The klfimpl package[1]

Philippe Faist *philippe.faist@bluewin.ch*

August 20, 2020

[1] *This document corresponds to klfimpl v0.2, dated 2020/08/20. It is part of the klfengine tool, see https://github.com/klatexformula/klfengine.*

klfimpl—LaTeX helper code for the implementation of the klfengine (the engine of KLatexFormula 5).

■ 1 Introduction

KLatexFormula 5 will ship with a new engine, *klfengine*. This implementation converts a LaTeX equation to different formats such as PDF or PNG. While KLatexFormula up to version 4 required a few runs of gs and other manual EPS fixes after running `latex` and `dvips`, the new workflow aims at directly doing as muh as possible from the latex end of things—get the correct paper size, correct background color, page margins, and so on—therefore cleaning up the workflow and speeding up the whole compilation.

This package provides an environment `\begin{klfcontent}...\end{klfcontent}` which typesets the given content in a box, measures the box dimensions, then resizes the page to fit the box as requested including margins and/or alignment to some fixed size, and renders the whole thing.

This package expects that there is a single such environment in the entire document and no other content.

■ 2 Implementation

2.1 Some general declarations

The box in which the LaTeX content will be typeset.

```
\newbox\klf@eqnbox
```

Some dimensions etc. that we will track. The main box dimensions (width, height, depth, total height = height + depth):

```
\newdimen\klf@w
\newdimen\klf@h
\newdimen\klf@d
\newdimen\klf@th
```

The paper size (width/height):

```
\newdimen\klf@ppw
\newdimen\klf@pph
```

Any offset that should be used to display the box (useful if margins are requested, or a fixed page width and/or height are requested):

```
\newdimen\klf@hshift
\newdimen\klf@vshift
```

And record some font dimensions:

```
\newdimen\klf@em
\newdimen\klf@ex
\newdimen\klf@capxhgt
```

We will compute any user input dimensions as proper dimension *inside* the equation box to make sure font settings are taken into account correctly. The dimensions will be stored here.

```
\newdimen\klf@dim@fixedwidth
\newdimen\klf@dim@fixedheight
\newdimen\klf@dim@topmargin
\newdimen\klf@dim@rightmargin
\newdimen\klf@dim@bottommargin
\newdimen\klf@dim@leftmargin
```

Remember which engine we're running under. To speed up things and to avoid problems with like `iftex` package not existing, we simply require the caller to tell us via a package option what `latex` engine is being run (`latex` with DVI output, `pdflatex`, `xelatex`, or `lualatex`).

```
\newif\ifklf@ltxengine@latex \klf@ltxengine@latexfalse
\newif\ifklf@ltxengine@pdflatex \klf@ltxengine@pdflatexfalse
```

```
\newif\ifklf@ltxengine@xelatex \klf@ltxengine@xelatexfalse
\newif\ifklf@ltxengine@lualatex \klf@ltxengine@lualatexfalse
```

If, for some reason, this package is called in a different context where we wouldn't want the layout to be set to zero by default, then there is a package option for this (`keeplayoutsizes`). This is the corresponding `\newif` flag:

```
\newif\ifklf@keeplayoutsizes
\klf@keeplayoutsizesfalse
```

The user can specify a fixed width and/or a fixed height for the resulting layout. These will be stored here, if applicable (or they will remain empty). The user input should be stored as a *macro*, not as a *dimen*, because we want a dimension given in font-specific metrics (e.g. `4.2em`) to be computed correctly relative to the equation font.

```
\def\klf@set@fixedwidth{}
\def\klf@set@fixedheight{}
```

Enable user to specify margins around the equations.

```
\def\klf@set@topmargin{0.1ex}
\def\klf@set@rightmargin{0.1ex}
\def\klf@set@bottommargin{0.1ex}
\def\klf@set@leftmargin{0.1ex}
```

Enable user to specify global scaling factors for the full resulting box. Scale will be applied to margins as well as to fixed size.

```
\def\klf@set@xscale{1}
\def\klf@set@yscale{1}
```

User can specify how the equation is aligned inside the box, in case we have a fixed width or a fixed height. The horizontal (resp. vectical) alignment coefficient is `0` for left (resp. top) alignment, `0.5` for middle alignment, and `1` for right (resp. bottom) alignment. It can be any value that interpolates between these.

```
\def\klf@set@xaligncoeff{0.5}
\def\klf@set@yaligncoeff{0.5}
```

User can specify the reference points for top and bottom alignment. The top can be aligned to the natural height of the box (`bbox`, the default) or to the height of a capital 'X' (`Xheight`). The bottom can be aligned to the natural depth of the box (`bbox`, the default) or to the baseline (`baseline`). (When you set `baseline`, make sure you have a fixed size or reasonable margins to accomodate any box depth.)

```
\def\klf@set@topalignment{bbox}
\def\klf@set@bottomalignment{bbox}
```

User can tell if they would like a line to be drawn where the baseline is, for instance as a visual reference for use in vector graphics editing software. This can be `none` or `line`.

```
\def\klf@set@baselineruletype{none}
```

2.2 Package options and settings

Set the LaTeX engine.

```
\DeclareOption{latex}{\klf@ltxengine@latextrue}
\DeclareOption{pdflatex}{\klf@ltxengine@pdflatextrue}
\DeclareOption{xelatex}{\klf@ltxengine@xelatextrue}
\DeclareOption{lualatex}{\klf@ltxengine@lualatextrue}
```

Package option to inhibit resetting the page layout to zero by default.

```
\DeclareOption{keeplayoutsizes}{\klf@keeplayoutsizestrue}
```

Now process those options

```
\DeclareOption*{\PackageError{klfimpl}{Unknown option `\CurrentOption'}{}}
\ProcessOptions\relax
```

The following could have been specified as package options, but for the sake of simplicity (and to avoid having to use keyval/xkeyval and so on, the information is provided via a simple macro call.

`\klfSetFixedWidth` `\klfSetFixedHeight` If applicable, set the fixed width and/or fixed height of the content to typeset.

```
\def\klfSetFixedWidth#1{%
  \xdef\klf@set@fixedwidth{#1}}
\def\klfSetFixedHeight#1{%
  \xdef\klf@set@fixedheight{#1}}
```

`\klfSetTopMargin` `\klfSetRightMargin` `\klfSetBottomMargin` `\klfSetLeftMargin` Same for equation margins:

```
\def\klfSetTopMargin#1{%
  \xdef\klf@set@topmargin{#1}}
\def\klfSetRightMargin#1{%
  \xdef\klf@set@rightmargin{#1}}
\def\klfSetBottomMargin#1{%
  \xdef\klf@set@bottommargin{#1}}
\def\klfSetLeftMargin#1{%
  \xdef\klf@set@leftmargin{#1}}
```

`\klfSetXScale` `\klfSetYScale` `\klfSetScale` Specify horizontal and vectical scaling factors. Values are a multipliying factor for dimensions: the value `1` means original size, `0.5` half size, `2` double size. If a value different than `1` was specified, the graphics package is loaded. *NOTE: This* *has to be the exact expression 1, not* `1.0` *or* `1.`. The `\klfSetScale` macro is a convenience macro that sets both horizontal and vectical scaling factors to the same value.

```
\def\klfSetXScale#1{%
  \xdef\klf@set@xscale{#1}%
  \ifx\klf@set@xscale\klf@macro@one
  \else
    \RequirePackage{graphics}%
  \fi
}
\def\klfSetYScale#1{%
  \xdef\klf@set@yscale{#1}%
  \ifx\klf@set@yscale\klf@macro@one
  \else
    \RequirePackage{graphics}%
  \fi
}
\def\klfSetScale#1{%
  \xdef\klf@set@xscale{#1}%
  \xdef\klf@set@yscale{#1}%
  \ifx\klf@set@xscale\klf@macro@one
  \else
    \RequirePackage{graphics}%
  \fi
}
\def\klf@macro@one{1}
```

`\klfSetXAlignCoeff`
`\klfSetYAlignCoeff`

```
\def\klfSetXAlignCoeff#1{%
  \xdef\klf@set@xaligncoeff{#1}%
}
\def\klfSetYAlignCoeff#1{%
  \xdef\klf@set@yaligncoeff{#1}%
}
```

`\klfSetTopAlignment`
`\klfSetBottomAlignment`

Macros for the user to set top and bottom alignment. See earlier for explanation of possible values. The possible values will be used to invoke a macro named e.g. `\klf@correctboxheight@@`⟨*top-alignment-name*⟩ and `\klf@correctboxdepth@@`⟨*bottom-alignment-name*⟩, see below in `{klfcontent}` environment.

```
\def\klfSetTopAlignment#1{%
  \xdef\klf@set@topalignment{#1}}
\def\klfSetBottomAlignment#1{%
  \xdef\klf@set@bottomalignment{#1}}
```

`\klfSetBaselineRuleType` Set which kind of baseline rule we would like, if any.

```
\def\klfSetBaselineRuleType#1{%
  \xdef\klf@set@baselineruletype{#1}}
```

2.3 Basic/common implementation macros

First of all, a simple macro to reset all LaTeX layout dimensions.

```
\def\klf@ZeroLayoutSizes{%
  \oddsidemargin=\z@\relax
  \evensidemargin=\z@\relax
  \topmargin=\z@\relax
  \voffset=-1in\relax
  \hoffset=-1in\relax
  \headsep=\z@\relax
  \headheight=\z@\relax
  \marginparsep=\z@\relax
  \footskip=\z@\relax
  \parindent=\z@\relax
  \parskip=\z@\relax
  \topskip=\z@\relax
}
```

`\klf@ZeroDisplaySkips` And define a routine that sets all the display-related skips to zero so that we can use this inside a `\vbox`.

```
\def\klf@ZeroDisplaySkips{%
  \abovedisplayskip=\z@\relax
  \belowdisplayskip=\z@\relax
  \abovedisplayshortskip=\z@\relax
  \belowdisplayshortskip=\z@\relax
}
```

By default, reset all these dimensions right away, unless the `keeplayoutsizes` package option was provided.

```
\ifklf@keeplayoutsizes
\else
  \klf@ZeroLayoutSizes
  \klf@ZeroDisplaySkips
\fi
```

`\klfSetPaperSize` Change the paper size. For `pdflatex` and `xe/luatex` this can be called after `\begin{document}` but for `latex` with traditional `dvi` output this must be issued in the preamble.

```
\def\klfSetPaperSize#1#2{%
  \@tempdima=#1\relax
  \@tempdimb=#2\relax
  \klf@SetPaperSize@FromDims\@tempdima\@tempdimb
}
\def\klf@SetPaperSize@FromDims#1#2{%
  \global\textwidth=#1\relax
  \global\textheight=#2\relax
  \global\hsize=#1\relax
  \global\vsize=#2\relax
  \global\paperwidth=#1\relax
  \global\paperheight=#2\relax
  \ifklf@ltxengine@pdflatex
    \global\pdfpagewidth=#1\relax
    \global\pdfpageheight=#2\relax
  \fi
  \ifklf@ltxengine@xelatex
    \global\pdfpagewidth=#1\relax
    \global\pdfpageheight=#2\relax
  \fi
  \ifklf@ltxengine@lualatex
    \global\pagewidth=#1\relax
    \global\pageheight=#2\relax
  \fi
}
```

2.4 Main implementation routine

`klfcontent` The argument should be a box command (e.g., `\hbox`, `\vbox`, `\vtop`, `\vcenter`). Example usage: `\begin{klfcontent}{\vcenter}{`⟨*initialization code*⟩`}`... or `\begin{klfcontent}{\hbox to 1cm}{`⟨*init. code*⟩`}`....

```
\def\klfcontent#1#2{%
  \unskip
  \samepage
  \setbox\klf@eqnbox=#1\bgroup
    \klf@ZeroDisplaySkips%
```

First run any user provided font commands.

```
    #2%
```

Do internal calculations now, after the user font commands, so that we have the correct em/ex dimension, etc.

```
    \global\klf@em=1em\relax
    \global\klf@ex=1ex\relax
    \setbox0=\hbox{X}%
    \global\klf@capxhgt=\ht0%
    \ifx\klf@set@fixedwidth\@empty\else
      \global\klf@dim@fixedwidth=\klf@set@fixedwidth\relax
    \fi
    \ifx\klf@set@fixedheight\@empty\else
      \global\klf@dim@fixedheight=\klf@set@fixedheight\relax
    \fi
    \global\klf@dim@topmargin=\klf@set@topmargin\relax
    \global\klf@dim@rightmargin=\klf@set@rightmargin\relax
    \global\klf@dim@bottommargin=\klf@set@bottommargin\relax
    \global\klf@dim@leftmargin=\klf@set@leftmargin\relax
```

If the user set a fixed width, then calculate the available horizontal space and update `\textwidth` & `\hsize`.

```
    \ifx\klf@set@fixedwidth\@empty\else
      \textwidth=\klf@dim@fixedwidth
      \advance \textwidth -\klf@dim@rightmargin
      \advance \textwidth -\klf@dim@leftmargin
      \hsize=\textwidth
    \fi
}
\def\endklfcontent{%
  \egroup
```

Now we record the box dimensions.

```
  \klf@w=\wd\klf@eqnbox\relax
  \klf@h=\ht\klf@eqnbox\relax
  \klf@d=\dp\klf@eqnbox\relax
```

If we shouldn't align to the bounding box, correct the height and the depth of the box to whatever is requested by the corresponding top/bottom alignment options.

```
  \csname klf@correctboxheight@@\klf@set@topalignment\endcsname
  \csname klf@correctboxdepth@@\klf@set@bottomalignment\endcsname
```

```
  \klf@th=\klf@h\relax
  \advance \klf@th \klf@d \relax
```

Determine page size and offsets. Take into account any possible fixed paper width or height and any margins.

```
  \ifx\klf@set@fixedwidth\@empty%
    \klf@ppw=\klf@w\relax
    \advance \klf@ppw \klf@dim@leftmargin \relax
    \advance \klf@ppw \klf@dim@rightmargin \relax
    \klf@hshift=\klf@dim@leftmargin\relax
  \else%
    \klf@ppw=\klf@dim@fixedwidth\relax
    \klf@hshift=\klf@set@xaligncoeff\klf@ppw\relax
    \advance \klf@hshift -\klf@set@xaligncoeff\klf@w\relax
    \advance \klf@hshift -\klf@set@xaligncoeff\klf@dim@rightmargin\relax
    \advance \klf@hshift -\klf@set@xaligncoeff\klf@dim@leftmargin\relax
    \advance \klf@hshift \klf@dim@leftmargin\relax
  \fi
  \ifx\klf@set@fixedheight\@empty%
    \klf@pph=\klf@th\relax
    \advance \klf@pph \klf@dim@topmargin \relax
    \advance \klf@pph \klf@dim@bottommargin \relax
    \klf@vshift=\klf@dim@topmargin\relax
  \else%
    \klf@pph=\klf@dim@fixedheight\relax
    \klf@vshift=\klf@set@yaligncoeff\klf@pph\relax
    \advance \klf@vshift -\klf@set@yaligncoeff\klf@th\relax
    \advance \klf@vshift -\klf@set@yaligncoeff\klf@dim@bottommargin\relax
    \advance \klf@vshift -\klf@set@yaligncoeff\klf@dim@topmargin\relax
    \advance \klf@vshift \klf@dim@topmargin\relax
  \fi
```

No scale has been applied yet. Call the rendering routine that will take into account scaling factors as necessary.

```
  \klf@RenderContentBox
```

Finally dump all meta-info to the standard output to provide additional information back to *klatexformula*.

```
  \klfDumpMetaInfo
  \ignorespaces
}
```

`\klf@RenderContentBox` The `\klf@RenderContentBox` macro sets the paper size (if the current latex engine allows this at this point), and displays the box accordingly.

If we're using a pdf-based engine (`pdflatex`, `xelatex` or `lualatex`), then we set the page size immediately (scaled correctly). If we're using the `latex` (LaTeX → DVI) engine, then we cannot set the page size at this point, because the page size needs to be set in the preamble. In this case, we don't do anything now, but *klatexformula* performs a second pass where the correct exact page size is set in preamble (via meta-data dumped by `\klfDumpMetaInfo`).

```
\newbox\klf@final@box
\def\klf@RenderContentBox{%
  \ifklf@ltxengine@latex% tough luck
  \else
    \@tempdima=\klf@ppw
    \@tempdima=\klf@set@xscale\@tempdima\relax
    \@tempdimb=\klf@pph
    \@tempdimb=\klf@set@yscale\@tempdimb\relax
    \klf@SetPaperSize@FromDims\@tempdima\@tempdimb%
  \fi
```

To render the contents, we check whether any scaling is applied. If so, we call `\klf@do@scale` which wraps the argument in an appropriate `\scalebox` (provided by the graphics package). (Otherwise we simply render the box without any `\scalebox`.)

```
  \let\klf@next\@firstofone
  \ifx\klf@set@xscale\klf@macro@one\else
    \let\klf@next\klf@do@scale\fi
  \ifx\klf@set@yscale\klf@macro@one\else
    \let\klf@next\klf@do@scale\fi
  \nobreak
  \hsize=\klf@ppw
  \setbox\klf@final@box=\vbox{\hbox to \z@{%
    \klf@next{%
```

Here, we actually render the box contents. There are three items to draw: (1) the background, (2) the baseline rule, if any, and (3) the actual equation box.

Begin with the background. The code in `\klf@DrawBackground` is designed so that it takes no horizontal or vertical space.

```
      \klf@DrawBackground
```

Draw the baseline rule and contents.

```
      \vbox to \z@{%
        \hrule \@height\z@\nobreak
        \vskip \klf@vshift\relax\nobreak
        \hbox{\vrule \@width\z@ \relax
          \raise \klf@d \hbox to \z@{%
            \csname klf@baseline@rule@@\klf@set@baselineruletype\endcsname
          }%
          \hskip \klf@hshift\relax
          \raise \klf@d \box\klf@eqnbox
        }%
      }%
    }%
  }}%
  \c@page=\z@
  \shipout\box\klf@final@box
}
\def\klf@do@scale#1{%
  \scalebox{\klf@set@xscale}[\klf@set@yscale]{#1}%
}
```

2.5 Background: color, frame, and/or custom elements

Enable the user to specify a custom background, and even draw stuff on it if they like. I'm not sure what the best API is to let the user draw what they like.

`\klfSetBackgroundColor`
`\klfSetBackgroundColorOpacity`

Let the user set a simple color, with a custom opacity (semitransparency is provided by the pgf package, so included it if necessary).

```
\newcommand\klfSetBackgroundColor[1]{%
  \klfEnsureColorPackageLoaded
  \definecolor{klfbgcolor}{RGB}{#1}%
  \def\klf@set@bgcoloropacity{1}%
}
\def\klf@set@bgcoloropacity{0}
\newcommand\klfSetBackgroundColorOpacity[1]{%
  \edef\klf@set@bgcoloropacity{#1}%
  \ifdim#1\p@=\z@\relax
  \else
    \ifdim#1\p@=\p@\relax
    \else
      \RequirePackage{pgf}%
    \fi
  \fi
}
\def\klfEnsureColorPackageLoaded{%
  \@ifpackageloaded{color}{}{%
    \@ifpackageloaded{xcolor}{}{%
      \RequirePackage{color}%
    }%
  }%
}
```

Some temporary registers.

```
\newdimen\klf@set@bgtmp@rectw
\newdimen\klf@set@bgtmp@recth
```

The background color will be drawn as a filled rectangle, extending on all sides with a bleed length stored in `\klf@set@bgcolor@bleed`.

```
\newdimen\klf@set@bgcolor@bleed
\klf@set@bgcolor@bleed=\p@
```

`\klf@DrawBackground@Color` The code to draw the background color. Draw the background color as a rectangle, with the required opacity. Of course, don't do this if the background rectangle is fully transparent.

```
\def\klf@DrawBackground@Color{%
  \ifdim\klf@set@bgcoloropacity\p@=\z@\relax
  \else
    \klf@set@bgtmp@rectw=\klf@ppw
    \advance\klf@set@bgtmp@rectw 2\klf@set@bgcolor@bleed
    \klf@set@bgtmp@recth=\klf@pph
    \advance\klf@set@bgtmp@recth 2\klf@set@bgcolor@bleed
    \ifdim\klf@set@bgcoloropacity\p@=\p@\relax
      \let\klf@tmp@pgfsetfillopacity\@gobble
    \else
      \let\klf@tmp@pgfsetfillopacity\pgfsetfillopacity
    \fi
    \begingroup
      \color{klfbgcolor}%
      \klf@tmp@pgfsetfillopacity{\klf@set@bgcoloropacity}%
      \hbox to \z@{%
        \hbox{}\hskip -\klf@set@bgcolor@bleed\relax
        \vbox to \z@{%
          \hrule \@height\z@
          \nobreak
          \vskip-\klf@set@bgcolor@bleed\relax
          \vskip\z@skip\relax
          \rule{\klf@set@bgtmp@rectw}{\klf@set@bgtmp@recth}%
        }%
      }%
      \klf@tmp@pgfsetfillopacity{1}%
    \endgroup
  \fi
}
```

`\klfSetBackgroundFrameXOffset`
`\klfSetBackgroundFrameYOffset`
`\klfSetBackgroundFrameOffset`
`\klfSetBackgroundFrameThickness`
`\klfSetBackgroundFrameColor`

Optionally draw a frame around the contents as a background decoration.

```
\newdimen\klf@set@bgframe@xoffset
\klf@set@bgframe@xoffset=\z@\relax
\newdimen\klf@set@bgframe@yoffset
\klf@set@bgframe@yoffset=\z@\relax
\def\klfSetBackgroundFrameXOffset#1{%
  \klf@set@bgframe@xoffset=#1\relax
}
\def\klfSetBackgroundFrameYOffset#1{%
  \klf@set@bgframe@yoffset=#1\relax
}
\def\klfSetBackgroundFrameOffset#1{%
  \klf@set@bgframe@xoffset=#1\relax
  \klf@set@bgframe@yoffset=\klf@set@bgframe@xoffset\relax
}
\newdimen\klf@set@bgframe@thickness
\klf@set@bgframe@thickness=\z@\relax
\def\klfSetBackgroundFrameThickness#1{%
  \klf@set@bgframe@thickness=#1\relax
}
\def\klf@set@bgframe@setcolor{}
\def\klfSetBackgroundFrameColor#1{%
  \klfEnsureColorPackageLoaded
  \definecolor{klffrmcolor}{RGB}{#1}%
  \def\klf@set@bgframe@setcolor{\color{klffrmcolor}}
}
```

Code to draw the background frame:

`\klf@DrawBackground@Frame`

```
\def\klf@DrawBackground@Frame{%
  \ifdim\klf@set@bgframe@thickness=\z@\relax
  \else
    \klf@set@bgtmp@rectw=\klf@ppw
    \advance\klf@set@bgtmp@rectw -2\klf@set@bgframe@xoffset
    \klf@set@bgtmp@recth=\klf@pph
    \advance\klf@set@bgtmp@recth -2\klf@set@bgframe@yoffset
    \hbox to \z@{%
      \hskip \klf@set@bgframe@xoffset\relax
      \vbox to \z@{%
        \vskip \klf@set@bgframe@yoffset\relax
        \begingroup
          \fboxsep=-\klf@set@bgframe@thickness\relax
          \fboxrule=\klf@set@bgframe@thickness\relax
          \klf@set@bgframe@setcolor
          \fbox{\phantom{\rule{\klf@set@bgtmp@rectw}{\klf@set@bgtmp@recth}}}%
        \endgroup
      }%
    }%
  \fi
}
```

`\klfAddBackgroundCommands`
`\klfAddBackgroundGraphics`

We also provide a generic hook, in case the user wants to draw more fancy stuff. The user can call `\klfAddBackgroundCommands` to append drawing commands to the background. The origin is the top left corner of the image. The user can get dimensions, etc., via the `\klf@ppw`/`\klf@pph`, etc. lengths (for now). The user code for each call to `\klfAddBackgroundCommands` is wrapped in a zero-sized box located at the top left point of the image.

The command `\klfAddBackgroundGraphics` is a shorthand for inserting a background graphic using the graphicx package's `\includegraphics[...]{...}`.

```
\newtoks\klf@set@bgextradrawcommands
\newcommand\klfAddBackgroundCommands[1]{%
  \klf@set@bgextradrawcommands=\expandafter{\the\klf@set@bgextradrawcommands
    \hbox to \z@{\vbox to \z@{%
      #1%
    }}%
  }%
}
\newcommand\klfAddBackgroundGraphics[2][]{%
  \RequirePackage{graphicx}%
  \klfAddBackgroundCommands{%
    \includegraphics[#1]{#2}%
  }%
}
```

`lf@DrawBackground@CustomCommands`

Code to render the custom commands. Remember to enclose code in a zero-sized box.

```
\def\klf@DrawBackground@CustomCommands{%
  \if\relax\detokenize\expandafter{\the\klf@set@bgextradrawcommands}\relax
  \else
    \the\klf@set@bgextradrawcommands
  \fi
}
```

`\klf@DrawBackground` Here's our main internal code macro that renders the background.

```
\def\klf@DrawBackground{%
  \klf@DrawBackground@Color
  \klf@DrawBackground@Frame
  \klf@DrawBackground@CustomCommands
}
```

2.6 Vertical bounding box adjustments (`bbox`, `Xheight`, `baseline`)

Now we define the top/bottom alignment correction routines. These "fix" the height and the depth of the box (rather, their values recorded in `\klf@h` and `\klf@d`) according to the given options.

The `bbox` top and bottom alignment options is the default, and leaves the box dimensions unchanged.

```
\def\klf@correctboxheight@@bbox{}
\def\klf@correctboxdepth@@bbox{}
```

The `Xheight` top alignment option sets the height of the "box" to be the height of a capital "X". WARNING: This assumes that the box only contains a single line of text.

```
\def\klf@correctboxheight@@Xheight{%
  \klf@h=\klf@capxhgt
}
```

The `baseline` bottom alignment option sets the bottom of the "box" to be the baseline, so sets the depth to zero.

```
\def\klf@correctboxdepth@@baseline{%
  \klf@d=\z@
}
```

2.7 Baseline rule

Now we define the baseline rule types. These simply draw whatever they want, typically a simple `\vrule` of a given width (because we're in horizontal mode).

First, we have the none rule type which simply typesets nothing.

```
\def\klf@baseline@rule@@none{}
```

Then we have the `line` rule, which draws a horizontal line throughout the box at the baseline height.

`\klfBaselineRuleLineSetup`
`\klfBaselineRuleLineThickness`

The line is controlled by the macros `\klfBaselineRuleLineSetup` and `\klfBaselineRuleLineThickness`:

```
\def\klfBaselineRuleLineSetup{}
\def\klfBaselineRuleLineThickness{0.05\p@}
```

You may redefine these to style the line as appropriate. For instance, this would give you a blue baseline that is 0.2pt thick:

```
\renewcommand\klfBaselineRuleLineSetup{\color{blue}}
\renewcommand\klfBaselineRuleLineThickness{0.2pt}
```

And finally this is the code that draws the line.

```
\def\klf@baseline@rule@@line{%
  \begingroup
    \klfBaselineRuleLineSetup
    \vrule width\klf@ppw height\z@ depth\klfBaselineRuleLineThickness\relax
  \endgroup
}
```

2.8 Dump meta-info on standard output

Define the routine that communicates back to *klatexformula* meta-info about the typeset content. This is automatically called in `\end{klfcontent}`.

`\klfDumpMetaInfo`

Dump meta-info on standard output to provide additional information to KLatexFormula. Careful, scaling factors have not been applied yet. So apply them here before displaying the quantities.

```
\def\klfDumpMetaInfo{%
  \begingroup
    \klf@em=\klf@set@xscale\klf@em\relax
    \klf@ex=\klf@set@yscale\klf@ex\relax
    \klf@capxhgt=\klf@set@yscale\klf@capxhgt\relax
    \klf@ppw=\klf@set@xscale\klf@ppw\relax
    \klf@pph=\klf@set@yscale\klf@pph\relax
    \klf@hshift=\klf@set@xscale\klf@hshift\relax
    \klf@vshift=\klf@set@yscale\klf@vshift\relax
    \klf@w=\klf@set@xscale\klf@w\relax
    \klf@h=\klf@set@yscale\klf@h\relax
    \klf@d=\klf@set@yscale\klf@d\relax
    \klf@th=\klf@set@yscale\klf@th\relax
    \message{%
^^J%
***-KLF-META-INFO-BEGIN-***^^J%
EM={\the\klf@em}^^J%
EX={\the\klf@ex}^^J%
CAP_X_HEIGHT={\the\klf@capxhgt}^^J%
PAPER_WIDTH={\the\klf@ppw}^^J%
PAPER_HEIGHT={\the\klf@pph}^^J%
HSHIFT={\the\klf@hshift}^^J%
VSHIFT={\the\klf@vshift}^^J%
BOX_WIDTH={\the\klf@w}^^J%
BOX_HEIGHT={\the\klf@h}^^J%
BOX_DEPTH={\the\klf@d}^^J%
BOX_TOTALHEIGHT={\the\klf@th}^^J%
^^J%
***-KLF-META-INFO-END-***^^J%
    }
  \endgroup
}%
```