AcroTEX.Net

# The grayhints Package

D. P. Story


Prepared: November 1, 2018

www.acrotex.net
Version 1.2, 2018/26/04

The grayhints Package

# Table of Contents

## 1. Introduction

We often see in HTML pages or in compiled executable applications, form fields (text fields, input fields) that require user input. The untouched field has text within it informing the user of the nature of the data to be entered into the field. This "grayed hint" immediately disappears when the user focuses the cursor on the field.[1] We illustrate the concept with an example or two.

First Name

yyyy/mm/dd Reset

Of course, the usual tool tips may also be provided.

It is not natural for Adobe form fields to do this, it takes some support code for it to work properly; scripts for the Keystroke, Format, OnFocus, OnBlur, and Calculate events are needed.

The grayhints package works as designed for Adobe Reader and PDF-XChange Editor.[2] All LaTeX workflows for PDF creation are also supported.

## 2. Package options

Without passing any options, the eforms package of AeB, dated 2017/02/27, is required; the document JavaScript function `AllowCalc()` and modified Adobe built-in functions are automatically embedded in the document. There are options, however, to modify this default setup.

`usehyforms` By default, this package requires eforms, dated 2017/02/27; however, if you are more comfortable using the form fields of hyperref, specify the option `usehyforms`.[3] When `usehyforms` is specified, insdljs dated 2017/03/02 or later is required.

`nocalcs` If this option is taken, the document JavaScript function `AllowCalc()` is not embedded in the document. The implications are that you are not using any calculation fields.

`nodljs` When this option is specified, there are no requirements placed on this package; that is, neither eforms nor insdljs are required. Use this option is you are not using any of the Adobe built-in formatting functions.

**Demo files:** `gh-eforms.tex, gh-hyperref.tex`. The latter file uses the `usehyforms` option (and hyperref form fields), while the former uses the eforms package. The demo files `gh-fmt-eforms.tex, gh-fmt-hyperref.tex` provided additional examples of the use of Adobe's built-in formatting functions.

[1] Focus on the field by clicking inside the form field.

[2] PDF-XChange Viewer, which is no longer supported by Tracker Software Products, will display the gray hints, but some of the needed JavaScript are not supported.

[3] eforms and hyperref form fields can be used in one document.

## 3. Creating a form field with a gray hint

In this documentation, we use eforms form fields to illustrate concepts, the demonstration file gh-hyperref.tex and gh-fmt-hyperref.tex has the form field markup for the case of hyperref forms.

There are two cases: (1) an ordinary variable text form field (this includes text fields and editable combo boxes) with no calculate script; (2) same as (1), but the field has a calculate script.

### 3.1. Variable text field, no calculate script

When there is no calculate script, to obtain a gray hint, it is necessary to supply scripts for the Format, Keystroke, OnFocus, and OnBlur events. The scripts are all defined in the grayhints package. In addition, the color of the text in the text field must be appropriate. We illustrate,

```
1 \textField[\TU{Enter your first name so I can get to know you better}
2     \textColor{\matchGray}\AA{%
3     \AAKeystroke{\KeyToGray}
4     \AAFormat{\FmtToGray{First Name}}
5     \AAOnFocus{\JS{\FocusToBlack}}
6     \AAOnBlur{\JS{\BlurToBlack}}
7     \AACalculate{\CalcToGray} % required if PDF-XChange Editor is used
8 }]{NameFirst}{2in}{11bp}
```

By default, the text color is black and the grayed hint text is light gray. The tool tip (\TU) is grayed out, as it is optional. In line (2) we match the color for the text to the gray color using the command \matchGray of grayhints. Within the argument of \AA, the \AAFormat, \AAKeystroke, \AAOnFocus, and \AAOnBlur scripts are inserted.

Keystroke Script: In line (3), \KeyToGray is placed within the argument of \AAKeystroke. This script changes the color of the text to gray when the field is empty.

Format Script: The script snippet \FmtToGray takes a single argument, which is the text of the hint. In line (4) the hint is 'First Name'.

OnFocus Script: The code snippet \FocusToBlack is inserted into the argument of \OnFocus, as seen in line (5). When the field comes into focus, this script changes the color to the normal color (usually black).

OnBlur Script: In line (6), the \BlurToBlack script is placed within the argument of \OnBlur, in the manner indicated. When the field loses focus (blurred), the script changes the color of text to gray if the field is empty or to its normal color (usually black), otherwise.

Calculate Script: In line (7), the Calculate event resets the color to gray. This is needed when the user presses the Enter key rather than exiting the field by tabbing out, or clicking or tapping an area outside the field. This script is required when (1) there is a possibility that PDF-XChange Editor is used; or (2) a "totals" calculation field.

The hyperref form field counterpart to the above example is,

```
\TextField[name={NameFirst},
    height=11bp,width=2in,charsize=9bp,
    color={\matchGray},
    keystroke={\KeyToGray},
    format={\FmtToGray{First Name}},
    onfocus={\FocusToBlack},
    onblur={\BlurToBlack},
    calculate={\CalcToGray}   % required if PDF-XChange Editor is used
]{}
```

The two fields appear side-by-side:

First Name First Name Reset

Both fields appear in their 'default' appearance.

### 3.2. Variable text field, with calculate script

If you want to make calculations based on entries in other fields, you will need the code snippet \CalcToGray as part of your calculate script.

```
\textField[\TU{The total for first and second integers}
    \textColor{\matchGray}\AA{%
    \AAKeystroke{AFNumber_Keystroke(0,1,0,0,"",true);\r\KeyToGray}
    \AAFormat{AFNumber_Format(0,1,0,0,"",true);\r\FmtToGray{Total}}
    \AACalculate{var cArray=new Array("Integer");\r
        if(AllowCalc(cArray))AFSimple_Calculate("SUM", cArray );\r
        \CalcToGray}
    \AAOnFocus{\JS{\FocusToBlack}}
    \AAOnBlur{\JS{\BlurToBlack}}
}]{TotalNumbers}{1in}{11bp}
```

The use of \r is optional, the author uses this to format the script within the user-interface of Acrobat (or PDF-XChange Editor).[4] The \textColor (line (2)), \AAOnFocus (line (8)), and \AAOnBlur (line (9)) are the same as earlier presented. Several comments are needed for the \AAKeystroke, \AAFormat and \AACalculate lines. The \AACalculate event above also shows, in general, how to integrate the gray hint methodology with other scripts; as a general rule, the gray hints commands should come last.

- This is a number field, so we use the built-in functions AFNumber_Keystroke and AFNumber_Format provided by the Adobe Acrobat and Adobe Acrobat Reader distributions. In lines (3) and (4), the \KeyToGray and \FmtToGray code snippets follow the built-ins.[5]

[4] The helper commands \r and \t are defined in eforms; if eforms is not loaded, use \jsR and \jsT instead.

[5] As a general rule, the code snippets \KeyToGray, \FmtToGray, and \CalcToGray should inserted after any built-in functions.

- For the Calculate event, special techniques are used. We define an array `cArray` (line (5)) consisting of the names of all the dependent fields we use to calculate the value of this field. In line (6), we make the calculation (`AFSimple_Calculate`) only if the document JavaScript function `AllowCalc(cArray)` returns true. The function returns true only if at least one of the fields is not empty. Following the calculation comes the code snippet `\CalcToGray`; this changes the text color to gray if the field is empty and to the normal color (usually black) otherwise.

  The function `AllowCalc()` is defined for all options except for the `nodljs` option.

Let's go to the examples. Build three fields (four actually), in the first two enter integers, the other two fields compute their sum.

① First Integer

② Second Integer

③ Total

④ 0 Reset

Enter numbers into the first two text fields (① and ②), the totals of these two fields appear in the last two fields (③ and ④). Total field ③ uses the recommended script `if(AllowCalc(cArray)` (see line (6) above), whereas field ④ does not. Initially, they both behave the same way until you press the reset button. For field ③ the gray hint appears, for field ④ the number zero (0) appears. This is because the calculation was allowed to go forward, and the calculated value is zero even through none of the dependent fields have a value. If you want the gray hint in the total field, you must use the conditional `if(AllowCalc(cArray)`.[6]

### 3.3. Changing the colors for gray hints

For the fields in which the gray hint scripts are used, there are two colors that are relevant, the normal color (defaults to black) and the gray color (defaults to light gray). The command `\normalGrayColors{⟨normalcolor⟩}{⟨graycolor⟩}` sets this pair of colors. The arguments for `\normalGrayColors` are JavaScript colors; they may be in any of the following four forms: (1) a JavaScript color array `["RGB",1,0,0]`; (2) a predefined JavaScript color, such as `color.red`; (3) a declared (or named) LaTeX color such as `red`; or (4) a non-declared LaTeX color such as `[rgb]{1,0,0}`. If the package `xcolor` is not loaded, only methods (1) and (2) are supported.

The package default is `\normalGrayColors{color.black}{color.ltGray}`. The predefined JavaScript colors are,

[6] Hence, don't use the `nodljs` option.

| Color Models | | |
|---|---|---|
| GRAY | RGB | CMYK |
| color.black | color.red | color.cyan |
| color.white | color.green | color.magenta |
| color.dkGray | color.blue | |
| color.gray | | |
| color.ltGray | | |

All these colors are defined in the LaTeX color packages, except for possibly dkGray, gray, and ltGray. These three are defined in grayhints.

We repeat the 'First Name' example with different declared colors. We begin by declaring,

```
\normalGrayColors{blue}{magenta}
```

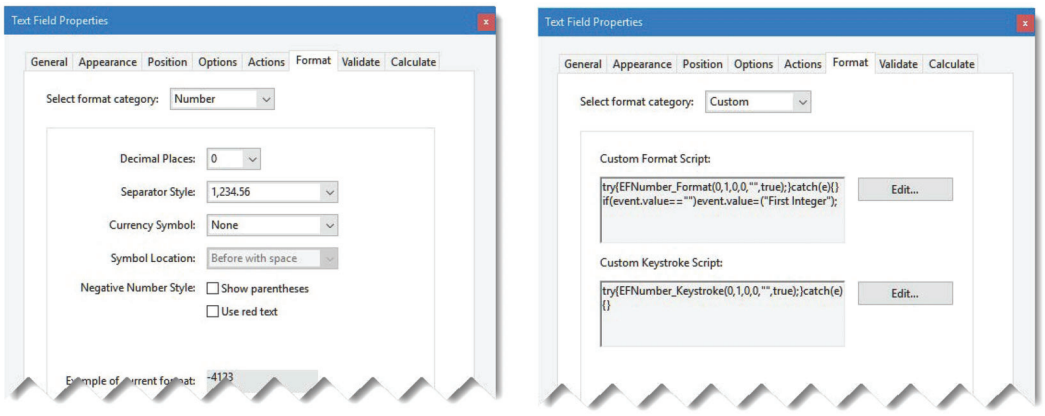
then build a 'gray hinted' field,

First Name Reset

## 3.4. Remarks on using Adobe built-in formatting functions

The grayhints package (as well as does insdljs) offers alternating naming of the Adobe built-in formatting functions. As a general rule, all Adobe built-in format, validate, and calculation functions that begin with 'AF' are given alternate names that begin with 'EF'. More specifically, Table 1 lists the Adobe built-in formatting functions and their alternative names. The purpose of these alternate names is to allow the JavaScript developer to access the scripts through the Acrobat user-interface.

You can learn more about the 'AF' versions and their arguments at the AcroTeX Blog web site; in particular, carefully read the article http://www.acrotex.net/blog/?p=218.




(a) Using AFNumber_Format (b) Using EFNumber_Format

Figure 1: Format tab: 'AF' versus 'EF' functions

| Adobe function name | Alternate function name | LaTeX command[7] |
|---|---|---|
| AFNumber_Keystroke | EFNumber_Keystroke | \NumKey |
| AFNumber_Format | EFNumber_Format | \NumFmt |
| AFPercent_Keystroke | EFPercent_Keystroke | \PercentKey |
| AFPercent_Format | EFPercent_Format | \PercentFmt |
| AFDate_Keystroke | EFDate_Keystroke | \DateKey |
| AFDate_Format | EFDate_Format | \DateFmt |
| AFDate_KeystrokeEx | EFDate_KeystrokeEx | \DateKeyEx |
| AFDate_FormatEx | EFDate_FormatEx | \DateFmtEx |
| AFTime_Keystroke | EFTime_Keystroke | \TimeKey |
| AFTime_Format | EFTime_Format | \TimeFmt |
| AFTime_FormatEx | EFTime_FormatEx | \TimeFmtEx |
| AFSpecial_Keystroke | EFSpecial_Keystroke | \SpecialKey |
| AFSpecial_Format | EFSpecial_Format | \SpecialFmt |
| AFSpecial_KeystrokeEx | EFSpecial_KeystrokeEx | \SpecialKeyEx |
| AFRange_Validate | EFRange_Validate | \RangeValidate |
| AFSimple_Calculate | EFSimple_Calculate | \SimpleCalc |
| AFMergeChange | EFMergeChange | \MergeChange |

Table 1: Built-in formatting commands

Figure 1 shows the impact of using the 'EF' functions. On the left, `AFNumber_Format` is used to format a number field that uses gray hints using the code

```
AFNumber_Format(0,1,0,0,"",true)\r\FmtToGray
```

As can be seen in sub-figure (a), or more accurately not seen, the code is not seen through the user-interface of Acrobat. In sub-figure (b) the underlying code is seen (and therefore editable through the user-interface) because the 'EF' version of the function was used:

```
\try{EFNumber_Format(0,1,0,0,"",true)}catch(e){}\r\FmtToGray
```

Note this code is wrapped in a `try/catch` construct; this is optional. The insdljs package defines a helper command `\dlTC` to do the wrapping for you:

```
\dlTC{EFNumber_Format(0,1,0,0,"",true)}\r\FmtToGray
```

When using pdflatex or xelatex, `try/catch` appears not to be needed, but when Adobe Distiller is used, Acrobat throws an exception when the file is first created. The `try/catch` suppresses (catches) the exception.

## 4. Extending the functionality of grayhints

The 'gray hints' technique works well with fields that require special formatting such as the ones provided by built-in formats of Adobe Acrobat, these formats are Number,

[7] These commands are explained in Section 4

Percentage, Date, Time, Special, and Custom. How 'gray hints' functions when one of these special formats is used may be acceptable, but if not, we offer an alternative. We begin by presenting two date text fields. The one on the left is the default, the one on the right is 'enhanced.'[8]

yyyy/mm/dd

```
1  \textField[\textColor{\matchGray}
2    \TU{Enter a date of your choosing}\AA{%
3    \AAOnFocus{\JS{\FocusToBlack}}
4  % Using the Adobe Built-in functions directly
5    \AAKeystroke{%
6      AFDate_KeystrokeEx("yyyy/mm/dd");\r
7      \KeyToGray}
8    \AAFormat{AFDate_FormatEx("yyyy/mm/dd");\r
9      \FmtToGray{yyyy/mm/dd}}
10   \AAOnBlur{\JS{\BlurToBlack}}
11   \AACalculate{\CalcToGray}
12 }]{Datefield1}{1in}{11bp}
```

Figure (a): Using Adobe built-in functions

yyyy/mm/dd

```
1  \textField[\textColor{\matchGray}
2    \TU{Enter a date of your choosing}\AA{%
3    \AAOnFocus{\JS{\FocusToBlack}}
4  % Using a customized version of Adobe built-in
5  % functions, with LaTeX access
6    \AAKeystroke{\DateKeyEx("yyyy/mm/dd");\r
7      \KeyToGray}
8    \AAFormat{\DateFmtEx("yyyy/mm/dd");\r
9      \FmtToGray{yyyy/mm/dd}}
10   \AAOnBlur{\JS{\BlurToBlack}}
11   \AACalculate{\CalcToGray}
12 }]{Datefield2}{1in}{11bp}
```

Figure (b): Using LaTeX format commands

Try entering a bogus date, such as the number '17'. The two fields operate identically when you then commit your date by clicking your mouse outside the fields. However, the behavior of these two fields differ when you commit your bogus date by pressing the Enter key. For Figure (a), the template 'yyyy/dd/mm' appears in black, whereas, for Figure (b), the phrase 'continue editing' appears in black. The latter is the 'enhanced' behavior of the fields that use the gray hints code.

**Remarks on the LaTeX format commands.** The difference in the two code snippets (Figures (a) and (b)) is seen in lines (6) and (8) of Figure (b). In these two lines we use the special `\DateKeyEx` and `\DateFmtEx` commands that ultimately call the Adobe built-in functions, but when these special commands are used, we can better manage what happens when the Enter key is pressed. The document author is strongly encouraged to use the LaTeX commands in the last column on the right of Table 1, instead of using the Adobe built-ins, as is done in Figure (a), or their 'EF' counterparts.[9]

The `grayhints` now defines `\EnterCommitFailEvent` and `\CommitSuccessEvent` to customize the user experience when Enter key is pressed (as opposed to leaving the field by clicking in white space, or tabbing away from the field). See the sample files `gh-fmts-eforms` for instructions to use these two commands. We present a final example, where the 'Enter' events have been changed.

*gh-fmts-eforms referenced*

yyyy/mm/dd (enter '17' then press enter)

This example appears in the `gh-fmts-eforms.tex` sample file.

## 5. My retirement

Now, I simply must get back to it. ĐŞ

[8] The example that follows is taken from `gh-fmts-forms.tex` where you will find additional discussion.

[9] Some of these LaTeX commands use the 'AF' built-in (mostly for the formate events) while others use the 'EF' versions of the built-ins.