

Page Layouting in DynaPDF

You can do a lot of PDF things with the [DynaPDF](#) functions in [MBS FileMaker Plugin](#). One of them is layouting text and images on a PDF page.

Let us show you the output of a new example we made for a client. On the left you see the final page and on the right side the same PDF but with visible boxes for the text output.

Our test story for today

>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Commodo quis imperdiet massa tincidunt. Quam vulputate dignissim suspendisse in est ante in. Purus semper eget dui at. Nibh ipsum consequat nisl vel pretium lectus quam. Integer quis auctor elit sed vulputate mi sit amet. Tempor id eu nisl nunc mi ipsum



faucibus. Lacus suspendisse faucibus interdum posuere lorem ipsum. Sit amet nulla facilisi morbi tempus iaculis urna id volutpat. Et malesuada fames ac turpis. Ut morbi tincidunt augue interdum velit euismod in pellentesque. At augue eget arcu dictum. Imperdiet sed euismod nisi porta lorem mollis. Donec ac odio tempor orci dapibus ultrices in iaculis nunc. Laoreet sit amet cursus sit amet dictum sit amet. Ut porttitor leo a diam.

More Details

Ut consequat semper viverra nam libero. Facilisis mauris sit amet massa vitae tortor condimentum. Pretium lectus quam id leo in vitae turpis massa. Ultrices gravida dictum fusce ut placerat. Neque egestas congue quisque egestas diam. Vehicula ipsum a arcu cursus vitae congue. Condimentum id venenatis a condimentum. Eget lorem dolor sed viverra ipsum nunc aliquet bibendum. Scelerisque viverra mauris in aliquam. Suspendisse faucibus interdum

posuere lorem ipsum dolor. Commodo quis imperdiet massa tincidunt nunc pulvinar. Dolor sed viverra ipsum nunc aliquet bibendum enim facilisis. In vitae turpis massa sed.

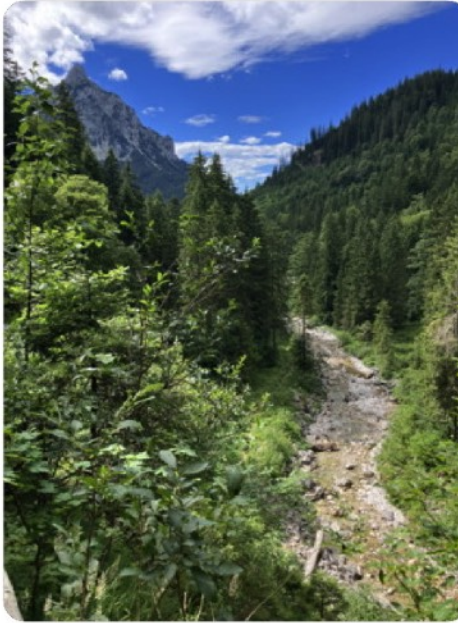
More Details

Enim sit amet venenatis urna. Ornare arcu dui vivamus arcu felis bibendum ut. Augue neque gravida in fermentum et sollicitudin ac. Nulla posuere sollicitudin aliquam ultrices. Blandit massa enim nec dui nunc mattis enim ut tellus. Scelerisque purus semper eget dui at. Consequat mauris nunc congue nisi. Tortor pretium viverra suspendisse potenti nullam ac tortor. Ut morbi tincidunt augue interdum velit euismod



in pellentesque. Nunc aliquet bibendum enim facilisis gravida neque convallis a cras. Nibh sit amet commodo nulla facilisi nullam vehicula ipsum. Nibh tortor id aliquet lectus proin nibh nisl condimentum. Ut faucibus pulvinar elementum integer enim neque. Interdum consectetur libero id faucibus nisl tincidunt. Tellus mauris a diam

As you see on the test page, we got a couple of cool things here:

<h1>Our test story for today</h1>	
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Commodus quis imperdiet massa tincidunt. Quam vulputate dignissim suspendisse in est ante in. Purus semper eget dui at. Nibh ipsum consequat nisl vel pretium lectus quam. Integer quis auctor elit sed vulputate mi sit amet. Tempus id eu nisl nunc mi ipsum</p>	<p>posuere lorem ipsum dolor. Commodus quis imperdiet massa tincidunt nunc pulvinar. Dolor sed viverra ipsum nunc aliquet bibendum enim facilisis. In vitae turpis massa sed.</p>
	<p>More Details Enim sit amet venenatis urna. Ornare arcu dui vivamus arcu felis bibendum ut. Augue neque gravida in fermentum et sollicitudin ac. Nulla posuere sollicitudin aliquam ultrices. Blandit massa enim nec dui nunc mattis enim ut tellus. Scelerisque purus semper eget dui at. Consequat mauris nunc congue nisi. Tortor pretium viverra suspendisse potenti nullam ac tortor. Ut morbi tincidunt augue interdum velit euismod</p>
<p>faucibus. Lacus suspendisse faucibus interdum posuere lorem ipsum. Sit amet nulla facilisi morbi tempus iaculis urna id volutpat. Et malesuada fames ac turpis. Ut morbi tincidunt augue interdum velit euismod in pellentesque. At augue eget arcu dictum. Imperdiet sed euismod nisi porta lorem mollis. Donec ac odio tempor orci dapibus ultrices in iaculis nunc. Laoreet sit amet cursus sit amet dictum sit amet. Ut porttitor leo a diam.</p>	
<p>More Details Ut consequat semper viverra nam libero. Facilisis mauris sit amet massa vitae tortor condimentum. Pretium lectus quam id leo in vitae turpis massa. Ultrices gravida dictum fusce ut placerat. Neque egestas congue quisque egestas diam. Vehicula ipsum a arcu cursus vitae congue. Condimentum id venenatis a condimentum. Eget lorem dolor sed viverra ipsum nunc aliquet bibendum. Scelerisque viverra mauris in aliquam. Suspendisse faucibus interdum</p>	<p>in pellentesque. Nunc aliquet bibendum enim facilisis gravida neque convallis a cras. Nibh sit amet commodo nulla facilisi nullam vehicula ipsum. Nibh tortor id aliquet lectus proin nibh nisl condimentum. Ut faucibus pulvinar elementum integer enim neque. Interdum consectetur libero id faucibus nisl tincidunt. Tellus mauris a diam</p>

We have a header line on top, which depending on the height moves down the other text. Then we place two pictures, which we read from container fields and scale to meet the column width. We apply a clipping path to give them round edges. The styled text is converted from FileMaker and output in six different text boxes. The first letter of the text is drawn in a bigger font on purpose. Any extra text goes into the second page of the PDF.

All happens in a FileMaker script within a second on the click of a button. Imagine if you can script writing 1000 PDFs in a script looping over many records. Like for a real estate company which automatically creates nice PDF documents for each property on offer.

Let's start by showing some script snippets. The first one draws the styled text from the header and draw it. The [DynaPDF.WriteStyledText](#) function does the styled text conversion automatically, so you don't need to do it manually. Once the text is output, we query the coordinates where the text ended on the Y position, so we can place more text below. This allows the user to put in a two line header and still have the layout look good.

```
Set Variable [ $r ; Value: MBS("DynaPDF.SetFillColor"; $pdf; 0; 0; 0) ]
Set Variable [ $r ; Value: MBS("DynaPDF.SetTextRect"; $pdf; $PosX;
$pageHeight - $OneInch; $TextWidth; -1) ]
Set Variable [ $r ; Value: MBS("DynaPDF.WriteStyledText"; $pdf;
"center"; Page Layouting::Header) ]
Set Variable [ $TextY ; Value: MBS("DynaPDF.GetLastTextPosY"; $pdf) -
20 // a little bit distance ]
```

Next we place the picture on the left. For this we first use [DynaPDF.ReadImageFormat](#) to let DynaPDF read the image header and tell us the size in pixel. If your images are special formats like PSD or HEIF, please try a function like [Container.ReadImage](#) to convert first to PNG. With the size information, we can do the calculation for the picture height with the correct aspect ratio. Once we know the size, we can draw a round rectangle to create a clipping path. while the clipping path is active, we insert the image to get round edges. Using [DynaPDF.SaveGraphicState](#) and [DynaPDF.RestoreGraphicState](#) ensures the proper removal of the clipping path when the image is finished.

```
Set Variable [ $r ; Value: MBS("DynaPDF.SaveGraphicState"; $pdf) ]
Set Variable [ $r ; Value: MBS("DynaPDF.ReadImageFormat"; $pdf;
Page Layouting::Picture Left) ]
Set Variable [ $ImageWidth ; Value: GetAsNumber (LeftValues ( $r ;
1 )) ]
Set Variable [ $ImageHeight ; Value: GetAsNumber (MiddleValues ( $r ;
2; 1 )) ]
Set Variable [ $LeftPicWidth ; Value: $ColumnWidth ]
Set Variable [ $LeftPicHeight ; Value: $ColumnWidth / $ImageWidth *
```

```

$imageHeight ]
Set Variable [ $LeftPicTop ; Value: 400 ]
Set Variable [ $LeftPicLeft ; Value: $OneInch ]
Set Variable [ $r ; Value: MBS("DynaPDF.RoundRectEx"; $pdf;
$LeftPicLeft; $LeftPicTop; $LeftPicWidth; $LeftPicHeight; 10; 10;
"NoFill") ]
Set Variable [ $r ; Value: MBS("DynaPDF.ClipPath"; $pdf; "Winding";
"fill" ) ]
Set Variable [ $r ; Value: MBS("DynaPDF.InsertImage"; $pdf; Page
Layouting::Picture Left; $LeftPicLeft; $LeftPicTop; $LeftPicWidth;
$LeftPicHeight; 1) ]
Set Variable [ $r ; Value: MBS("DynaPDF.RestoreGraphicState"; $pdf) ]

```

Whenever you insert an image, you may want to set a few options first. The call above requests images stored as JPEG with 90% quality with a maximum resolution of 300 (instead of 150 as default), no transparent color and prefers pass through for JPEGs to avoid recompression.

```

Set Variable [ $r ; Value: MBS( "DynaPDF.SetJPEGQuality"; $PDF; 90 )
&
MBS( "DynaPDF.SetCompressionFilter"; $PDF; "jpeg" ) &
MBS( "DynaPDF.SetSaveNewImageFormat"; $PDF; 0 ) &
MBS( "DynaPDF.SetResolution"; $PDF; 300 ) &
MBS( "DynaPDF.SetUseTransparency"; $PDF; 0 ) ]

```

We do the same with the right picture on a slightly different position. Then we place the first letter in bigger font size. This is called an Initial and is a styling element, which we can use in FileMaker:

```

Set Variable [ $r ; Value: MBS("DynaPDF.SetFont"; $pdf; "Georgia"; 2;
20) ]
Set Variable [ $r ; Value: MBS("DynaPDF.SetTextRect"; $pdf; $PosX;
$TextY; 50; 50 ) ]
Set Variable [ $r ; Value: MBS("DynaPDF.WriteStyledText"; $pdf; "left";
TextSize ( Left(Page Layouting::Text; 1); 30)) ]
Set Variable [ $LastX ; Value: MBS("DynaPDF.GetLastTextPosX"; $pdf)
+ 3 ]
Set Variable [ $LastY ; Value: MBS("DynaPDF.GetLastTextPosY"; $pdf) -
3 ]

```

Once done we need to know where the text ended, so we can now define the rectangles for the text:

first box for text

```

Set Variable [ $SectionLeft ; Value: $lastX ]
Set Variable [ $SectionTop ; Value: $TextY ]
Set Variable [ $SectionWidth ; Value: $ColumnWidth - $lastX + $PosX ]
Set Variable [ $SectionHeight ; Value: $SectionTop[1] - $LastY + 10 ]

```

first box for text

```
Set Variable [ $SectionLeft[2] ; Value: $PosX ]
Set Variable [ $SectionTop[2] ; Value:
MBS("DynaPDF.GetLastTextPosY"; $pdf) - 6 ]
Set Variable [ $SectionWidth[2] ; Value: $ColumnWidth ]
Set Variable [ $SectionHeight[2] ; Value: $SectionTop[2] - $LeftPicTop -
10 - $LeftPicHeight ]
```

second box for text

```
Set Variable [ $SectionLeft[3] ; Value: $PosX ]
Set Variable [ $SectionTop[3] ; Value: $LeftPicTop - 10 ]
Set Variable [ $SectionWidth[3] ; Value: $ColumnWidth ]
Set Variable [ $SectionHeight[3] ; Value: $SectionTop[3] - $OneInch ]
```

third box for text

```
Set Variable [ $SectionLeft[4] ; Value: $RightPicLeft ]
Set Variable [ $SectionTop[4] ; Value: $TextY ]
Set Variable [ $SectionWidth[4] ; Value: $ColumnWidth ]
Set Variable [ $SectionHeight[4] ; Value: $TextY - $RightPicTop - 10 -
$RightPicHeight ]
```

forth box for text

```
Set Variable [ $SectionLeft[5] ; Value: $RightPicLeft ]
Set Variable [ $SectionTop[5] ; Value: $RightPicTop - 10 ]
Set Variable [ $SectionWidth[5] ; Value: $ColumnWidth ]
Set Variable [ $SectionHeight[5] ; Value: $SectionTop[5] - $OneInch ]
```

next page

```
Set Variable [ $SectionLeft[6] ; Value: $OneInch ]
Set Variable [ $SectionTop[6] ; Value: $pageHeight - $OneInch ]
Set Variable [ $SectionWidth[6] ; Value: $TextWidth ]
Set Variable [ $SectionHeight[6] ; Value: $TextHeight ]
```

This probably looks more complicate than it is. Section 1 is first two lines of text right of the initial letter. Then section 2 is the text below until the picture. We subtract 10 from the height as a distance to the picture. The 3rd section then fills text below the picture until the bottom of the page with a one inch distance. The 4th section is the text on the right top side and section 5 contains the text for the bottom right. We added a sixth section covering the whole page for the overflow pages.

The coordinates can be difficult to get sometimes. We here use the native coordinate system for the PDF page, although DynaPDF can swap it with [DynaPDF.SetPageCoords](#) function. So the top of a text box is from the bottom of the page and the height goes down from that. We need to take this into account for drawing the rectangles.

If you like to measure coordinates in a graphics app, please render the PDF with 72 DPI and then measure pixels. Some applications like

GraphicsConverter ask you for the resolution when you open a PDF document and then you can measure distances.

Now let's show you the final part here with the output of all text boxes in one call to [DynaPDF.WriteStyledText](#):

```
Set Variable [ $section ; Value: 1 ]
Set Variable [ $r ; Value: MBS("DynaPDF.SetTextRect"; $pdf;
$SectionLeft[$section]; $SectionTop[$section];
$SectionWidth[$section]; $SectionHeight[$section] ) ]
Set Variable [ $r ; Value: MBS("DynaPDF.AllowPageBreak"; $pdf; 1) ]
Set Variable [ $r ; Value: MBS("DynaPDF.SetPageBreakExpression";
$pdf; "Let ( [
    $section = Min($section + 1; 6);
    r = If( $section = 6; MBS("\DynaPDF.EndPage\"; $pdf) &
MBS("\DynaPDF.Append\"; $pdf); 0);
    a = MBS( "\DynaPDF.SetTextRect\"; $pdf; $SectionLeft[$section];
    $SectionTop[$section]; $SectionWidth[$section];
$SectionHeight[$section] )
]; 0 )") ]
Set Variable [ $r ; Value: MBS("DynaPDF.WriteStyledText"; $pdf; "left";
Middle(Page Layouting::Text; 2; Length(Page Layouting::Text))) ]
```

As you see, we use a page break expression to decide where to continue the text if a text box is full. The expression can use the variables defined in the script. Let's unquote the expression to inspect it:

```
Let ( [
$section = Min($section + 1; 6);
r = If( $section = 6; MBS("DynaPDF.EndPage"; $pdf) &
MBS("DynaPDF.AppendPage"; $pdf); 0);
a = MBS( "DynaPDF.SetTextRect"; $pdf; $SectionLeft[$section];
$SectionTop[$section]; $SectionWidth[$section];
$SectionHeight[$section] )
]; 0 )
```

As you see, we count \$section up on each call. Then we check if we reach 6, the last section and stay there. And if we hit section 6, we append a new page. Then we set the text rectangle for the text output and continue writing text.

You may use the sample file as a base and extend it for automatically generating thousands of custom PDFs based on records and their information. That may include conditions to skip images if there are none in the container or adding more information like a barcode or a weblink to the page.

Please try the example, included in the next pre-release of [MBS FileMaker Plugin](#) 14.2. And please do not hesitate to contact us with your questions.

See also

[PDF Attachments in FileMaker with DynaPDF](#)

[DynaPDF Parser for FileMaker](#)

[MBS Plugin Advent calendar: 15 - DynaPDF](#)

[Merge documents with DynaPDF](#)

[Add page links for FileMaker](#)

[Things you can do with DynaPDF](#)

[Render pictures from PDF](#)

[Adding cutting lines for PDF in FileMaker](#)

[Swiss QR-Codes for invoices as vector graphics](#)

[Example Script for DynaPDF.FindText and DynaPDF.WebLink](#)