

Retract text from a PDF with DynaPDF

If you like to remove sensitive text from a PDF file in FileMaker, you can use the [DynaPDF](#) functions. We have parser functions to find text on a page, delete the text and we can draw a black bar on top of the word.

Let's go through the script part by part. First you need to initialise DynaPDF once. This can happen early in the start script of your solution or later in the first script needing DynaPDF.

Next we create a new PDF environment and open an existing PDF from a container. Then we import the first page to edit it. You could of course use [DynaPDF.ImportPDFFile](#) instead to import the whole PDF file and then later make a loop over the pages.

Once the page is loaded, we can create a parser object. When we write text with the parser, we need to provide an alternative font in case the font used for the text is not available. Fonts may be embedded in the PDF with font subsetting. That means only the used characters are in the font definition and if you replace text, you may not have all characters available.

To later draw rectangles, we need to collect all the coordinates. We use our [QuickList](#) functions and just append new values and later pop them back. The QuickList can store a list of values in each list element. Basically this allows you to have a plugin managed list of values or even a list of lists.

Use the [DynaPDF.Parser.ParsePage](#) function to start parsing a page. We pass the text selection flag, so we can query the selection bounding box for the found text. This way we can use the selection to call [DynaPDF.Parser.DeleteText](#) right in the loop to delete the text and use the QuickList for the rectangle drawing later.

Whenever you modify a page with the parser, please use [DynaPDF.Parser.WriteToPage](#) to write the changes back to the page. You can pass optimisation flags for the parser and the write function, so you can do some changes like our [DynaPDF.Optimize](#) function. For example to have DynaPDF delete alternate images.

If we found some coordinates, we can edit the page, loop over the coordinates and draw rectangles. There are a few special cases to handle. For example the page may have a crop box. We ask DynaPDF to use visible coordinates, so it automatically adjusts for a crop box offset. And then the page may be rotated, but we can just rotate it up right, so we can just draw into the page with correct coordinates.

When finished, we store the PDF in a container field and free both the QuickList and the DynaPDF environment.

Please try the script. We'll include it in the Find and Replace Text.fmp12 example file for next version. The example database has multiple scripts for various operations.

Below the whole script:

Find Text and remove it and draw rectangle in file Find and Replace Text

```
# Find text and draw rectangles on their position
#
# Initialize DynaPDF if needed
If [ MBS("DynaPDF.IsInitialized") ≠ 1 ]
    Perform Script [ "InitDynaPDF" ; Specified: From list ; Parameter:  ]
End If
# Clear current PDF document
Set Variable [ $pdf ; Value: MBS("DynaPDF.New") ]
# Load PDF from container
Set Variable [ $r ; Value: MBS("DynaPDF.OpenPDFFromContainer"; $pdf;
DynaPDF Replace Text::InputPDF) ]
# import a page
Set Variable [ $r ; Value: MBS("DynaPDF.ImportPDFPage"; $pdf; 1) ]
#
# initialize parser
Set Variable [ $OptimizeFlags ; Value: 0 ]
Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.Create"; $pdf; $OptimizeFlags ) ]
Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.SetAltFont"; $pdf; "Helvetica"; 0;
12) ]
#
# We collect positions in a quicklist
Set Variable [ $list ; Value: MBS("QuickList.New") ]
#
Set Variable [ $page ; Value: 1 ]
Set Variable [ $r ; Value: MBS("DynaPDF.SetPageCoords"; $pdf; "TopDown") //
parser delivers top down coordinates ]
#
# Now parse a page
Set Variable [ $r ; Value: MBS("DynaPDF.Parser.ParsePage"; $pdf; $page;
"EnableTextSelection") ]
If [ MBS("IsError") ]
    Show Custom Dialog [ "Failed to parse page" ; $r ]
Else
    # Run a find
```

```

Set Variable [ $continueFind ; Value: 0 ]
Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.FindText"; $pdf; DynaPDF
Replace Text::SearchText; "CaseInsensitive"; $continueFind) ]
If [ $r = 1 ]
    Loop [ Flush: Defer ]
        # store positions for later
        Set Variable [ $rectangle ; Value:
MBS( "DynaPDF.Parser.SelectionBBox"; $pdf ) ]
        Set Variable [ $r ; Value: MBS("QuickList.Push"; $list;
$rectangle) ]
        #
        # delete text
        Set Variable [ $x1 ; Value: GetAsNumber(GetValue($rectangle;
1)) ]
        Set Variable [ $y1 ; Value: GetAsNumber(GetValue($rectangle;
2)) ]
        Set Variable [ $x2 ; Value: GetAsNumber(GetValue($rectangle;
3)) ]
        Set Variable [ $y2 ; Value: GetAsNumber(GetValue($rectangle;
4)) ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.DeleteText";
$pdf; $x1; $y1; $x2; $y2) ]
        #
        # Continue search
        Set Variable [ $continueFind ; Value: 1 ]
        Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.FindText";
$pdf; DynaPDF Replace Text::SearchText; "CaseInsensitive"; $continueFind) ]
        Exit Loop If [ $r ≠ 1 ]
    End Loop
    #
    # Save changes back
    Set Variable [ $r ; Value: MBS( "DynaPDF.Parser.WriteToPage"; $pdf;
$OptimizeFlags) ]
End If
End If
#
# now edit page and draw the rectangles
If [ MBS("QuickList.Count"; $list) > 0 ]
    #
    If [ MBS("DynaPDF.EditPage"; $pdf; $page) = "OK" ]
        Set Variable [ $r ; Value: MBS("DynaPDF.SetUseVisibleCoords"; $pdf;
1) // just in case crop box is used ]
        Set Variable [ $r ; Value: MBS("DynaPDF.SetLineWidth"; $pdf; 1) ]
        Set Variable [ $r ; Value: MBS("DynaPDF.SetOrientationEx"; $pdf;
MBS("DynaPDF.GetOrientation"; $pdf)) // rotate coordinate system as needed ]

```

```

Loop [ Flush: Defer ]
    Set Variable [ $rectangle ; Value: MBS("QuickList.Pop"; $list) ]
    Set Variable [ $x1 ; Value: GetAsNumber(GetValue($rectangle;
1)) ]
    Set Variable [ $y1 ; Value: GetAsNumber(GetValue($rectangle;
2)) ]
    Set Variable [ $x2 ; Value: GetAsNumber(GetValue($rectangle;
3)) ]
    Set Variable [ $y2 ; Value: GetAsNumber(GetValue($rectangle;
4)) ]
    Set Variable [ $r ; Value: MBS("DynaPDF.Rectangle"; $pdf; $x1;
    $y1; $x2-$x1; $y2-$y1; "FillStroke") ]
    #
    Exit Loop If [ MBS("QuickList.Count"; $list) < 1 ]
End Loop
End If
End If
#
Set Variable [ $r ; Value: MBS("QuickList.Free"; $list) ]
# save
Set Field [ DynaPDF Replace Text::OutputPDF ; MBS( "DynaPDF.Save"; $pdf;
"output.pdf" ) ]
# Cleanup
Set Variable [ $r ; Value: MBS("DynaPDF.Release"; $pdf) ]

```