Table of Contents

Short ones this month

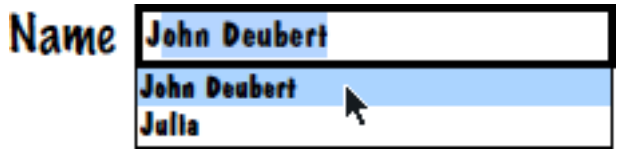
This month's articles are very short. I'm working on a book with a very close deadline. I'll tell you about the book in the next *Journal* issue; the software hasn't been announced yet.

PostScript folks: I know we're in the middle of a two-part series. The second part will come next time. Sorry.

– John

Acrobat Form Auto-Completion

Acrobat supports an Auto-Completion feature that can be very convenient when you are filling out a form. Acrobat records all of the text that you type into your form fields. Whenever you fill out another text field, Acrobat notes what you are typing and presents a drop-down menu of possible matches. You can use your keyboard's Down arrow or your mouse to select from this menu the entry you want.



Acrobat auto-completion has two modes:

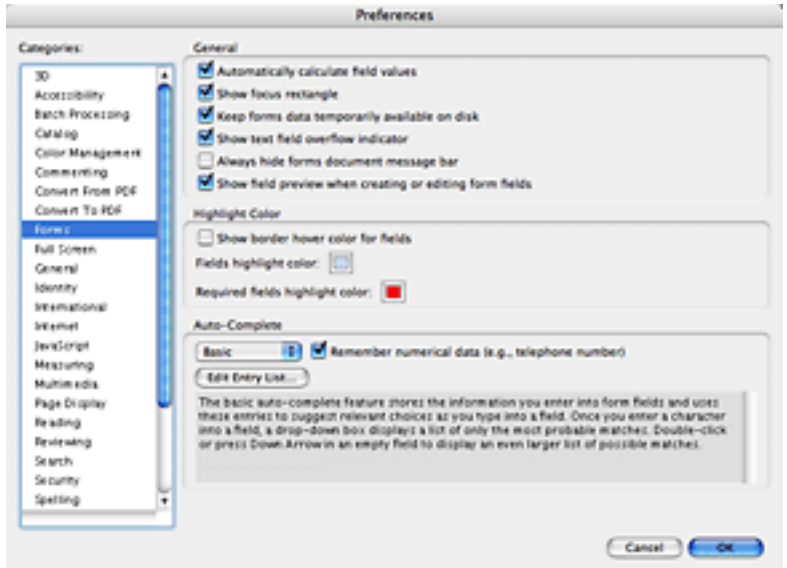
- *Basic mode,* which behaves exactly as I have described.
- *Advanced mode,* in which Acrobat automatically enters text directly into your form field if it decides one of its recorded entries matches what you are typing.

The feature defaults to Basic mode.

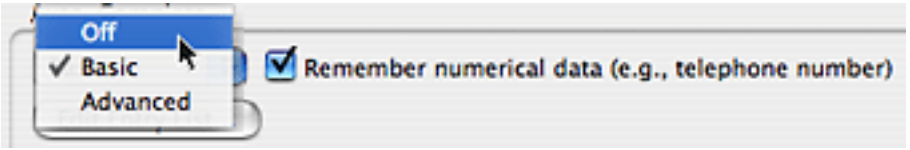
Next Page ->

Turning Auto-Completion On

Auto-completion can be turned on or off and set to basic or advanced mode through the Acrobat Preferences dialog box. The Forms preferences (at right) include an Auto-Completion pop-up menu that lets you select among Off, Basic, and Advanced (below).



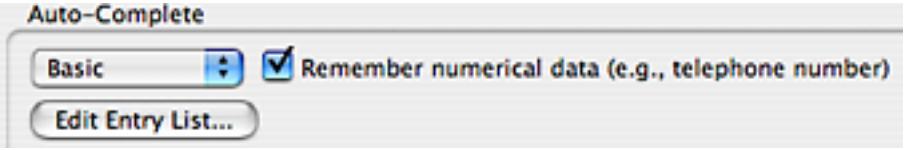
Note that there is also a checkbox that specifies whether Acrobat should record text that consists only of numbers. This is a good idea if you want it to be able to auto-enter telephone numbers; it's a bad idea if you *don't* want it to auto-enter your credit card numbers.

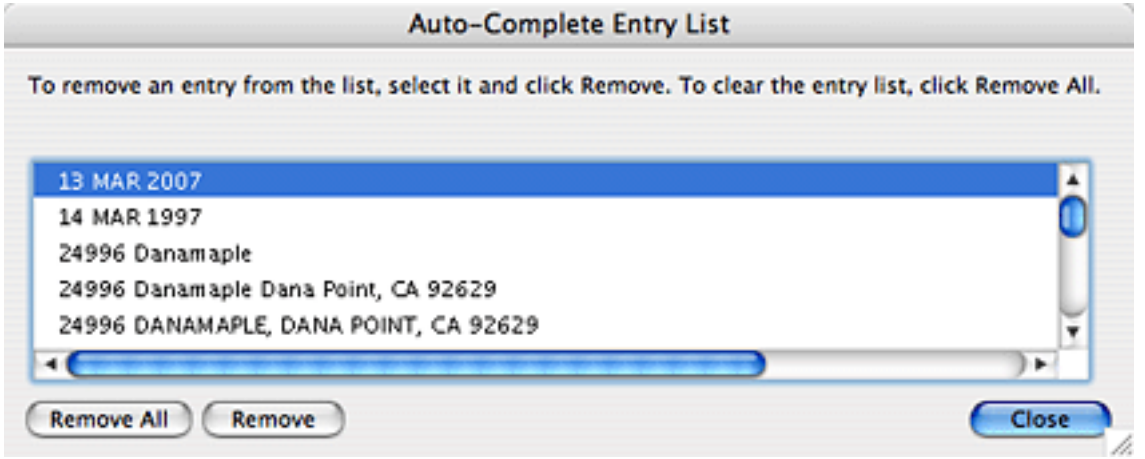


Next Page ->

Removing Auto-Completion Entries

You can remove entries from Acrobat's list of auto-completion text. Click the *Edit Entry List* button and Acrobat will present you with the dialog box pictured below.





Select the entry that you want to delete and click the *Remove* button. Acrobat will delete the entry from the list. Note that if you use that same text in a form, Acrobat will record it again.

Next Page ->

Good for Frequent Form Users

The Auto-Completion feature is very useful for people who fill out a lot of Acrobat forms. The only warning I can think of is to periodically review the list of Auto-Completion entries, because Acrobat keeps them indefinitely and, I believe, across versions of Acrobat. In the illustration on the first page of this article, you can see Acrobat is offering "Julia" as a suggestion for the text field; Julia is a colleague whom I have not seen since 1999, back in the Acrobat 5 era.

Return to Main Menu

Making Styled Text With *makefont*

You may remember the *makefont* operator from your PostScript Foundations or PostScript for Support Engineers class. This alternative to *scalefont* takes as its arguments a font dictionary and an array of six numbers (*i.e.*, a PostScript transformation matrix); it returns a font dictionary representing the original font transformed by the matrix:

```
<<fdict>> [ a b c d e f ] makefont  =>  <<fdict'>>
```

In this short article, we're going to look at how to use *makefont* to create a variety of types of styled text: condensed, $_{\text{sub-}}$ and $^{\text{super}}$script, and *obliqued*.

A Brief Review

The six-element array of numbers we hand to *makefont* is a transformation matrix. A review of transformation matrices is far beyond what we can do in this article, but the net effect each of the six numbers has on the resulting font is easily listed:

a, d — These are the horizontal and vertical scaling multipliers that should be applied to the font.

e, f — These are the offsets that should be applied to position of the characters relative to the current point.

b, c — These skew the *x* and *y* axes of the font's characters.

We shall discuss these in more detail as we create our styled text.

Next Page ->

Acumen Journal: PostScript Tech 6

Condensed Text

Condensed text has had a scale applied to the character shapes in the horizontal direction. This is easily done by choosing suitable values for *a* and *d* in a call to *makefont*.

For example,

```
/Helvetica findfont [ 10 0 0 12 0 0 ] makefont setfont
```

The above line of code sets the current font to a version of Helvetica that is as wide as a 10 point font, but as tall as a 12 point font. That is, our current font is a condensed 12-point Helvetica, as at right.

This is 12-pt Helvetica

This is 12-pt Helvetica-Condensed

Condensing the current font

You can do this a bit more generically by using the current font, rather than getting a fresh, 1-point font with *findfont*. In this case, the scale values *a* and *b* are interpreted as multiples of the current point size:

```
currentfont [ .85 0 0 1 0 0 ] makefont setfont
```

The above sets the current font to a version of the previous current font scaled to 85% of its original width. You would probably want to use *gsave* and *grestore* so you can revert to your original font.

Next Page ->

Acumen Journal: PostScript Tech 7

Superscript & Subscript

The *f* value in *makefont* is the key to creating a superscript or subscript font. This value offsets the text characters vertically, positive values raising the text above the baseline, negative values dropping them below. Thus,

```
/Helvetica findfont [ 12 0 0 12 0 6 ] makefont setfont
```

The above code will set the current font to a 12-point Helvetica that prints 6 points above the baseline. Again, we can generically make a superscript version of the current font as follows:

```
currentfont [ 1 0 0 1 0 3 ] makefont setfont
```

This changes the current font so that text prints 3 points above the baseline. Of course, superscript text is often reduced in size, so a better version might be:

```
currentfont [ .7 0 0 .7 0 3 ] makefont setfont
```

Not surprisingly, you get subscript text by using a negative value for *f*.

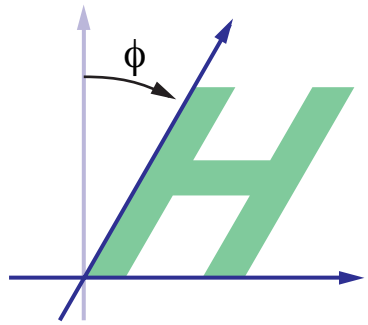
```
currentfont [ .7 0 0 .7 0 -3 ] makefont setfont
```

Next Page ->>

Oblique Fonts

This is 12-pt Helvetica-PseudoOblique

An oblique font has vertical strokes that are not actually vertical, as in the example at right. It is always best to use the actual oblique version of a font (for example, the official Helvetica-Oblique). In a pinch, however, you can create your own oblique font using the *c* value in the *makefont* array. (Professional typographers should skip this page; according to individual temperaments, they find the results of the following technique to be painful, ugly, or enraging. It generally looks acceptable to me, if not overdone.)



The *c* value in the *makefont* array skews the font's vertical axis by some angle, ϕ in the diagram at right. It would be nice if *c* were the angle in degrees, but it's not; instead, here's the calculation:

$$c = d \times \tan(\phi)$$

where *d* is the vertical point size. Thus, to create a 12-point font skewed 8°, you would do the following:

```
/Helvetica findfont [ 12 0 1.69 12 0 0 ] makefont setfont
```

where 1.69 is 12tan(8°).

Again, working with the current font, this would be:

```
currentfont [ 1 0 .141 1 0 0 ] makefont setfont
```

The resulting Helvetica appears in the box at the top of this page.

Next Page ->

Acumen Journal: PostScript Tech 9

A Style Procedure

You might find it convenient to define procedures that implement these styles. For example, here is a definition and use of a pair of procedures—*BeginOblique* and *EndOblique*—that wrap a *makefont* call.

```
/BeginOblique
{    gsave
     currentfont                          % Get the current font
     [ 1  0  .141  1  0  0 ] makefont     % Oblique it
     setfont                              % Make it the current font
} bind def

/EndOblique
{    currentpoint            % Push the current position on the stack
     grestore                % Do a grestore
     moveto                  % Restore the current point
} bind def

/Helvetica 20 selectfont     % Now we use our new procedures
72 600 moveto
(Here we have some ) show
BeginOblique
(oblique) show
EndOblique
( text) show
```

Here we have some *oblique* text

I'll leave a detailed examination of the example as an Exercise for the Reader.

Return to Main Menu

Schedule of Classes, September-November 2006

Following are the dates of Acumen Training's upcoming PostScript and PDF Technical classes. Clicking on a class name below will take you to the description of that class on the Acumen training website.

These classes are taught in Orange County, California and on corporate sites world-wide. See the Acumen Training web site for more information.

Technical Classes

	PDF File Content and Structure 1	Sept 19–22		Sept 18–21
NEW!	PDF File Content and Structure 2		Oct 30–Nov 2	
	PostScript Foundations		Oct 16–20	Sept 4–8
	Variable Data PostScript		Oct 2–6	
	Advanced PostScript			
	PostScript for Support Engineers			Nov 28–Dec 1

Course Fee The PostScript and PDF classes cost $2,000 per student.

Registration Info

Acrobat Class Schedule

Regretfully, I have suspended teaching Acrobat classes.

They'll be back.

Return to Main Menu

Contacting John Deubert at Acumen Training

For more information

For class descriptions, on-site arrangements or any other information about Acumen's classes:

Web site: http://www.acumentraining.com **email:** john@acumentraining.com

telephone: 949-248-1241

mail: 24996 Danamaple, Dana Point, CA 92629

Registering for Classes

To register for an Acumen Training class, contact John any of the following ways:

Register On-line: http://www.acumentraining.com/registration.html

email: registration@acumentraining.com

telephone: 949-248-1241

mail: 24996 Danamaple, Dana Point, CA 92629

Back issues

All issues of the *Acumen Journal* are available at the Acumen Training website:
http://www.acumenjournal.com/AcumenJournal.html

Return to First Page

What's New at Acumen Training?

Working on a New Book

This month's Acumen Journal is short because I'm writing a new book for Peachpit Press. I can't disclose the topic, since it involves something-or-other that hasn't been announced yet.

Tell you all about it next time.

Return to First Page

Journal Feedback

If you have any comments regarding the *Acumen Journal,* please let me know. In particular, I am looking for three types of information:

Comments on usefulness. Does the Journal provide you with worthwhile information? Was it well written and understandable? Do you like it, hate it? Did it make you reflect on the brevity and futility of life?

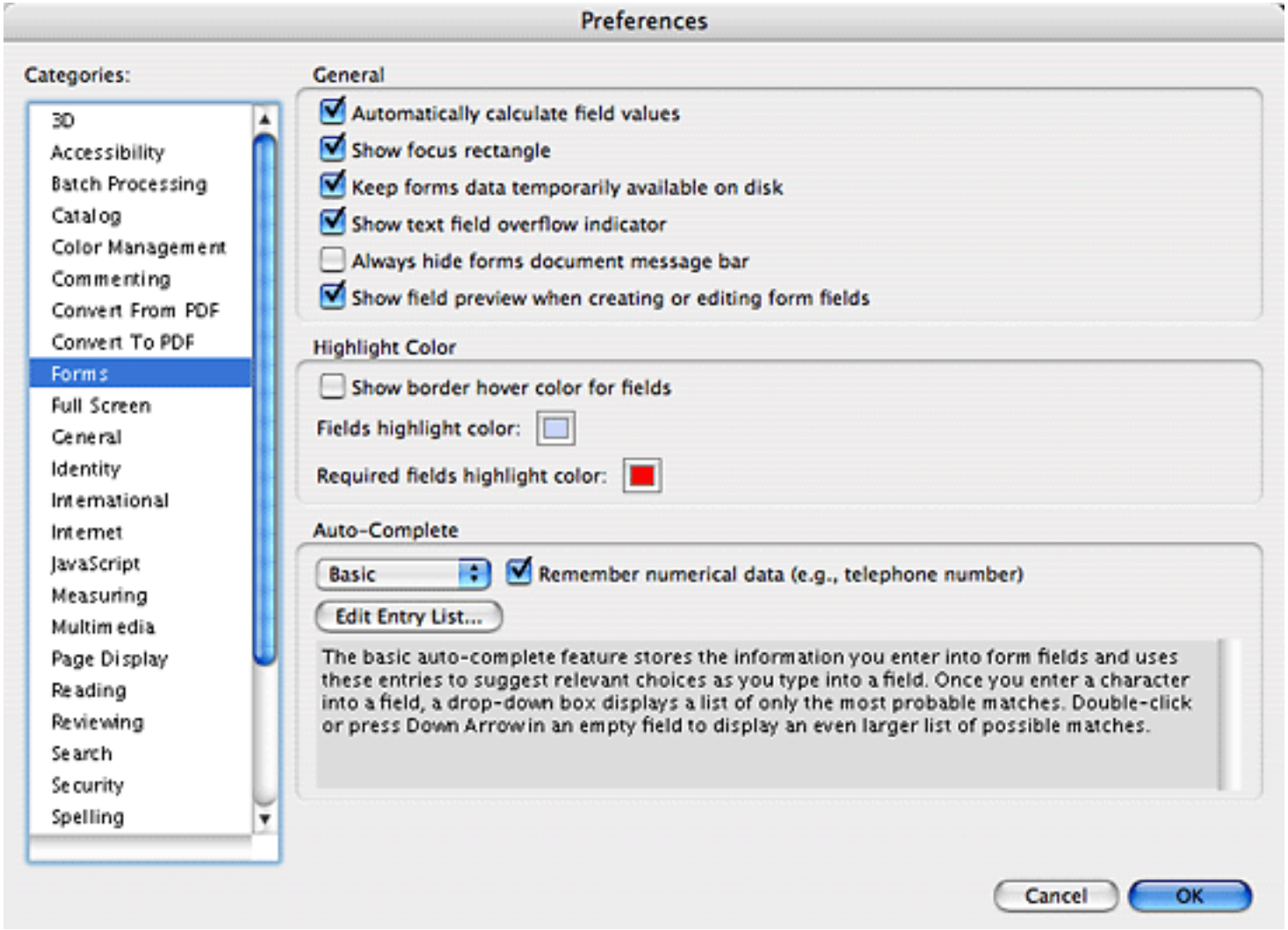
Suggestions for articles. Each Journal issue contains one article each on PostScript and Acrobat. What topics would you like me to write about?

Questions and Answers. Do you have any questions about Acrobat, PDF, or PostScript? Feel free to email me about. I'll answer your question if I can. (If enough people ask the same question, I can turn it into a Journal article.)

Please send any comments, questions, or problems to:

journal@acumentraining.com

Return to Menu


Preferences
Categories:
3D
Accessibility
Batch Processing
Catalog
Color Management
Commenting
Convert From PDF
Convert To PDF
Forms
Full Screen
General
Identity
International
Internet
JavaScript
Measuring
Multimedia
Page Display
Reading
Reviewing
Search
Security
Spelling
General
Automatically calculate field values
Show focus rectangle
Keep forms data temporarily available on disk
Show text field overflow indicator
Always hide forms document message bar
Show field preview when creating or editing form fields
Highlight Color
Show border hover color for fields
Fields highlight color:
Required fields highlight color:
Auto-Complete
Basic
Remember numerical data (e.g., telephone number)
Edit Entry List...
The basic auto-complete feature stores the information you enter into form fields and uses these entries to suggest relevant choices as you type into a field. Once you enter a character into a field, a drop-down box displays a list of only the most probable matches. Double-click or press Down Arrow in an empty field to display an even larger list of possible matches.
Cancel
OK