The fancytooltips package[*][†]

Robert Mařík
marik@mendelu.cz

April 8, 2010

1 Introduction

The package `fancytooltips` is a package for LaTeX. The pdf can be created by pdflatex or by latex + dvips + AdobeDistiller[1] + Adobe Acrobat[2]. It allows to create tooltips in a similar way like `cooltooltips` package, but the tooltip is a page from another PDF file. In this way you can use mathematics, pictures and animations in your tooltips. The resulting PDF file can be used also with free Abobe Reader.

The tooltips are activated by clicking the active area on the screen and deactivated after closing page or by moving mouse outside the link. You can try the links here (Einstein's formula) and also here (animation – numbers from 1 to 6). You have to use the free Adobe Reader or nonfree Adobe Acrobat to see the effect (xpdf, evince and others fail to work with JavaScripts). For more examples how the presentation may look like see the `examples` subdirectory.

Similar functionality can be achieved using `pdfcomment` (smaller size of the resulting PDF, but restricted to plain text), `ocgtools` (works with layers, only `pdflatex` is supported) and AcroTeX (works with layers, only `latex` + `dvips` + Adobe Distiller + Adobe Acrobat are supported). All these "similar" packages allow to write the text for he tooltips and the text for the main document in one file. In contrast to this apporach, `fancytooltips` package makes use of an external file. However, if you prefer to write both tooltips in one file, you can preprocess your file using `preview` package and exteract tooltips automatically, as shown in the `examples/fancy-preview` directory.

The buttons are created using `eforms.sty` which is a part of AcroTeX bundle.

[*] This document corresponds to fancytooltips v1.6, dated 2010/04/10.

[†] Supported by grants 18/2006, 99/2008 and 131/2010 of Higher Education Development Fund (FRVŠ)

[1] not free ps2pdf

[2] not free Adobe Reader.

2 Usage

2.1 The file with tooltips

The file with tooltips is an ordinary pdf file, one tooltip per page, tooltips should be in the top right corner at the page, in a colored box and the rest of the page should be transparent. If you consider to use `movetips` option (see below), then every page should have the dimensions equal to the dimensions of the colored box with tooltip[3]. We also provide simple cross referencing mechanism to reffer to the tooltips. If the pdf file is created by LaTeX, `\keytip` you can define keywords to reffer to the pages using `\keytip` command. Simply put `\usepackage[createtips]{fancytooltips}` into preamble and write `\keytip{`⟨*foo*⟩`}` in document. This writes information about keyword ⟨*foo*⟩ and the pagenumber into file `\jobname.tips`.

2.2 The file with presentation – pdfLaTeXusers

In the file with presentation, the user is responsible

- input either `color` or `xcolor` package in the preamble
- LaTeX the file two times (we write some macros into `aux` file).

This is not comfortable for the user, but everybody uses different set of packages and from this reason, this part is left to the user. (And among others, the `color` or `xcolor` package is probably inputted by the package which is used to build the presentation.)

`filename option` To input the tooltips from file ⟨*foo.pdf*⟩ call the package with `filename` option: `\usepackage[filename=foo]{fancytooltips}`.

`movetips option` By default, tooltip appears in the top right corner of the page (use View–PageLayout-Single Page in your Adobe Reader, please). If the option `movetips` is used, then tooltip appears close to the mouse pointer. More precisselly, tooltip appears with left down corner at the mouse position, if there is enough place. If not, tooltip appears with right down corner at the mouse position. Finally, the tooltip is shifted down to fit the page, if nesessary[4].

`mouseover option` If you use `mouseover` option, then tooltip appears if you move the mouse pointer to the active area (no clicking is necessary).

`inactive option` This option makes the package inactive.

`active option` This option forces the package active even if `inactive` option is loaded.

`debug option` Prints two alerts reporting success or problems when opening PDF file in Adobe Acrobat (Adobe Reader). Use this option to find possible source of problems. See also the section Troubleshooting . . . in this manual.

`\tooltip` The user can put the tooltip into her or his presentation using the com-

[3] Look at the files `tooltipy.tex` and `tooltipy.pdf` from `examples` subdirectory for a simple example how to meet this condition under pdfLaTeX

[4] This option works in this way if every page of the file with tooltips has dimensions of the box with tooltip. See the `examples` subdirectory.

`\ToolTipExtratext` `noextratext option`

mand `\tooltip{`⟨*stuff*⟩`}{`⟨*keyword-or-pagenumber*⟩`}` where ⟨*stuff*⟩ is the printed text in ⟨*tooltipcolor*⟩ color and ⟨*keyword-or-pagenumber*⟩ is either the pagenumber of the tooltip in the external file or the keyword defined by `\keytip` command and stored in `\TooltipFilename.tips` file, where `\TooltipFilename` is set automatically from the `filename` option. The printed text ⟨*stuff*⟩ is followed by `\TooltipExtratext` command. The default value is small blue soap, as you have seen in the second paragraph of this documentation. There is a package option `noextratext` which defines `\TooltipExtratext` to be empty. If `{`⟨*keyword-or-pagenumber*⟩`}` is not recognized as valid keyword for tooltips,

`\tooltip*`

The text `{`⟨*stuff*⟩`}` is inserted in `\hbox` by `\tooltip`. With starred version of the `\tooltip` macro the text `{`⟨*stuff*⟩`}` is not inserted into the box and the active button dos not cover the text `{`⟨*stuff*⟩`}`, but covers the mark produced by `\TooltipExtratext`.

`\tooltipanim` `\delayinterval`

The user can put a series (animation) of tooltips into the presentation by using `\tooltipanim{`⟨*stuff*⟩`}{`⟨*start*⟩`}{`⟨*end*⟩`}` command, where ⟨*start*⟩ and ⟨*end*⟩ are keywords defined by `\keytip` command or page numbers. The delay between two frames is `\delayinterval` milliseconds. The default value is 200, you can change it by command `\def\delayinterval{100}`.

`\FancyHook`

Redefines `\ref` and `\eqref` commands to work with preview, see the demo files in `example/fancy-preview` directory.

`\TooltipRefmark`

Extra text added to `\ref` and `\eqref` commands with previews, see the demo files in `example/fancy-preview` directory. The default value is the same as for `\TooltipExtratext`.

2.3 Changes for dvips users

`pages option`

dvips users have to specify option `dvips` in `fancytooltips` package. They have to use also a `pages` option with the number of pages in the PDF file with tooltips. You have to call the package by something like this:

```
\usepackage[dvips,filename=tooltipy,pages=27]{fancytooltips}
```

You have to `latex` (two times) and `dvips` your file first. This produces `filename.ps` and `Tooltipsdljs.fdf` files. Distill the `filename.ps` file into `filename.pdf` and open this file by Adobe Acrobat Professional – this imports macros from `Tooltipsdljs.fdf` file. In Acrobat's JavaScript console (open by `Crtl+J`) run (using `Ctrl+Enter`) the command `ImportTooltips();` which is defined for the document. This command inserts invisible buttons on the first page, imports icons (the file with icons specified as ⟨*filename*⟩ parameter when loading fancytooltips must be in working directory) and returns 1. Then save the file under another name.

3 Troubleshooting and known problems

The source code is in Mercurial repository at `http://bitbucket.org/robert.marik/fancytooltips/`. You can also report problems and issues in the forum at this site.

- The package works only with the last `eforms.sty`, version 2006/10/03 v1.0a. You can download this version from www.arotex.net site.

- If the graphics included by `\TooltipExtratext` and `\TooltipRefmark` has colors with custom opacity, Adobe Acrobat Pro sometimes renders the pictures bad. No problems of this type have been observed with Adobe Reader.

- For **dvips** users: In some cases the file `Tooltipsdljs.fdf` is not imported automatically (probably some setting in Adobe Acrobat). In this case you do not see any message even if you use `debug` option. You have troubles of this type if you see in the Javascript console (Ctrl+J) error messages like "`aebTrustedFunctions is not defined 3:Page:Open CloseTooltips is not defined 1:Page:Open`". In this case you have to import the file `Tooltipsdljs.fdf` manually from menu "Form". Then go to the JavaScript console and run `ImportToltips();` command.

Follow the points below if you want to find the source of your problems.

- For dvips users it is a good idea to check that AcroTeX is properly installed. Do the demo files from AcroTeX work for you?

- Try to use `debug` option, prepare the PDF file and open it in Adobe Acrobat or Adobe Reader.

  - You should see two messages when opening the file. If not, the Javascript do not work in your document (are not inserted properly).
  - Both messages should report success for pdflatex users. For dvips users one of the message should report error and if you run `ImportToltips();` command in Javascript console, you should see a message in Javascript console which confirms that the pictures from external PDF file have been imported. If you save the PDF file and open again, both messages should report success.

Change History

v1.0

General: Initial version 1

v1.1

General: Clener code, better documentation, fix for bug with delayinterval macro, support for dvips and Distiller users. 1

v1.2

General: Better support for dvips users, buttons for icons are created automatically using esopic package and have the name "ikona.pagenumber", new option `movetips`, tooltip appears on the current page only and, as a consequence, animations are faster. 1

v1.3

General: Fixed bug which appears in dvips route. Improved documentation – added a note about the required version of eforms.sty 1

v1.4

General: Added support for presentations (like `pause` command for texpower). 1

v1.5

General: Handling number of pages via aux file (fixes some problems issued by previous modification and page break near tooltip). . . 1

v1.6

General: Added options `debug`, `active` and `inactive`, added an example with automatical previews for equations and floats. Added handling of this.dirty and app.focusRect. Changed the name of the file with tooltips. Added starred version of `tooltip` command. 1

4 Implementation

```
⟨*package⟩
\RequirePackage{everyshi}
\RequirePackage{graphicx}
\RequirePackage{xkeyval}
\RequirePackage{eso-pic}

\newif\ifcreatetips\createtipsfalse
\DeclareOptionX{createtips}{\createtipstrue}

\newif\ifTooltip@usepdftex\Tooltip@usepdftextrue
\DeclareOptionX{dvips}{\Tooltip@usepdftexfalse}

\newif\ifextratext\extratexttrue
\DeclareOptionX{noextratext}{\extratextfalse}

\newif\ifmovetips\movetipsfalse
\DeclareOptionX{movetips}{\movetipstrue}

\newif\ifmouseover\mouseoverfalse
\DeclareOptionX{mouseover}{\mouseovertrue}

\newif\if@fancytooltips@inactive\@fancytooltips@inactivefalse
\DeclareOptionX{inactive}{\@fancytooltips@inactivetrue}

\newif\if@fancytooltips@active\@fancytooltips@activefalse
\DeclareOptionX{active}{\@fancytooltips@activetrue}

\DeclareOptionX{filename}{\xdef\TooltipFilename{#1}}
\DeclareOptionX{pages}{\xdef\TooltipPages{#1}}

\def\fancytooltipsdebugmsg{}
\DeclareOptionX{debug}{\def \fancytooltipsdebugmsg
{
if (this.getField("ikona") == null)
 {app.alert("No buttons for placing tootlips are available. Contact the author. The file may
 else
```

```
  {app.alert("Buttons for placing tooltips are available. Congratulations! Hope everything will
if (this.getField("animtiph") == null)
  {app.alert("No hidden buttons containing tooltips available. The interactive features will no
  else
  {app.alert("Hidden buttons containing tooltips are available. Congratulations! Hope everythin
}}

\ProcessOptionsX

\newdimen\buttontipwidth
\newdimen\buttontipheight
\newdimen\fancy@a
\newdimen\fancy@b

\if@fancytooltips@active\@fancytooltips@inactivefalse\fi

\if@fancytooltips@inactive
\newcommand{\tooltip}{\@ifstar\tooltip@Star\tooltip@NoStar}%
\newcommand{\tooltip@Star}[2]{#1}%
\newcommand{\tooltip@NoStar}[2]{#1}%

\newcommand{\tooltipanim}{\@ifstar\tooltipanim@Star\tooltipanim@NoStar}%
\newcommand{\tooltipanim@Star}[2]{#1}%
\newcommand{\tooltipanim@NoStar}[2]{#1}%
\def\keytip#1{}%
\def\FancyHook{}%
\def\TooltipPage#1#2{}%
\let\TooltipExtratext\relax
\let\TooltipRefmark\relax
\PackageWarning{fancytooltips}{Fancytooltips inactive}%
\expandafter\endinput\fi

```

Each `\ref` command has associated a number and writes new label into aux file. If the target page for the ref is different from the page with this `\ref` command, the corresponding tooltip is inserted.

```
\newcount\fancycheckcount\fancycheckcount=0
\def\fancy@second#1#2#3#4{#2}

\def\FancyHook{
\global\let\oldref\ref
\gdef\ref##1{\oldref{##1}\global\advance\fancycheckcount by 1\relax
\edef\templabel{fancyanchorref:\the\fancycheckcount}%
\expandafter\label\expandafter{\templabel}%
\expandafter\ifx \csname FancyToolTip@##1\endcsname\relax \else\hbox to 0 pt{%
\expandafter\ifx \csname r@##1\endcsname \relax\else
\edef\temp{\csname r@##1\endcsname}%
\expandafter\edef\expandafter\fan@tempa\expandafter{\expandafter\fancy@second\temp}%
\fi
\expandafter\ifx \csname r@fancyanchorref:\the\fancycheckcount\endcsname \relax\else
\edef\fan@ttemp{r@fancyanchorref:\the\fancycheckcount}%
\expandafter\let\expandafter\fan@temp@w\csname \fan@ttemp\endcsname
\edef\fan@temp@ww{\expandafter\fancy@second\fan@temp@w}%
\fi
\ifx\fan@temp@ww\undefined\else
\fi
\ifx\fan@tempa\fan@temp@ww\else\smash{%
  \let\TooltipExtratext\relax\tooltip{\strut\TooltipRefmark}{##1}}%
\fi
\hss}%
\fi}
\ifTooltip@usepdftex
\def\TooltipRefmark{\hbox {\raisebox{0.4em}{\includegraphics[width=0.7em]%
        {fancytipmark.pdf}}}}%
\else
\def\TooltipRefmark{\hbox {\raisebox{0.4em}{\includegraphics[width=0.7em]%
        {fancytipmark.eps}}}}%
\fi%\ifTooltip@usepdftex
}

\ifcreatetips
```

This part (three lines) is processed if the option `createtips` is used. In the opposite case we process the second part, up to the end of the package.

```
\newwrite\tipfile
\immediate\openout\tipfile \jobname.tips
\def\keytip#1{\write\tipfile{\string\tooltipname{#1}{\arabic{page}}}}
\expandafter\endinput\fi
```

This part is processed if the option `createtips` is not used. We define macros which put the hidden button with the name `ikona.n` in the backgoud of the page `n`, if one of the commands `\tooltip` or `\tooltipanim` has been used on this page. Javascripts defined by `\tooltip` and `\tooltipanim` commands then unhide this button and show the corresponding picture.

```

\ifx\TooltipFilename\undefined
\PackageWarning{fancytooltips}{** The filename with tooltips is not given. **}
\fi

\ifTooltip@usepdftex
\RequirePackage[pdftex]{eforms}
\def\TooltipExtratext{\hbox{\raisebox{0.4em}{\includegraphics[width=0.7em]%
        {fancytipmark.pdf}}}}
\else
\RequirePackage[dvips]{eforms}
\def\TooltipExtratext{\hbox{\raisebox{0.4em}{\includegraphics[width=0.7em]%
        {fancytipmark.eps}}}}
\fi%\ifTooltip@usepdftex
\ifextratext\else\let\TooltipExtratext\relax\fi

\AtBeginDocument{
```

```
\global\buttontipwidth=\paperwidth
\global\buttontipheight=\paperheight
}

\ifTooltip@usepdftex
\def\frametip@{%
  \pdfstartlink user{%
    /Subtype /Widget
    /F 6
    /T (ikona.\thepage)
    /FT /Btn
    /Ff 65536
    /H /N
    /BS << /W 1 /S /S >>
    /MK << /TP 1 /IF <</A[1.0 1.0]/SW /B>>  >>
  }%
  \vbox to \buttontipheight {\vss\hbox to \buttontipwidth{\hss}}\pdfendlink}
\else
```

For dvips users we use the macros from eqxerquiz.sty package.

```
\def\everyeqIcon#1{\def\every@eqIcon{#1}}
\def\every@eqIcon{}
\newcommand\eqIconFTT[4][]
{%
  \push@@Button{#1}{#2}{#3}{#4}{}{\eq@setButtonProps\eq@Button@driver}%
  {\eqIconDefaults\every@ButtonField\every@eqIcon}%
}
\def\eqIconDefaults
{%
  \rawPDF{}\S{}\mkIns{/TP 1 /IF<</A[1.0 1.0]/SW/B>>}\R{0}
  \CA{}\RC{}\AC{}\BC{}\BG{}\H{B}
  \textColor{0 g}\Ff{\FfReadOnly}
}
\def\frametip@{\eqIconFTT[\BC{}\BG{}\F{\FHidden}]%
  {ikona.\thepage}{\paperwidth}{\paperheight}}%
\fi%\ifTooltip@usepdftex

\def\frametip{%
  \expandafter\ifx \csname TooltipPage\thepage\endcsname\relax
  \else
  \setbox0=\hbox{\frametip@}%
  \hbox{\raise \dp0 \box0}
  \fi}%
\AddToShipoutPicture{\hbox to 0 pt{\frametip\hss}}
```

In the macros `\tooltip` and `\tooltipanim` we print the text into box with zero dimensions and then we build a button which covers this text and has an associated JavaScript action. The important part is the `\PushButton` macro. You can adjust these macros or write similar macros which do what you need.

```
\definecolor{tooltipcolor}{rgb}{0,0,1}

```

```
\newcount\tooltip@count
\newtoks\tooltip@toks
\newtoks\tooltip@pagetoks
\tooltip@pagetoks={\thepage}
\def\tooltippage{}

\def\TooltipPage#1#2{%
\expandafter\gdef\csname TooltipPage#2\endcsname{#2}%
\expandafter\gdef\csname Tooltipcount2page#1\endcsname{#2}%
}
```

We use the hack for Adobe Acrobat suggested by DPS and Jorg at http://www.acrotex.net/forum/showthre

```

\def\fancy@JS#1{\JS{var DirtyBeforeTooltip=this.dirty; #1
  this.dirty=DirtyBeforeTooltip;}}

\newcommand{\tooltip}{\@ifstar
                     \tooltip@Star%
                     \tooltip@NoStar%
}

\newcommand{\tooltip@Star}[2]{{\color{tooltipcolor}#1}%
  {\let\SaveTooltipExtratext\TooltipExtratext
   \let\TooltipExtratext\relax
   \hbox to 0 pt{\tooltip@NoStar{\strut\SaveTooltipExtratext}{#2}\hss}}}

\newcommand{\tooltip@NoStar}[2]{%
  \global\advance\tooltip@count by 1%
  \edef\act{\write\@auxout{\noexpand\string\noexpand\TooltipPage{\the\tooltip@count}{\the\toolt
  \edef\tooltippage{\csname Tooltipcount2page\the\tooltip@count \endcsname}%
  \checkTipNumber{#2}\edef\TipNumber{\FindTipNumber{#2}}%
  \def\tempfancytooltips{}%
  \ifmovetips\edef\tempfancytooltips{nastav(\TipNumber,\tooltippage);}\fi
  \leavevmode
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{\copy0\smash{\TooltipExtratext}\hss}%
  \fancy@a=\ht0
  \advance\fancy@a by \dp 0
  \fancy@b=0.5\fancy@a
  \advance\fancy@b by -\dp0
  \advance\fancy@b by -2bp
  \lower-\fancy@b\vtop{\hbox{%
  \pushButton[\BC{}\BG{}\S{}\AA{\AAMouseExit{\fancy@JS{CloseTooltips();}}%
  \ifmouseover
  \AAMouseEnter{\fancy@JS{this.getField("ikona."+(\tooltippage)).hidden=false;
      try {app.clearInterval(animace);}catch (e) {}
      \tempfancytooltips
      zobraz(\TipNumber,\tooltippage);
    }}
  \fi}
  \A{\fancy@JS{this.getField("ikona."+(\tooltippage)).hidden=false;
```

```
      try {app.clearInterval(animace);}catch (e) {}
      \tempfancytooltips
      zobraz(\TipNumber,\tooltippage);
    }}]
  {TooltipField}{\wd0}{\fancy@a}}}}

\def\delayinterval{200}

\newcommand{\tooltipanim}{\@ifstar
                     \tooltipanim@Star%
                     \tooltipanim@NoStar%
}

\newcommand{\tooltipanim@Star}[3]{{\color{tooltipcolor}#1}%
  {\let\SaveTooltipExtratext\TooltipExtratext
   \let\TooltipExtratext\relax
   \hbox to 0 pt{\tooltipanim@NoStar{\strut\SaveTooltipExtratext}{#2}{#3}\hss}}}

\newcommand{\tooltipanim@NoStar}[3]{%
  \global\advance\tooltip@count by 1%
  \edef\act{\write\@auxout{\noexpand\string\noexpand\TooltipPage{\the\tooltip@count}{\the\toolt
  \edef\tooltippage{\csname Tooltipcount2page\the\tooltip@count \endcsname}%
  \checkTipNumber{#2}\edef\TipNumberA{\FindTipNumber{#2}}%
  \checkTipNumber{#3}\edef\TipNumberB{\FindTipNumber{#3}}%
  \leavevmode
  \setbox0=\hbox{{\color{tooltipcolor}{#1}}}\hbox to 0 pt{\copy0\smash{\TooltipExtratext}\hss}%
  \def\tempfancytooltips{}%
  \ifmovetips\edef\tempfancytooltips{nastav(\TipNumberA,\tooltippage);}\fi
  \fancy@a=\ht0
  \advance\fancy@a by \dp 0
  \fancy@b=0.5\fancy@a
  \advance\fancy@b by -\dp0
  \advance\fancy@b by -2bp
  \lower-\fancy@b\vtop{\hbox{%
  \pushButton[\BC{}\BG{}\S{}\AA{\AAMouseExit{\fancy@JS{CloseTooltips();}}
  \ifmouseover
  \AAMouseEnter{\fancy@JS{
      try {app.clearInterval(animace);}catch (e) {}
      var cislo=\TipNumberA;
      \tempfancytooltips
      function animuj()
      {
        var DirtyBeforeTooltipanim=this.dirty;
        if (cislo<\TipNumberB) cislo=cislo+1;
        this.getField('ikona.'+(\tooltippage)).buttonSetIcon(this.getField("animtiph."+cislo).b
        this.dirty=DirtyBeforeTooltipanim;
      };
      this.getField('ikona.'+(\tooltippage)).buttonSetIcon(this.getField("animtiph."+\TipNumber
      this.getField("ikona."+(\tooltippage)).hidden=false;
      animace=app.setInterval('animuj();', \delayinterval);
    }}
  \fi}
  \A{\fancy@JS{
      try {app.clearInterval(animace);}catch (e) {}
      var cislo=\TipNumberA;
      \tempfancytooltips
      function animuj()
      {
        var DirtyBeforeTooltipanim=this.dirty;
        if (cislo<\TipNumberB) cislo=cislo+1;
        this.getField('ikona.'+(\tooltippage)).buttonSetIcon(this.getField("animtiph."+cislo).b
        this.dirty=DirtyBeforeTooltipanim;
      };
      this.getField('ikona.'+(\tooltippage)).buttonSetIcon(this.getField("animtiph."+\TipNumber
      this.getField("ikona."+(\tooltippage)).hidden=false;
      animace=app.setInterval('animuj();', \delayinterval);
    }}
  ]{TooltipField}{\wd0}{\fancy@a}}}}
```

This code closes tooltip if the page is closed.

```
\ifTooltip@usepdftex
\def\TooltipPageopencloseJS{  \global\pdfpageattr{%
    /AA << /O << /S /JavaScript /JS (
    var DirtyBeforeCloseTooltip=this.dirty; CloseTooltips(); this.dirty=DirtyBeforeCloseTooltip
}
\pdfximage{\TooltipFilename.pdf}%
\edef\TooltipPages{\the\pdflastximagepages}%
\else
\def\TooltipPageopencloseJS{
\literalps@out{%
     [ {ThisPage} << /AA <<
    /O << /S /JavaScript /JS (var DirtyBeforeCloseTooltip=this.dirty; CloseTooltips(); this.dir
    >> >> /PUT pdfmark}}
\OpenAction{/S /JavaScript /JS (var DirtyBeforeCloseTooltip=this.dirty; CloseTooltips(); this.d
\fi%\ifTooltip@usepdftex
\EveryShipout{\TooltipPageopencloseJS}%

\ifTooltip@usepdftex
\begin{insDLJS}[fancyTooltipsLoaded]{Tooltipsdljs}{DLJS for Tooltips}
  \fancytooltipsdebugmsg

  var animace;
  var fancyTooltipsLoaded = true;

  function CloseTooltips()
  {
    try {this.getField("ikona").hidden=true;}catch (e) {}
    try {app.clearInterval(animace);}catch (e) {}
  }

  function nastav(cislo,strana)
  {
    var f=this.getField("ikona."+(strana));
    var g=this.getField("animtiph."+cislo);
    var sourf=f.rect;
    var sourg=g.rect;
    if ((mouseX+sourg[2]-sourg[0])<sourf[2])
    var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
    else
    var percX=100*(mouseX-sourf[0]-(sourg[2]-sourg[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]
    var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourg[3]));
    if (percX>100) percX=100;
    if (percY>100) percY=100;
    if (percX<0) percX=0;
    if (percY<0) percY=0;
    f.buttonAlignX=percX;
    f.buttonAlignY=percY;
  }

  function zobraz(cislo,strana)
  {
    var f=this.getField("ikona."+(strana));
    var g=this.getField("animtiph."+cislo);
    f.hidden=false;
    f.buttonSetIcon(g.buttonGetIcon());
  }

  this.dirty=false;
  app.focusRect = false;
\end{insDLJS}
\else
\begin{insDLJS}[fancyTooltipsLoaded]{Tooltipsdljs}{DLJS for Tooltips}
  var animace;
  var fancyTooltipsLoaded = true;

  \fancytooltipsdebugmsg

  function CloseTooltips()
  {
    try {this.getField("ikona").hidden=true;}catch (e) {}
    try {app.clearInterval(animace);}catch (e) {}
  }

  function ImportTooltips()
  {
    console.println("importing pictures");
    for (var i=1;i<=\TooltipPages;i++)
    {
      this.insertPages(this.numPages-1,"\TooltipFilename.pdf",(i-1),(i-1));
      var rozm=this.getPageBox("Crop",this.numPages-1);
      this.deletePages(this.numPages-1);
      var p=this.addField("animtiph."+i,"button",0,rozm);
      p.buttonPosition=position.iconOnly;
      p.hidden=true;
      this.getField("animtiph."+i).buttonImportIcon("\TooltipFilename.pdf",(i-1));
    }
    console.println("Imported \TooltipPages pictures, save the PDF file.");
    return(1);
  }

  function nastav(cislo,strana)
  {
    var f=this.getField("ikona."+(strana));
    var g=this.getField("animtiph."+cislo);
    var sourf=f.rect;
    var sourg=g.rect;
    if ((mouseX+sourg[2]-sourg[0])<sourf[2])
    var percX=100*(mouseX-sourf[0])/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]));
    else
    var percX=100*(mouseX-sourf[0]-(sourg[2]-sourg[0]))/((sourf[2]-sourf[0])-(sourg[2]-sourg[0]
    var percY=100*(mouseY-sourf[3])/((sourf[1]-sourf[3])-(sourg[1]-sourg[3]));
    if (percX>100) percX=100;
    if (percY>100) percY=100;
    if (percX<0) percX=0;
    if (percY<0) percY=0;
    f.buttonAlignX=percX;
    f.buttonAlignY=percY;
  }

  function zobraz(cislo,strana)
  {
    var f=this.getField("ikona."+(strana));
    var g=this.getField("animtiph."+cislo);
    f.hidden=false;
    f.buttonSetIcon(g.buttonGetIcon());
  }

  this.dirty=false;
  app.focusRect = false;
\end{insDLJS}
\fi
```

A cycle is used to create hidden buttons. Each button has associated a page from the file with tooltips as icon. These icons are invoked by JavaScripts defined in `\tooltip` and `\tooltipanim` macros.

```
\newcount\tooltip@count
\ifTooltip@usepdftex
\newcommand*{\TooltipHidden}{%
  \count@=0
  \@whilenum\count@<\TooltipPages \do{%
```

```
    \tooltip@count=\count@
    \advance \tooltip@count by 1%
    \bgroup
    \immediate\pdfximage
    page \the\tooltip@count{\TooltipFilename.pdf}%
    \mbox{\leavevmode
      \vbox to 0 pt{\vss\hbox to 0 pt{\pdfstartlink user{%
        /Subtype /Widget
        /F 6
        /T (animtiph.\the\tooltip@count)
        /FT /Btn
        /Ff 65536
        /H /N
        /BS << /W 1 /S /S >>
        /MK <<
        /TP 1
        /I \the\pdflastximage\space 0 R
        /IF << /SW /A >>
        >>
      }%
      \phantom{\pdfrefximage \pdflastximage}%
      \pdfendlink\hss}}}%
    \egroup
    \advance\count@\@ne}%
}
\AddToShipoutPicture*{\hbox to 0 pt{\TooltipHidden}}
\else
\let\TooltipHidden\relax
\fi
```

The keywords for the tooltips can be stored in the file `\TooltipFilename.tips`. The topics in this file are created by `\keytip` macro (see the first part of the code).

```
\AtBeginDocument{\IfFileExists{\TooltipFilename.tips}%
  {\input{\TooltipFilename.tips}
  \PackageInfo{fancytooltips}{Inputting \TooltipFilename.tips.}}%
  {\PackageWarning{fancytooltips}{No file \TooltipFilename.tips!
      Your keywords for tooltips will not work!}}}

\def\tooltipname#1#2{\expandafter\xdef\csname FancyToolTip@#1\endcsname{#2}}

\def\FindTipNumber#1{\expandafter\ifx \csname FancyToolTip@#1\endcsname\relax
  #1\else\csname FancyToolTip@#1\endcsname\fi}

\def\checkTipNumber#1{\expandafter\ifx
  \csname FancyToolTip@#1\endcsname\relax \PackageWarning{fancytooltips}{No
    framenumber is assigned to keyword #1. I assume that #1 is the
    number of the frame.}%
  \fi}

⟨/package⟩
```