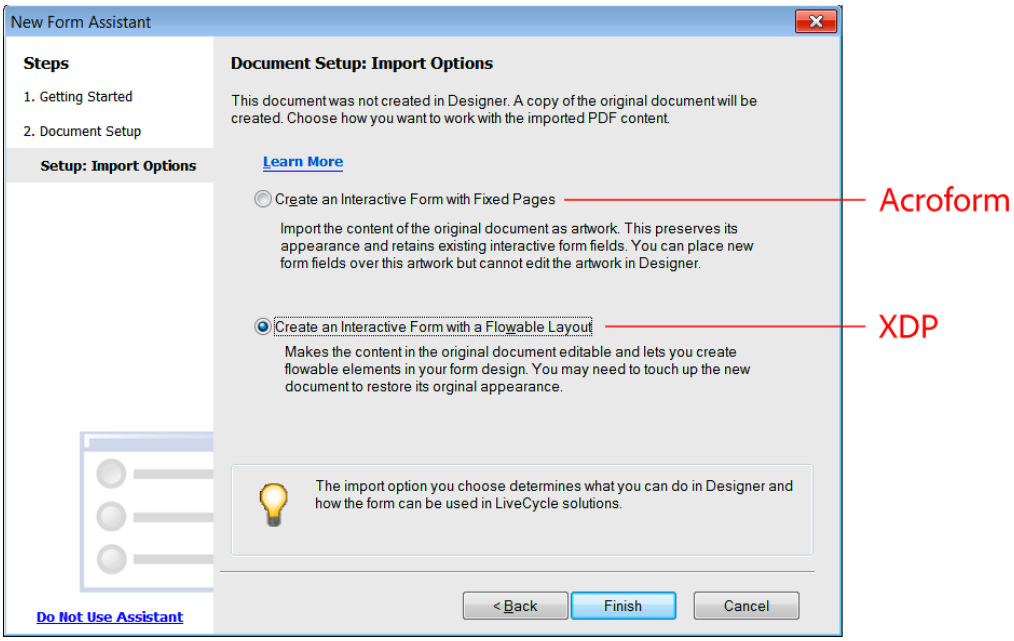


Acrobat will create text fields and even a signature field on your form.

40. Click **Close** in the upper right-hand corner.
41. These are interactive fields, and you can enter data.



42. Select **File – Save As** and navigate to your working folder.
43. Enter **<yourname>-Appointment-Acroform.pdf** for the *File name* and click **Save**.
44. Click **Save**.
45. Open **Designer** if it is not already opened.
46. Select **File – New** and the *New Form Assistant* will open.
47. Select **Import a PDF document** and click **Next**.
48. Click **Browse** and navigate to your working folder.
49. Select the **<yourname>-Appointment-Acroform.pdf** file and click **Open**.
50. Click **Next** to continue.
51. Select **Create an Interactive Form with Fixed Pages**. This is the proper choice for an Acroform.



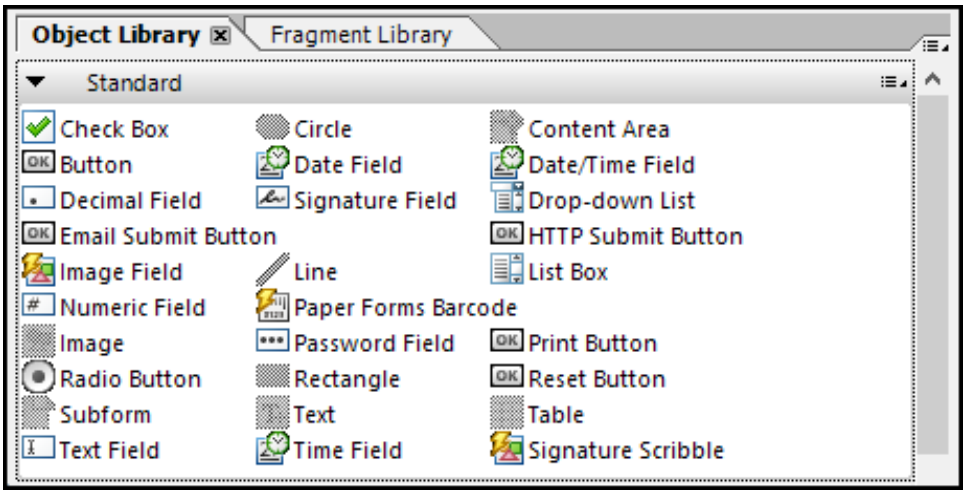
Note: *The first option will create an Acroform, and the second option will create an XDP.*

52. Click Finish.

53. If you don't have Helvetica, map it to a san-serif font like Arial.

54. Click **OK**.

55. Notice that you have seven text fields and one signature field in your form's hierarchy. You can continue to work on this form in Designer but notice that not all of the Designer standard objects are available for this Acroform *(see illustration)*. Also, this Acroform will not render as HTML because it is not an XFA form.



Convert a Print-Stream PDF File to XFA

56. Select **File – New** and the *New Form Assistant* will open.

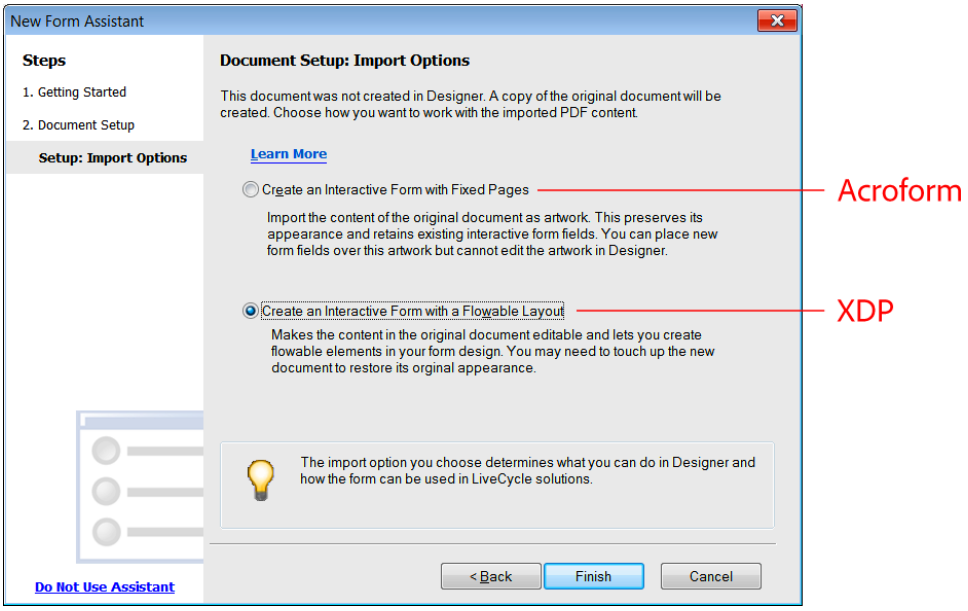
57. Select **Import a PDF document** and click **Next**.

58. Click **Browse** and navigate to your Student Files.

59. Select *Appointment-Printstream.pdf* and click **Open**.

60. Click **Next** to continue.

61. Select **Create an Interactive Form with a Flowable Layout**. This will be a true XFA form. You will be able to save this as an XDP and it will render as HTML.



Note: *The first option will create an Acroform, and the second option will create an XDP.*

62. Click **Finish**.

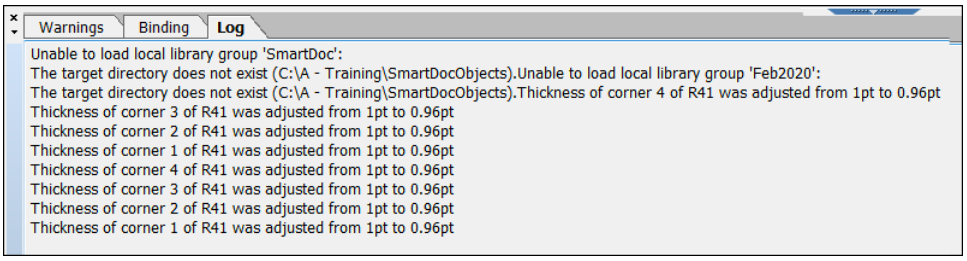
63. If you don't have Helvetica, map it to a san-serif font like Arial.

64. Select **Permanently Replace Unavailable Fonts**.

65. Click **OK** to convert the file.

66. You will see the **Conversion Summary** dialog box.

67. Select the **Show this message in the report inspector** option. In the future, your conversion summaries will be shown in the report inspector.



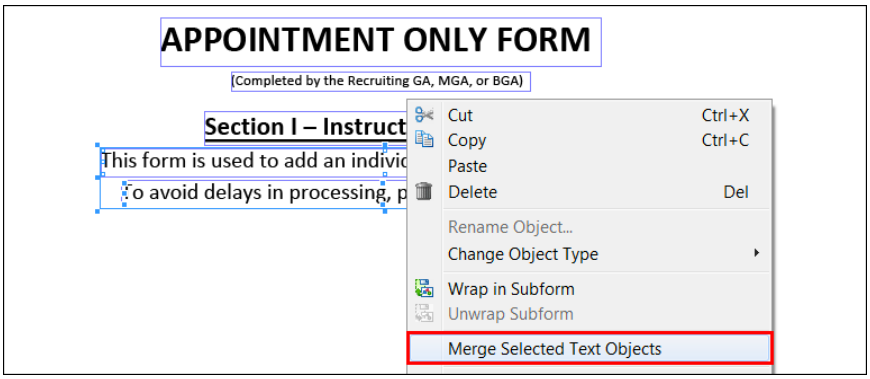
68. Click **OK** to close the **Conversion Summary** dialog box.

Update the Content Area on the Master Page

69. Expand **(Master Pages) – PageArea1** in the Hierarchy palette.
70. Select **ContentArea1** and open the **Layout** palette on the right.
71. Change the Width to **7.5in**.
72. Change the Height to **10in**.
73. Change the X position to **0.5in**.
74. Change the Y position to **0.5in**.

Clean-up the text

75. Select **Design View**.
76. Delete the large black rectangle in the middle of the form.
77. Select the separate text items, right-click and select **Merge Selected Text Objects** *(see illustration)*.



78. Open the Drawing Aids palette.
79. Expand your **Drawing Aids** palette. If you don't see your **Drawing Aids** palette, you can select Window – Drawing Aids to display the palette.
80. Set the X Interval to 16/in and the Y Interval to 16/in. This will produce a 1/16-inch grid.
81. You can close the **Drawing Aids** palette now to save space.
82. Select **View – Grids & Guidelines – Show Grid**.
83. Select **View – Grids & Guidelines – Snap to Grid**.

You can now use the Snap to Grid feature to align your form objects.

Other Form Improvements

There are other improvements that can be made to this PDF form. Here are some ideas.

– Notice that the underlined text is actually a rectangle underneath a text object. This is not an ideal structure. Instead, delete the rectangles and use the underline property of the text object.

– Notice that the fill-in fields are text objects and not Text Field objects. Select each Text object and change the Type to Text Field object. Delete the underline in the caption value and set the Appearance property to Underlined.

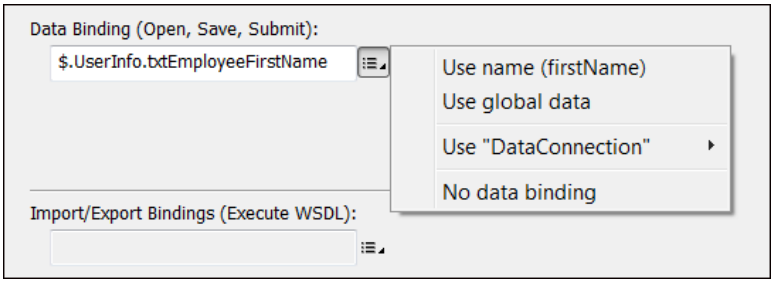
– Delete the 4 separate rectangles that form the outlined box. Add a new rectangle with a solid black border to replace the outlined box.

Working with Data

Designer enables you to integrate your forms with data by binding your form objects to a data source or schema, formatting your data fields with patterns, and validating your user's data directly in the form.

Data binding

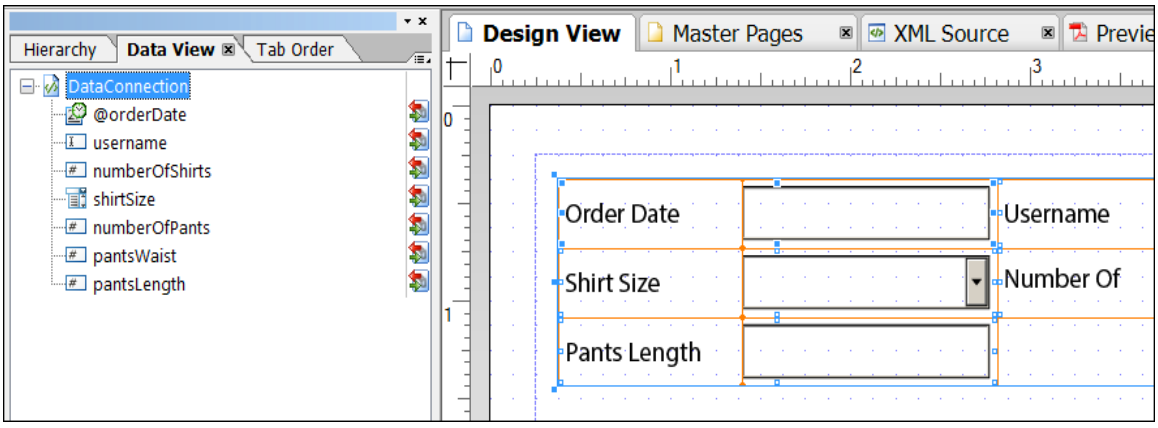
Data binding is the process of mapping your form objects to elements in a data file. Typically, you'll bind your form objects to elements in an XML schema or an XML data file. You can see a field's data binding with the Binding tab of the Object palette *(shown here)*. The Binding tab is one way to set a form field's data binding.



Here are the options for data binding.

- **Use name:** When you drag and drop form objects onto a blank form; Designer will create an implicit data binding, and the XML data associated with your form will use the form structure and field names in your form. Designer refers to implicit binding as Use name which means you are using the names of your form fields for the names of the data elements.
- **Use global data:** If you set a form object's binding property to *Use global data*, all other form objects with the same name will have the same data value. For example, if you have four Text Field objects with identical names and one of the objects is set to *Use global data*, they'll all be set to *Use global data*. This is referred to as global binding. If a form filler changes the data in any one of these fields, the data in all the others will be replicated to match.
- **Use DataConnection:** Data Connection is the process of binding your form fields directly to a data connection like an XML schema or other data source. The exported data from this type of form will have the same structure as the data source. This type of binding is often called *explicit binding* because you're explicitly connecting a field to an external data source.
- **No data binding:** When a field is set to *No data binding*, it doesn't participate in the data binding; data will not flow into the field, and data will not flow out. All fields and subforms that don't need to have data imported or exported should be set to *No data binding* to improve the performance of importing and exporting data.

This illustration shows an example of DataConnection binding. The icons to the right of each data node indicate that the node is bound to a form field.



There are two ways to set a form field's binding to a data connection.

- You can use the Binding tab of the Object palette as shown previously.
- You can drag and drop data elements from the DataConnection to your form fields (as shown above).

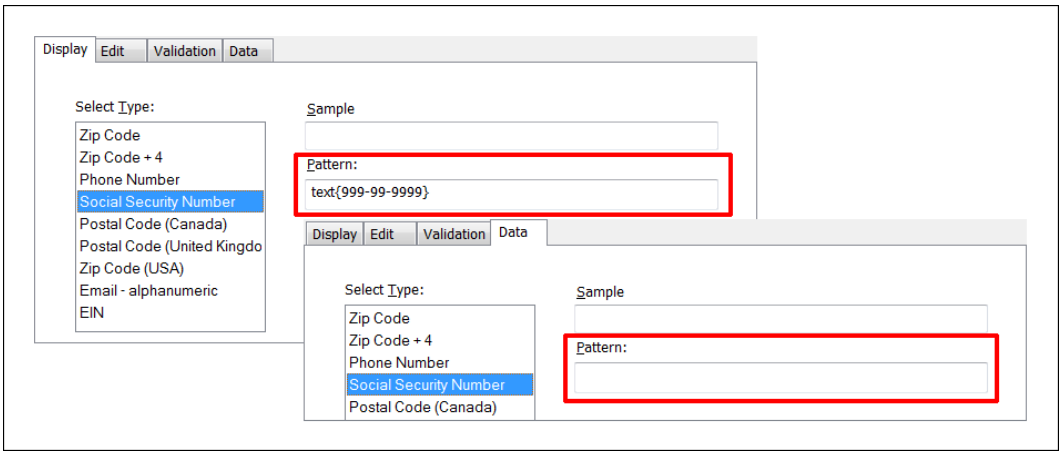
Note: *In addition to these typical binding methods, you can also use JavaScript to import data and bind it to your form objects. This is helpful when you need to perform some scripting logic on the data in addition to binding it to the form objects.*

Data formatting

Designer enables you to control how data is displayed in your form and how it's formatted when it's exported from your form.

- You can add a Display Pattern to a form field to format the display of the data for the form filler.
- You can also add a Data Pattern to format the exported data that comes from the form.

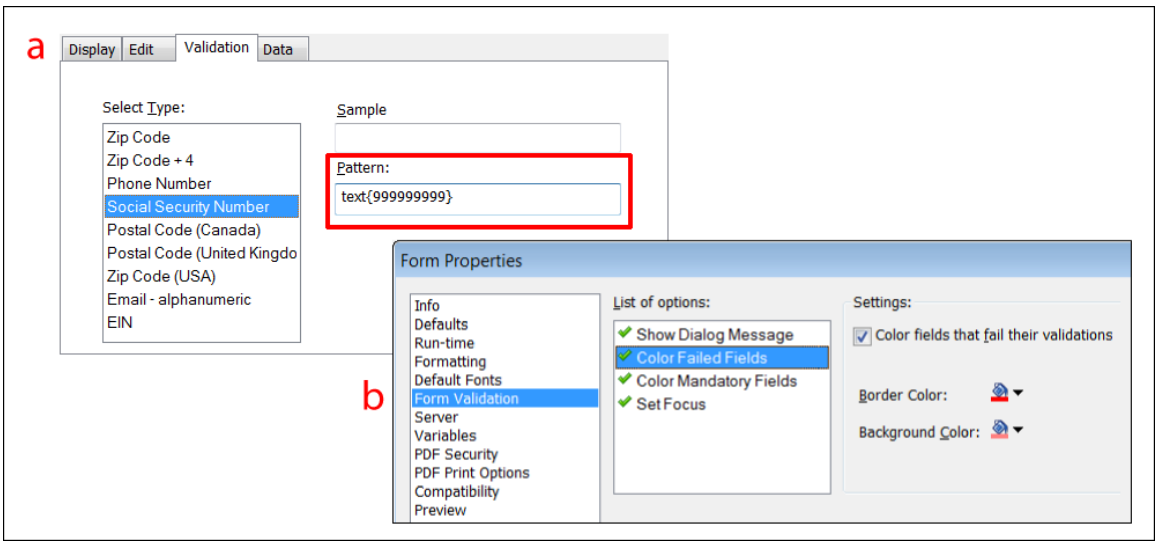
As shown here, the US Social Security number has a Display Pattern but not a Data Pattern. So, the form filler will see a social security number with the dashes, but the exported data will just be the numbers.



Data validation

Data binding and data formatting are closely related to data validation in Designer. Data validation is the process of checking the data on an interactive form, and Designer provides many methods and tools. You can validate data with patterns and with JavaScript. Designer also includes powerful visual tools to help you with form validation and you will use these tools in the exercises. There are two primary strategies for data validation.

- **Field validation:** You can validate data as a user fills in each field. You can do this with Validation patterns *(a in the illustration below)* or with JavaScript.
- **Form validation:** You can also validate the entire form when a user submits the form. In this case, you are typically validating that the user has completed all required fields. You can create form validation with Designer's *Form Validation* tool in the *Form Properties* dialog box *(b in the illustration below)*.

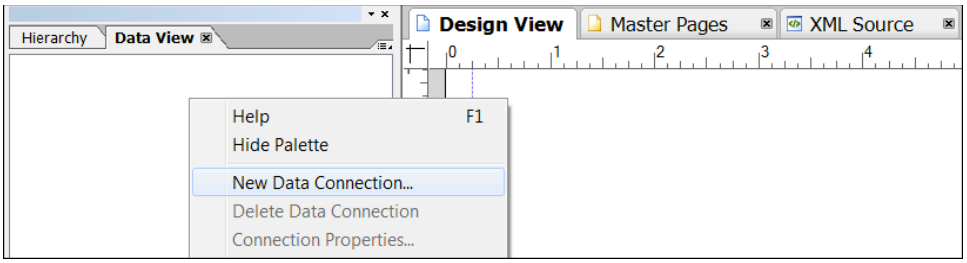


Exercises

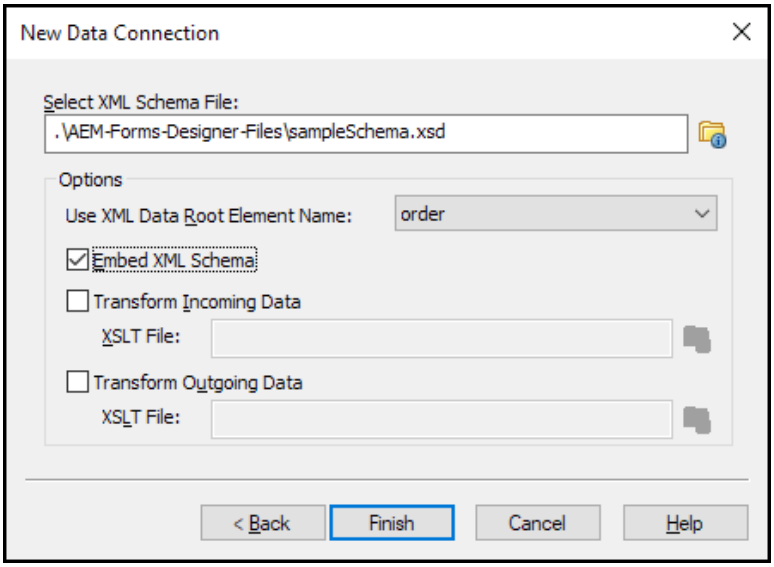
Binding to an XML schema

If you're new to Designer, this is a simple but effective demonstration of the remarkable integration that Designer has with data files. In a corporate environment, this is important because your forms will produce data that meets all the stringent requirements of your enterprise systems and databases. Follow these steps to bind a new blank form to an XML schema:

1. Create a new blank form by choosing **File – New**. Designer opens the *New Form Assistant*.
2. Select **Use A Blank Form**, and click Next to continue.
3. Keep the defaults on all subsequent screens, and keep clicking **Next** until you get to the **Finish** button.
4. Click **Finish** to create your new form.
5. Switch from the Hierarchy palette to the **Data View** palette on the left side of your workspace. If you don't see the Data View tab, choose Window - Data View. The Data View palette should be empty.
6. **Right-click** on the empty Data View palette, and choose **New Data Connection** *(see illustration)*. Designer launches the New Data Connection wizard.



7. Enter **sampleSchema** as the name of your connection.
8. Select **XML Schema** and click **Next**.
9. Click **Browse** *(the folder icon)* and navigate to your Student Files.
10. Select *sampleSchema.xsd* and click **Open**.
11. Select **Embed XML Schema**. This will include the schema in your form file.

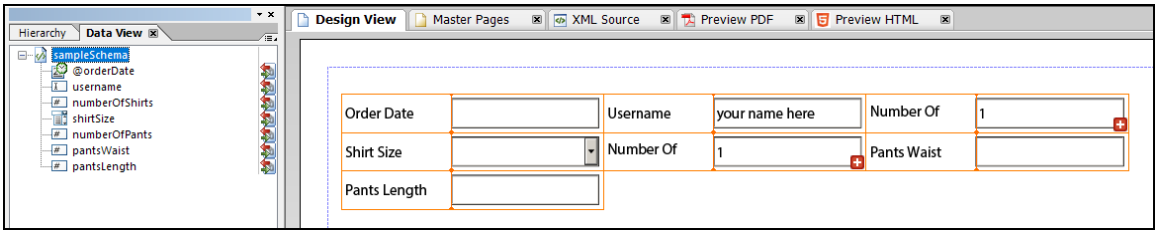


Note: *Although you won't use these options in this exercise, you can transform the XML data your Designer forms use at runtime with an XSLT file.*

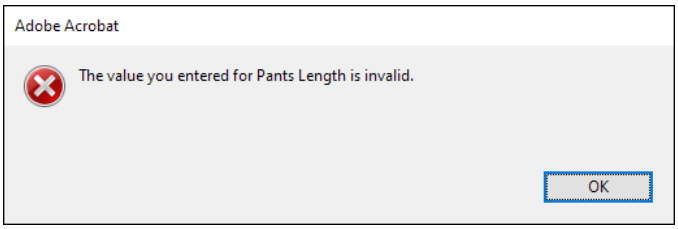
12. Click **Finish** to create your connection.

Your Data View palette will be populated with the data objects illustrated below.

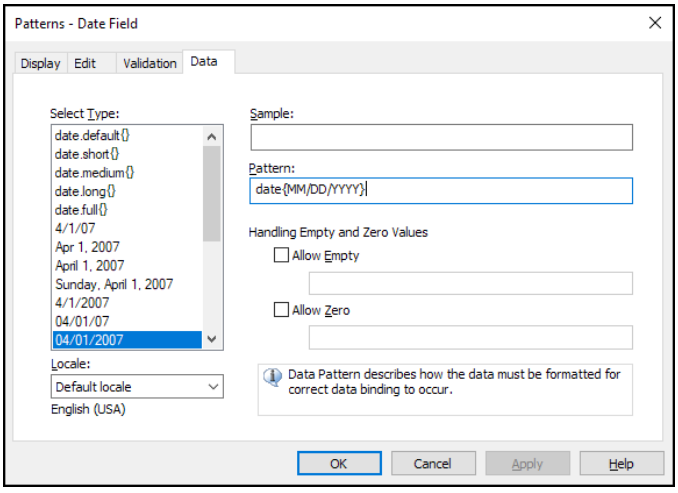
13. **Drag and drop** the data connection *(sampleSchema)* onto the upper-left corner of your form. Designer automatically creates the data-bound form fields for you. Your screen should now look like this.



14. Select **File – Save As** and navigate to your working folder.
15. Enter **mySampleSchemaForm.pdf** for the *File name*.
16. Click the *Save as type* dropdown list and select **Adobe Dynamic XML Form (*.pdf)**.
17. Click **Save**.
18. Select **Preview PDF** in the Layout Editor to see that attributes from the schema are now in your form.
19. Click the **Shirt Size** dropdown list. The three available choices—*Small, Medium, and Large*—came directly from the XML schema file. These values are an enumeration in the schema, and this example shows you one way your forms can benefit from explicit data binding.
20. Enter a length of **27** in the Pants Length Text Field and press Enter. You'll receive a *"The value you entered for pants length is invalid."* message. This data validation JavaScript was automatically generated by Designer based on a restriction in the schema file.

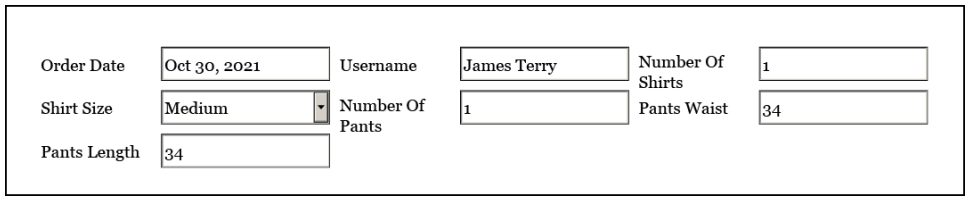


21. Select **Design View** to close the PDF preview.
22. Select the *Order Date* field. This is an example of a Designer Date field, and it contains a data pattern.
23. Click the **Patterns** button on the *Field* tab of the *Object* palette to bring up the *Patterns* dialog box.
24. Select the *Data* tab to see the data pattern **date{YYYY-MM-DD}.** The XML data that's output from this field will correspond to this pattern.
25. Delete the pattern and select the **04/01/2007** pattern.
26. Click **OK**.



27. Select **File – Save**.

28. Open **Adobe Acrobat**.
29. Select **File – Open** and navigate to your working folder.
30. Select *mySampleSchemaForm.pdf* and click **Open**.
31. Complete the form *(see illustration)*.



32. Select **Edit – Form Options – Export Data**.
33. Click **Save** and open the XML file in *XML Spy* or a similar XML Editor. Note: You can also use a Text Editor like NotePad if you don't have an XML Editor.

Your data will look like this. Notice how the data pattern has stylized the *orderDate* value.

```
<?xml version="1.0" encoding="UTF-8"?>
<order orderDate="10/30/2021">
    <username>James Terry</username>
    <numberOfShirts>1</numberOfShirts>
    <shirtSize>Medium</shirtSize>
    <numberOfPants>1</numberOfPants>
    <pantsWaist>34</pantsWaist>
    <pantsLength>34</pantsLength>
</order>
```

Floating Fields

34. Go back to Designer if you are in a different application.
35. Select **File – Open** and navigate to your Student Files.
36. Select *AdobeDisclaimerWithFloatingFields-Start.xdp* and click **Open**.
37. Select **Preview PDF**.
38. Notice the first paragraph is missing some important data. Also, notice the date for the promotion is hard-coded to be on September 1st, 1995 *(see illustration)*. We will correct both of these issues.

BOILERPLATE DISCLAIMER FOR PROMOTIONAL/SALES/DISCOUNT OFFERS

*Offer starts at on September 1st, 1995 and expires at on . Offer available in

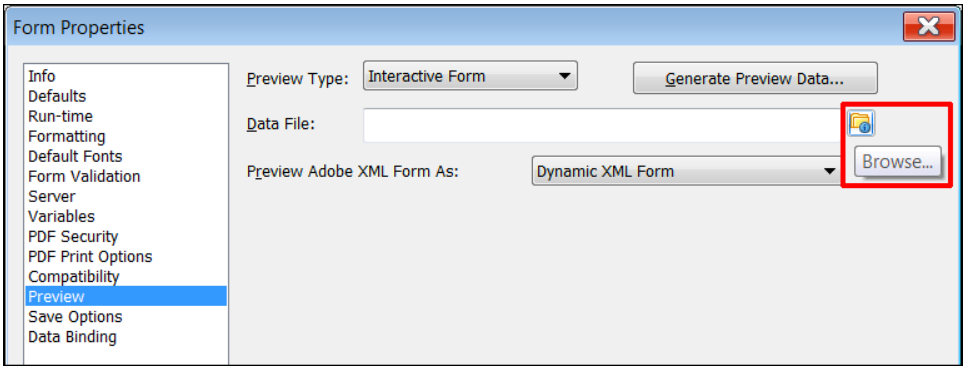
General Terms: Valid only for eligible persons who are 18+. National, state or local governmental and political officials and residents or persons in embargoed countries or countries subject to U.S. or local export restrictions, are not eligible. Adobe authorized reseller prices may vary from prices quoted above which are estimated street prices only. Offer and prices subject to change without notice due to unforeseen circumstances. Offer may not be assigned, exchanged, sold, transferred, combined, or redeemed for cash or other goods and services not expressly stated here as included. Subject to availability and U.S. export controls law and laws where the recipient resides. Additional terms and conditions may apply. VOID WHERE PROHIBITED OR RESTRICTED BY LAW.

39. Select **Design View**.

40. Notice the first paragraph is made up of many floating fields. These floating fields are bound to the variable data to produce many different types of promotions.

41. Select **File – Form Properties**. Designer opens the Form Properties dialog box.

42. Select **Preview** and click the **Browse** icon to right of the *Data File* field *(see illustration)*.

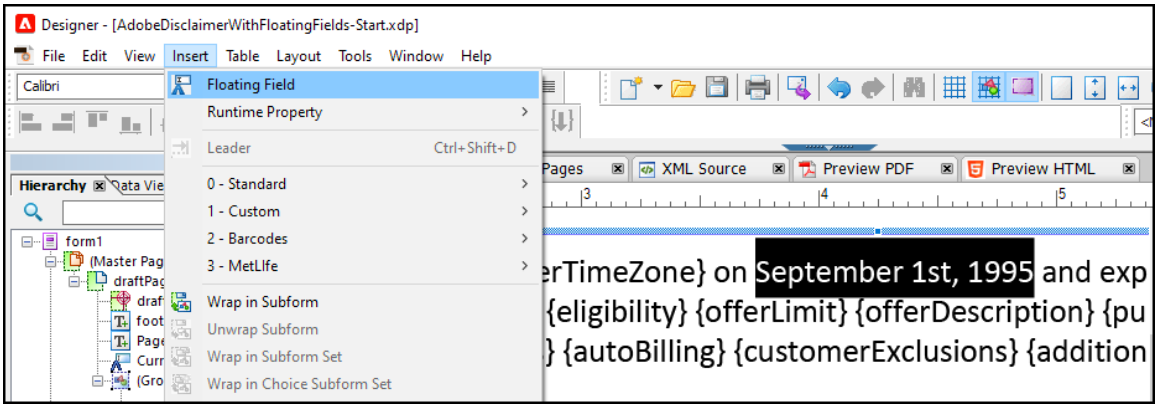


43. Navigate to your Student Files.

44. Select *AdobeDisclaimer-Data.xml* and click **Open**.

45. Click **OK** on the Form Properties dialog box.

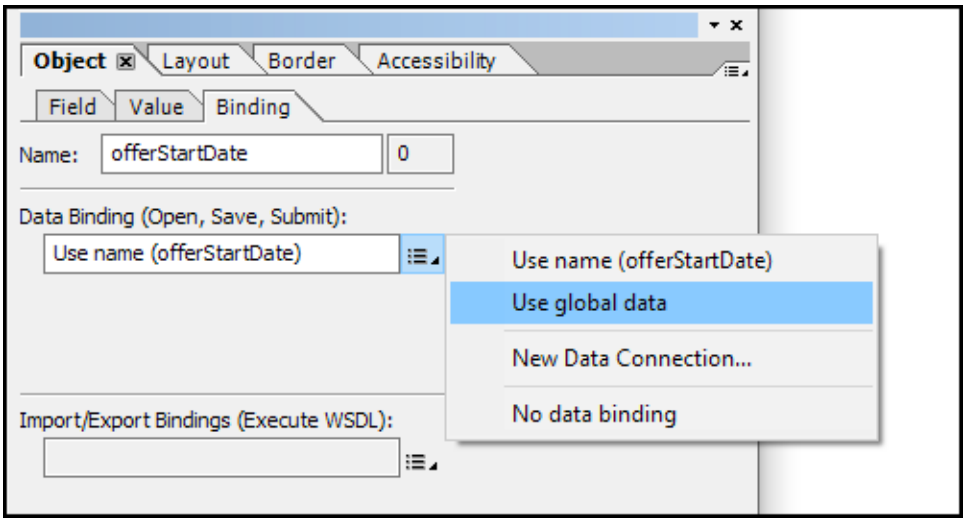
46. Select the *September 1st, 1995* text item *(see illustration)* and select **Insert – Floating Field**.



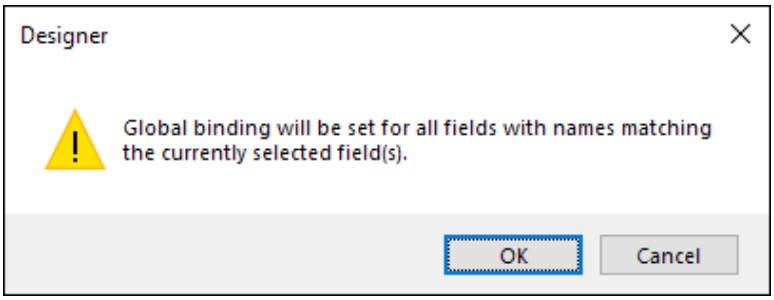
47. Select the *Binding* tab of the *Object* palette.

48. Enter **offerStartDate** as the Name *(see illustration)*.

49. Click the *Data Binding* dropdown and select **Use global data**.



50. Click **OK** on the global data message box.



51. Select **Preview PDF** to see the template fill with data. Notice the variable data from the XML file and the static text from the template work perfectly together to create the first paragraph *(see illustration)*.

BOILERPLATE DISCLAIMER FOR PROMOTIONAL/SALES/DISCOUNT OFFERS

*Offer starts at 9:00 AM EST on Jun 15, 2018 and expires at 9:00 AM EST on Jun 15, 2018. Offer available in the United States and Canada (excluding Quebec) only to existing subscribers who receive this email directly from Adobe. Get 40% off regular price on AEM Forms Designer 6.2 and AEM Forms Workbench 6.2 when you purchase this summer through the Adobe store. Must use the promotion code: ADOBE40 when prompted. Products must be purchased in the same sales transaction to be eligible for the savings. Limit: 1 subscribing customer. 12-month subscription required. At the end of your offer term, your subscription will be automatically billed at the standard subscription rate, unless you elect to change or cancel your subscription. Offer not available to volume licensing customers. For more information, call 800-222-3333.

General Terms: Valid only for eligible persons who are 18+. National, state or local governmental and political officials and residents or persons in embargoed countries or countries subject to U.S. or local export restrictions, are not eligible. Adobe authorized reseller prices may vary from prices quoted above which are estimated street prices only. Offer and prices subject to change without notice due to unforeseen circumstances. Offer may not be assigned, exchanged, sold, transferred, combined, or redeemed for cash or other goods and services not expressly stated here as included. Subject to availability and U.S. export controls law and laws where the recipient resides. Additional terms and conditions may apply. VOID WHERE PROHIBITED OR RESTRICTED BY LAW.

52. Select **Design View** to close the PDF preview.

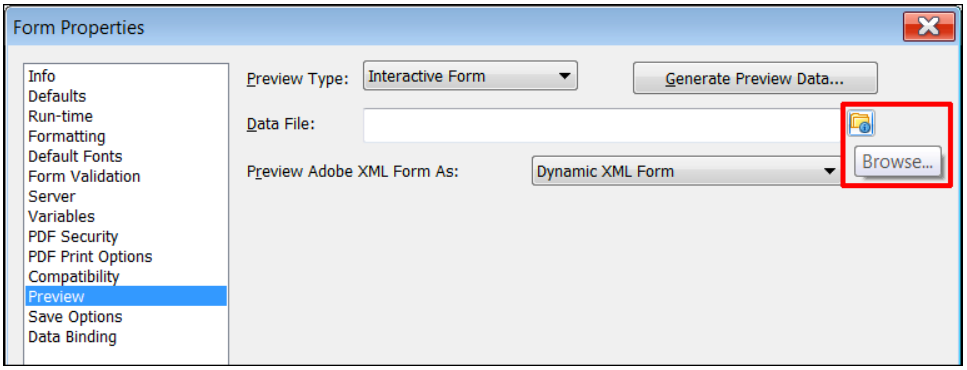
Data Driven Documents

53. Select **File – Open** and navigate to your Student Files.

54. Select *dataDrivenDocument.xdp* and click **Open**.

55. Select **File – Form Properties**. Designer opens the Form Properties dialog box.

56. Select **Preview** and click the **Browse** icon to right of the *Data File* field *(see illustration)*.



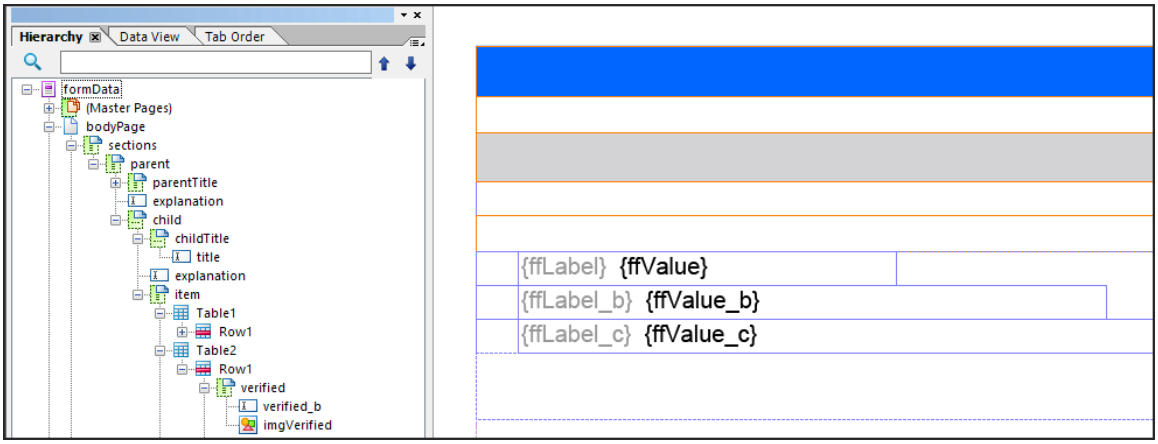
57. Navigate to your Student Files.

58. Select **dataDrivenDocument-1.xml** and click **Open**.

59. Click **OK** on the *Form Properties* dialog box.

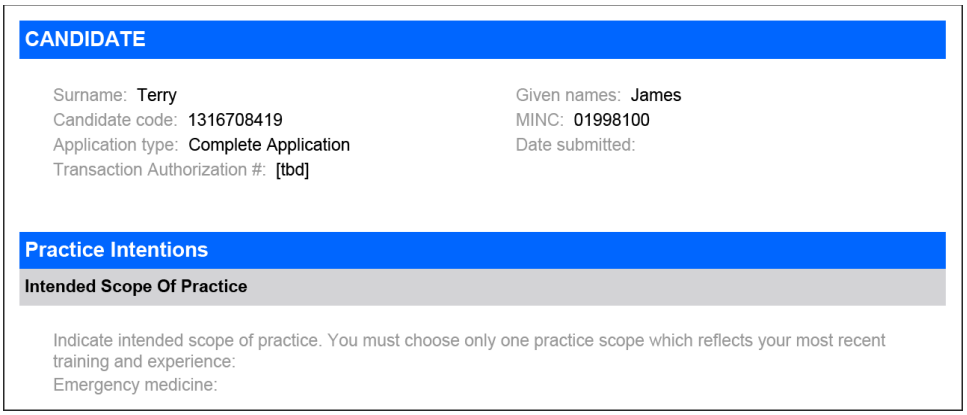
60. Notice that this template is about ¼ of a page *(see illustration)*.

61. Expand **bodyPage –Sections –Parent – Child –item** *(see illustration)*. Notice the sophisticated hierarchy indicates that this template likely has some dynamic behavior.

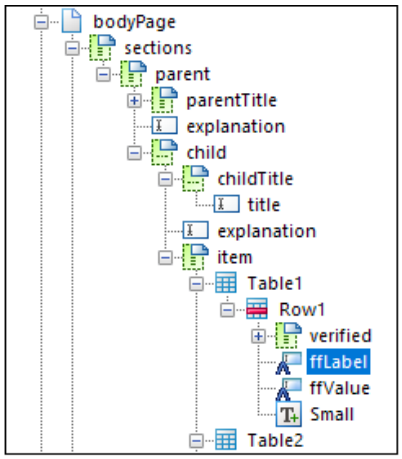


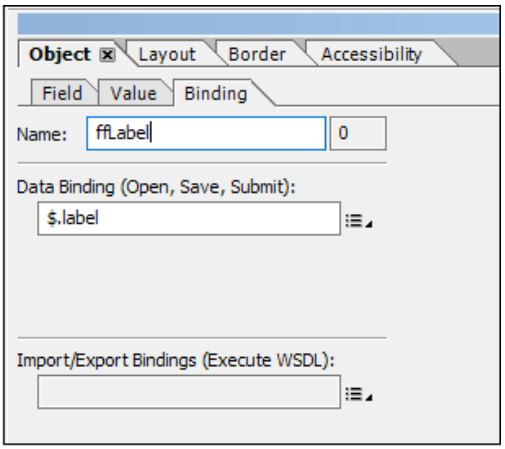
62. Select Preview **PDF**.

63. Notice that our ¼ of a page template has been used to generate 11 PDF pages *(see illustration)*.



64. Select *Design View*.
65. Expand the Hierarchy *(see illustration)* so you can see *Row1* in *Table1*.
66. Select the *ffLabel* floating field *(see illustration, left)*.
67. Select the *Binding* tab of the *Object* palette *(see illustration, right)*.
68. Notice the floating field is bound to the **$.label** data node.



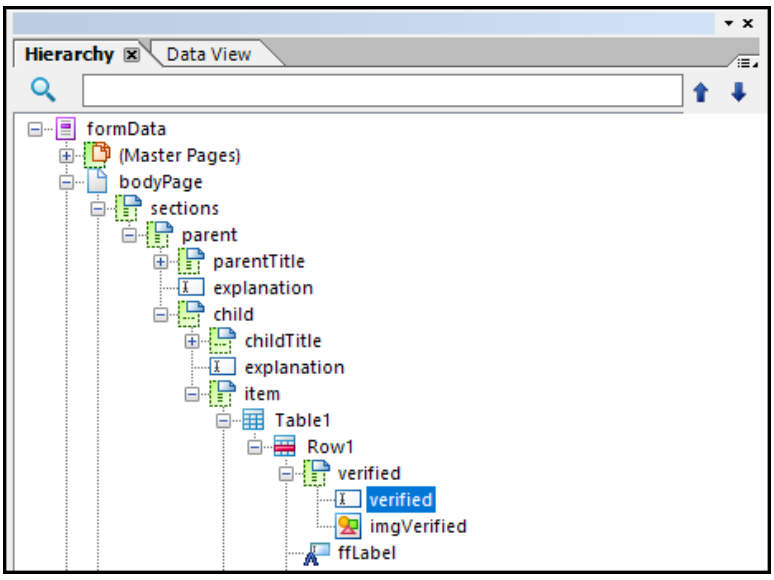


69. Select the *ffValue* floating field in the *Hierarchy* palette.
70. Notice the floating field is bound to the **$.value** data node in the *Binding* tab of the *Object* palette.
71. Also expand *Table2* and *Table3* and notice the bindings on the floating fields are bound to different data nodes.

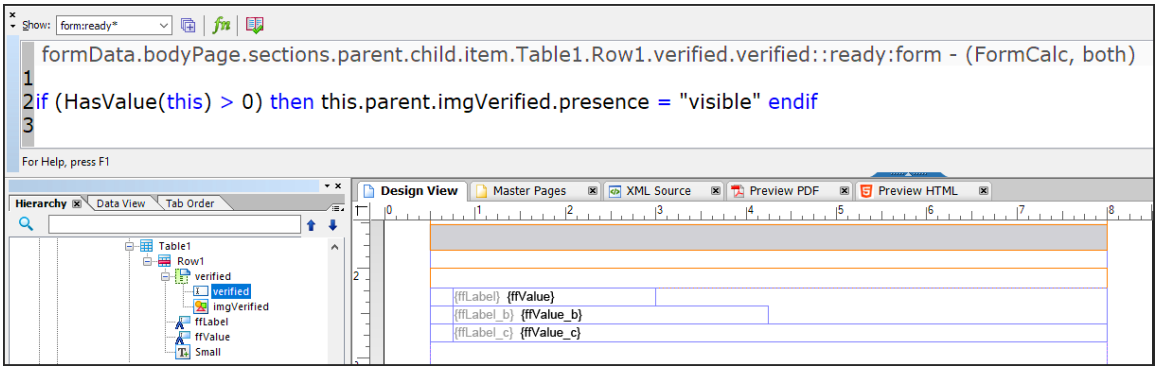
– Table2 uses *$.label_b* and *$.value_b*.

– Table3 uses *$.label_c* and *$.value_c*.

72. Expand the *verified* subform in any of the tables *(see illustration)*.
73. Select the *verified* text field *(see illustration)*.



74. Open the *Script Editor* and select the *form:ready* event *(see illustration)*.



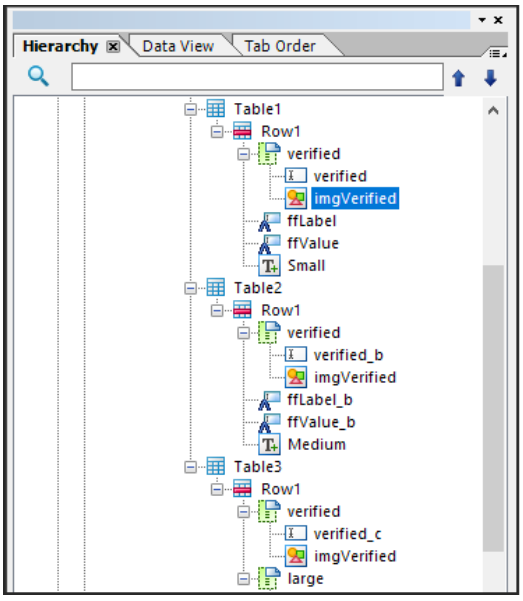
if (HasValue(this) > 0) then this.parent.imgVerified.presence = "visible" endif

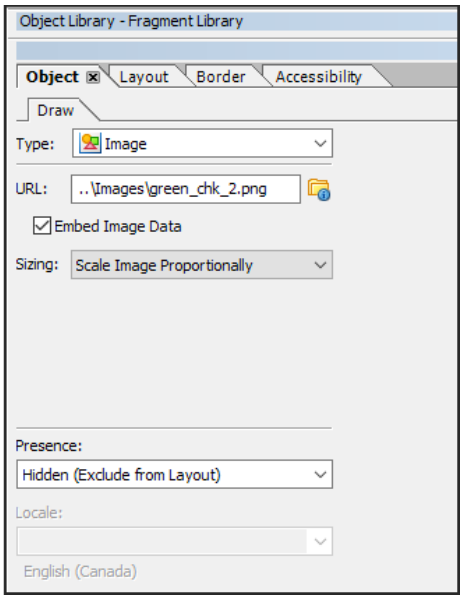
75. The script checks to see if there is a value in the field and if there is it sets the *imgVerified* image field's presence property to "visible".

76. Select the *imgVerified* image field *(see illustration)*.

77. Select the *Field* tab of the *Object* palette *(see illustration)*.

78. Notice the initial state for this image is *Hidden (Exclude from Layout)*.

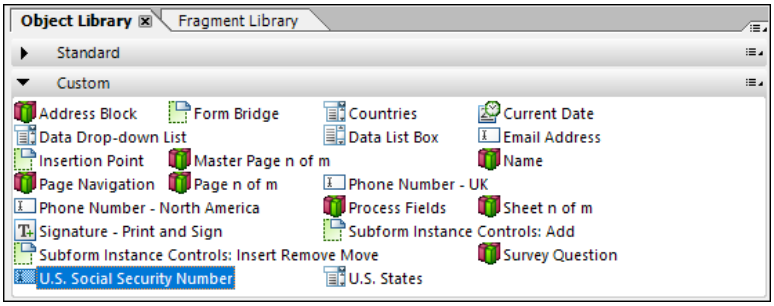




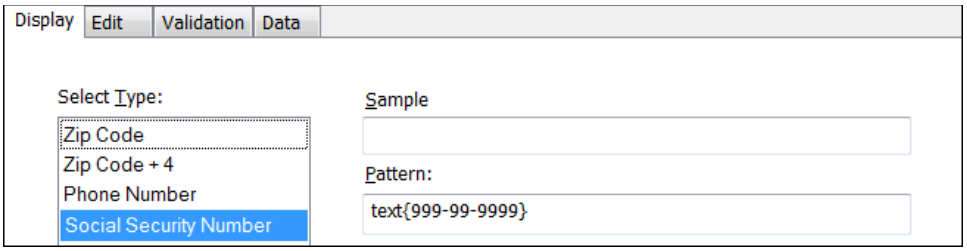
Data Formatting

Follow these steps to learn more about data formatting in Designer:

1. Create a new blank form by choosing **File – New**. Designer launches the New Form Assistant.
2. Select **Use A Blank Form**, and click **Next** to continue.
3. Keep the defaults, and click the **Finish** button to create your new form.
4. Expand the *Custom* tab of the *Object* Library palette so you can see the custom objects *(see illustration)*.



5. Drag and drop the **U.S. Social Security Number** object from the *Custom Object Library* to your form.
6. With the U.S. Social Security Number *(SSN)* object selected, select the *Field* tab of the *Object* palette.
7. Click the **Patterns** button. The Patterns dialog box appears. The Display tab shows the pattern that will be assigned to the nine digits that are entered into this field at runtime.



8. Select the **Validation** tab.
9. The validation pattern indicates that the field requires 9 digits.
10. Click **OK** to close this dialog box.
11. Click **Preview PDF** and enter nine digits into the SSN field. Upon field exit, the display of your nine digits will match the data format that was set in the Patterns dialog box.
12. Select **Design View** to close the PDF preview.
13. Select **File – Save As** and navigate to your working folder.
14. Enter **<yourname>Validation** as the *File name*.
15. Click the *Save as type* dropdown and select **Adobe XML Form (*.xdp)**.
16. Click **Save**.

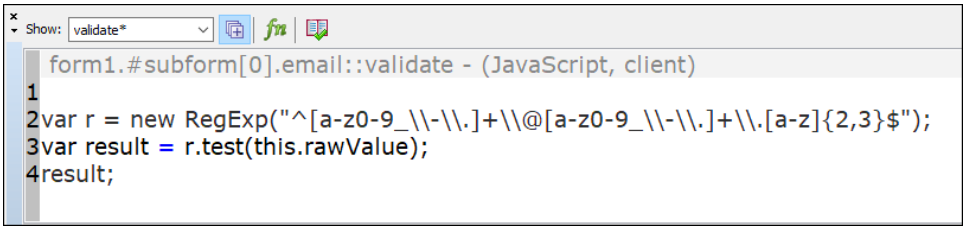
    Note: *The built-in Designer patterns work fine in PDF forms and documents, but not all of them work when you render your form as HTML.*

Data Validation

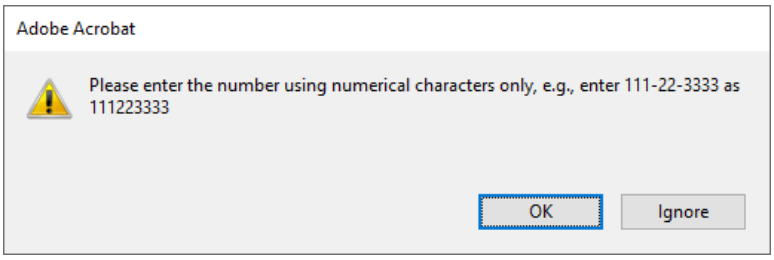
17. Drag and drop the **Email Address** object from the *Custom Object Library* to your form and put it below your SSN field.
18. With the Email Address selected, open or expand the **Script Editor**.
19. Select the **validate** event.
20. You will see a script that is more complicated than it needs to be. The same functionality can be found with this much simpler script.

```
var r = new RegExp("^[a-z0-9_\\-\\.]+\\@[a-z0-9_\\-\\.]+\\.[a-z]{2,3}$");
var result = r.test(this.rawValue);
result;
```

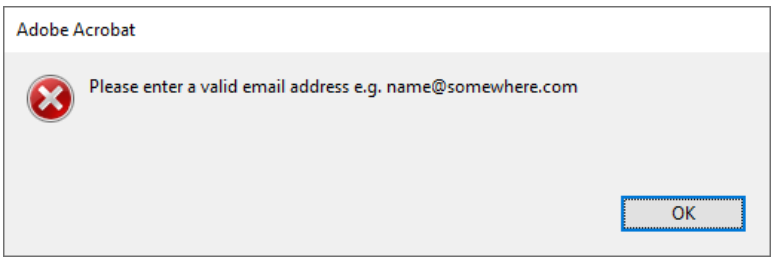
21. Copy the script above and paste it into the validate event in your Script Editor *(see illustration)*.



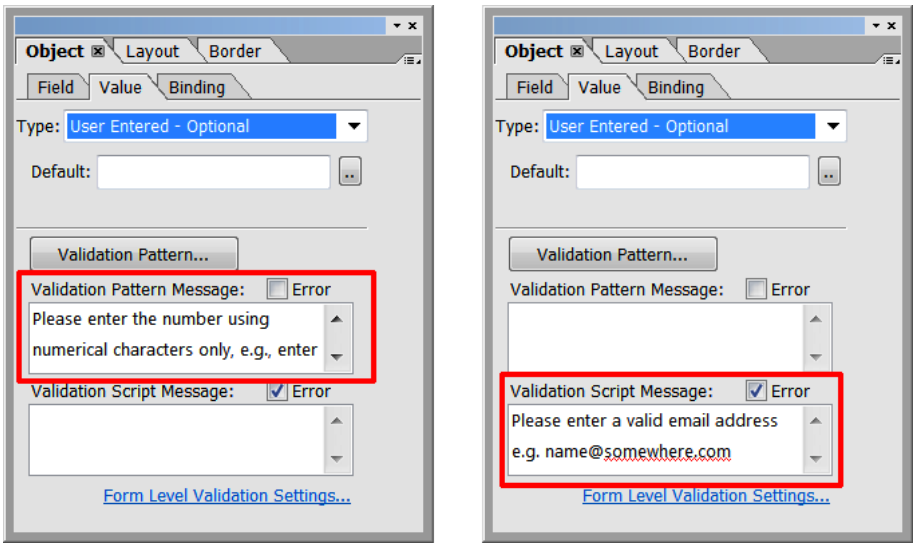
22. Click **Preview PDF**.
23. Enter eight digits into the SSN field and hit Tab or Enter on your keyboard.
24. Notice the message and the Warning icon *(see illustration)*.



25. Click **Ignore** to close the message box.
26. Enter **NotAnEmailAddress** in the email address field and hit Enter on your keyboard.
27. Notice the message and the Error icon *(see illustration)*.



28. Click **OK** to close the message box.
29. Select **Design View** to close the PDF preview.
30. Select the **U.S. Social Security Number (SSN)** object.
31. Select the *Value* tab of the *Object* palette.
32. Notice the **Validation Pattern Message** *(see illustration on left)*.
33. Select the **Email Address** object.
34. Select the **Value** tab of the Object palette.
35. Notice the **Validation Script Message** *(see illustration on right)*. Notice the Error checkbox is selected.

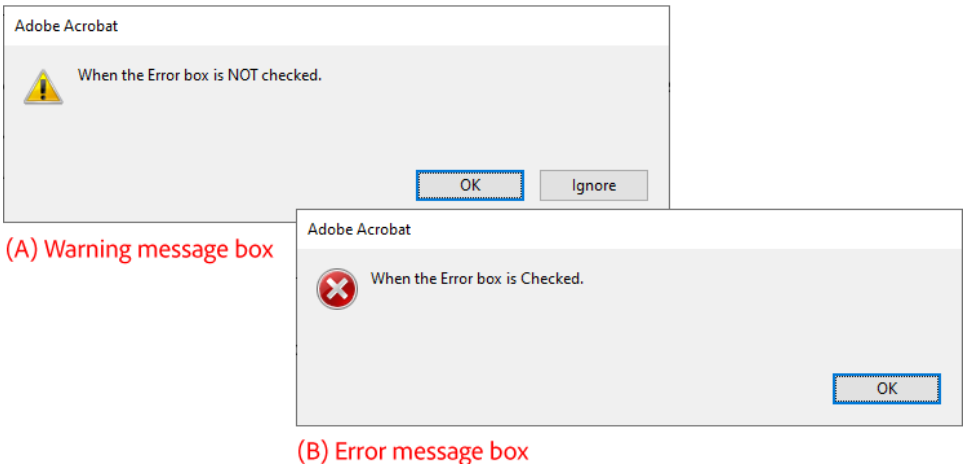


Note: *You can customize these messages. To start a new line in a message, press Ctrl+Enter.*

Error or Suggestion

These next few steps will further examine the difference between checking and not checking the Error checkbox. When Error is checked, the user will see an Error icon appear instead of a Warning icon.

1. Select **File – Open** and navigate to your Student Files.
2. Select *formValidation-Start.xdp* and click **Open**.
3. Select **File – Save As** and navigate to your working folder.
4. Enter **<yourname>FormValidation** as the *File name*.
5. Click the *Save as type* dropdown and select **Adobe XML Form (*.xdp)**.
6. Click **Save**.
7. Click the first SSN field and select the *Value* tab in the *Object* palette. Notice that the Error checkbox is NOT checked. This is also true for the first email field.
8. Click the second SSN field and select the *Value* tab in the *Object* palette. Notice that the Error checkbox is checked. This is also true for the second email field.
9. Select **Preview PDF**.
10. When you fail a validation for the first two fields, you will see the **Warning message box (A)**. You can choose to ignore the warning by clicking the *Ignore* button.
11. When you fail a validation for the second two fields, you will see the **Error message box (B)**.



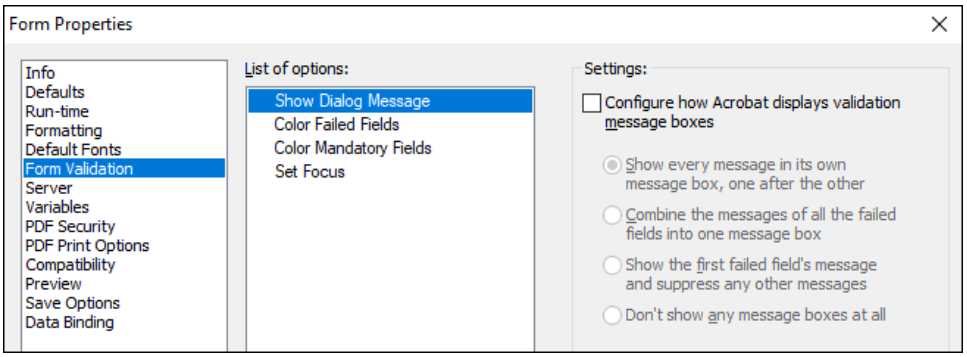
12. Notice there is no *Ignore* button on the **Error message box**

13. Select **Design View** to close the PDF preview.

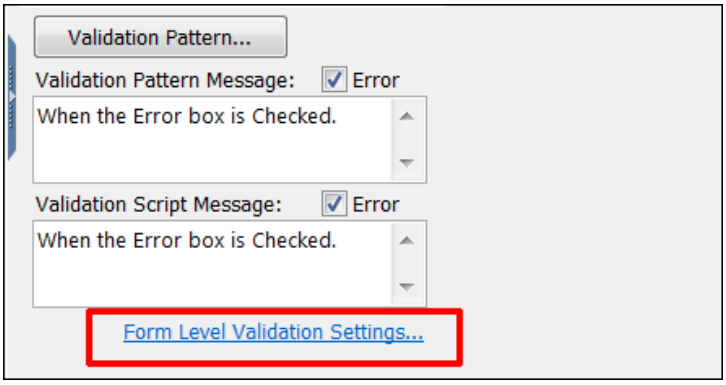
Note: *The Form Level Validation Settings (**File – Form Properties – Form Validation**) can override the Field Level Settings. For instance, if you select **Combine the messages of all the failed fields into one message box** in your Form Level Validation Settings, all the message boxes will default to the Error icon.*

Form Level Validation Settings

14. Select **File – Form Properties – Form Validation** to open the Form Level Validation settings.

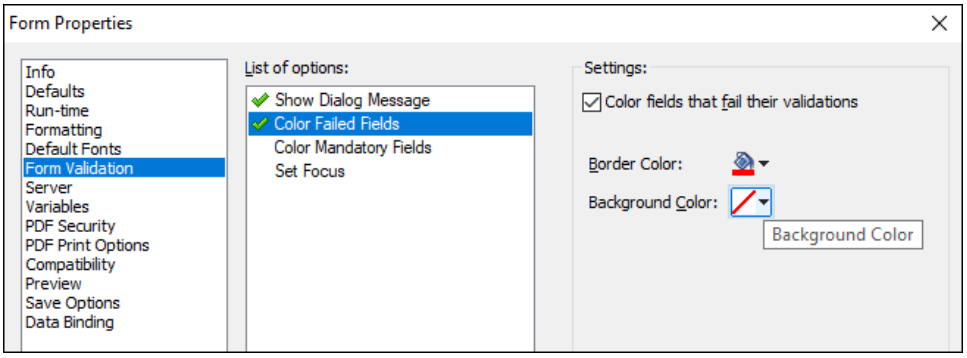


Note: *You can also open these settings by clicking the blue link at the bottom of the Value tab in the Object palette.*

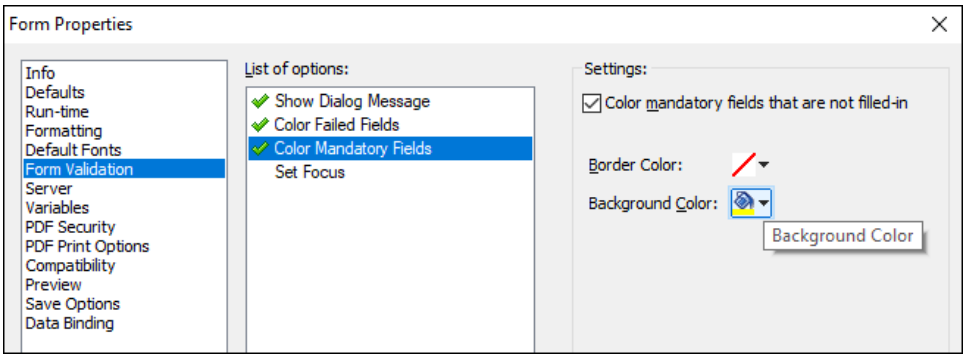


15. Select **Show Dialog Message** in the *List of options*. This is likely the default option.

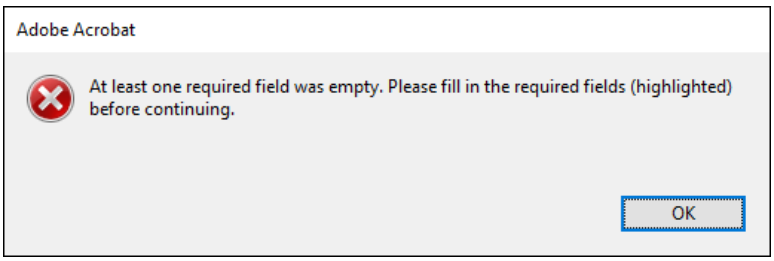
16. Select **Configure how Acrobat displays validation message boxes**.
17. Select **Show every message in its own message box, one after the other**.
18. Select **Color Failed Fields** in the *List of options*.
19. Select **Color fields that fail the validations**.
20. Set the Border Color to **red**.
21. Set the Background Color to **None** *(see illustration)*.
22. Your settings should now look like this *(see illustration)*.



23. Select **Color Mandatory Fields** in the *List of options*.
24. Select **Color mandatory fields that are not filled-in**.
25. Set the Border Color to **None**.
26. Set the Background Color to **yellow**.
27. Your settings should now look like this *(see illustration)*.

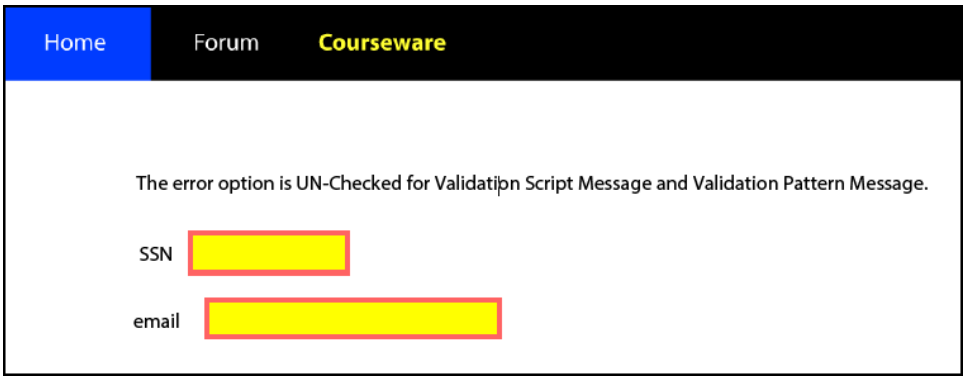


28. Click **Set Focus** in the *List of options*.
29. Select **Set Focus to the first field that fails to validate**.
30. Click **OK** in the *Form Properties* dialog.
31. Select **Preview PDF** to see how your form validation performs.
32. Click the **Submit by Email** button. You should see a message box *(see illustration)*.



33. Click **OK** in the message box.

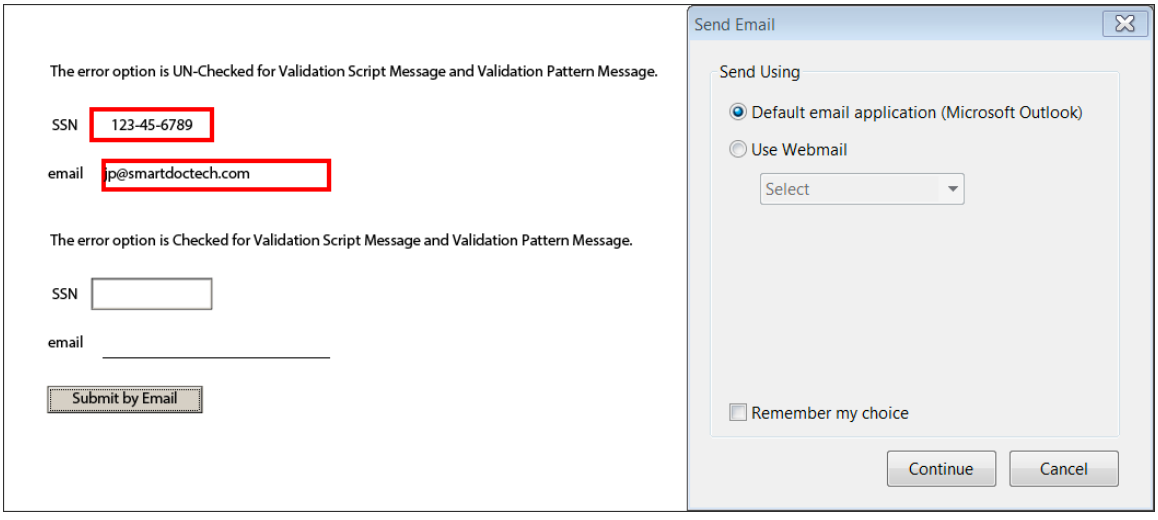
34. Notice the first two fields are colored yellow (see illustration).



35. Enter **123456789** for the SSN and hit Enter on your keyboard. Notice the yellow highlight color is removed because the field now has data.

36. Enter your email address hit Enter on your keyboard. Notice the yellow highlight color is removed because the field now has data. The changes in these fields are an indication that your mandatory fields have met the validation.

37. When your required fields are completed, you can click the **Submit by Email** button again. You should see the *Send Email* Dialog. You do not need to submit the form. The purpose is to show you how the form validation is invoked and how it is satisfied.

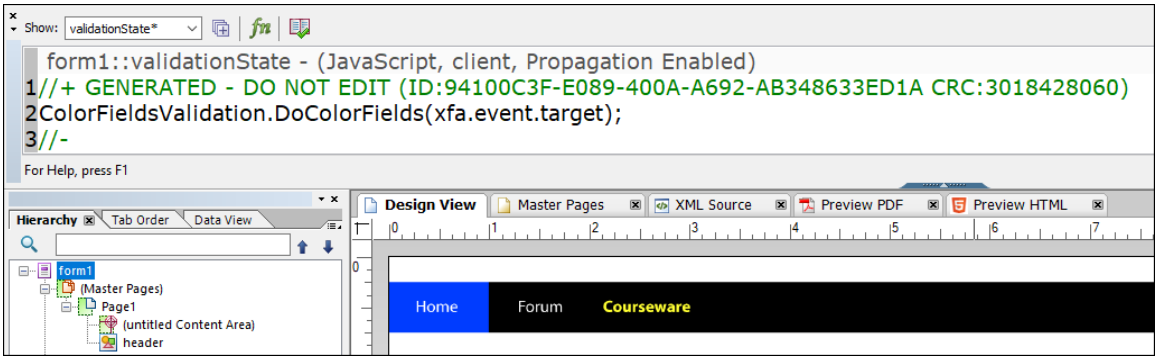


38. Click **Cancel** in the *Send Email* dialog.

39. Select **Design View**.

40. Select the **form1** node in the *Hierarchy* palette.

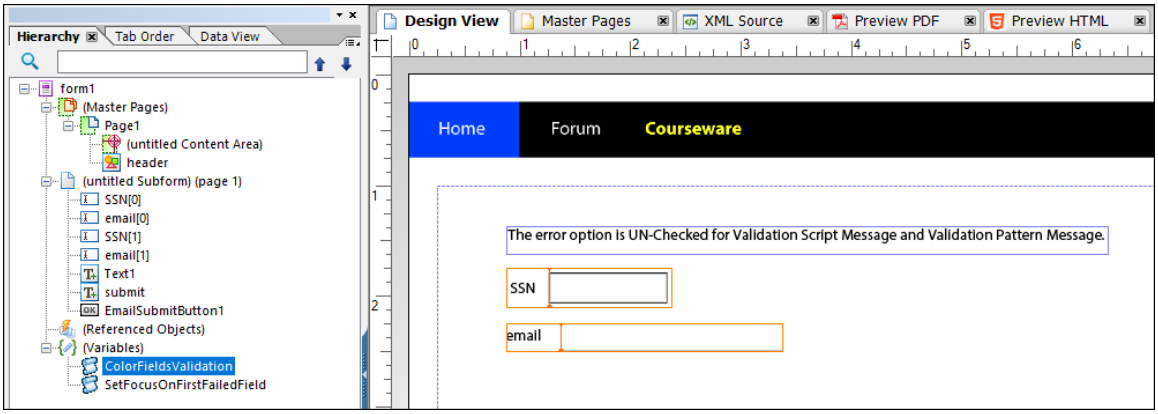
41. Click the *Show Events* dropdown and select **validationState**. You will see the script that was generated for the form validation *(see illustration)*.



42. Notice that this script calls a method of the *ColorFieldsValidation* script object.

ColorFieldsValidation.DoColorFields(xfa.event.target);

43. Select the *ColorFieldsValidation* script object in the Hierarchy *(see illustration)*.



44. Scroll down and notice the *DoColorFields* method. This is the method we are calling.

```
function DoColorFields(oInvalidNode) {
    // If this form is running on a client other than Acrobat
    // (like on the server) then don't run this script
    if (xfa.host.name != "Acrobat") {
        return;
    }
    ...
```

45. You will also see a second method called *InitializeColorFields()* in this Script Object.

```
function InitializeColorFields() {
    // Disable Acrobat's field highlighting. The Color
```

```
    // Failed Fields action takes care of highlighting fields.
    if (xfa.host.name == "Acrobat") {
        app.runtimeHighlight = false;
    }
}
```

46. Also notice the second script object called *SetFocusOnFirstFailedField*. The *DoSetFocusOnFirstFailedField* method of this object is called from a different event in the root node.
47. Select the **form1** node in the *Hierarchy* palette again.
48. Click the *Show Events* dropdown and select **postSubmit**. This is the script that calls the *DoSetFocusOnFirstFailedField* method of the *SetFocusOnFirstFailedField* object.

```
//+ GENERATED - DO NOT EDIT (ID:DA035025-30E2-437C-8219-20F5C876097B CRC:3815899415)
SetFocusOnFirstFailedField.DoSetFocusOnFirstFailedField(this);
//-
```

Important Form Topics

This section covers topics that are very important to form developers.

Fonts

The fonts you choose and the technical approaches you take will greatly affect the readability and usability of your forms. There are two basic types of fonts: serif and sans serif.

Serif fonts feature a short line, often perpendicular, that finishes off the main stroke of the letterform. They have a shorter x-height than San Serif fonts and as the older of the two styles, are generally considered to be more traditional. A font's x-height is the height of a lowercase x.

Serif fonts, as you might have guessed, do not have serifs. Sans serif fonts also have very little variation in the thickness of the stroke, a taller x-height, and a more modern appearance. You can see the main differences between serif and sans serif fonts illustrated below.

Adobe Garamond Pro (left) is a serif font, and Adobe Myriad Pro (right) is a sans serif font. Because quality design will improve the success of your forms, they should be designed by a professional graphic designer who's experienced with typography and typographic layout. This section focuses on the best technical approaches for controlling typographic quality in Acrobat, Designer, and on the server.

The following fonts are installed by Adobe with every installation of Acrobat and Reader, so they're always available:

- Courier Standard, Courier Standard Bold, Courier Standard Bold Oblique, Courier Standard Oblique
- Minion Pro Bold, Minion Pro Bold Italic, Minion Pro Italic, Minion Pro Regular
- Myriad Pro Bold, Myriad Pro Bold Italic, Myriad Pro Italic, Myriad Pro Regular
- Symbol (Type 1)

If you choose one of these fonts, you'll have the best of both worlds: typographic and layout fidelity and a small PDF file size.

Fonts Strategies

Here are some good font strategies to consider to ensure your fonts look good on an end-user's machine.

Use the standard fonts

The following fonts are installed by Adobe with every installation of Acrobat and Reader, so they're always available:

- Courier Standard, Courier Standard Bold, Courier Standard Bold Oblique, Courier Standard Oblique

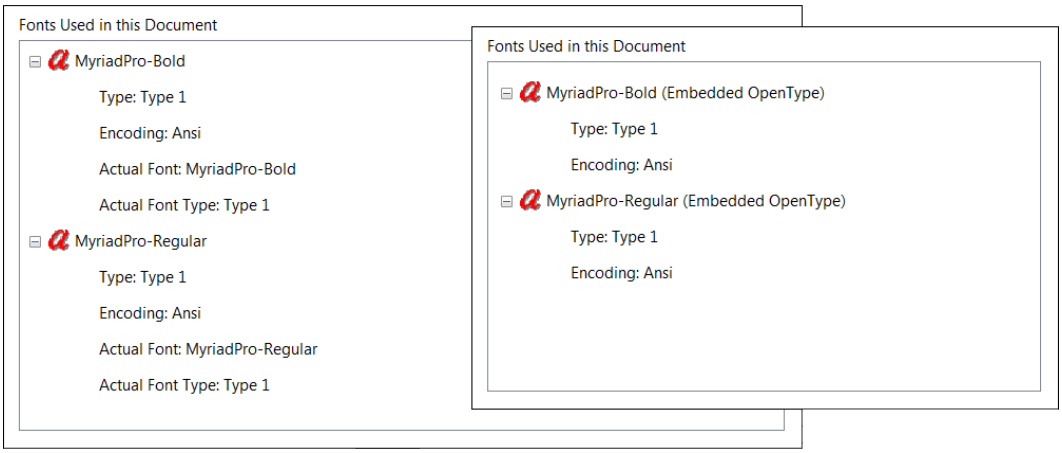
- Minion Pro Bold, Minion Pro Bold Italic, Minion Pro Italic, Minion Pro Regular
- Myriad Pro Bold, Myriad Pro Bold Italic, Myriad Pro Italic, Myriad Pro Regular
- Symbol (Type 1)

If you choose one of these fonts, you'll have the best of both worlds: typographic and layout fidelity and a small PDF file size. You can also choose a popular Windows font, or a standard font in your organization. Popular Windows fonts include *Arial*, *Calibri*, and *Times New Roman*.

Trust Acrobat's font substitution

If you can't use one of the standard fonts, you can always trust Acrobat's font substitution feature. Adobe Acrobat was designed to open PDF files and render them properly even when the file's fonts don't exist on the viewer's computer. Acrobat uses an Adobe Multiple Master typeface as a substitute for the missing font. Two Adobe Multiple Master font families are included in Acrobat—AdobeSerifMM for serif fonts, and AdobeSansMM for sans serif fonts. Acrobat's font substitution works best if your form is using a font that's stylistically similar to one of these fonts.

However, you may have a font that isn't similar. In these cases, the layout can be changed, and captions can be truncated. This is a problem for complex layouts with many fields that are packed into a small space. Once part of the layout changes, it will have a ripple effect and graphic imperfections will cascade down your form. In these cases, you may want to embed your font into your form.



Embed your fonts

This option includes specific font information in your PDF file. On one hand this is beneficial because you can be assured that your fonts will render properly on a form-filler's machine. On the other hand, selecting this option makes your file much larger, which results in longer download times for your user. Consider the Purchase Order form. The poFontsEmbedded.pdf file has embedded fonts, and its file size is 188 KB. However, if you resave this same form without embedding fonts, the file size decreases to 56 KB. The performance advantage of smaller PDF forms is so substantial that you should think twice before choosing to embed fonts.

However, if you must embed fonts, try to minimize the number of different fonts you use in your form. Embedding fonts is an all or nothing proposition; there's no option to select the specific fonts or characters that you'd want to embed.

Fonts Mapping

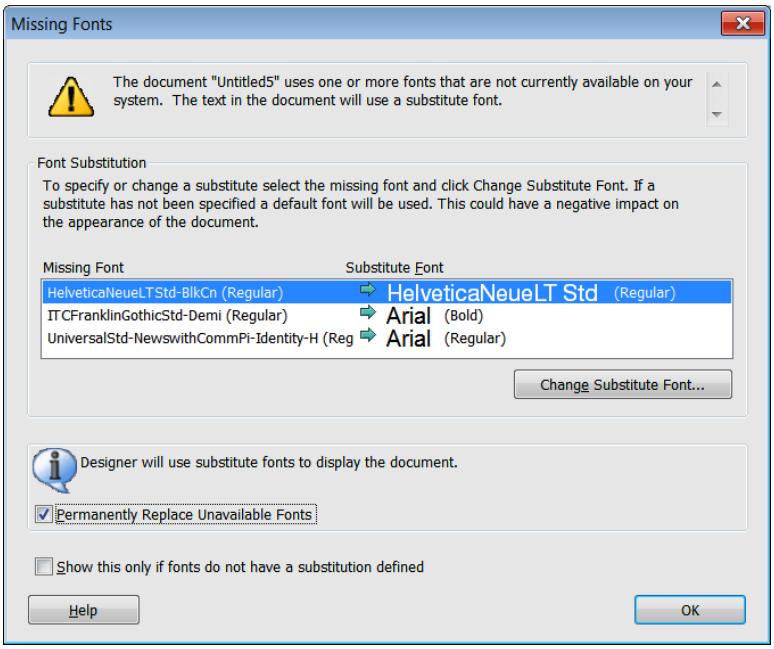
You can also control the font substitution within your Designer program. This is beneficial if you want all your form creators to map fonts on input PDFs to certain corporate standard fonts available on their systems. Designer controls

this font-mapping with a table stored in your Designer.xci file that maps available system fonts to the fonts in your files.

Replace a form's font

You can permanently replace a form's fonts with the substitute fonts on your system. You can do this with the Missing Fonts dialog box, which you can access at either of the following times:

- When you open a form with missing fonts, you will be prompted with this dialog box. Select the option Permanently Replace Unavailable Fonts. After you save the file, the fonts will be changed, and you'll no longer see this dialog box when you open the file.
- You can also access this dialog box at any time by selecting Tools – Missing Fonts. If this option is grayed out, that means your form doesn't have any missing fonts.



Tabbing

The tabbing order of your form is important because many of your form users will navigate your form by tabbing through the fields. Tabbing order is also critical if your forms need to be accessible to users who rely on keyboard commands because of vision or mobility impairments.

We recommend setting the tabbing order last because it can often be affected by other form changes that occur during development. After completion of all form changes, you'll have to apply the tabbing order only once. Otherwise, you may duplicate your efforts by re-creating the tab order after a new set of form changes.

How Tabbing Order Works

Designer's default tabbing order works from left to right and top to bottom, starting from the upper-left corner of the page. The default order is based on the precise X and Y coordinates of each object. A difference of 0.0001 inches in the Y coordinate may not be seen by the naked eye, but it will affect the tabbing order.

The tabbing order takes into account complex container objects like subforms and content areas. For instance, if you have two vertical subforms side by side on a form and each is filled with text fields, the tabbing order will work through each text field in the subform on the left before it moves to the subform on the right.

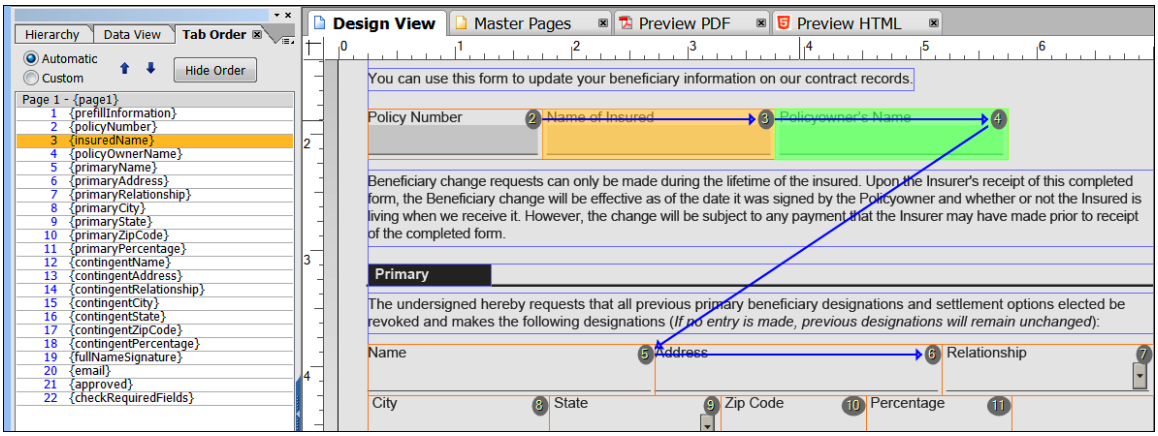
The following form objects aren't tabbed to when a form filler is navigating the form with the Tab key:

- Circle, Line, or Rectangle objects.
- Objects with a Presence property of Invisible or Hidden.
- Objects with an Access property of Protected. This property can be set with the following JavaScript: **TextField1.access = "protected";**

Designer treats a RadioButtonList object as a single stop in the tabbing order. A form filler must use the Up Arrow and Down Arrow keys on the keyboard to navigate through the radio buttons in a radio button list.

The Tab Order Palette

You can use the Tab Order palette to override the default tabbing order for your form. If you're working on a long and complicated form, you can edit the tabbing order one page at a time and save your changes intermittently. You rearrange the order of the accessible fields within the Tab Order palette by dragging and dropping the object names or by selecting an item and using the up and down arrows at the top of the palette. Designer will highlight each field in the tab order with a numbered icon in the Design View.



Some objects that aren't tabbed to are still part of the tabbing order because the tabbing order is also used to determine the reading order in accessible forms. If your objects are set to Invisible, Hidden, or Protected, the form filler won't tab to them when tabbing through the form, but the objects will still be part of the form's tabbing order.

Accessibility

Different people will use our forms in different ways. Some may navigate our forms with a mouse while others may navigate them with a keyboard. Some users may have difficulty with low contrast typography while others may have challenges with certain colors. And some users will access our forms through an audible screen reader. In all of these cases, our forms will be greatly improved by adding Accessibility features.

Note: *This topic is covered in more detail in the Accessibility – PDF course on gemforms.training.*

Localization

In addition to making sure your forms have the proper fonts, it's also important to make sure they are displaying the proper currency symbols and date formatting. If you're developing forms for an international audience, consider the expectations and requirements that different cultures have. For instance, Americans see a date formatted as 10/12 and think it is October 12th. However, Europeans will see the same date format and think it is December 10th.

Designer enables you to address these differences with the locale property. You can set this property at the field or form level. The following table shows the patterns and symbols that different locales will use by default. Setting a specific display pattern on these fields will override the default locale setting.

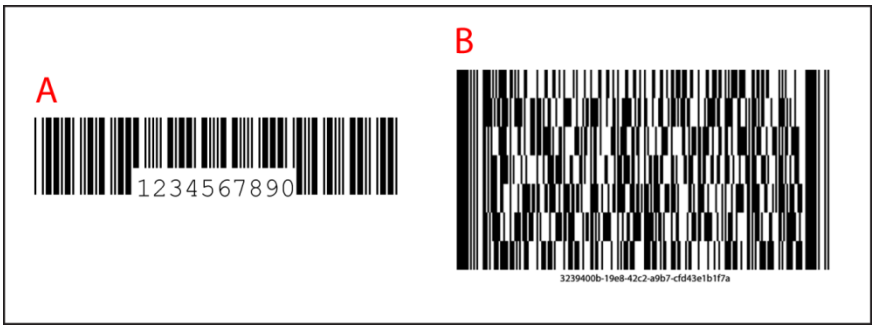
Locale Settings Defaults

	Default Locale (English USA)	English UK	Spanish (Spain)	Chinese (China)
Dates	MMM DD, YYYY	DD MM YYYY	DD/MM/YYYY	YYYY-MM-DD
Currency	$9,999.99	£9,999.99	9.999,99 €	¥ 9,999.99
Numbers	9,999.99	9,999.99	9.999,99	9,999.99
Time	hh:MM:SS A	HH:MM:SS	HH:MM:SS	Ahh:MM:SS

Barcodes

Even when you supply an interactive form to your users, it's possible that they'll send back a printed form to complete the workflow. In these cases, you can encode some or all of the form data into barcodes, making it easier to bring the encoded data into your database via a barcode scanner.

Designer supports two software barcode types: the one-dimensional or linear type and the two-dimensional type. You can find Designer's barcode objects in the Barcodes tab of the Library palette. One-dimensional barcodes enable you to encode a single value into a barcode object. If you need to encode more than one value into a single barcode, use the two-dimensional Paper Forms Barcode. This barcode object encodes data that a user types into a form and works with a server module called Adobe Barcoded Forms.



Designer supports one-dimensional barcodes (A) and two-dimensional barcodes (B).

One-dimensional barcodes

You can encode a single value into a one-dimensional barcode at design time or at runtime. At design time, you can enter a value into the Default field of the Value tab. If you need to encode a single value at runtime, you can use the rawValue property of your barcode:

PostUS5ZipBarCode1.rawValue = "90210";

Two-dimensional barcodes

The Paper Forms Barcode is an example of a two-dimensional barcode. This barcode object implements the industry standard PDF417 format. You can add this barcode object to a form to encode the user's data at runtime. To see this functionality, the form must be opened in Acrobat Professional or Reader Extended for use in Adobe Reader. Follow these steps to see an example.

PDF417 Barcodes

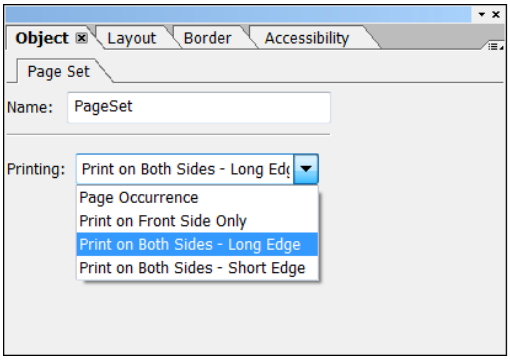
In this case, PDF stands for Portable Data File, which is not the same as Adobe's PDF (Portable Document Format). Barcodes of this type are typically used in identification cards and inventory management. PDF417 is an industry-standard, two-dimensional barcode format with the following features:

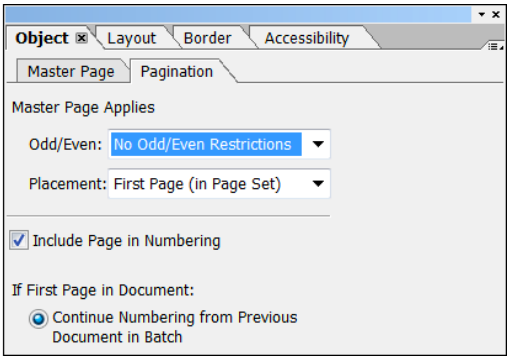
- **More storage:** PDF417 barcodes can store up to 2710 characters of data, which is more than one-dimensional barcodes can store.
- **Fast reading:** PDF417 barcodes can be read nearly as fast as one-dimensional barcodes.
- **Redundancy:** Even if part of the barcode becomes damaged through faxing or some other activity, the data that the barcode represents can be reconstructed.

Designer includes two PDF417 barcodes: the Paper Forms Barcode and the PDF417 barcode object. Unlike the Paper Forms Barcode, the PDF417 barcode accepts only one value. However, unlike the other one-dimensional barcodes, the PDF417 barcode includes the redundancy of the PDF417 standard.

Pagination

There are various ways to control the pagination of a dynamic document. You learned one technique in the Expense Report example. In this section, you will learn two more techniques. In the first, you will use *Page Sets* to control the pagination. And, in the second, you will use JavaScript to control the pagination of a dynamic document. When you use *Page Sets*, you can set pagination properties on the master pages. As shown here, you can set properties on the *Page Set* tab and on the *Pagination* tab when you work with *Page Sets*.





Exercises

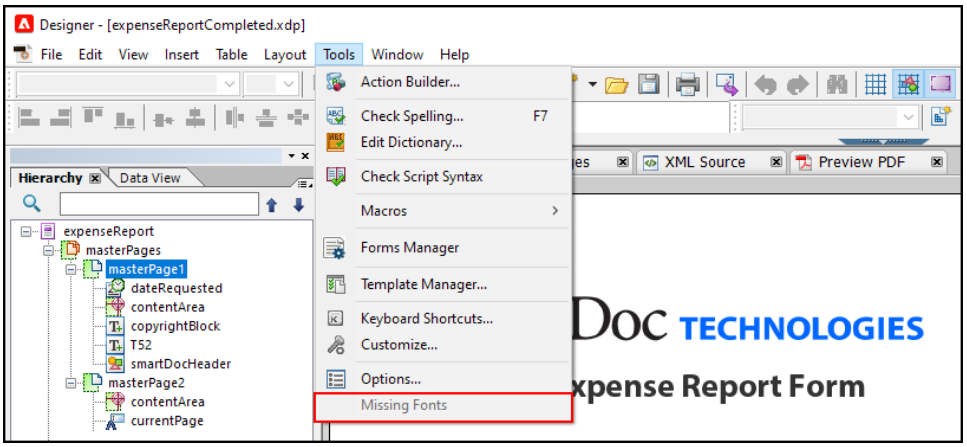
These exercises cover all of these important form topics.

Work with Fonts

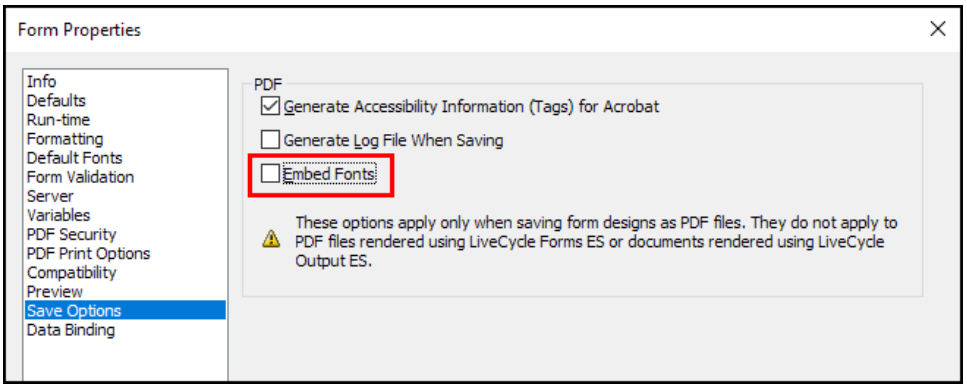
Font Embedding

Note: *If you must embed fonts, try to minimize the number of different fonts you use in your form. Embedding fonts is an all or nothing proposition; there's no option to select the specific fonts or characters that you'd want to embed.*

1. Select **File – Open** and navigate to your Student Files.
2. Select *expenseReportCompleted.xdp* and click **Open**.
3. Select the **Tools** menu and notice that *Missing Fonts* is grayed-out. This is an indication that your form doesn't have any missing fonts.



4. Select **File – Save As** and navigate to your working folder.
5. Enter **expenseReportFontsEmbedded.pdf** as the *File name*.
6. Click the *Save As type* dropdown, select *Adobe Dynamic XML Form (*.pdf)* and click **Save**.
7. Select **File – Form Properties – Save Options**.
8. Deselect the **Embed Fonts** option *(see illustration)*. Note: The ability to choose to embed or not embed fonts is available only to a PDF file.



9. Click **OK**.

10. Select **File – Save As** and navigate to your working folder.

11. Enter **expenseReportFontsNotEmbedded.pdf** as the *File name* and click **Save**.

Here are some important points about these two files.

– expenseReportFontsNotEmbedded.pdf is approximately 37 KB in size, so it will download and render much quicker for the user. In this case, this is all you need because the SmartDoc Expense Report uses MyriadPro-Bold and MyriadPro-Regular, which are both included with Adobe Acrobat.

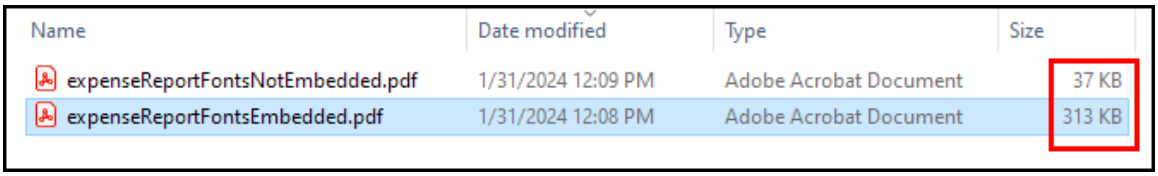
– expenseReportFontsEmbedded.pdf is almost 10X larger because it includes the fonts. When this form is opened, Acrobat or Reader will use the embedded fonts even though there's already a valid version of these fonts available on the system.

Let's view these forms in Acrobat or Reader.

12. Open Acrobat or Reader if they are not already open.

13. Select **File – Open** and navigate to your working folder.

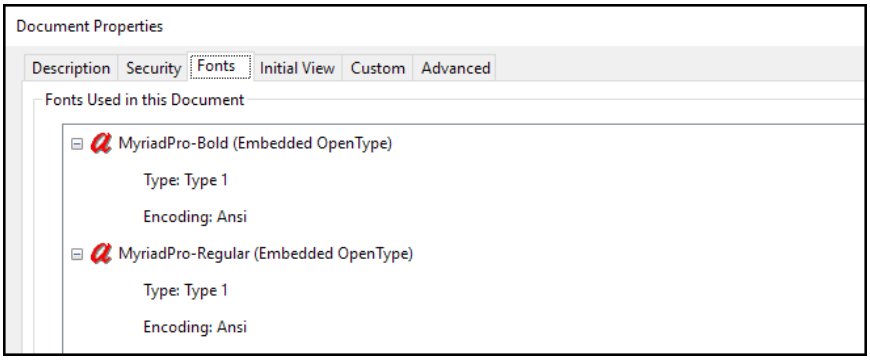
14. Notice the difference in the file size of the two PDFs *(see illustration)*.



15. Select your *expenseReportFontsEmbedded.pdf* and click **Open**.

16. Select **File – Properties** to view the *Document Properties* dialog box.

17. Select the **Fonts** tab to view the fonts used in this document. Notice that these are both *Embedded OpenType* fonts *(see illustration)*.

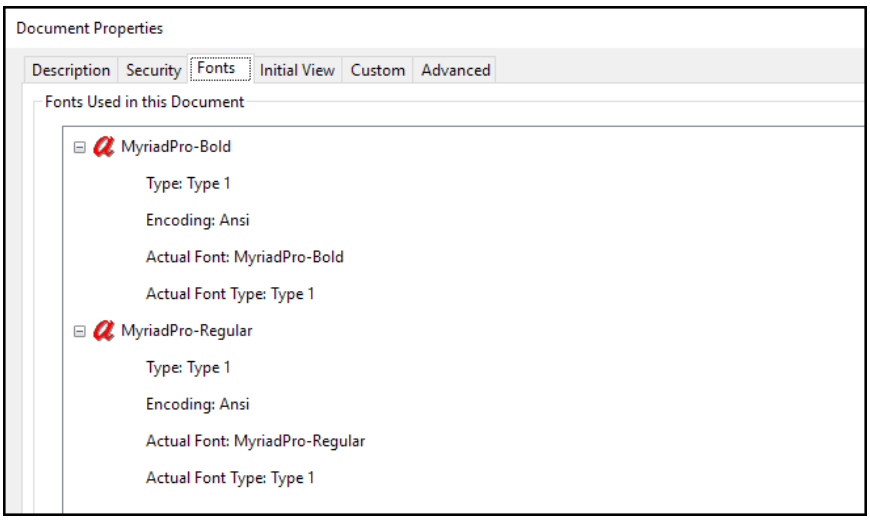


18. Click **Cancel** to close the *Document Properties* dialog.

19. Select **File – Open**, select your *expenseReportFontsNotEmbedded.pdf* and click **Open**.

20. Select **File – Properties** to view the *Document Properties* dialog box.

21. Select the **Fonts** tab to view the fonts used in this document. Notice that neither font is embedded but they both display perfectly fine in the PDF. That is because Adobe includes the Myriad font with Acrobat and Reader.

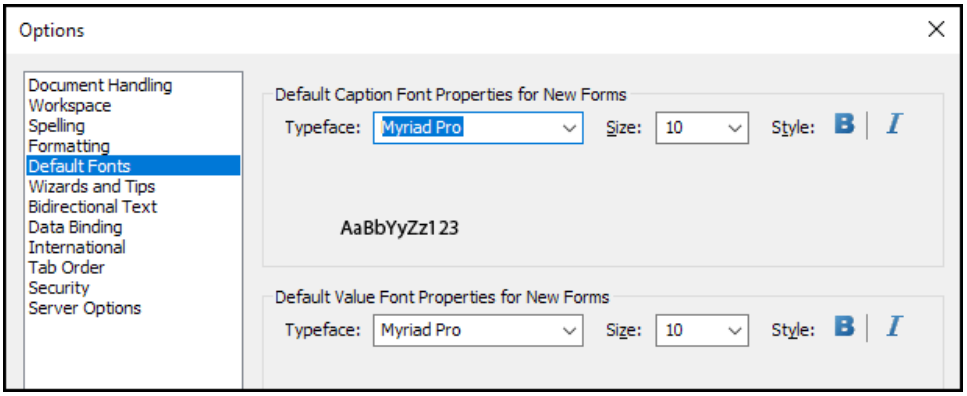


22. Click **Cancel** to close the *Document Properties* dialog.

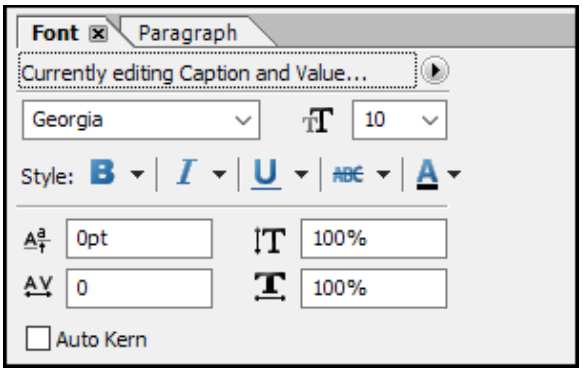
In this exercise, you will see the two different ways you can set default fonts in Designer. You can set default fonts in general for your Designer program, or you can set them specifically for your current form.

Set the Default Fonts in Designer

1. Go back to your Designer application.
2. Select **Tools – Options – Default Fonts**.
3. Notice there is a default typeface for captions and a default typeface for values *(see illustration)*.



4. Click the *Default Caption Typeface* dropdown and select Georgia.
5. Click the *Default Value Typeface* dropdown and select Georgia.
6. Click **OK** in the *Options* dialog box.
7. Select **File – New**, and create a blank form.
8. Drag and drop a **Text Field** object from the Object Library onto your form.
9. Open the Font palette. You'll notice the caption and value properties of the Text Field are set to *Georgia* because this is now Designer's default font *(see illustration)*.

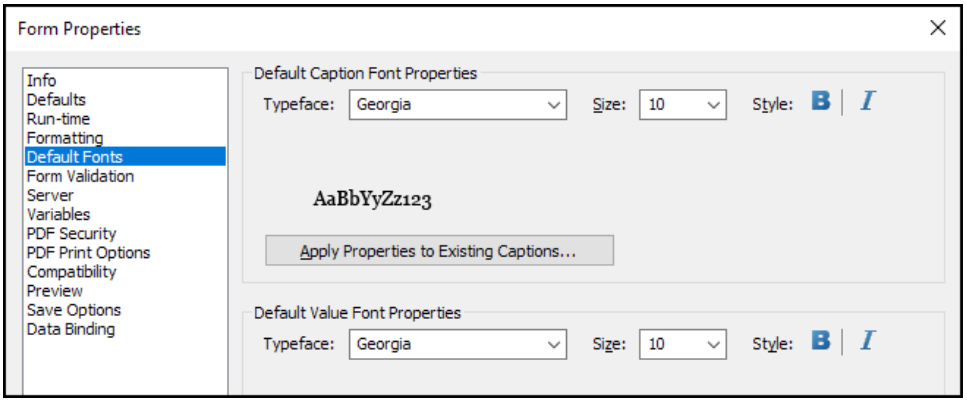


10. Select **File – Save As** and navigate to your working folder.
11. Enter **The Designer Default Font is Georgia.pdf** as the File name.
12. Click the *Save as type* dropdown and select **Adobe Dynamic XML Form (*.pdf)**.
13. Click Save.

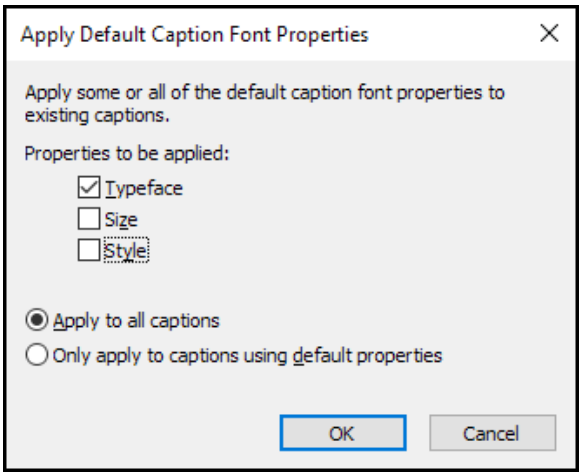
Set the Default Fonts in your form

You can also set your current form's default fonts in the Form Properties dialog box.

14. Select **File – Form Properties – Default Fonts**.
15. Notice this form also has a default typeface for captions and a default typeface for values *(see illustration)*. The form's default fonts were determined based on the Designer default fonts you set in the last exercise.



16. Click the *Default Caption Font Properties Typeface* dropdown and select **Arial**.
17. Click **Apply Properties To Existing Captions**, and the *Apply Default Caption Font Properties* dialog will appear.
18. Deselect *Size* and *Style (see illustration)*.



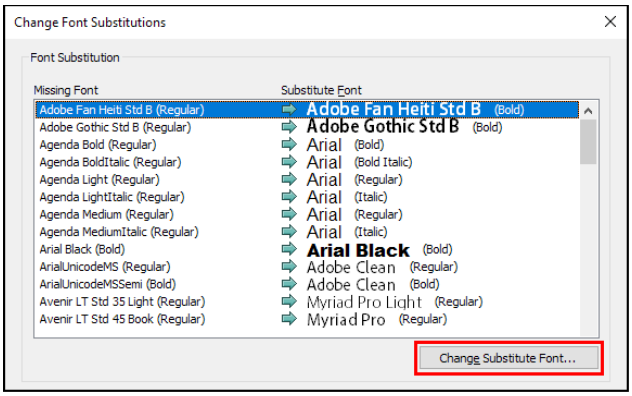
19. Click **OK**.
20. Click the *Default Value Font Properties Typeface* dropdown and select **Arial**.
21. Click **Apply Properties To Existing Field Values**, and the *Apply Default Value Font Properties* dialog will appear.
22. Deselect *Size* and *Style*.
23. Click **OK**.
24. Click **OK** to close the *Form Properties* dialog box.
25. Drag and drop a **Text Field** object from your Object Library to your form.
26. Notice that the default fonts for this new object are now Arial.

    Note: *These default form fonts will override Designer's default font settings, so if you send this form to a different user, they'll see these same default fonts when working with this form. Also, when you create a template with a specific default font, that template's default font will be used on all forms derived from the template, regardless of the defaults set in Tools – Options.*

Modify the font map

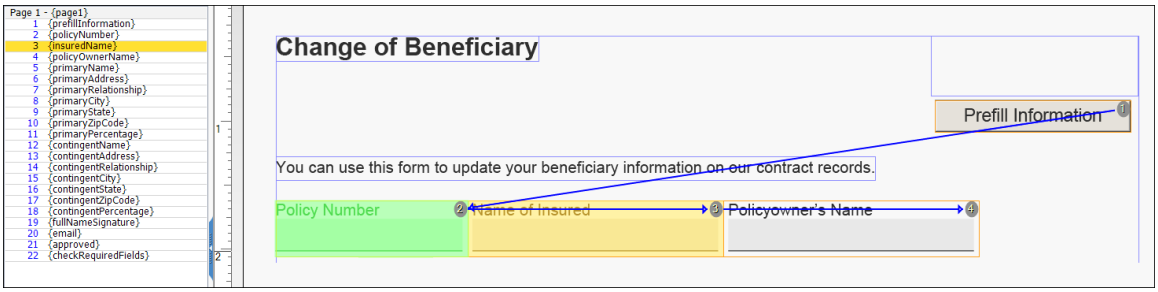
You can modify this font-mapping table interactively in Designer by following these steps.

1. Select **Tools – Options** to open the Options dialog box.
2. Select **Document Handling**, and click the **Modify Font Substitutions** button.
3. The *Change Font Substitutions* dialog box will appear *(see illustration, left)*.
4. Select the mapping that you want to change, and click **Change Substitute Font** *(see illustration, left)*.
5. The *Specify Substitute Font* dialog box will appear *(see illustration, right)*.
6. Select the substitute font and style that you want *(see illustration, right)*.
7. Click OK to close the *Specify Substitute Font* dialog box.
8. Click **OK** to close the *Change Font Substitutions* dialog box.



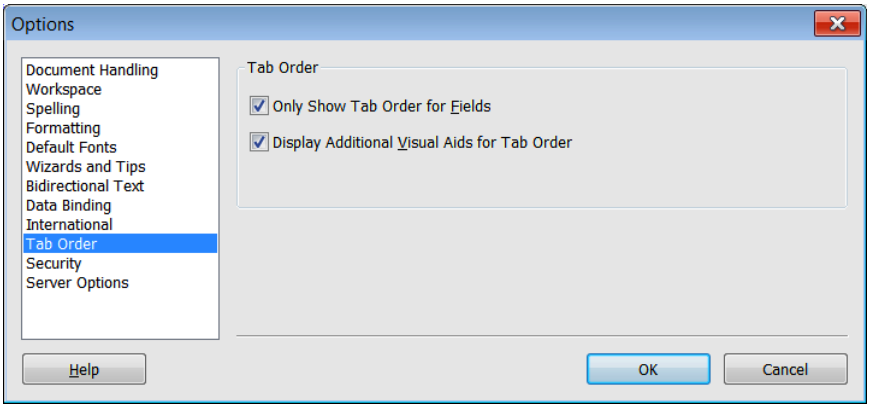
Note: *This table enables you to control the font mappings that Designer uses to map your system fonts to the fonts in your input PDFs.*

Review the Tabbing



Designer's Display Additional Visual Aids For Tab Order option enables you to view the flow of your tab order.

1. Select **File – Open** and navigate to your student files.
2. Select the *changeOfBeneficiaryTabbing.xdp* file and click **Open**.
3. Select **Tools – Options** and select the *Tab Order* option *(see illustration)*.



4. Select both of these options.

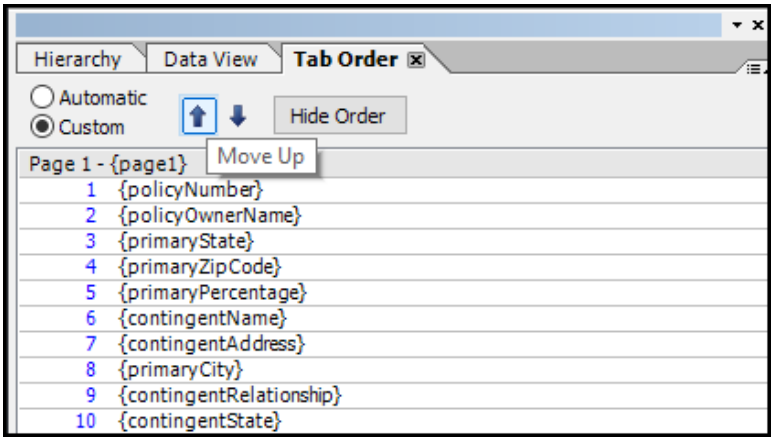
– **Only Show Tab Order For Fields:** By default, the tab order includes all the fields and objects on the form. However, if you're creating a custom tab order for your form's interactive fields, it's more cumbersome if all the

form's objects are in the tab order. You can select this option to hide static Text objects and Image objects in the tab order view.

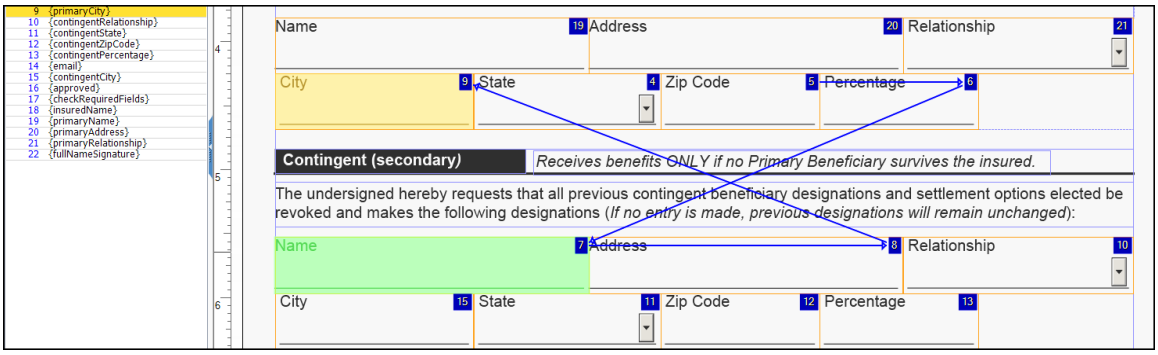
– **Display Additional Visual Aids For Tab Order:** Selecting this option will provide you with additional visual aids for your tab order. Designer will show blue arrows displayed on your form when you hover over a field. The arrows show the two previous fields and the two subsequent fields in your tab order relative to the field you're hovering over.

5. Click **OK** to close the *Tools – Options* dialog.

6. Select the **Tab Order** palette on the left *(see illustration).* Note If you do not see the Tab Order palette, select *Window – Tab Order*.



7. Notice how the tab order is less than ideal.



8. Switch from **Custom** to **Automatic**.

9. Notice how this change improved the tab order.

View Features in an Accessible Form

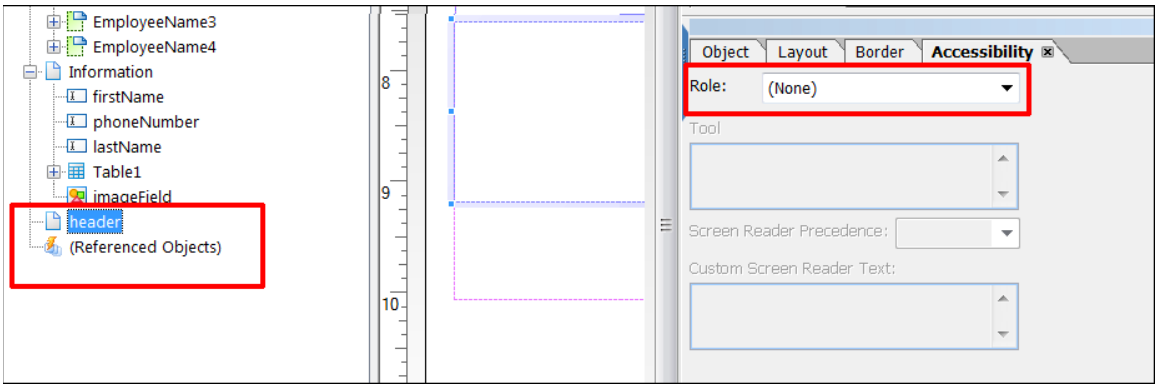
Note: *This exercise is taken from the Accessibility – PDF course on gemforms.training.*

1. Select **File - Open** in Designer and navigate to your Student files.

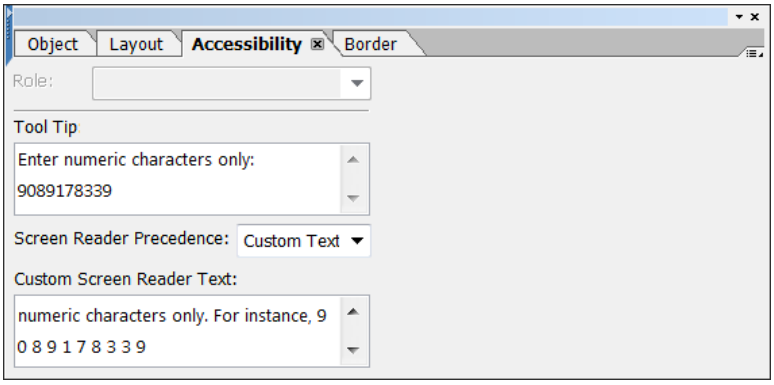
2. Select *accessibleForm.pdf* and click **Open**.

3. Select **File – Save As…** and navigate to your working folder.

4. Enter <yourname>AccessibleForm.pdf for the *File name*.
5. Click the *Save as type* dropdown and select **Adobe Static PDF Form (*.pdf)**.
6. Click **Save**.
7. Select the *Hierarchy* tab on the left.
8. Select the *header* subform at the very bottom of the Hierarchy *(see illustration)*.
9. Click the *Accessibility* palette to see its properties. If you don't see *the Accessibility* palette, select *Window – Accessibility*. The *Role* option of this subform is set to *(None)*, which means that it won't be read by the screen reader *(see illustration)*.



10. Select the **Information** subform, and click the *Accessibility* palette to see its properties. The *Role* option of this subform is also set to *None*, so this also will not be read by the screen reader. However, the screen reader will check the nested child subforms that make up the list to determine how many items are in the list, how the list is structured, and where the list ends. This information *(the list of child nodes)* will then be read by the screen reader.
11. Select the **phoneNumber field**, and click the *Accessibility* palette to see its properties. Notice that the *Tool Tip* and the *Custom Screen Reader Text* have different messages. The *Tool Tip* text will be shown when interactive users place their mouse over the phone number field at runtime. Notice that the integers in the phone number are spaced out so the screen reader will read them one at a time as opposed to announcing a single 10 digit number *(see illustration)*.



The *Tool Tip* text will also be read by screen readers if your Screen Reader Precedence option is set to *Tool Tip*. In our example, the precedence is set to *Custom Text* because we need to provide a more extended message for the users of our audible form. The *Screen Reader Precedence* dropdown list determines which of the following text will be read to the user when this field becomes active:

- **Custom Text:** The message that you enter in the Custom Screen Reader Text field.
- **Tool Tip:** The message that you enter in the Tool Tip field.
- **Caption:** The caption that appears in the Field tab of the Object palette.
- **Name:** The name of the field displayed in the Hierarchy palette.
- **None:** No message is read.

Note the importance of adding meaningful captions and names for our form fields. When there is not custom text or tool tips, screen readers may read our captions and field names.

Set Custom Reader Text

You should also set custom screen reader text for the images on your form. Follow these steps to add this to your form and save it with accessibility tags for Acrobat:

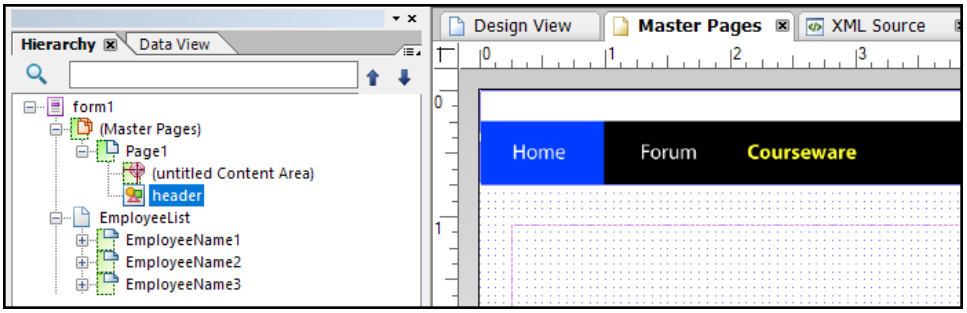
12. Select the **imageField** object.
13. Select the **Accessibility** palette.
14. Enter **This is a bar chart of quarterly results for Year 2023** as your *Custom Screen Reader Text*.
15. Make sure your *Screen Reader Precedence* property for the image field object is set to **Custom Text**.
16. This will add the following XFA tags to your imageField object.

<assist>

<speak>This is a bar chart of quarterly results for Year 2023.</speak>

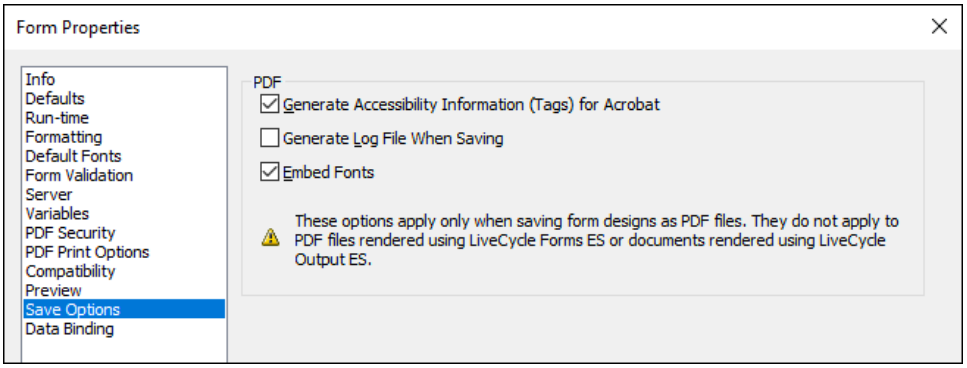
</assist>

17. Select the **Master Pages** tab and select the **header** imageField object *(see illustration)*.



18. Select the **Accessibility** palette and enter **This is a decorative action bar from the aemforms.training website** as your *Custom Screen Reader Text*.
19. Select **Design View**.
20. Select **File – Form Properties**.
21. Select **Info** and enter **<yourname> Accessible Form** for the *Title*.
22. Enter **<yourname>** for the *Author*.
23. Select **Defaults** in the list on the left.

24. Make sure your default *Form Locale* is set.
25. Select **Save Options** in the list on the left.
26. Make sure the **Generate Accessibility Information (Tags) for Acrobat** check box is selected. You must have this property selected for screen readers to navigate your form.

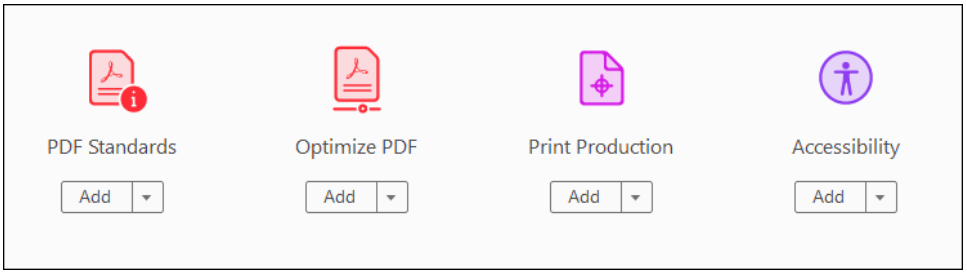


27. Click **OK** to close the *Form Properties* dialog box.
28. Select **File – Save**. You can now test this form in Acrobat.

Test with the Screen Reader in Acrobat

Follow these steps to use the screen reader in Acrobat.

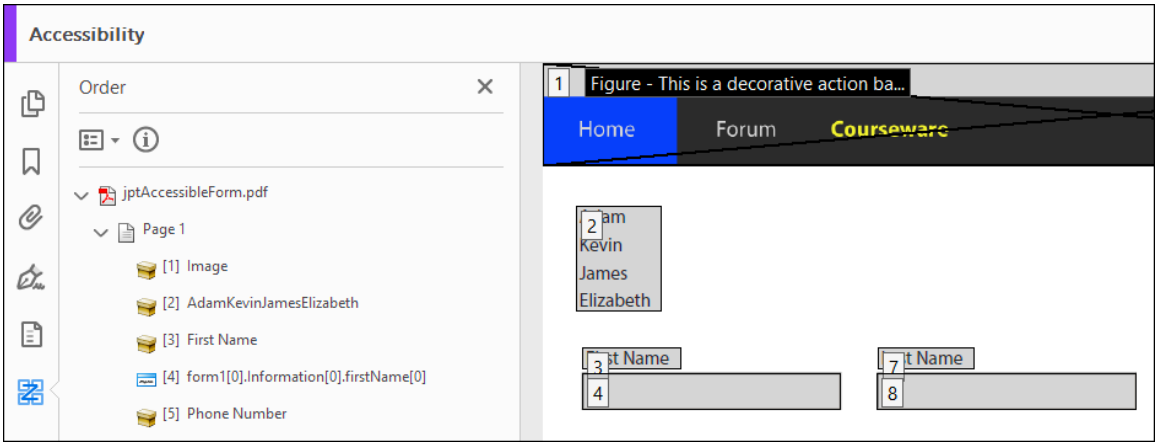
29. Open Acrobat if it is not already open.
30. Select **File – Open** and navigate to your working folder.
31. Select the **<yourname>AccessibleForm.pdf** file and click **Open**.
32. Select **View – Read Out Loud – Activate Read Out Loud**.
33. Select **View – Read Out Loud – Read This Page Only**. You will hear the screen reader audibly translate your form.
34. After you experience the audible screen reader, select **Tools**.
35. Select **Accessibility** *(see illustration)*.



36. Select **Reading Order** in the panel on the right.
37. Click **Show Order Panel** in the bottom right of the *Reader Order* dialog.

38. You will see the *Reading Order* panel on the left *(see illustration)*. This shows you the order that a screen reader will audibly read the document.

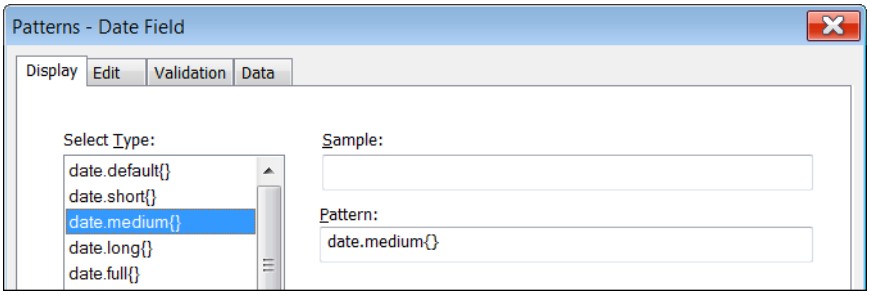
39. Click the X in the upper right of the panel to close it.



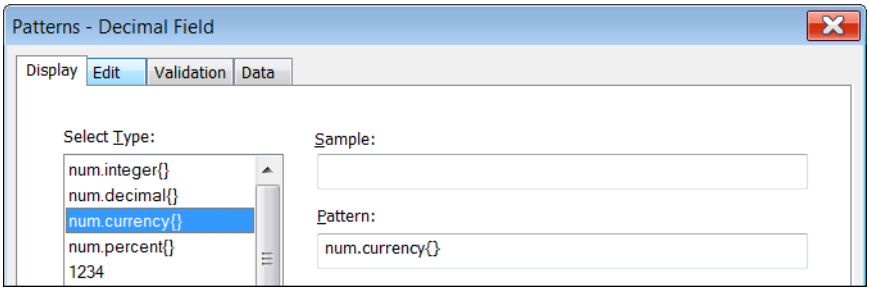
Localization with a Specific Locale

You can see how different locale settings affect your form by changing the locale property in the SmartDoc Expense Report:

1. Select **File – Open** and navigate to your Student Files.
2. Select the **expenseReportCompleted.xdp** file and click **Open**.
3. Select **File – Save As…** and navigate to your working folder.
4. Enter **<yourname>Locale.xdp** for the File name.
5. Click the *Save As type* dropdown, select *Adobe XML Form (*.xdp)* and click **Save**.
6. Select the **date field** in the expensesRow subform.
7. Open the Field tab of the Object palette.
8. Click the Locale dropdown list and select **Spanish (Spain)**. Locales are listed by language first and then by country or region.
9. Click the **Patterns** button, and remove the prefilled *Display*, *Edit*, and *Data* patterns.
10. Select the *Display* tab in the Patterns dialog.
11. Double-click the **date.medium{}** pattern so it gets added as the *Display* pattern *(see illustration)*.

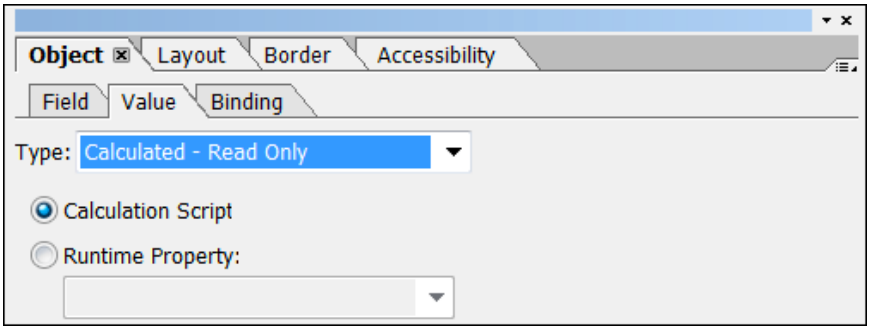


12. Click **Apply**, and then click **OK**.
13. Select the **cost** field in the expensesRow subform.
14. Open the Field tab of the Object palette.
15. Click the Locale dropdown list and select **Spanish (Spain)**.
16. Click the **Patterns** button, and remove the prefilled *Display* and *Edit* patterns.
17. Select the *Display* tab in the Patterns dialog.
18. Double-click the **num.currency{}** pattern so it gets added as the *Display* pattern *(see illustration)*. The new currency pattern will put the Euro symbol after the amount.



19. Select the *Edit* tab in the Patterns dialog.
20. Double-click the **num.currency{}** pattern so it gets added as the *Edit* pattern.
21. Click **Apply**, and then click **OK**.
22. Select the **total** field in the expensesRow subform.
23. Open the Field tab of the Object palette.
24. Click the Locale dropdown list and select **Spanish (Spain)**.
25. Click the **Patterns** button, and remove the prefilled *Display* pattern.
26. Double-click the **num.currency{}** pattern so it gets added as the *Display* pattern.

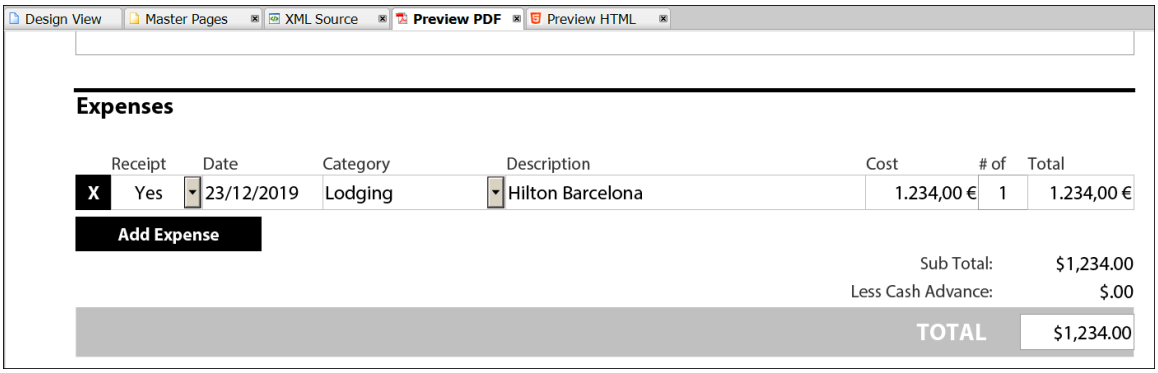
    Note: *There is no Edit pattern to update because this field is set to Calculated – Read Only.*



27. Click **Apply**, and then click **OK**.
28. Select **Preview PDF**.

29. Select **December 23, 2020** from the calendar dropdown list of the Date field of the Expenses row. Note how the date is now formatted as 23/12/2020, which is the proper formatting for Spain.

30. Enter **1234** in the Cost field of the Expenses row. Note how the value is now formatted as 1.234,00€. The calculated Total field at the end of the row is also displayed as 1.234,00€. However, the Sub Total and Total fields at the bottom of the form are all still formatted as $1,234.00 because the form's default locale is English (USA).



The previous steps showed how to set the locale on individual fields. This is a valid approach, but it's more common for the entire form to use the same locale.

31. Select **Design View** to close the PDF preview.

32. Select **File – Form Properties**, and click **Defaults**.

33. Select **English (United Kingdom)** in the *Form Locale* dropdown list.

34. Click **OK** to close the Form Properties dialog box.

35. Select the **date** field in the expensesRow subform.

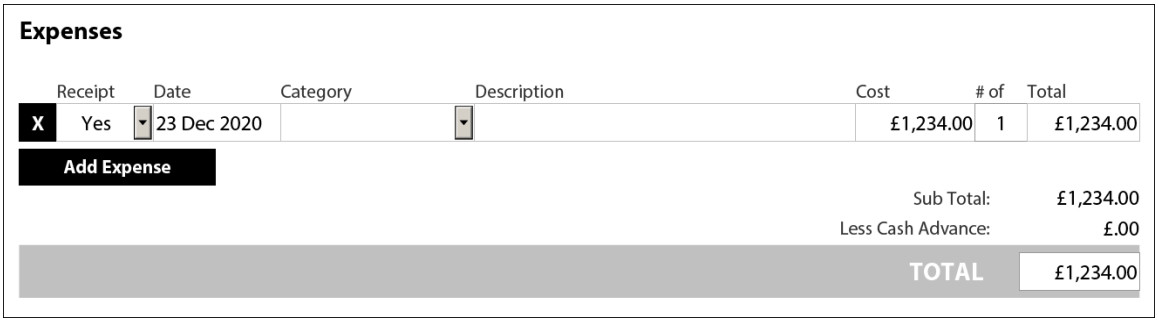
36. In the Field tab of the Object palette, change your Locale from Spanish (Spain) to Default locale.

37. Repeat the previous step for the cost field and total field in the expensesRow subform. You previously gave each one of these fields a unique locale that will override your form's locale. You are now setting them to use the form's default locale.

38. Select **Preview PDF** and enter some data into your form.

39. Select **December 23, 2020** from the calendar dropdown list of the Date field of the Expenses row.

40. Enter **1234** in the Cost field of the Expenses row. Your form is now consistently using the English *(United Kingdom)* locale *(see illustration)*.

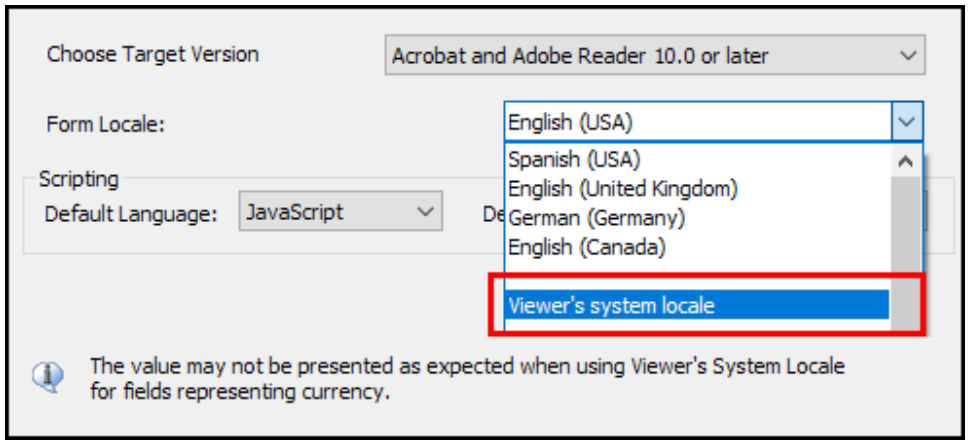


Note: *There's another option to set the locale to your viewer's system locale. This isn't a recommended practice because it doesn't deliver reliable and predictable results.*

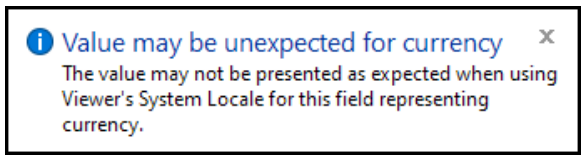
Localization with the Viewer's System Locale

> ***Note:*** *This exercise includes changing your Windows System Locale to simulate a computer in a different country and it also requires a system reboot. If you are unable to do this during the training time, simply read through the steps to see the benefits of changing your form to the Viewer's system locale.*

41. Select **File – Open** and navigate to your Student Files.
42. Select the **expenseReportCompleted.xdp** file and click **Open**.
43. Select **File – Save As…** and navigate to your working folder.
44. Enter **viewersSystemLocale.pdf** for the *File name*.
45. Click the *Save As type* dropdown, select *Adobe Dynamic XML Form (*.pdf)* and click **Save**.
46. Select **File – Form Properties**, and click **Defaults**.
47. Click the *Form Locale* dropdown and select **Viewer's system locale** *(see illustration)*.
48. Notice the warning message about currency fields.



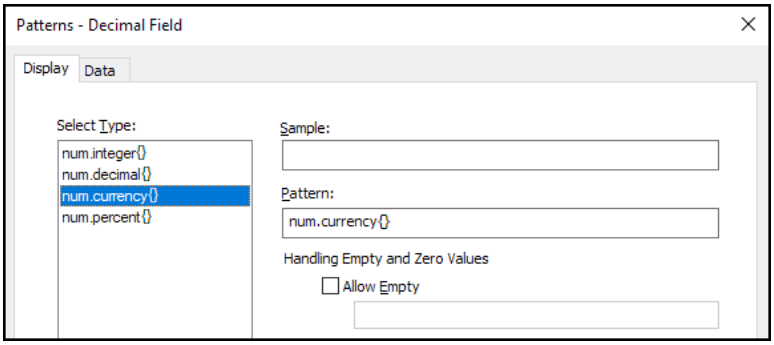
49. Click **OK** to close the *Form Properties* dialog box.
50. You will see more warnings about currency fields.



51. Select **File – Save**.
52. Select the Date field and click Patterns in the Field tab of the Object palette.
53. Select the **date.default{}** pattern and click **Apply**.

54. Select the Cost field and click Patterns in the Field tab of the Object palette.

55. Select the **num.currency{}** pattern *(see illustration)* and click **Apply**.



56. Update the other 4 Decimal Fields to the same **num.currency{}** display pattern.

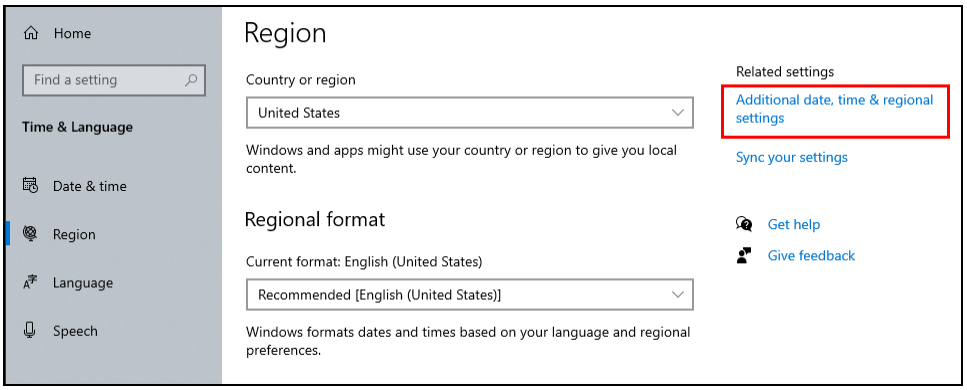
Change your Computer's Locale

57. Click *Windows Start* and select *Settings*.

58. Select *Time & Language*.

59. Select *Region (see illustration)*.

60. Select *Additional date, time, & regional settings* in the *Related settings* panel.



61. Select *Change date, time, or number formats*.

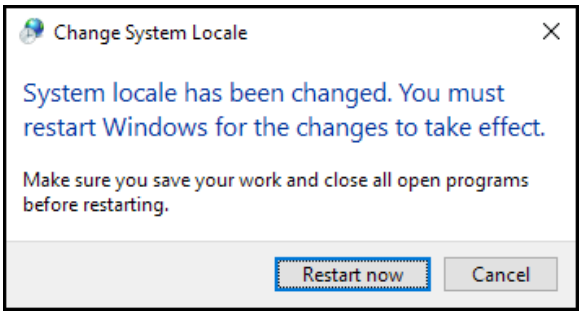
62. In the *Region* dialog box, select the *Administrative* tab.

63. Select *Change system locale*.

64. Click the *Current system locale* dropdown and select a new locale. I am switching from *English (United States)* to *English (United Kingdom)*.

65. Click **OK**.

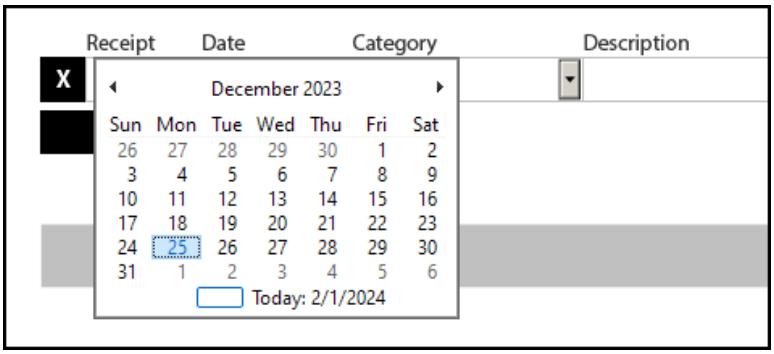
66. We must restart our computers to apply the changes.



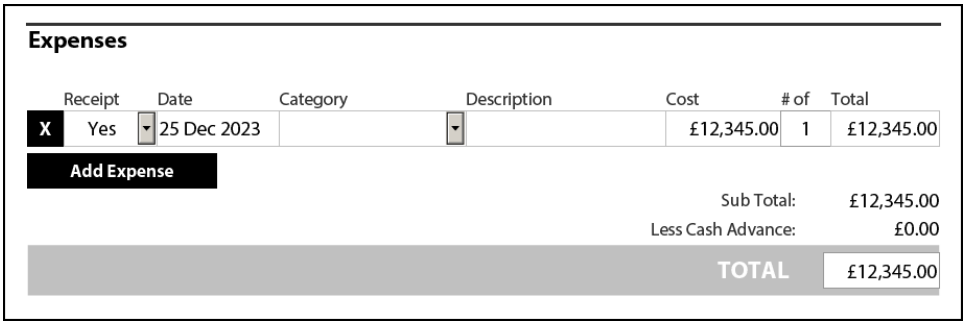
Note: *We are simulating a computer in the UK. Users in the UK will have this setting by default and will not have to go through these steps. When you are done with the simulation, make sure to reset your computer back to your standard locale. My standard is English (United States).*

View the PDF with a new System Locale.

67. Once the computer is restarted with the new system locale, open up Adobe Acrobat.
68. Select **File – Open** and navigate to your working folder.
69. Select *viewersSystemLocale.pdf* and click **Open**.
70. Click the *Date* calendar control *(see illustration)* and select this past Christmas.



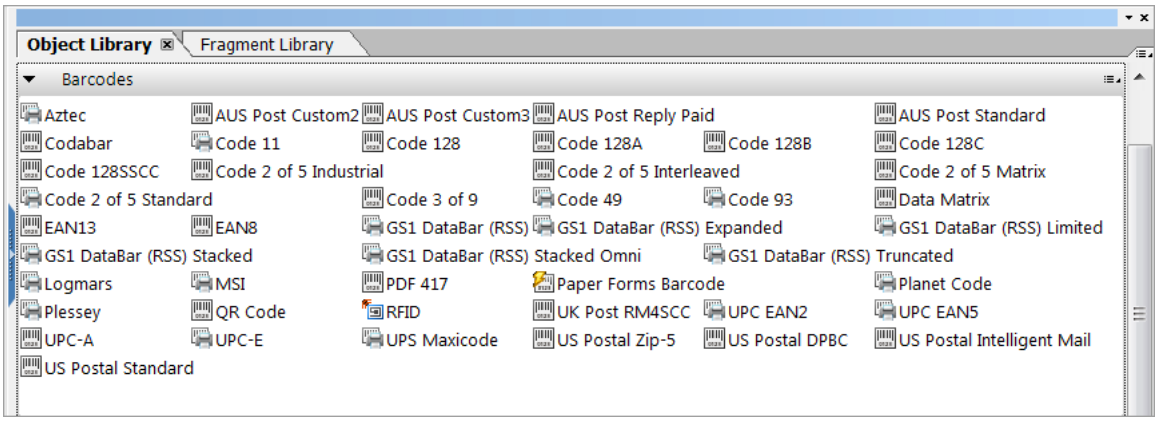
71. Enter **12345** in the *Cost* field.
72. You should see patterns for the date and currency reflect your new system locale *(see illustration)*. My system locale is set to *English (United Kingdom)*.



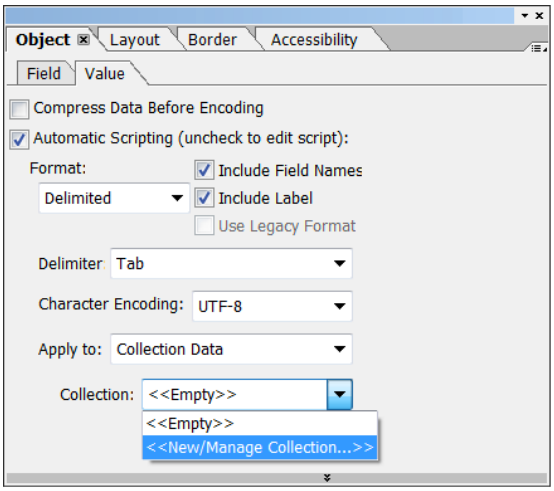
Work with Barcodes

1. Create a new form in Designer by selecting **File – New** to launch the New Form Assistant.

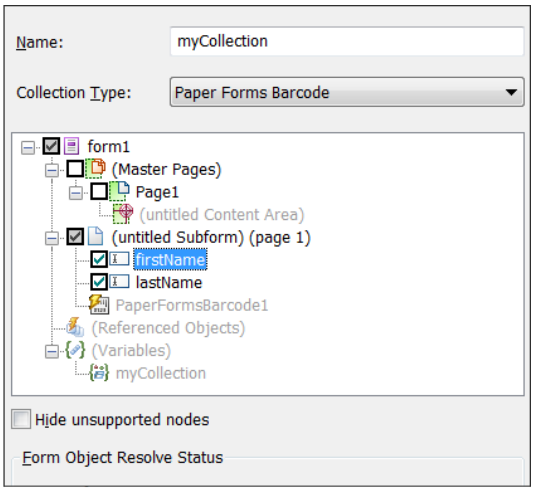
2. Select **Use a blank form** and click **Next**.
3. Click **Finish**.
4. Select **File – Save As…** and navigate to your working folder.
5. Enter **<yourname>Barcode.pdf** for the File name.
6. Click the *Save as type* dropdown and select **Adobe Dynamic XML Form (*.pdf)**.
7. Click **Save**.
8. Drag and drop two **Text Field** objects from the Standard Object Library to the form.
9. Name the first Text Field **firstName** and enter **First Name** for the caption.
10. Name the first Text Field **lastName** and enter **Last Name** for the caption.
11. Expand the Barcodes tab in your Object library so you can see all the barcodes *(see illustration)*.



12. Drag and drop a **Paper Forms Barcode** object from the Barcodes Object Library to the form.
13. Click **OK** if you receive a *Tips and Hints* message box.
14. With the barcode selected, highlight the **Value** tab of the **Object** palette.
15. Click the in the *Collection* dropdown at the bottom of the palette and select **<<New/Manage Collection>>** (A).

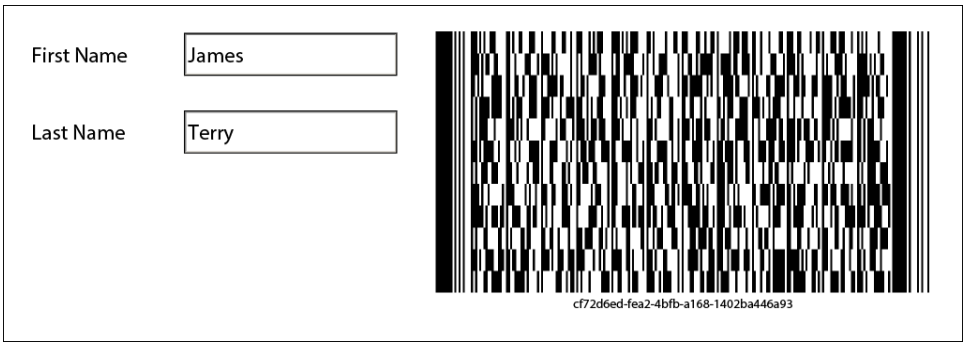


A



B

16. In the *Collection List* dialog box, click **New**, and enter a new collection named **myCollection**.
17. Select **myCollection** in the list and click **Modify**. The Collection Editor dialog box appears.
18. Select the **firstName** and **lastName** nodes *(B in above illustration)*.
19. Click **OK** and **Close** the Collection List dialog box.
20. Select **Preview PDF**.
21. Enter your first name and last name. You will see that the *Paper Forms Barcode* object is automatically updated when you exit the text field after updating its value *(see illustration)*.



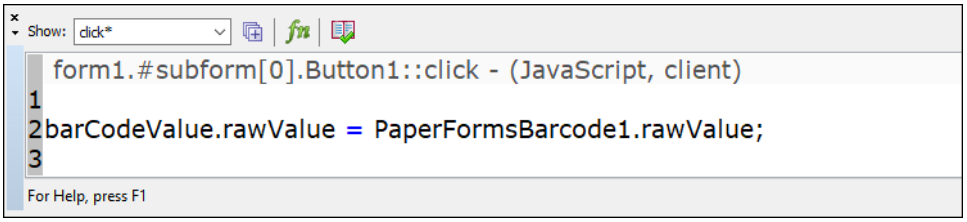
Note: *If you do not see the barcode in Preview PDF, try opening the PDF directly in Acrobat. You may be using Reader as your default PDF viewer. If this is the case, the bar code will appear as a solid gray rectangle.*

22. Select **Design View** to close the PDF preview.
23. Drag and drop another **Text Field** object from the Standard Object Library to the form.
24. Enter **barCodeValue** for the name.
25. Select the field and click **Allow Multiple Lines** on the Field tab of the Object palette.
26. Enter **Barcode Value** for the Caption.
27. Select the Layout palette and set the Height to **1.75in** and the Width to **3in**.

28. Drag and drop a **Button** object from the Standard Object Library to the form.
29. Set the button's caption to **View the contents of the barcode**.
30. Select the button and open the Script Editor.
31. Select the **click** event.
32. Add this script.

barCodeValue.rawValue = PaperFormsBarcode1.rawValue;

33. Your *Script Editor* should now look like this.



34. Select **Preview PDF**.
35. Enter your first name and last name. You should see the paper forms barcode updating.
36. **Click** the button. You should see the contents of the barcode.

Label firstName lastName
cf72d6ed-fea2-4bfb-a168-1402ba446a93 James Terry

Note: *If you do not see the barcode in Preview PDF, it is likely that you have Reader and not Acrobat.*

37. Select **Design View**.
38. Select **File – Save**.

View a Reader Extended PDF

If you do not have Acrobat, you can view a functioning barcode in a Reader Extended file by following these steps.

39. Open **Reader**.
40. Select **File – Open** and navigate to your Student Files.
41. Select *BarCodeDemo-RE.pdf* and click **Open**.
42. Enter your first name and last name. You should see the paper forms barcode updating *(see illustration)*.

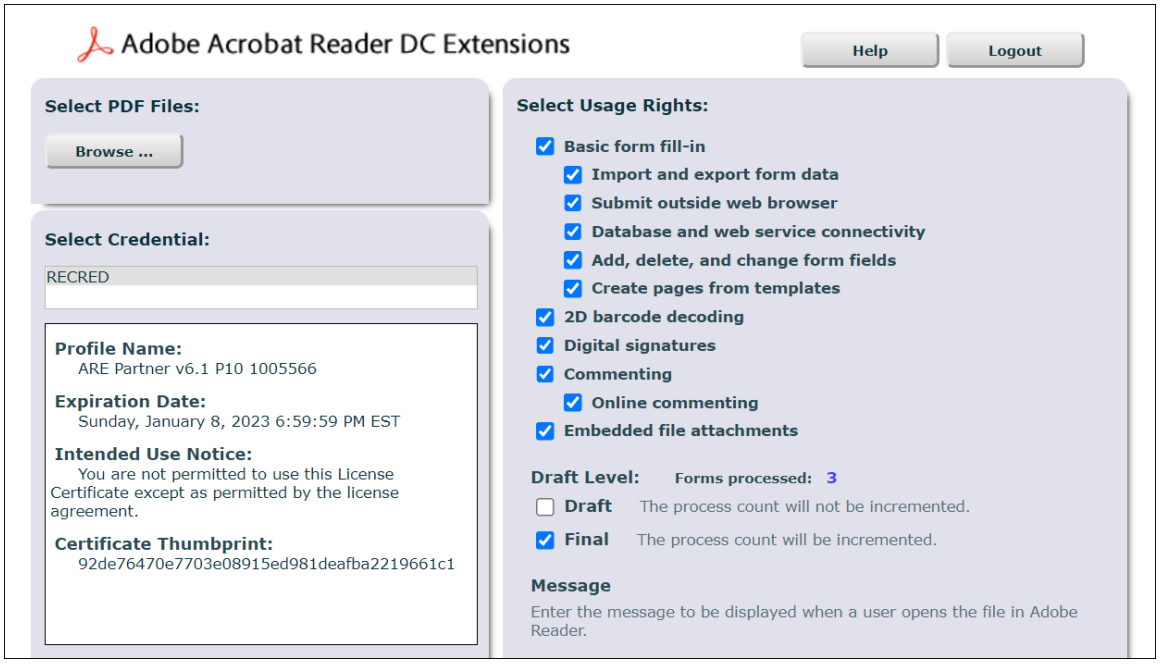
43. Enter **Hello World** in the *Text Field* and hit Enter on your keyboard. Your barcode should update.

44. **Click** the button. You should see the contents of the barcode.

Label firstName lastName
cf72d6ed-fea2-4bfb-a168-1402ba446a93 James Terry

Note: *The expiration date for this Reader Extensions certificate is Sunday, January 8, 2023 6:59:59 PM EST. If you review this file after this date, it will not work.*

Note: *This PDF was extended with Reader DC Extensions, not with Acrobat Pro (see illustration). Extending a file in Acrobat Pro will not enable this dynamic barcode functionality.*

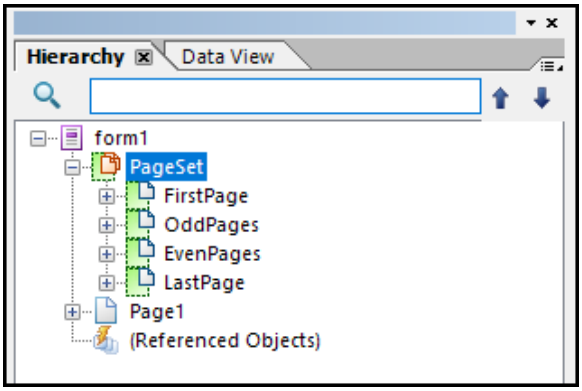


Control Pagination

1. Select **File – Open** and navigate to your Student Files.

2. Select *pageSets.xdp* and click **Open**.

3. Select the **PageSet** node in your Hierarchy palette *(see illustration)*.



4. Open the *Page Set* tab of the *Object* palette and notice that the *Printing* property is set to **Print on Both Sides – Long Edge** *(see illustration A)*.

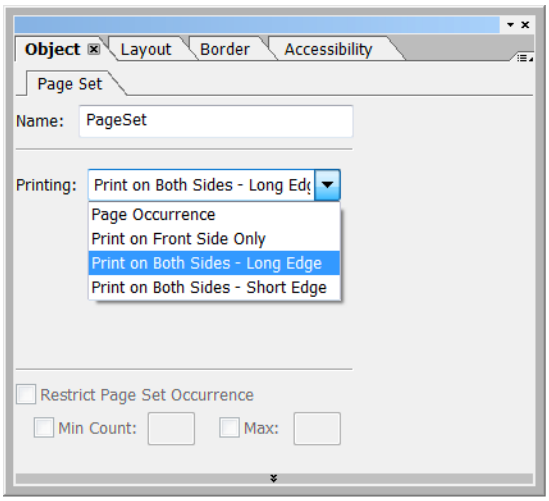
5. Select the **FirstPage** master page node in your Hierarchy palette.

6. Open the *Pagination* tab of the *Object* palette notice the *Master Page Applies To* section on the top of the *Pagination* tab *(see illustration B)*.

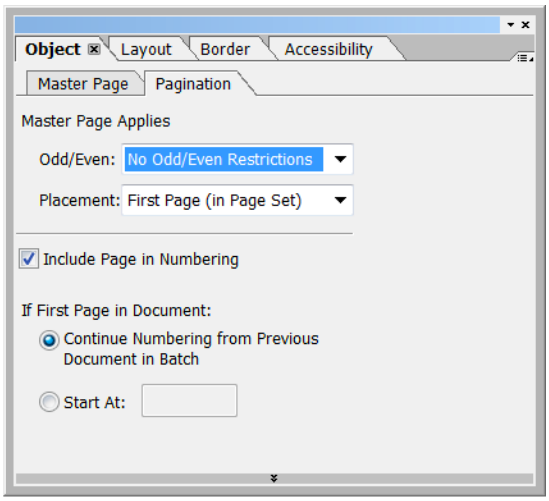
7. Notice there are **No Odd/Even Restrictions** *(see illustration B)*.

8. Notice the *Placement* is set to **First Page (in Page Set)** *(see illustration B)*.

9. Notice that pages that use this master page will be included in the page numbering *(see illustration B)*.



A



B

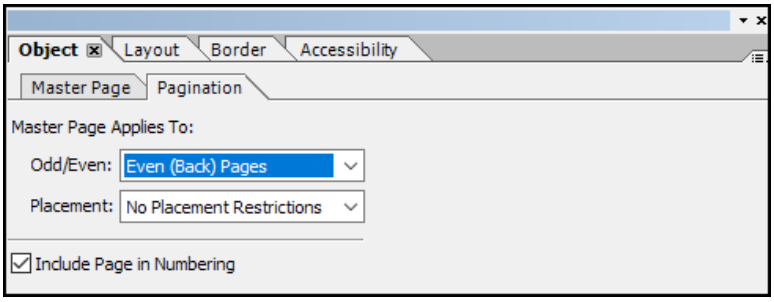
10. Select the **OddPages** master page node in your Hierarchy palette.

11. Open the *Pagination* tab of the *Object* palette.

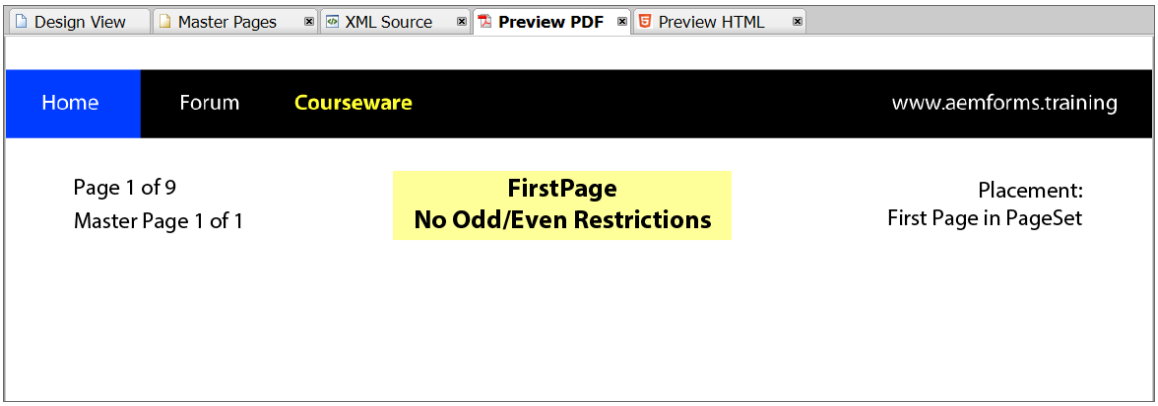
12. Notice that this master page is applied to **Odd (Front) Pages**. Our PageSet will not use this for the first page because we have an explicit FirstPage master page.

13. Notice there are no other placement restrictions.

14. Select the **EvenPages** master page node in your Hierarchy palette.
15. Open the *Pagination* tab of the *Object* palette.
16. Notice that this master page is applied to **Even (Back) Pages**. Our PageSet will not use this for the last page because we have an explicit LastPage master page.

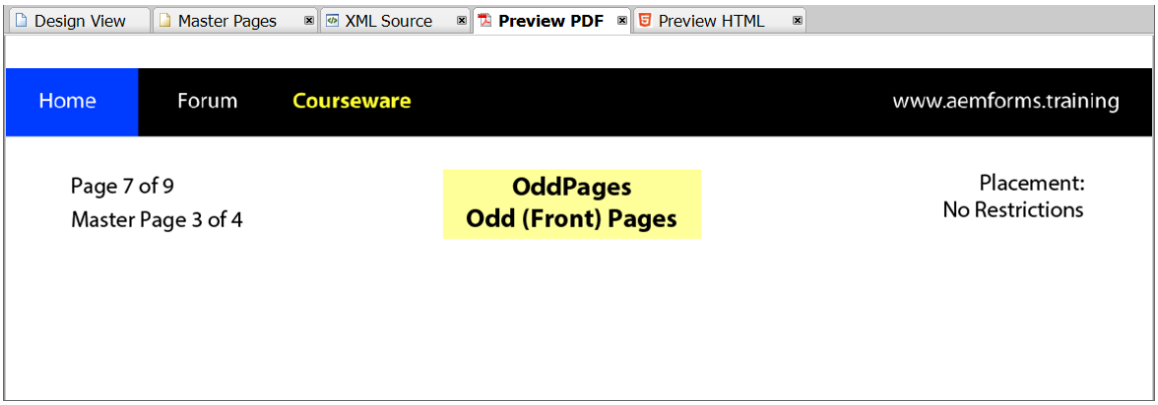


17. Select the **LastPage** master page node in your Hierarchy palette.
18. Open the *Pagination* tab of the *Object* palette.
19. Notice that there are **No Odd/Even Restrictions**.
20. Notice the *Placement* is set to **Last Page (in Page Set)**.
21. Select **Preview PDF** to see your PageSet and master pages in action.
22. You will see 9 pages and the top of each page shows the master page that was chosen from your PageSet along with some other pertinent information.
23. This is the first page *(Page 1 of 9)*, and it uses the FirstPage master page *(Master Page 1 of 1)*.



24. Scroll to the second page *(Page 2 of 9)*. Notice it says, *"Master Page 1 of 4"*. This is the first instance of the *EvenPages* master page in the
25. Scroll through the pages to see how the other pages are handled.

This is the seventh page *(Page 7 of 9)*, and it uses the OddPages master page *(Master Page 3 of 4)*. This is the third time the OddPages master page is being used because the first page *(page 1)* did not use it.



26. Select **Design View** to close the *PDF Preview*.

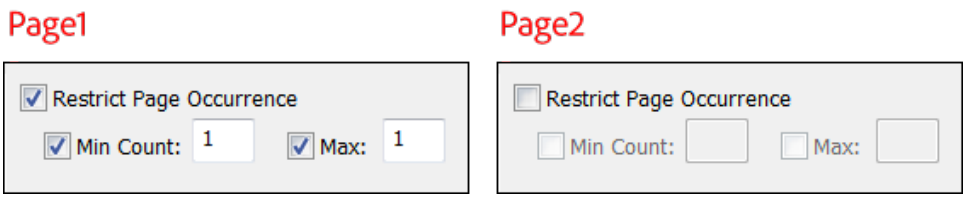
Always Produce an Even Number of Pages

Sometimes it is necessary to always produce an even amount of pages regardless of the content in a dynamic form or document. Follow these steps to review an XDP template that will always produce an even number of pages by inserting a blank page when it is required.

27. Select **File – Open** and navigate to your Student Files.

28. Select *Always produces an even amount of pages.xdp* file and click **Open**.

29. Select the **Master Pages** tab and review the two master pages. The Restrict Page Occurrence property has the same values as the *SmartDoc Expense Report*.

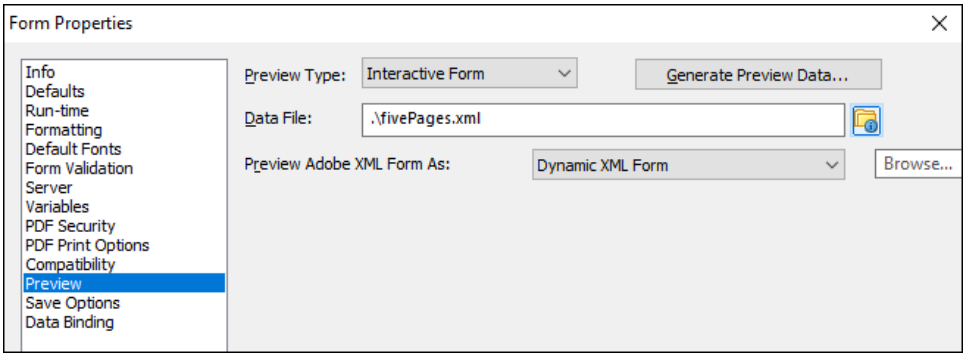


30. Select **File – Form Properties**.

31. Select **Preview**.

32. Click **Browse** *(see illustration)* and navigate to your Student Files.

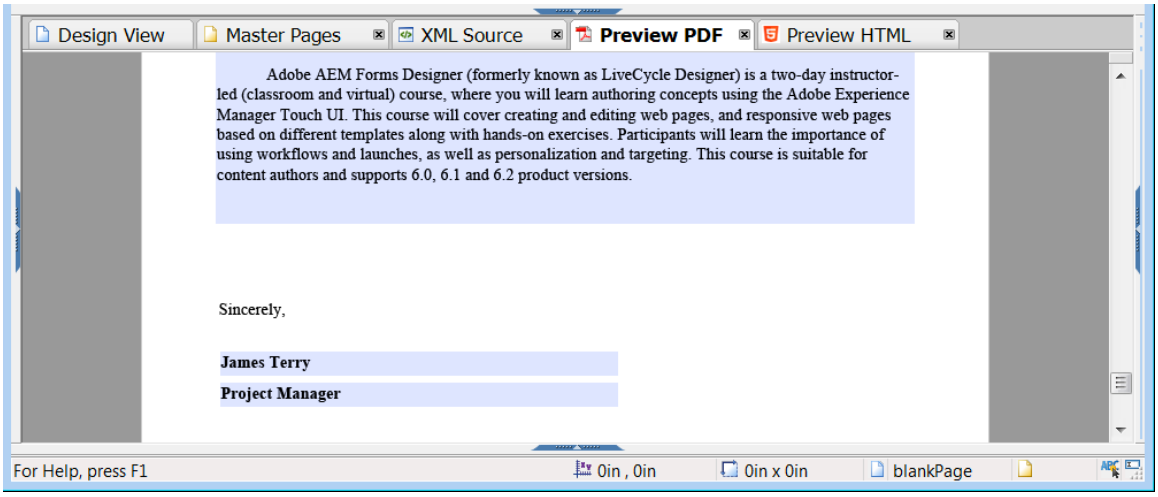
33. Select *sixPages.xml* and click **Open**.



34. Click **Open** and click **OK**.

35. Click **Preview PDF**.

36. Scroll to page **6 of 6** in your document. Notice that you do not see the additional page at the end because your document has an even number of pages.



37. Select **Design View** to change your data file.

38. Select **File – Form Properties**.

39. Click **Preview**.

40. Click **Browse** and navigate to your Student Files.

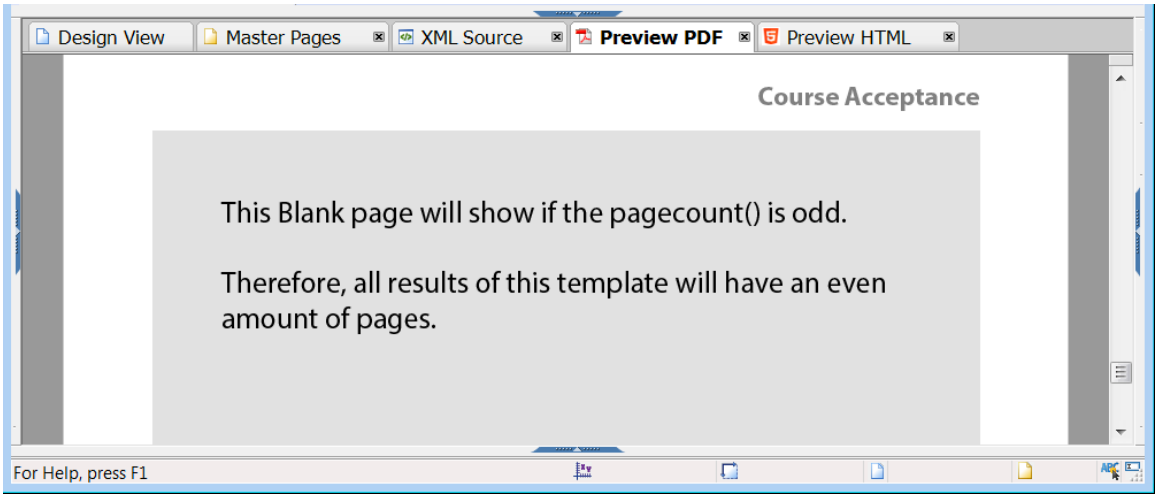
41. Select *fivePages.xml* and click **Open**.

42. Click **OK** and click **OK**.

43. Click **Preview PDF**.

44. Scroll to page **5 of 6** in your document. Notice the signature block indicating this is the end of the letter. In this case, the dynamic data only produced 5 pages.

45. Scroll to page 6. You will now see the Blank page that will show whenever your pagecount() is odd.

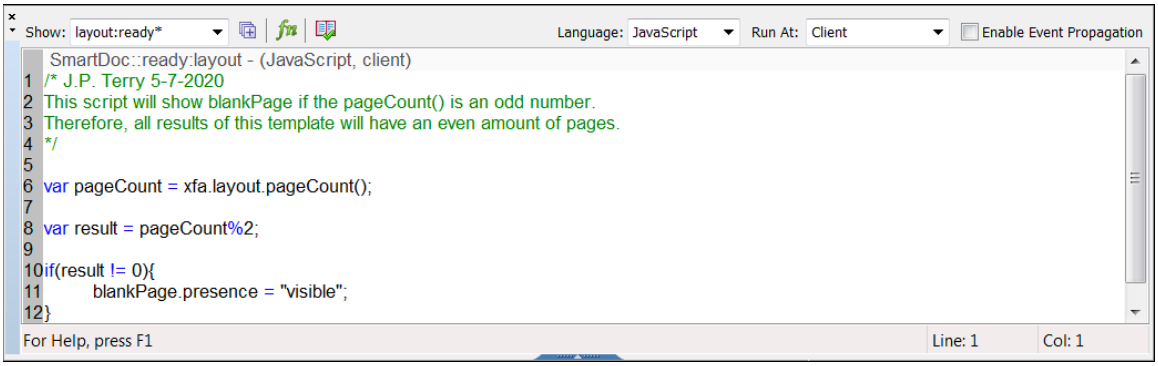


How it works

46. Select **Design View** to close the PDF preview.

47. Select the top-most form node called **SmartDoc**.

48. Expand the **Script Editor** and select the **layout:ready** event. You will see this script.



Here are the important details about the script. Line 6 creates a variable called pageCount and sets the value to the pageCount of the document. Line 8 creates a variable called result and sets the value to the modulus (remainder) of pageCount when pageCount is divided by 2. If it is an even number, the modulus will be 0. If it is an odd number, the modulus will be 1. Lines 10 – 12 will display the blankPage if result is not equal to 0. Therefore, odd numbered documents will display the additional blank page at the end.

blankPage must be initialized to hidden

In order for this to work, blankPage must be initialized to hidden. You can do this at Design time by setting the Presence property in the Subform tab of the Object palette. Or you can follow these steps to add a script.

49. Select **Design View** if it is not already selected.

50. Select the **blankPage** subform.

51. Expand the **Script Editor** and select the **initialize** event.

52. Enter this script.

this.presence = "hidden";

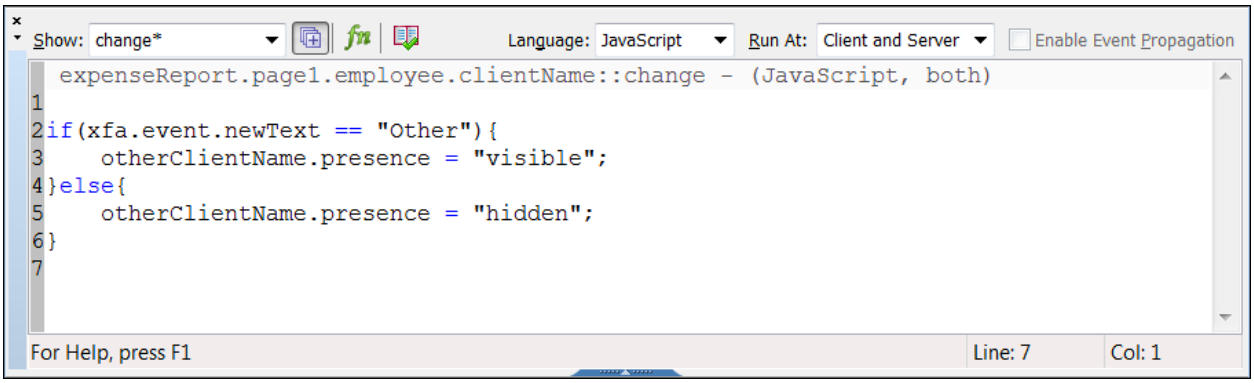
This will ensure that blankPage will be initialized as hidden even if the property is mistakenly set to Visible at design time in Designer.

Scripting in Designer

Scripting gives you full control over your form's functionality at runtime. Through scripting, you have the power to manipulate your form's interactive controls to provide your users with a richer and more intuitive experience. Designer scripting is similar to HTML scripting in the following ways.

- **Use of JavaScript:** You use JavaScript to write custom scripts that add functionality to your forms like you use JavaScript to add custom functionality to a web page. Designer also includes another language, FormCalc, which is covered later in this section.
- **Use of an object model:** Like web programming, Designer has an object model that you can use for your scripting. You can call methods and properties of the object model to provide rich functionality to your forms.
- **Range of scripts:** Because JavaScript is a rich, object-oriented scripting language; you can write an incredible range of scripts, from very basic one-line scripts to complex and detailed, object-oriented scripts that can span hundreds and even thousands of lines.

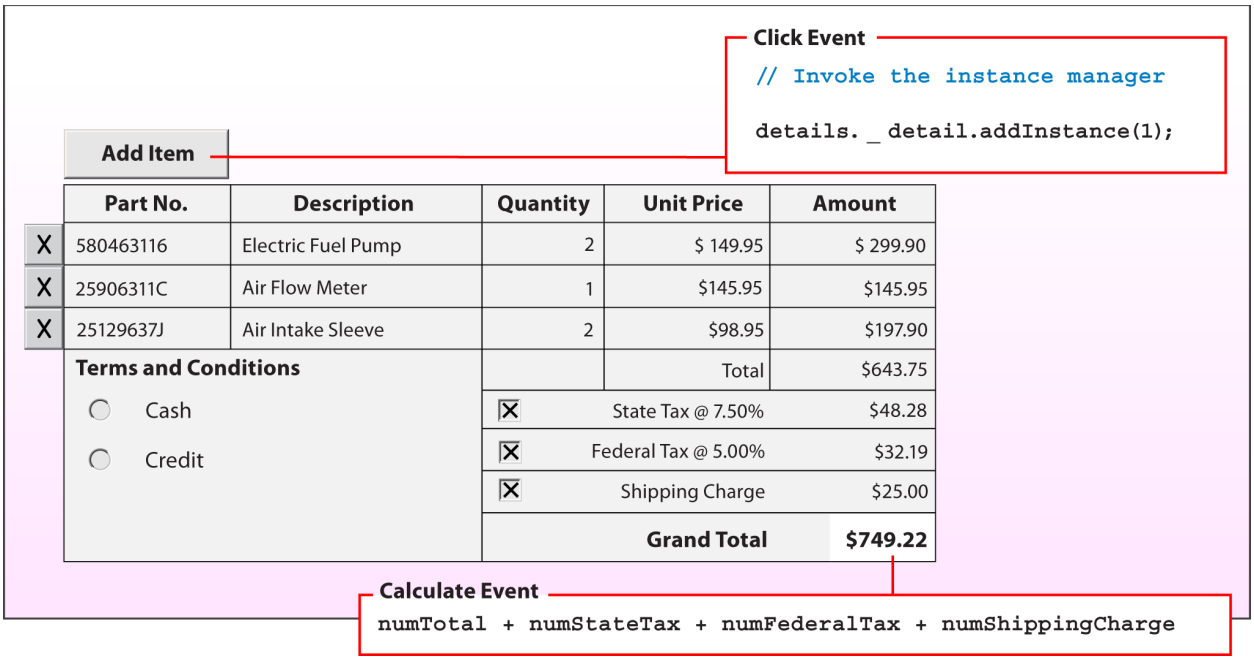
If you're familiar with JavaScript, you'll find the scripting concepts in Designer to be very familiar. However, since Designer uses the XFA object model, it is not exactly the same as writing JavaScript for web pages.



The Benefits of Scripting

In many cases, scripting improves the functionality and usability of your forms. A number of useful scripts are included in the Purchase Order form in your Student Files. One of these handy scripts is found in the Add Item button's **click** event. This script creates a new line item in the purchase order each time the button is clicked.

Another practical script is found in the Grand Total text field (**numGrandTotal**). This script automatically calculates the grand total for the purchase order every time a financial event on the purchase order occurs. When a new item is entered or when a new tax is applied, the grand total is recalculated automatically. This is a useful script for form fillers because they can see how all their changes and additions to the Purchase Order affect the bottom line. Unlike the previous script, which ran on the **click** event, this script runs on the **calculate** event of the numeric field.



Two scripts from the Purchase Order form show the benefits of adding scripting to your forms.

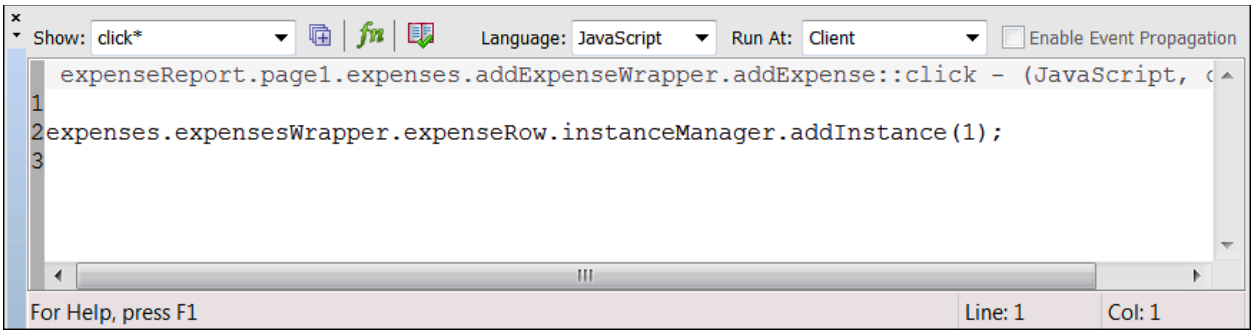
Although scripting is not a requirement in Designer forms, a small investment in scripting will yield the following benefits to you and your form filler.

- **Automatic calculations at runtime:** As described in the Purchase Order example, the addition of automatic calculations will make your forms much more useful to your form fillers.
- **Ability to control the appearance of form objects at runtime:** In the previous section, you set the visual properties of your form objects at design time. With scripting, you can make changes to the visual properties of your form at runtime to provide assistance to your form fillers. For instance, before your user submits a form, you can use a validation script to make sure that all the required fields have been filled in with data. If there are required fields without data, you can call a script to highlight them in yellow. This way, the form filler's attention will be drawn visually to the required task.
- **Enforcement of business rules:** You can use form scripting to enforce the business rules of your company or industry. For instance, the Purchase Order example could have a script that takes an action whenever the grand total goes above a certain threshold. You could create a script that limits users from going above a certain amount, or you could allow them to go above the amount and automatically route the purchase order for a senior manager's approval.
- **Data validation and formatting:** In addition to using patterns and properties to enforce data validation, you can use custom scripting. This approach is valuable because you'll be able to correct many data-entry mistakes before they get to your back-end system.

Using Scripting

Virtually every form object in a Designer form is scriptable at any step in your process. This is a big advantage because scripting will give you full control to respond to events and user actions in your smart forms at runtime.

For the most part, you'll put your scripts into form object events like the **click** event of a button or the **initialize** event of a text field. When these events fire, the scripts behind the events are executed. You can see an example of this type of scripting in the SmartDoc Expense Report. The addExpense button on the form has a script in its **click** event.

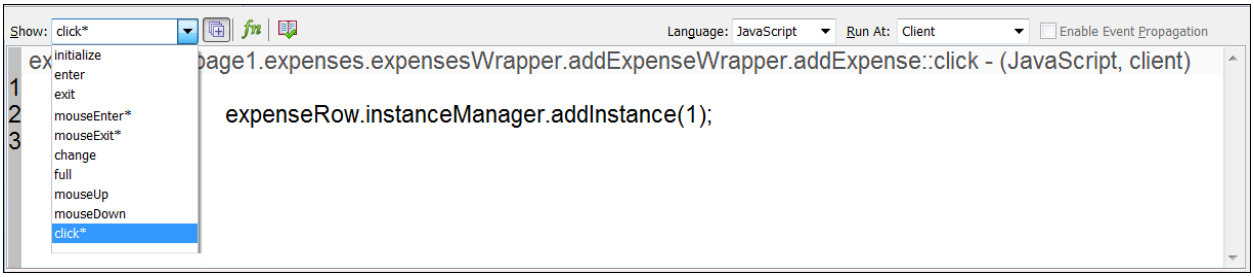


The Script Editor showing the **click** event of the addExpense button.

You can see this script by selecting the addExpense button in Design View. The Designer Script Editor shows you a few important aspects of this script:

- **Show Events:** This dropdown list shows you that the script will run in the **click** event of the button. The asterisk indicates that the event has scripting associated with it.
- **Language:** The (scripting) Language property is JavaScript, so the script must contain JavaScript with correct structure and syntax.
- **Run At:** The Run At specification is Client so the script will be processed on a user's local machine.
- **Editing window:** The editing window shows the actual script.

The addExpense button also has other scripts on the *mouseEnter* and *mouseExit* events.



FormCalc and JavaScript

Designer supports two scripting languages: FormCalc and JavaScript. A form can use both languages at the same time, but you can't mix the two languages in one object event. FormCalc is an easy-to-use calculation language that Adobe developed for XFA forms. It's very similar to the calculation language that you use in a spreadsheet program like Microsoft Excel. JavaScript, on the other hand, is a powerful, industry standard scripting language that many programmers are already familiar with. And with the advent of HTML forms, JavaScript is even more ideal because it runs natively in web browsers.

Surprisingly, FormCalc worked well in many of my HTML form tests, including when I set the scripts to run on the client. However, since JavaScript has so many advantages over FormCalc, I recommend JavaScript for most smart form applications. FormCalc is best suited for PDF-only applications developed by nonprogrammers. Also, Adobe advises against using it for HTML forms in Designer's Help system. This book focuses on JavaScript but includes introductory information about FormCalc in this section.

> ***Note:*** *The following is from Designer's Help System.—If you are developing forms for use with a server-based process (for example, using Forms) with the intent of rendering your forms in HTML, you should develop your calculations*

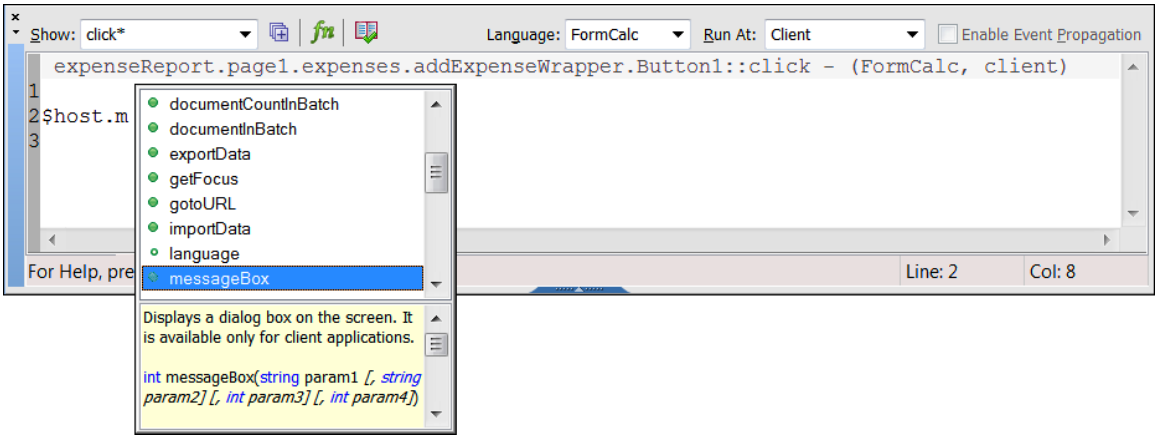
and scripts in JavaScript. FormCalc calculations are not valid in HTML browsers, and are removed prior to the form being rendered in HTML.

JavaScript is now the default scripting language in Designer. It offers the following advantages:

- **JavaScript is ubiquitous:** It's likely that you'll be able to leverage your organization's existing JavaScript knowledge because JavaScript is a scripting standard that many programmers and designers already know. Designer uses real and complete JavaScript, unlike the earlier versions of Adobe Acroforms, which used a subset of JavaScript.
- **JavaScript is object-oriented:** You can create JavaScript objects and custom functions and use them throughout your form. This feature of JavaScript isn't available in FormCalc, and it enables you to eliminate work by allowing to you reuse your code.
- **50,000 JavaScript fans can't be wrong:** Because JavaScript has been a scripting standard for a long time, you'll be able to find example scripts for almost everything you need to do. You can find these example scripts on the web, in JavaScript books, and in this book.
- **JavaScript works for PDF *and* HTML forms:** JavaScript will work for both PDF and HTML forms. You can use JavaScript to create forms that function in a similar fashion when they're rendered as HTML or PDF.

Script Editor Configuration

You can configure your Script Editor by selecting *Tools – Options – Workspace*. There are many features you can select including the Show Statement Completion Options and the Add Statement Completion Method Signatures. After you select these options, the Editor will provide completion options and method signatures as shown here.



Action Builder

If you need the benefits of scripting but don't have the time to master JavaScript, Designer's Action Builder may be the tool you're looking for. Action Builder is a script-creating assistant that you can access in Designer by selecting Tools – Action Builder. You select form objects, conditions, and results and Action Builder generates the script for you. For instance, you can select a Button object and indicate that you want an action performed when the button is clicked.

You can define the result of the action as a message box. Action Builder will generate the script and add it to your Button object's click event in the Script Editor. When a user clicks the button at runtime, the pop-up message box will appear.

How it works

You use the Action Builder to specify one or more conditions and one or more results that should occur when the conditions are met. Designer will interpret your actions and associated results and create JavaScript for your form. This JavaScript is added at the beginning of the event in your Script Editor before any script that you manually add. Designer treats an Action Builder script as a managed script, and it monitors the script for changes. If there are no changes, you can go back to the Action Builder and edit the action that generated the script. However, if you edit the script manually in the Script Editor, Designer will break the link between the script and the action in your Action Builder dialog box. In this case, you are free to work with the script as if it was entered manually. Designer will stop monitoring the script and display a message in the Log tab of the Report palette.

The Limits of Action Builder

Although Action Builder is a useful tool, an experienced JavaScript programmer can create JavaScript that is more efficient and easier to manage. Here is an example of the type of script that Action Builder creates.

```
//+ GENERATED - DO NOT EDIT (ID:98ED712F-1855-448B-99FF-C08B70BFB1EE CRC:1298018225)
//+ Type: Action
//+ Result1: Custom("$Node2","requiredValidation","","noreturn")
//+ Node2: form1[0].#variables[0].required[0]
//+ Node1: form1[0].page1[0].signature[0].callSOFunction[0]
//+ Condition1: Button("$Node1","click")
//+ ActionName: callSOMethod
this.resolveNode("required").requiredValidation();
//-
```

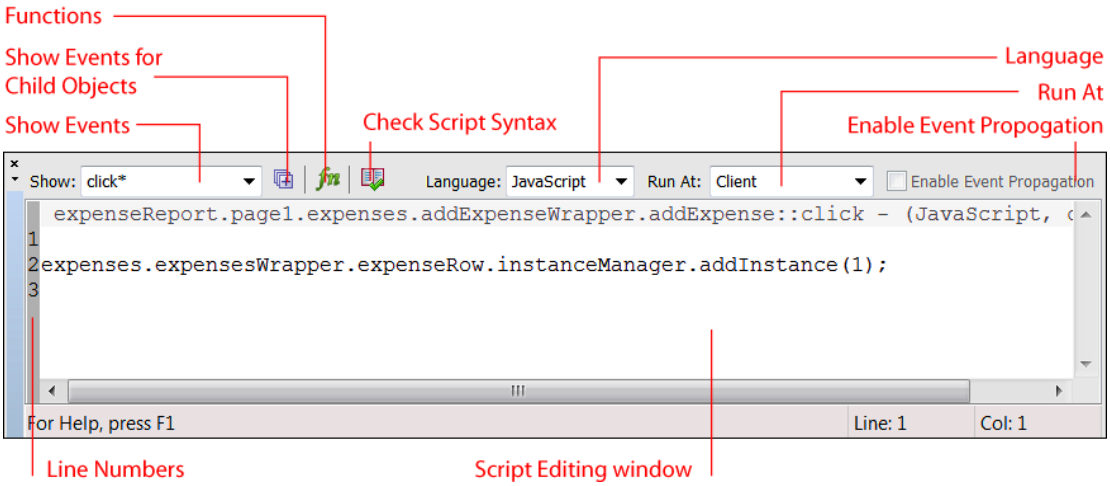
Although this script works, it relies on the **.resolveNode()** method which is less efficient than calling the function of the script object directly with this line of JavaScript.

```
required.requiredValidation();
```

As a best practice, you should limit the number of times you call the **.resolveNode()** and **.resolveNodes()** methods in your scripts. This is particularly important when your form has many objects and a deep subform hierarchy. By using direct paths in your JavaScript, your host application *(Acrobat or Reader)* will not need to iterate through a complex form structure in order to resolve the node reference. As you work with Action Builder, you will see that it relies heavily on the **.resolveNode()** method. However, if you are new to JavaScript and you want to add automated feature to your Designer forms, Action Builder is a very useful tool. You can think of it as your personal JavaScript Assistant.

The Script Editor

As you have seen in previous examples, the Script Editor is the tool you'll use to create and edit scripts. It contains a number of features to help you write, test, and organize your scripts.



If you can't see the Script Editor on your screen, select Window – Script Editor. The Script Editor shows the following information:

- **Script Editing window:** This is where you write and edit your scripts.
- **Show Events dropdown:** In this case the event is the button's **click** event. The asterisk is an indication that there's script in this event. If you select the Show Events dropdown list, you'll see all the events for this object. The events that are grayed out are not available for this object. Choosing the right event for your script is very important.
- **Show Events for Child Objects:** You can see all the scripts for a subform, a page, or for your entire form at once. This option will show all the scripts for the object that you currently have selected and all its child objects. If you select the top form object in the hierarchy, this option will show you all the scripts associated with a specific event. As an alternative, you can change from a specific event to view all Events With Scripts to see every script in your form.
- **Functions:** Click this button to view a list of FormCalc or JavaScript functions, depending on your language setting. You will use this in the FormCalc exercise.
- **Check Script Syntax tool:** This tool provides some basic syntax checking for FormCalc and JavaScript. It is not a very robust debugger at all. The Advanced Scripting class has a section on the Acrobat JavaScript debugger, which is a more fully realized JavaScript debugger for PDF forms.
- **Language:** In this case, the language is set to **JavaScript**. If you put proper FormCalc script into this box, it won't work. Remember, both languages can be used in the same form, but not in the same event.

– **Run At:** This option specifies where your script will run. If it is set to **Client**, the script will be processed by Acrobat, Reader, or a web browser. If the specification is set to **Server**, the script will be processed by the server. If your workflow doesn't include a server, you should always use the **Client** option. The third option is **Client and Server**. Use this setting when you aren't sure if your form will be rendered on a client or a server, or for forms that may be rendered on both the client and the server at different times.

Scripting in Designer is simple enough that you can begin today, but it's robust enough that you won't be bored even after years of programming.

Variables

A *variable* is a symbolic representation that refers to a place in computer memory that holds a value. Variables are advantageous because they enable you to define a value in one location but refer to it from many locations on your form. When you need to change the value, you don't need to go back to each variable reference; you only need to go back to the one location where the variable is defined to make your change. All other locations that refer to the variable will be updated. This section focuses on two primary variable types:

– **Form variables:** Form variables are declared and assigned in the Variables tab of the form's properties dialog box. Form variables are global in nature, which means you can access them from any part of your form.

– **Script variables:** Script variables are declared and assigned in your scripts. Script variables are local in nature, so you can only access a script variable from within the script where it is declared.

Variable names are case-sensitive and can't contain spaces. You can't begin a variable name with a number, but a variable name can contain numbers, letters, and underscores. At runtime, naming conflicts occur when the names of variables are identical to those used as XML Form Object Model properties, methods, or form design field names. These conflicts can cause scripts to return unexpected values; therefore, it's important to give each variable a unique name. Here are examples:

– Use the variable name **fieldWidth** and **fieldHeight** instead of **w** and **h.**

– Use the form design object name **clientName** instead of name.

Form variables

A form variable typically acts as a placeholder for a value that you might have to change in the future. For instance, you can create form variables for state tax rates. When you need to calculate the sales tax, you reference the form variable in your script. When you need to change the tax rate, all you have to do is open the Variables panel in the Form Properties dialog box and make the change. Because you referenced the form variable, not a specific numeric value, you don't need to update the tax rate in multiple locations on your form. The values of your form variables will reset each time the form is opened. All form variables are stored as strings; therefore, you may need to convert it to a number before it's used in a calculation.

Script variables

Unlike form variables, script variables are local in nature. A script variable is within the scope only while the script is running. If a variable is declared in one script and then referenced in another script, an error will occur. Once the script is finished, the memory that was assigned to the local variable is released and is able to be used for other purposes.

Object References

It is important to create accurate object references in your scripts.

Fully qualified

A *fully qualified* reference uses the complete form hierarchy beginning with the **xfa** root node. The advantage of a fully qualified statement is that it will work every time regardless of where the script that contains the reference is located. Here is an example of a fully qualified reference. The visible root node of the form in Designer's Hierarchy palette is form1. A fully qualified reference goes two levels above this, all the way to the root node of xfa.

xfa.form.form1.purchaseOrder.commentsHeader.exampleButton

Abbreviated reference

Even though fully qualified references will always work, they usually take longer to write than abbreviated references. The syntax for an abbreviated reference is shorter because it starts with the first object the two objects have in common. For instance, this abbreviated reference works in the change event of the clientName dropdown object of the SmartDoc Expense Report: **otherClientName.presence = "visible";**. The reason it works, and the reason it is a clear object reference is that the clientName object and the otherClientName object have the same parent. If two objects do not have the same parent in common, you can go up a level to see if they have grandparents or great-grandparents in common.

The current object

Both FormCalc and JavaScript use shortcuts to reference the current object. Here is the syntax

- JavaScript: **this.assist.toolTip.value**
- FormCalc: **$.assist.toolTip.value**

Other Techniques

Unnamed objects

I recommend providing meaningful names to your form objects. However, occasionally you may come across unnamed objects in Designer. Here is the syntax for referring to unnamed objects

xfa.form.form1.#subform.PrintButton1.presence = "invisible"

Multiple objects with the same name

Designer also supports the creation of multiple objects with the same name. When multiple objects have the same name, each has a different occurrence number represented by a number in brackets directly following the object name. Here is the syntax for FormCalc.

xfa.form.form1.#subform.TextField1[0].rawValue = "First"
xfa.form.form1.#subform.TextField1[1].rawValue = "Second"
xfa.form.form1.#subform.TextField1[2].rawValue = "Third"

You'll also need to use the **xfa.resolveNode** method if you're using JavaScript with multiple objects of the same name, because JavaScript can't interpret the occurrence number syntax. The following code will work in JavaScript:

```
xfa.resolveNode("xfa.form.form1.#subform.TextField1[0]").rawValue = "First";
xfa.resolveNode("xfa.form.form1.#subform.TextField1[1]").rawValue = "Second";
xfa.resolveNode("xfa.form.form1.#subform.TextField1[2]").rawValue = "Third";
```

FormCalc Shortcuts

FormCalc has built-in shortcuts that reduce the effort required to write references. For instance, the **$host** shortcut makes reference to the host object. You can replace the script of your exampleButton with the following line of code:

```
$host.messageBox("hello world")
```

Hidden XFA Properties that you can script

Due to the way the XML Form Object Model is structured, some object properties and methods exist on child objects of the objects on the form. These child objects exist only as part of the XML Form Object Model and don't appear in the Hierarchy and Data View palettes. To access these properties and methods, you must include the child objects in the reference syntax. For example, the following reference syntax sets the tool tip text for the txtCondition field:

```
txtCondition.assist.toolTip.value = "Conditions of purchase.";
```

Events

The forms you create in Designer are based on an event-driven programming model. As you've seen in this section, the scripts you write are triggered by events. Events are occurrences or actions that change the state of a form. When these events occur, the scripts behind the events are executed. Because this event-driven model determines how your scripts will perform, you need to consider which events you want to use, when the events will fire, and how often the events will fire.

Types of events

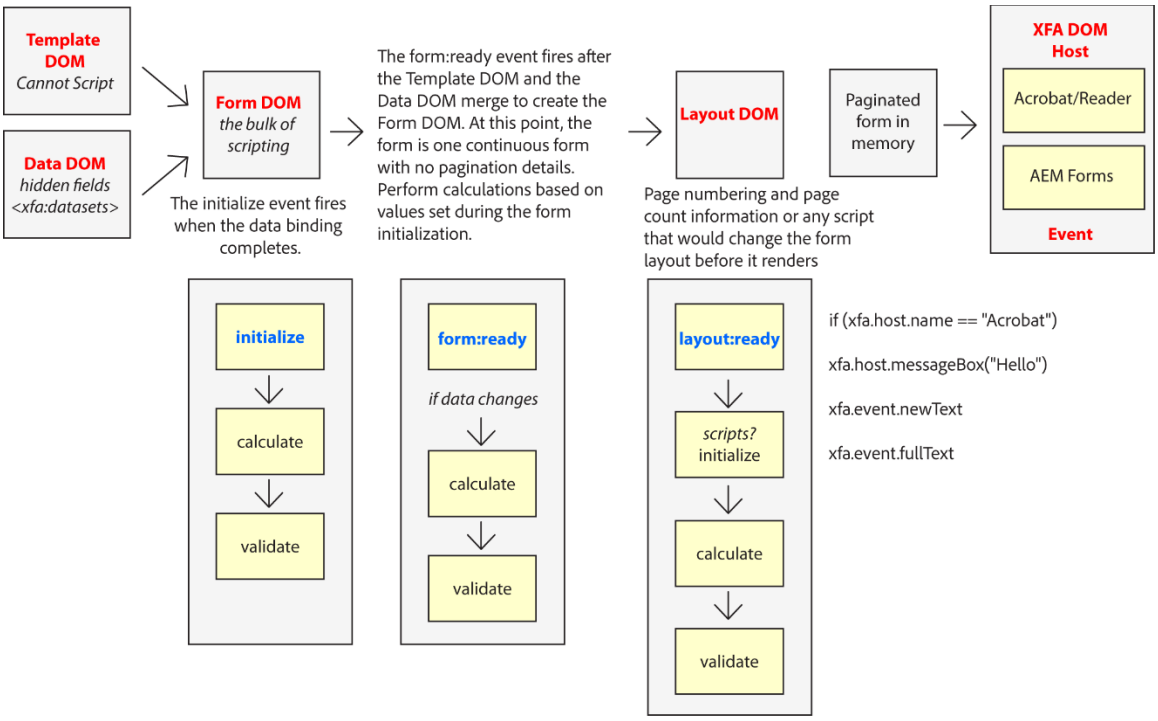
There are three categories of events in Designer forms: process events, interactive events, and application events. Most of your scripts will run in the interactive events, but the process and application events will also be useful to you.

Process events

The following is a list of the process events:

- calculate
- form:ready
- indexChange
- initialize
- layout:ready
- validate

Process events will fire automatically based on an internal process or in response to the firing of an interactive event. As illustrated below, process events are fired following a major form change like the form merging process and the form layout process.



This diagram shows the general flow of process events leading up to the PDF form opening in the application. The process events are in bold type. The last event, **docReady**, is an application event. The Template DOM and Data DOM are merged into the Form DOM. You can put scripting in events of the Form DOM but make sure to put page related scripting in the layout:ready event which fires after the Layout DOM is complete.

Process events are also fired after interactive events are fired. For instance, a button **click** event will also fire the **layout:ready** event for each object on the form. Process events may fire multiple times based on the situation. For instance, a script in a button's **calculate** event will fire twice when an interactive form is opened.

Interactive events

The following is a list of the interactive events:

- change
- click
- enter
- exit
- mouseDown
- mouseEnter
- mouseUp
- preOpen
- postOpen
- postSign
- preSign
- mouseExit

Interactive events are useful for scripting because they fire as a direct result of a user's action.

Application events

The following is a list of the application events:

- docClose
- docReady
- postPrint
- postSave
- prePrint
- preSave
- preSubmit
- postSubmit

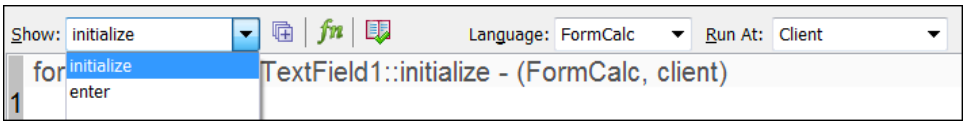
Application events fire as a result of the actions of a client application like Acrobat or a server application like AEM Forms. For example, the **postPrint** event will fire immediately after Acrobat or Reader has sent the form to a printer or spooler.

Exercises

FormCalc Scripting

FormCalc has many built-in functions that cover a range of areas, including finance, logic, dates/times, and mathematics. FormCalc is supported in Designer, Acrobat, and Reader. Like Microsoft Excel, most FormCalc scripts are only one line long. Follow these steps to see some of FormCalc's functions in action:

1. Open Designer if it is not already open.
2. Select **File – Open** and navigate to your Student Files.
3. Select *formcalc.xdp* and click **Open**.
4. Select **File – Save As** and navigate to your working folder.
5. Enter **<yourname>FormCalc.xdp** as the *File name*.
6. Click the *Save As type* dropdown, select **Adobe XML Form (*.xdp)**.
7. Click **Save**.
8. Select **TextField1** in the Hierarchy palette and open the Script Editor.
9. Select the **initialize** event in the *Show Events* dropdown, make sure *Language* is set to **FormCalc** and *Run At* is set to **Client** *(see illustration)*.



10. Enter this script.

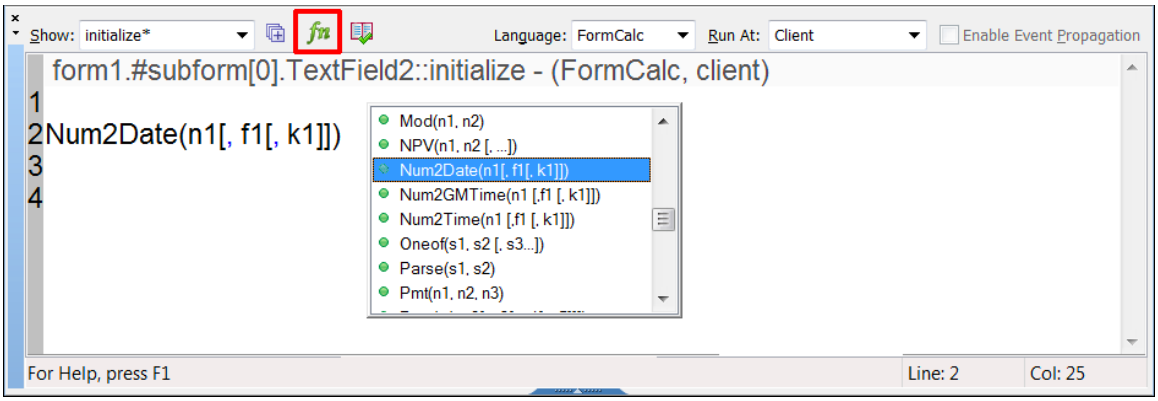
TextField1.rawValue = Sum(1,2,3,4)

11. Select **TextField2** and go to the Script Editor.

12. Select the **initialize** event in the Show Events dropdown.

13. Click the **Functions** icon and select the **Num2Date()** function.

    Note: *You can use this approach to stub-out your functions. This will ensure that your function syntax is correct.*



14. Complete your script so it matches this. You can also use the Functions icon for the **Date()** and **DateFmt([n1[, k1]])** functions.

TextField2.rawValue = Num2Date(date(), DateFmt(3))

15. Select **NumericField1** and go to the Script Editor.

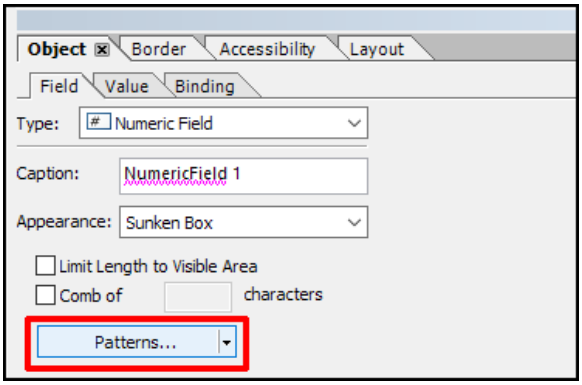
16. Select the **initialize** event and enter this script. You can also use the Functions icon for the Apr() function.

NumericField1.rawValue = Apr(35000, 269.50, 360)

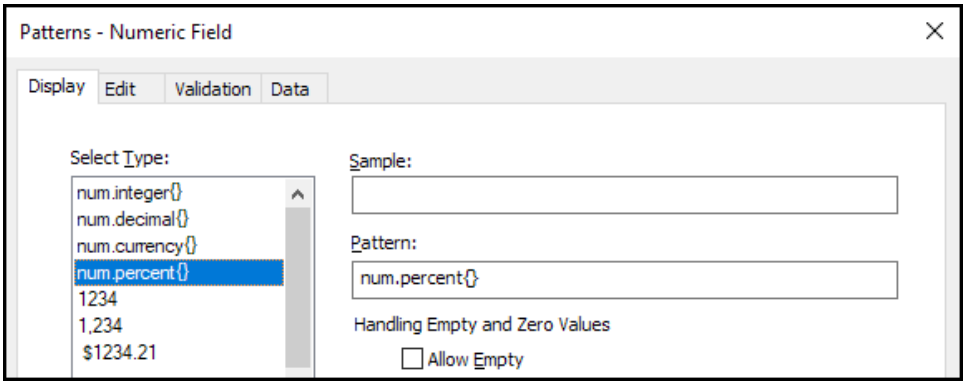
Note: *If you're working on financial forms, FormCalc has many built-in financial functions that you can add to your form. For instance, it's easy to calculate the APR (annual percentage rate) of a loan with FormCalc. In this example, the principal amount of the loan is $35,000, the monthly payment amount is $269.50, and the number of months is 360. This script will return an annual percentage rate of 0.08515404566 to NumericField1. In the next step, you'll add a display pattern to your field so that this calculation is more relevant to the user.*

17. Select **NumericField1** and select the *Field* tab of the *Object* palette.

18. Click **Patterns** *(see illustration)*.



19. Select the **num.percent{}** display pattern to show the calculation as a percentage.



20. Click **OK** to close the *Patterns* dialog box.

21. Select **Button1** and go to the Script Editor.

22. Click the Show Events dropdown *(see illustration)* and select **click** event.

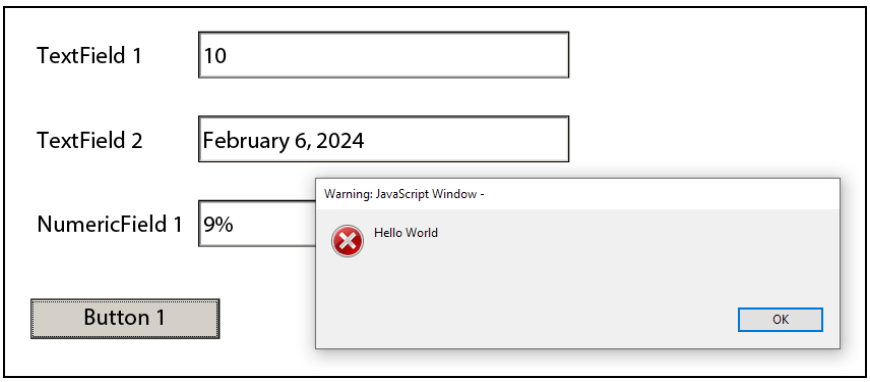
23. Enter this script.

$host.messageBox("Hello, World!")

> ***Note*** *The* ***$host*** *item is a FormCalc shortcut that represents the host object, which is either Adobe Acrobat or Adobe Reader, the host of your PDF form.*

24. Select **Preview PDF** to view your new scripts in action. The form fields should now show 10 (the sum of [1, 2, 3, 4]), today's date, and 9%.

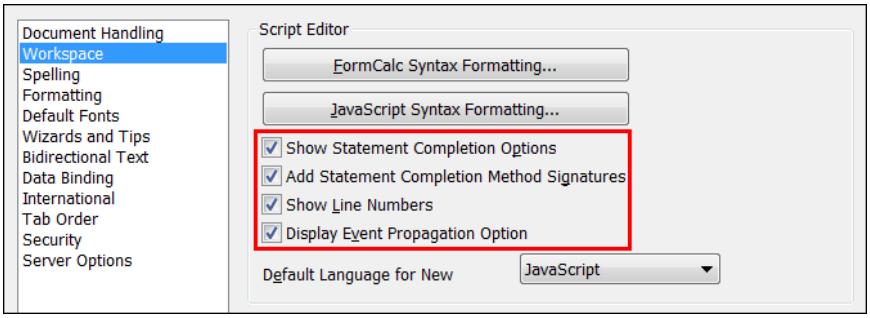
25. Click **Button1** and you will see a message box with a **Hello, World** message.



26. Click **OK** to close the message box.
27. Select **Design View** to close the PDF preview.

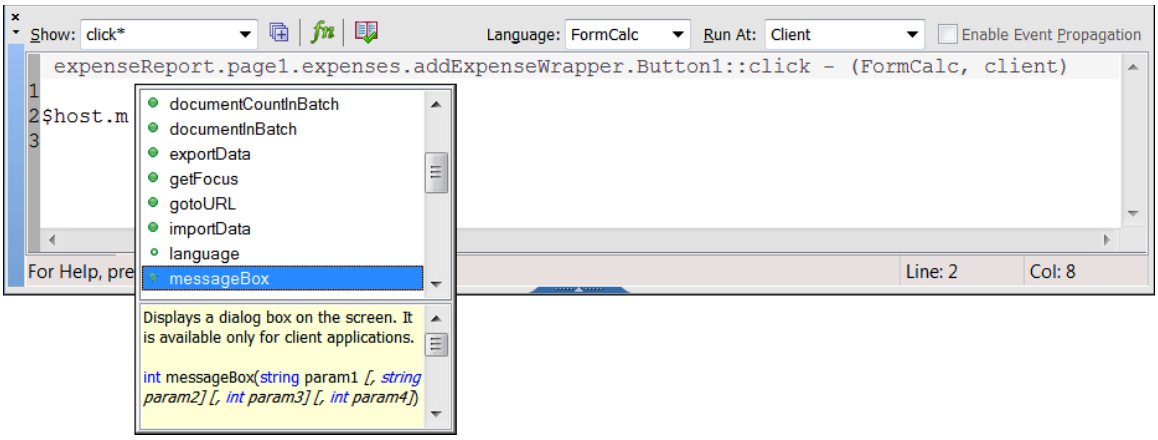
Script Editor Configuration

28. Select **Tools – Options – Workspace** to configure your *Script Editor*.
29. Select all of the options outlined in red *(see illustration)*.



30. Click **OK** to close the dialog.
31. Select **Button1** and go to the Script Editor.
32. Click the Show Events dropdown and select **click** event.
33. Click the *Language* dropdown and select **JavaScript**.
34. Delete the existing FormCalc script.
35. Type in this line of JavaScript. Notice the statement completion feature of Designer working as you enter this script *(see illustration)*.

xfa.host.messageBox("Hello, World!");



Note: *The statement completion feature of Designer helps you with context-sensitive information.*

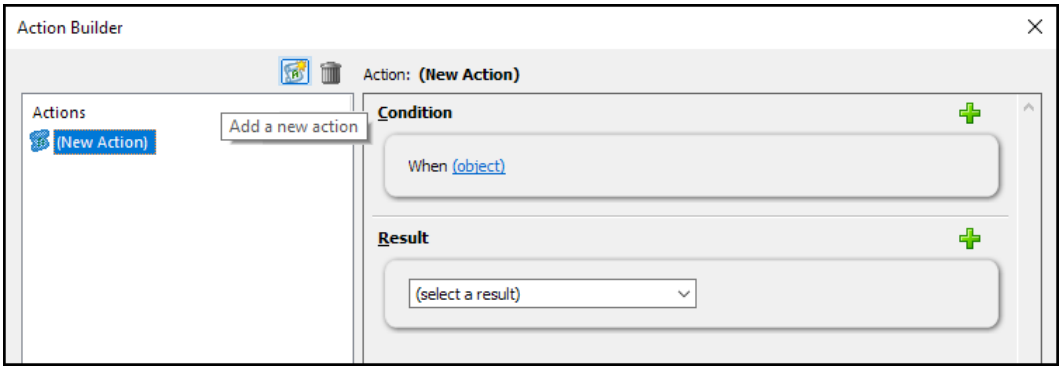
36. Select **Preview PDF**.
37. Click **Button1** to see the same pop-up message box. The host object is still either Acrobat or Reader, but the JavaScript syntax for referring to the host object is **xfa.host.**
38. Select **Design View**.

Action Builder exercises

Add a Message Box for a Form Submission

Follow these steps to use Action Builder to create a script that will display a message box when the form is ready to be submitted.

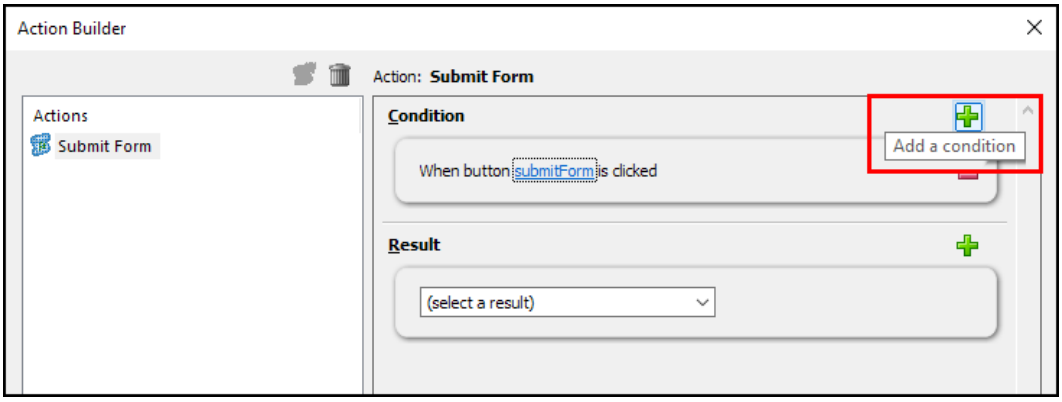
1. Select **File – Open** and navigate to your Student Files.
2. Select *changeOfBeneficiaryActions.xdp* and click **Open**.
3. Select **Tools – Action Builder** to open the Action Builder dialog.
4. Click the **Add a new action** button *(see illustration)*.



5. Double click on **(New Action)** in the Actions list and rename the action **Submit Form**.
6. Create a condition by clicking the **(object)** link in the *Condition* panel.
7. Scroll down, select the **submitForm** button.

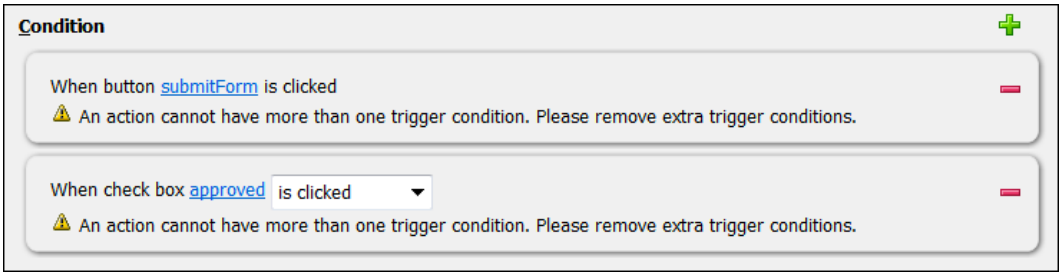
8. Click **OK**. A condition has been created *(see illustration)*.

9. Click the top green plus sign next to *Condition* to add a second condition *(see illustration)*.



10. Create a second condition by clicking the **(object)** link.

11. Scroll down, select the **approved** check box, and click **OK**. A second condition has been created. A warning has also been added to the two conditions to let you know that you now have multiple trigger conditions for one action.



12. Change the **is clicked** property to **is checked** in the second condition. By changing this condition, you are eliminating the second trigger condition, therefore the warning is removed.

13. Click the *Result* dropdown and select **Show a Message Box**.

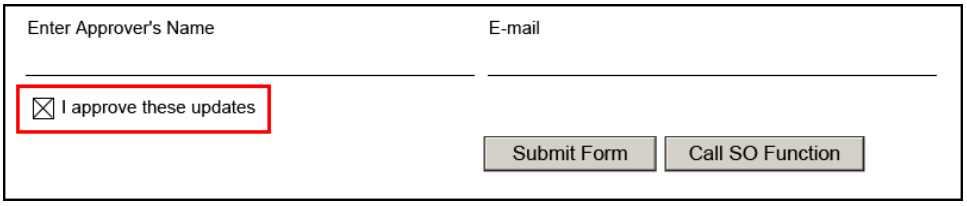
14. Enter **Your form is ready to be submitted** as the message.

15. Enter **Ready for submission** as the title and click **OK**.

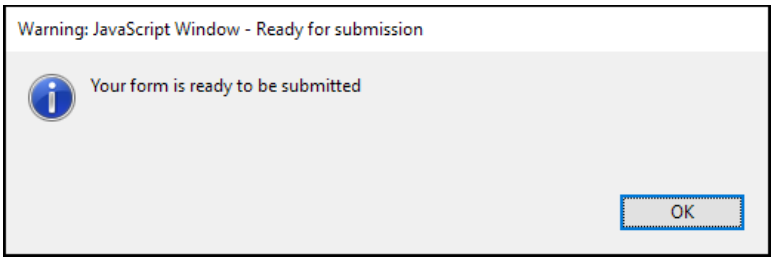
16. Select **Preview PDF** to see your script in action.

17. Scroll down and select the **I approve these updates** check box *(see illustration)*.

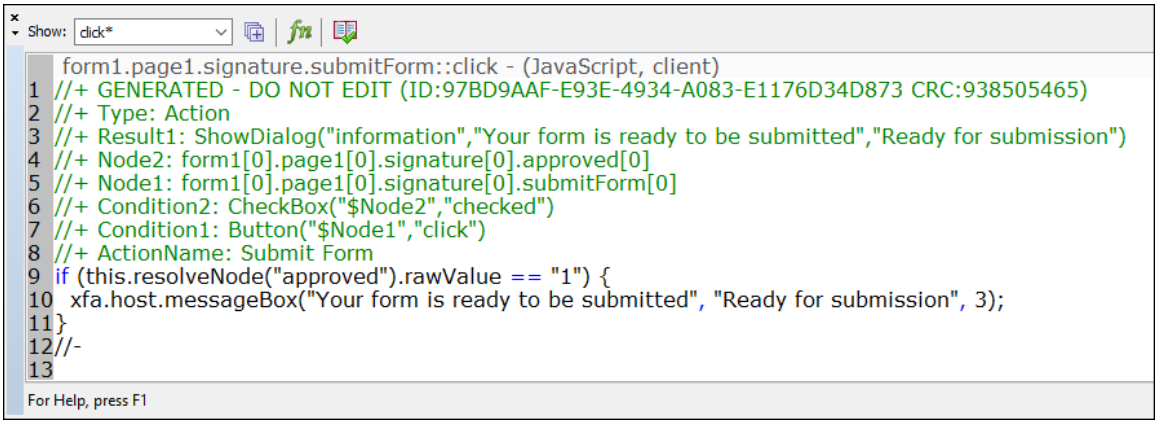
18. Click the **Submit Form** button.



19. You will see a message box stating that your form is ready to be submitted.

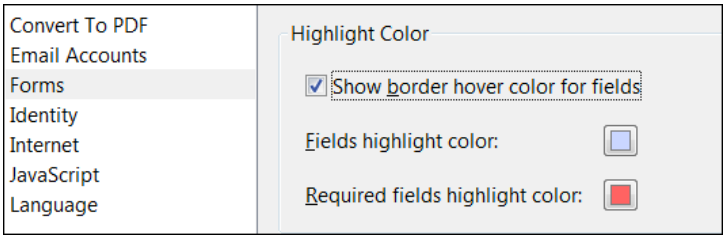


20. Click **OK** to close the message box.
21. Select **Design View** to close the PDF preview.
22. Scroll to the bottom of your form and select the *Submit Form* button.
23. Open the *Script Editor* and notice the *click* event is chosen *(see illustration).*
24. You will see the script that Action Builder created for you *(see illustration).*



Highlight Required Fields

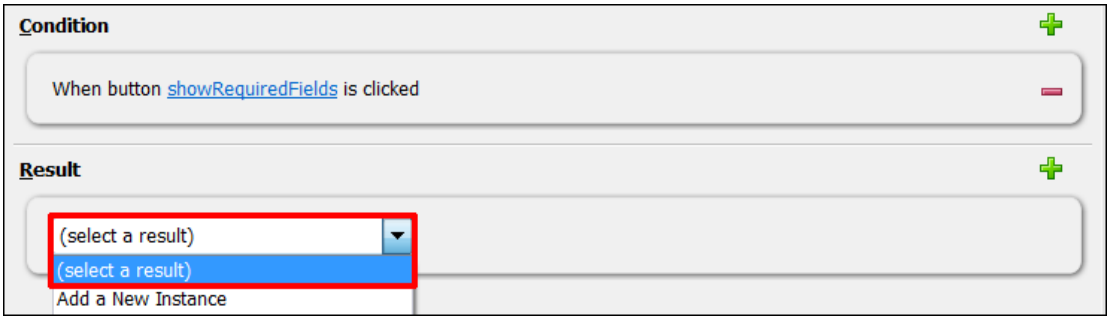
> ***Note:*** *In this exercise, we will use Action Builder to highlight required fields with yellow. There is a setting in Acrobat that will affect how you see the yellow highlight. You can see this setting by selecting Preferences – Forms in Acrobat. If the Show border hover color for fields is selected, you will not see the yellow highlighting.*



25. Select **Tools – Action Builder**.
26. Click **Add a new action**.
27. Double click on **(New Action)** in the Actions list and rename the action **showRequiredFields**.
28. Create a condition by clicking the **(object)** link in the *Condition* panel.
29. Select the **showRequiredFields** Button object in the header subform of page1.

30. Click **OK**.

31. Click the *Result* dropdown and select **Set the Background Color of a Field** in the dropdown list.



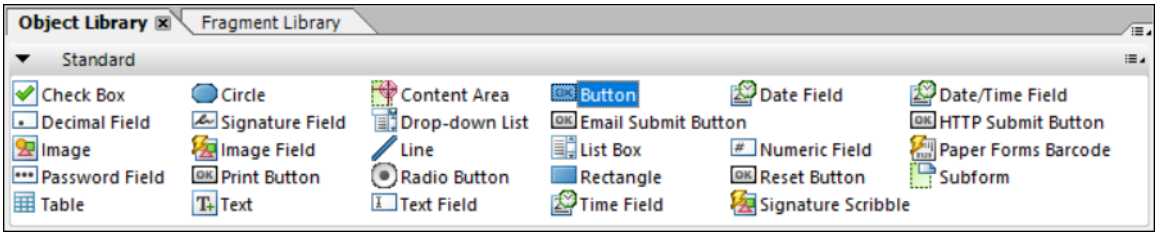
32. Click the **(object)** link and select the **primaryName** field.

33. Click **OK**.

34. Click the *Color* dropdown and select **yellow**.

35. Click **OK** to complete your action. Designer automatically adds custom JavaScript to your button's click event.

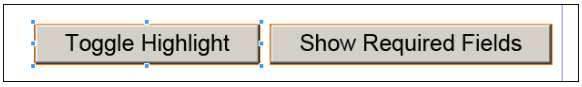
36. Locate the *Button* object in the Standard Object Library *(see illustration)*.



37. Drag and drop a new button object to the header subform on your form.

38. Name the button **toggleHighlight**.

39. Enter **Toggle Highlight** as the caption. You should now have two buttons that look like this.

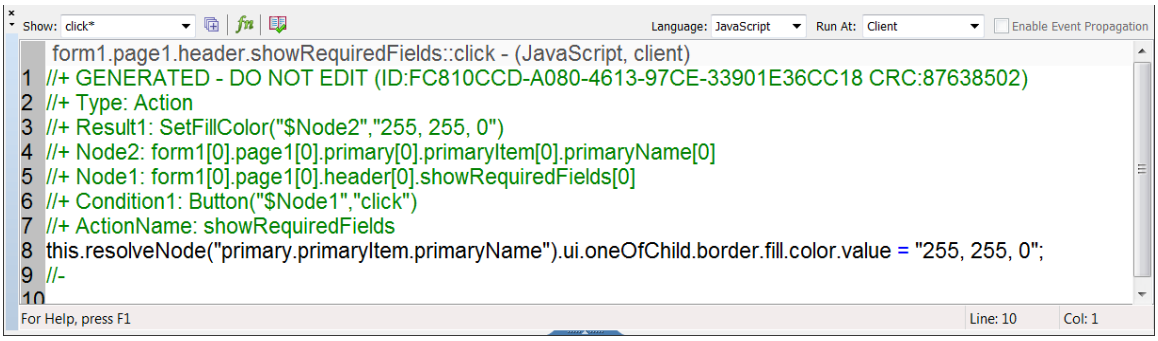


40. Enter this script in the click event of the toggleHighlight button.

app.runtimeHighlight = !app.runtimeHighlight;

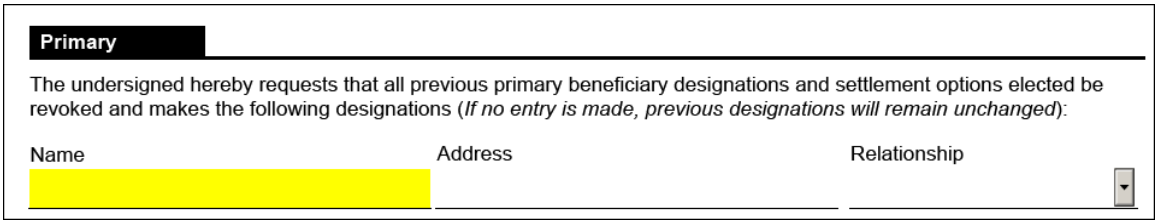
Note: *This button will enable you to toggle the runtime property value.*

41. Select the **showRequiredFields** button and review the JavaScript in the button's click event. The majority of the lines are JavaScript comments.



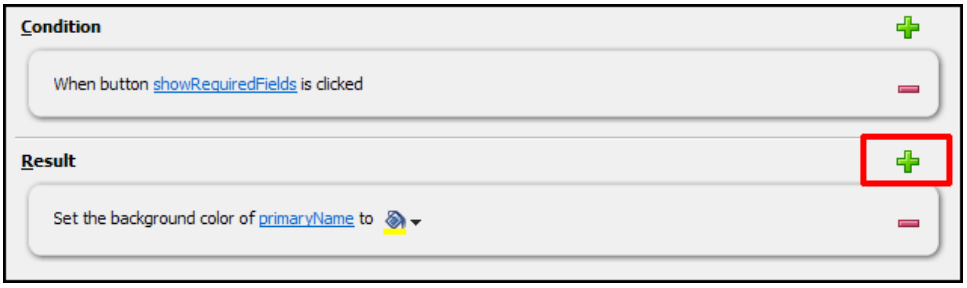
42. Select the **Preview PDF** tab to see the new script in action.
43. Click the **showRequiredFields** button.
44. The primaryName field should now be yellow *(see illustration)*.

    Note: *If your highlighting is not working, use your Toggle Highlight button.*

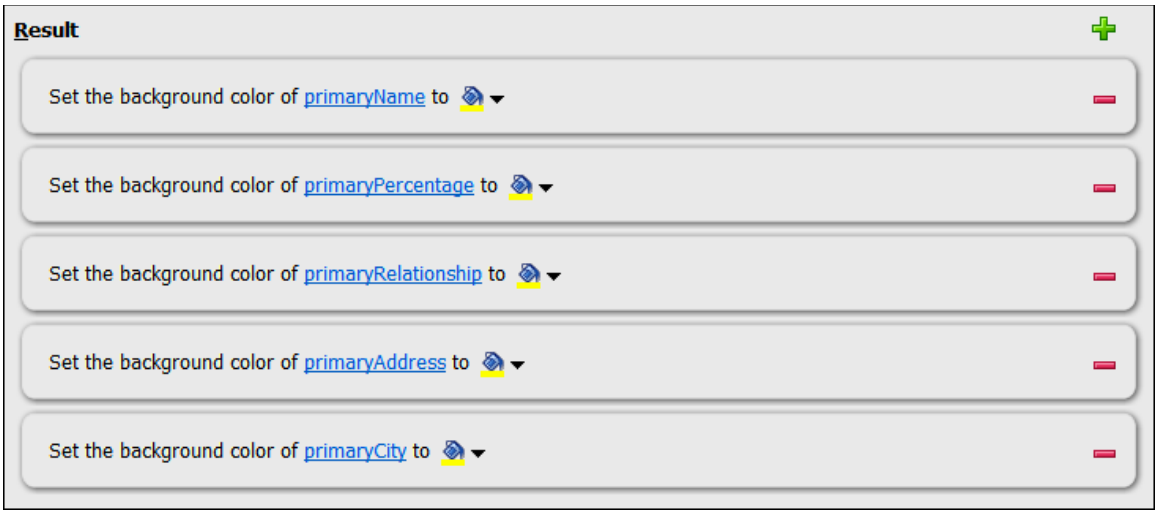


You can add additional results to the same condition by returning to the Action Builder.

45. Select **Design View** to close the PDF preview.
46. Click **Tools – Action Builder**.
47. Select the **showRequiredFields** action.
48. Click the green plus sign on the right of the *Result* panel *(see illustration)* to add a new Result.



49. Repeat the previous steps for the other fields in the primaryItem subform. Set their background color to yellow so you have multiple results *(see illustration)*.



50. Click **OK** in the Action Builder dialog box when you are done.
51. Select the **showRequiredFields** button and review the JavaScript in the button's click event of the Script Editor. If you don't see the Script Editor, select *Window – Script Editor*.
52. You should see this line of JavaScript for each field.

this.resolveNode("primary.primaryItem.primaryName").ui.oneOfChild.border.fill.color.value = "255, 255, 0";

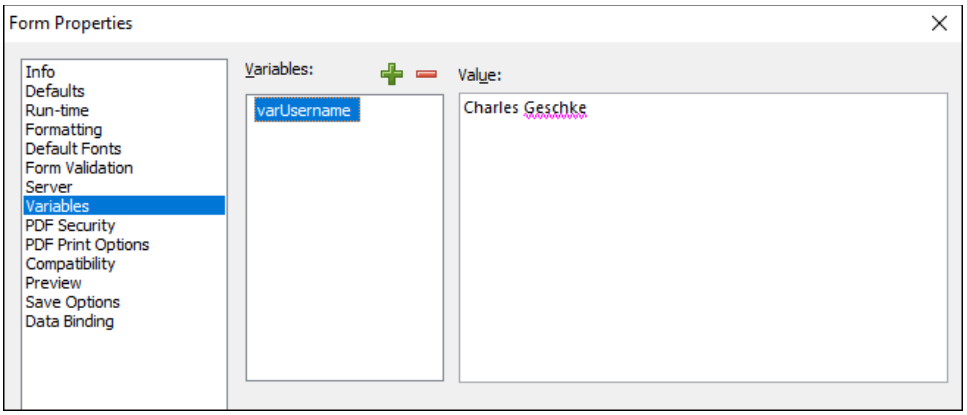
53. Select the **Preview PDF** tab to see the new script in action.

    Note: *The highlighting will not work with the "Highlight Existing Fields" button selected.*

Work with Variables

In this exercise, you will work with form and script variables.

1. Select **File – Open** and navigate to your Student Files.
2. Select *changeOfBeneficiaryStart.xdp* and click **Open**.
3. Select **File – Form Properties – Variables**.
4. Click the **green plus** to enter a new variable.
5. Enter **varUsername** as your variable name, and then enter **Charles Geschke** as your variable value *(see illustration)*.

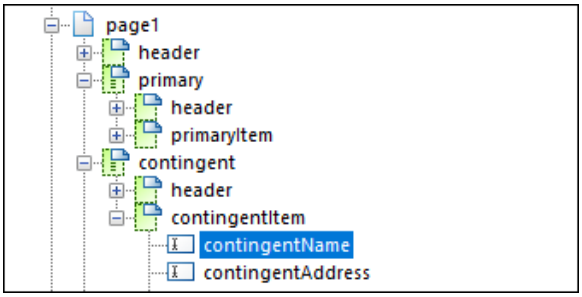


6. Click **OK** to close the *Form Properties* dialog box.

   Note: *Even though you were able to create a form variable without scripting, you must use scripting to apply the value of your variable to a form object at runtime.*

You can now make a reference to this variable from anywhere on your form.

7. Expand the *contingent* subform until you can see the *contingentName* field *(see illustration)*.



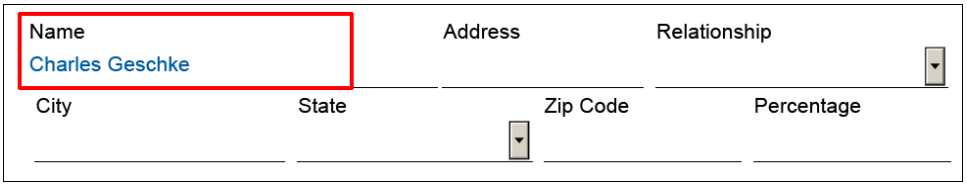
8. Select the **contingentName** field.

9. Expand the **Script Editor**.

10. Select the **initialize** event, and add this script:

this.rawValue = varUsername.value;

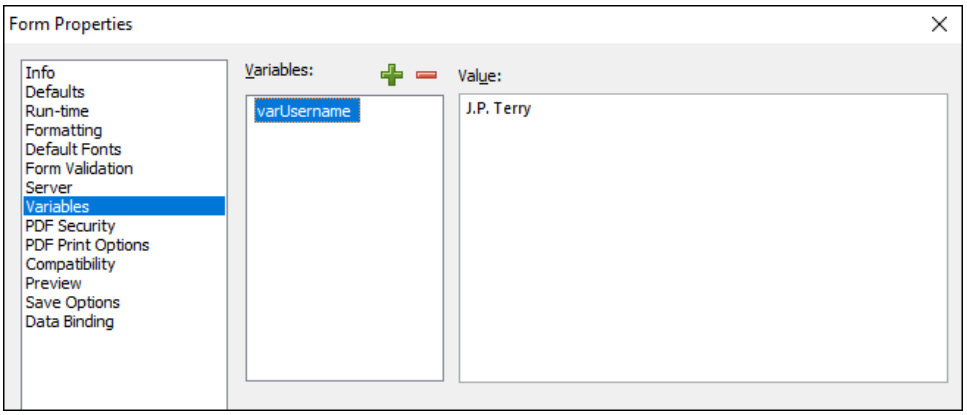
11. Select **Preview PDF**. The value **"Charles Geschke"**, from your form variable is prefilled in the form.



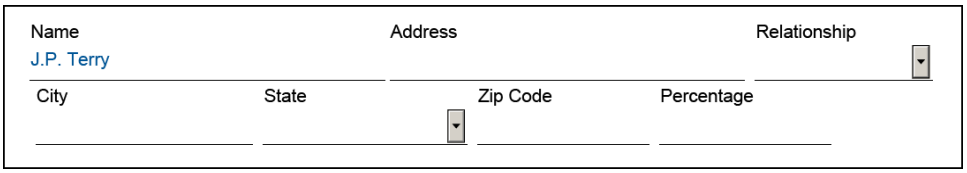
Note: *This simple script uses two different property values to refer to similar information. When you need to access the value of an XFA field or form object, use the* ***.rawValue*** *property. When you need to access the value of a form variable, use the* ***.value*** *property. If you're working in FormCalc, you can retrieve the value of the variable without explicitly referencing a value property, although adding the value property will also work.*

order.#subform[0].fromVariable::click: - (FormCalc, client)
$.rawValue = varUsername

12. Select **Design View** to close the PDF preview.
13. Select **File – Form Properties – Variables**.
14. Select the *varUsername* variable *(see illustration)*.
15. Change the variable's value to your name *(see illustration)*.



16. Click **OK**.
17. Click **Preview PDF**.
18. You will now see the *Name* field populated with your name *(see illustration)*.



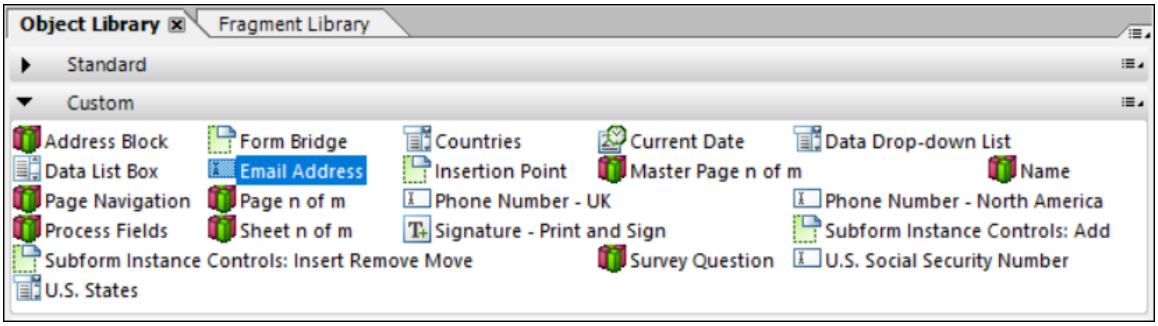
19. Select **Design View**.

Note: *Your form variable values will be reset to their original values each time the form is opened.*

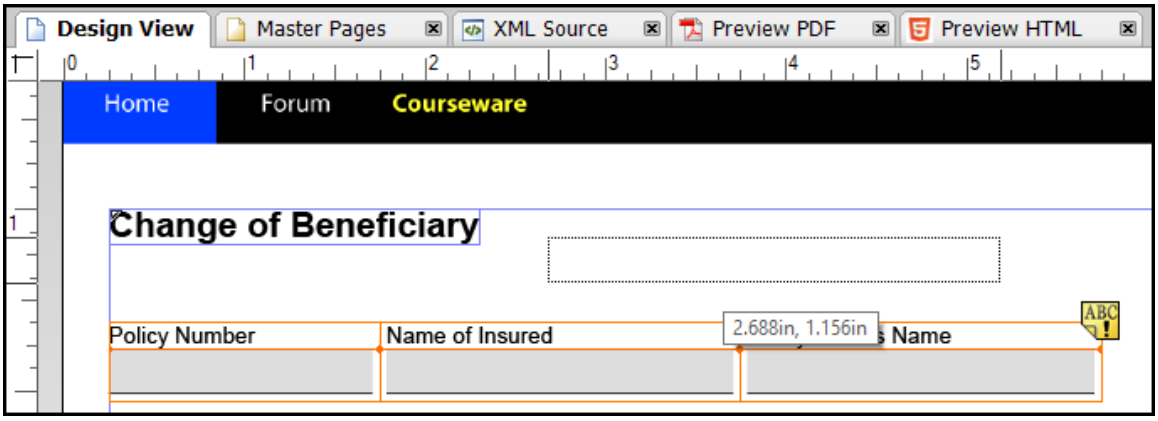
Script variables

Unlike form variables, script variables are local in nature. A local variable is used only within the scope of the script in which it's declared. Follow these steps to see a script variable.

1. Locate the **Email Address** object in the *Custom* tab of the *Object Library (see illustration)*.



2. Drag and drop the **Email Address** object from the *Custom* tab of the *Object Library* to the top of your form in Design View *(see illustration)*.

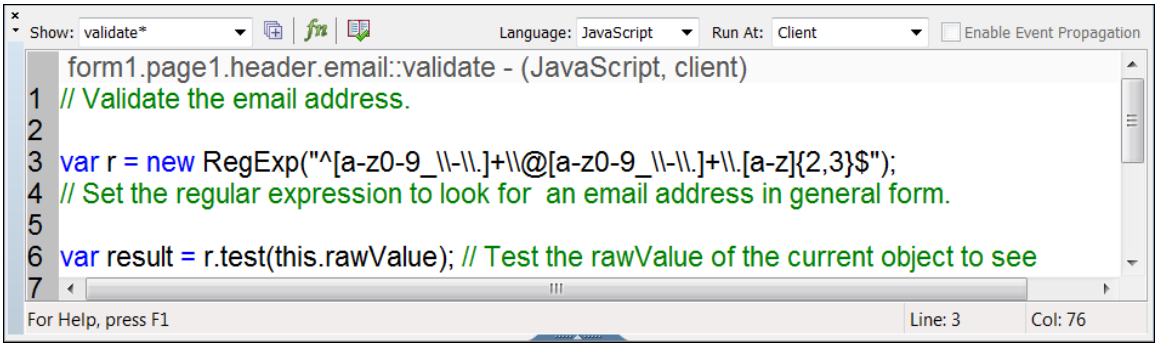


3. With the Email Address selected, open or expand the **Script Editor**.

4. Click the *Show Events* dropdown and select the **validate** event.

5. You will see a script variable named **r** and a script variable named **result** defined with the JavaScript keyword **var**.

6. These variables are local to this script in this event.



Create Object References

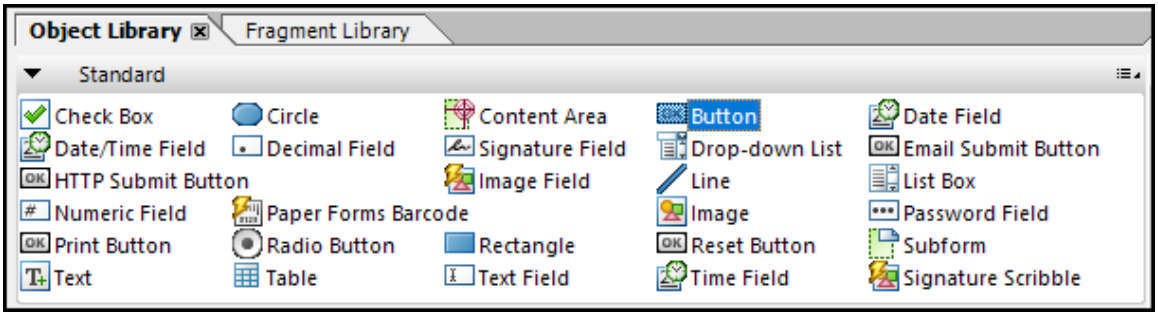
Follow these steps to reference objects.

1. Select **File – Open** and navigate to your Student Files.

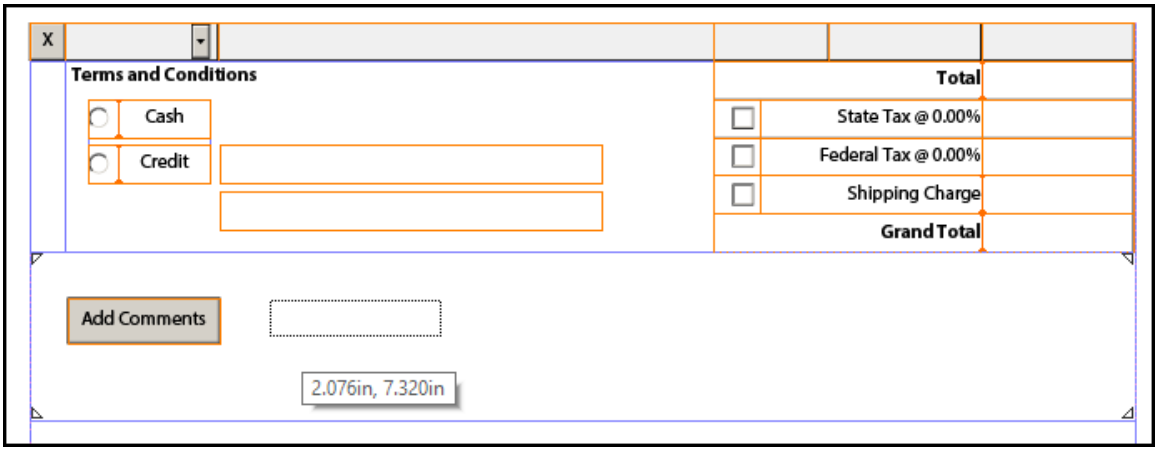
2. Select *purchaseOrder.xdp* and click **Open**.

3. Locate the *commentsHeader* subform; it is toward the bottom of the form.

4. Locate the **Button** object in the *Standard* tab of the *Object Library (see illustration).*



5. Drag and drop two new **Button** objects onto the *commentsHeader* subform next to the addComments button *(see illustration).*

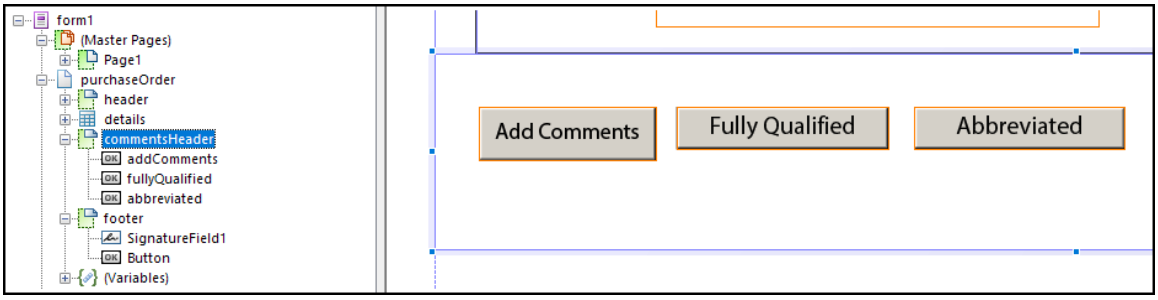


6. Name your first button **fullyQualified** and set its caption to **Fully Qualified**.

7. Name your second button **abbreviated** and set its caption to **Abbreviated**.

   Note: *It is very important for the examples in this section that you put this button in the commentsHeader subform.*

8. Your *commentsHeader* subform should now look like this *(see illustration).*



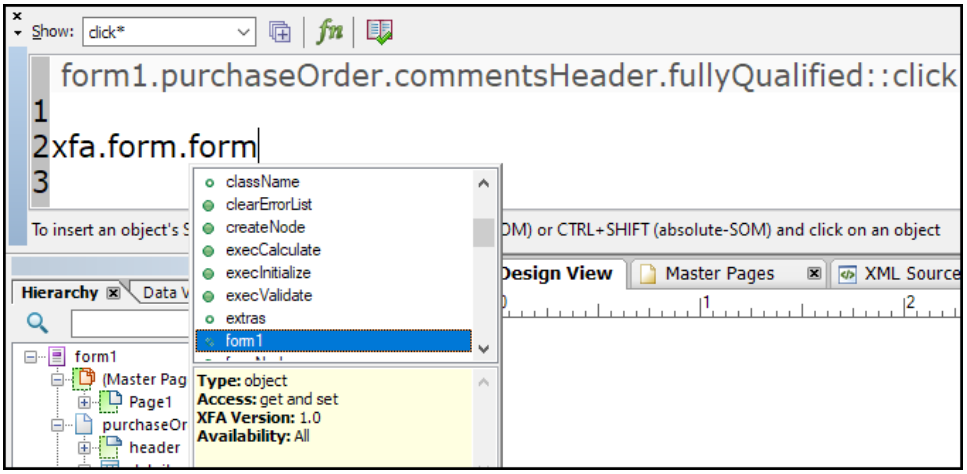
Fully Qualified References

Follow these steps to create a fully qualified object reference.

9. Select the **fullyQualified** button and open the Script Editor.

10. Select the click event.

11. Start typing **xfa.form.form1** and you will see the statement completion options window *(see illustration)*.



12. Although you cannot see the lower level nodes *(xfa.form)*, you can see all of the remaining nodes starting with *form1* node in the Hierarchy palette.

13. The statement completion options will guide you to this completed object reference.

xfa.form.form1.purchaseOrder.header.txtOrderedByCity

14. Complete the script so the value will be set to **Chicago**.

xfa.form.form1.purchaseOrder.header.txtOrderedByCity.rawValue = "Chicago";

Easy Way to Create Abbreviated References

Follow these steps to use Designer's automatic insert object reference feature.

15. Select the **abbreviated** button and open the Script Editor.

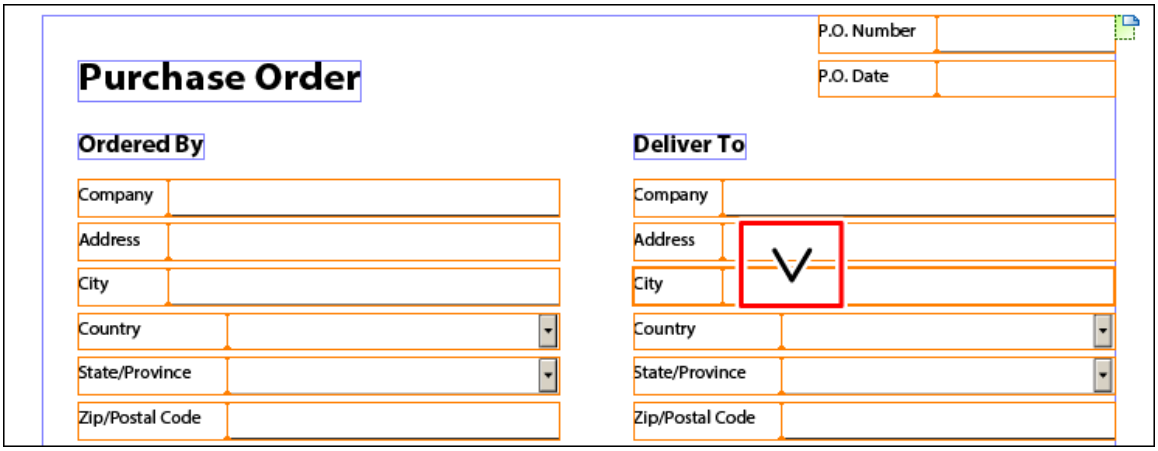
16. Select the **click** event.

17. Place your cursor in the Script Editor under the click event header.

18. Press and hold the **Ctrl key**.

19. Hover over your form in Design View. Your cursor will turn into a **V** *(see illustration)*.

20. Place the **V** over the City field *(see illustration)* in the Deliver To column of the Purchase Order header subform.



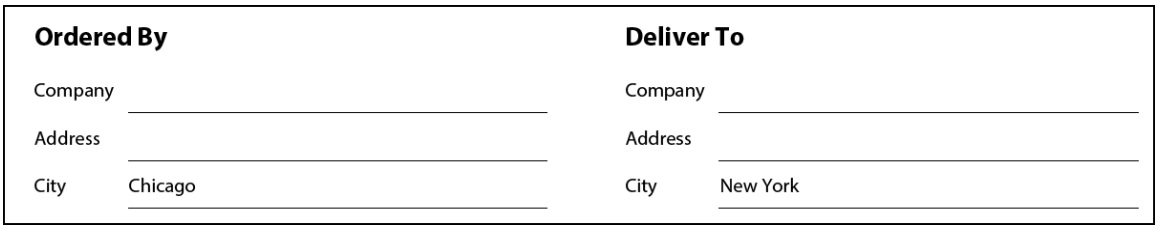
21. Click the field. A new object reference, relative to the selected object, will be created automatically in your Script Editor. It will look like this.

header.txtDeliverToCity

22. Complete the script so the value will be set to New York.

header.txtDeliverToCity.rawValue = "New York";

23. Select **Preview PDF**.
24. Click the **Fully Qualified** and the **Abbreviated** buttons.
25. You will see both fields populate *(see illustration)*.



Note: *The fully qualified path will work from any location on the form because it starts at xfa.form which is the root node of every XFA form.*

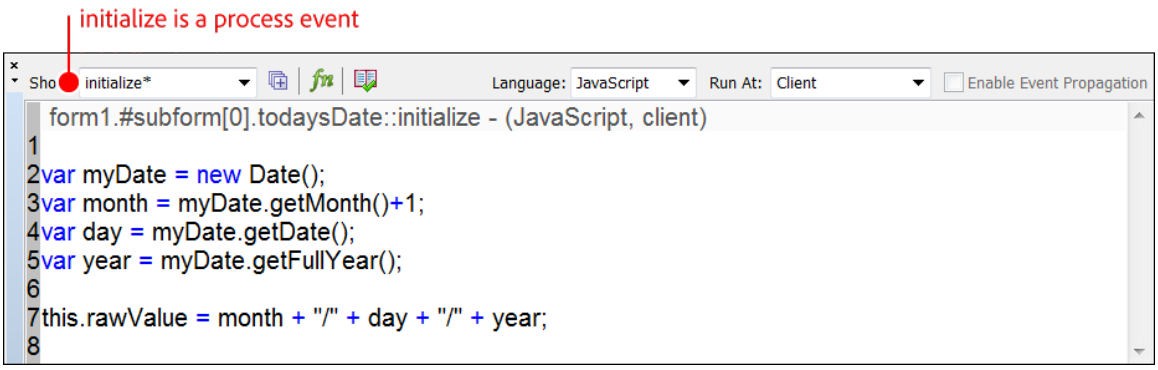
26. Select Design View to close the PDF Preview.

Work with Events

Follow these steps to see scripts in each of the three types of events.

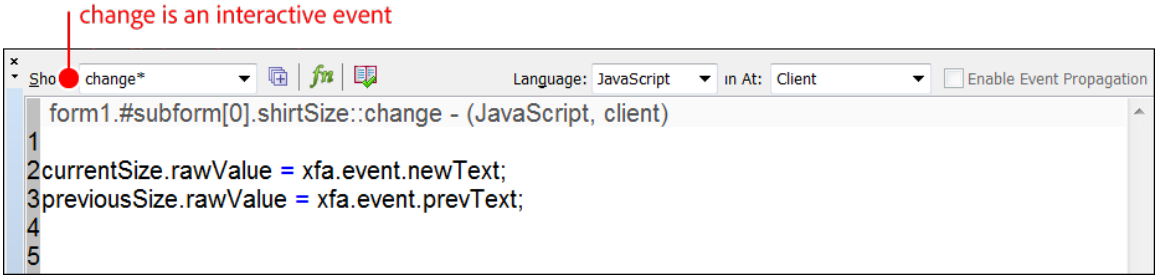
1. Select **File – Open** and navigate to your Student Files.
2. Select *events.xdp* and click **Open**.
3. Select the **todaysDate** field and open the **Script Editor**.
4. Expand the **Script Editor** by selecting the handle and dragging the window downwards.

5. Click the *Show Events* dropdown and select the **initialize** event. This is an example of a process event. The event will fire when the form initializes.

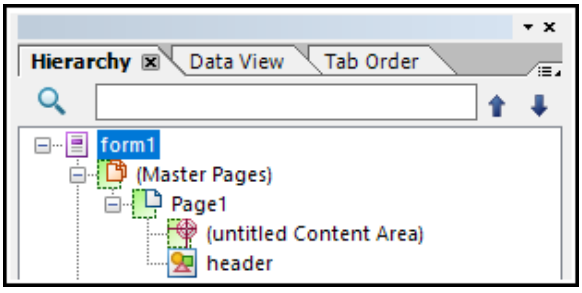


6. Select the **shirtSize** dropdown list.

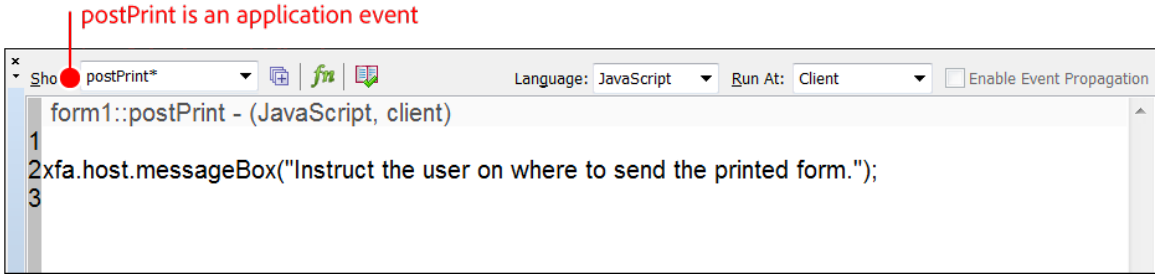
7. Click the *Show Events* dropdown and select the **change** event. This is an example of an interactive event.



8. Select the root node of the form *(form1 in the Hierarchy panel)*.



9. Click the *Show Events* dropdown and select the **postPrint** event. This is an example of an application event.



10. Select **Preview PDF** to see these scripts in action.

11. The form will open with the value of the Today's Date field set to the current date. This is because you used the **initialize** *process event*.

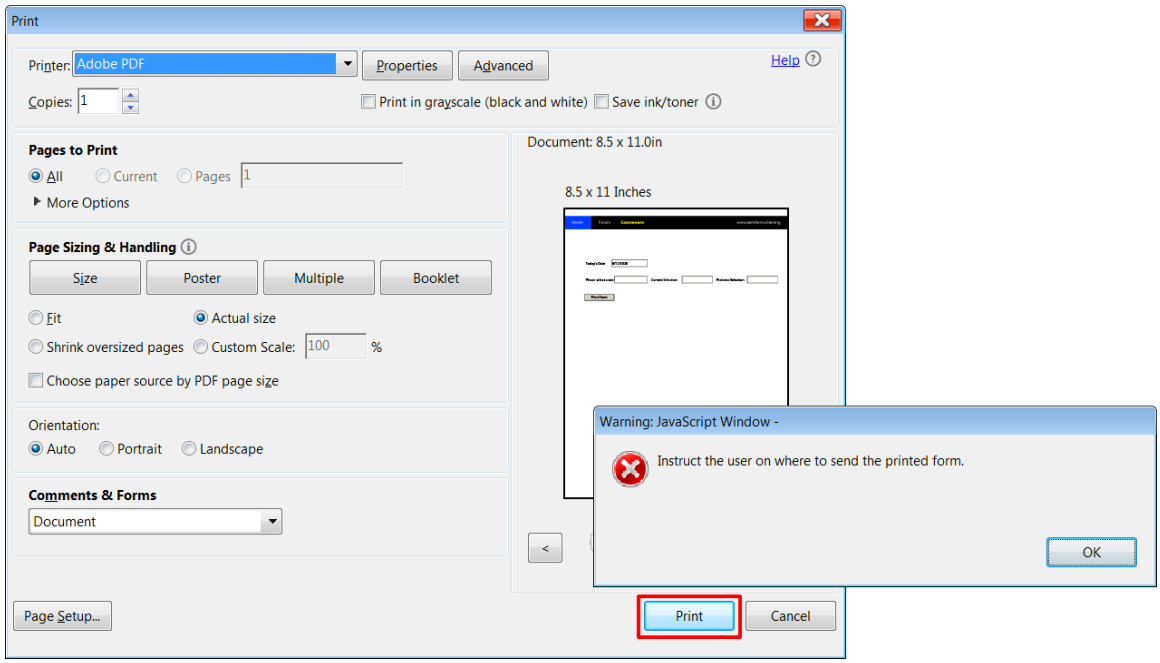
12. Select a **shirt size** in the dropdown list and the **Current Selection** field will populate with your selection.

13. Select a different size and the **Previous Selection** field will populate as well. Your script will be executed every time there is a change in the dropdown list.

14. Click the **Print Form** button to see an example of an application event.

15. Acrobat will display the Print dialog box.

16. If you click the **Print** button in the bottom right, Acrobat will print your file. Immediately after the print action, the XFA **postPrint** event will fire, and your script will run.



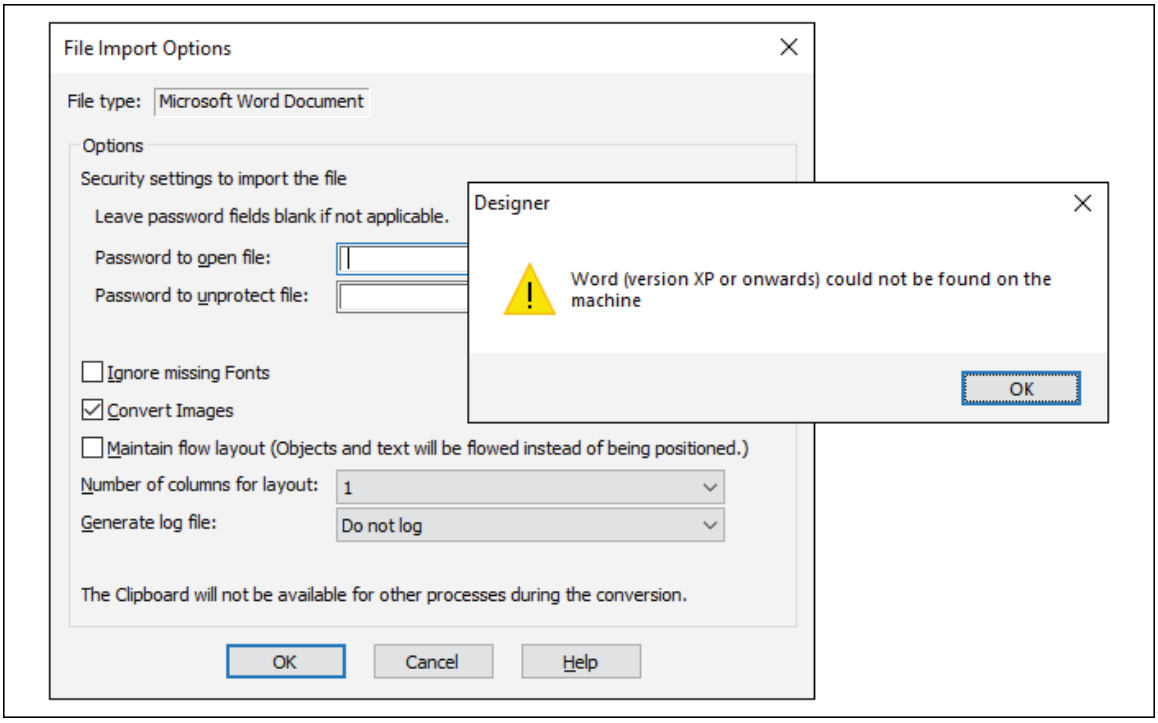
17. Select **Design View** to close the PDF preview.

Trouble Shooting

This section details some known issues and trouble-shooting steps.

Word (version XP or onwards) could not be found on this machine

This error occurs when importing a Microsoft Word document into AEM Forms Designer.



Designer needs to access Microsoft Word libraries to convert the Word file to XDP. In this case, Designer in unable to find Microsoft Word because it is not looking in the correct location stipulated in the Windows Registry.

In the past, we have solved this problem by making edits to the Windows Registry. However, this is NOT the ideal approach. Adobe now has a fix for this issue in AEM Forms Designer. You can download and run this Service & Cumulative Fix Pack:

Designer6.5.0_English_Cumulative_QF.msp

If you are unable to install this update, you can use the **ConvertedFromMSWordFile.xdp** in your Student Files. If you need to convert future MS Word files you will need to first create a PDF file from your Microsoft Word file and then convert the PDF file in Designer.

About this Courseware

SmartDoc Technologies supplied the official *Adobe-Certified* AEM Forms Training courseware to Adobe and Adobe's clients from 2016 – 2021. During that time, our SmartDoc Courseware was battle-tested by thousands of students worldwide. Our SmartDoc Technologies courseware has been peer-reviewed and certified by the Adobe *Engineering*, *Product*, and *Curriculum* teams and by thousands of students like you.

In addition to having the highest *quality* AEM Forms courseware, the SmartDoc library also has the highest *quantity* of AEM Forms courseware. You will find the perfect course for your specific AEM Forms needs in the SmartDoc library. You can always find a current listing of our Adobe AEM Forms training courses on our website.

www.smartdoctech.com