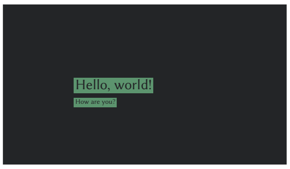
`ppt-slides`: LaTeX Package for Slide Decks à la Power Point[*]

Yegor Bugayenko
yegor256@gmail.com

2024/10/18, 0.4.0

1 Introduction

This package helps you render slide decks in PowerPoint™ style:

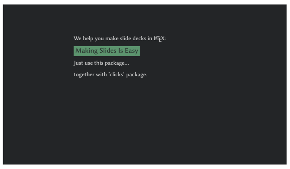


Hello, world!

How are you?

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\begin{pptMiddle}
\pptTitle{Hello, world!}{How are you?}
\end{pptMiddle}
\end{document}
```

It is recommended to use this package together with clicks:



We help you make slide decks in LaTeX:

Making Slides Is Easy

Just use this package...

together with 'clicks' package.

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\usepackage[static]{clicks}
\begin{document}
We help you make slide decks in \LaTeX:
\plick{\pptBanner{Making Slides Is Easy}}
\plick{Just use this package...}
\plush{together with 'clicks' package.}
They will look similar to what PowerPoint can
make, but with the precision of \LaTeX. We
recommend using us together with ``clicks.''
\end{document}
```

2 Layout

`\pptToc`
`\pptChapter`

First, split your story into chapters:

```
\documentclass{article}
\usepackage{clicks}
\usepackage[template,scheme=light]{ppt-slides}
\begin{document}
```

[*] The sources are in GitHub at yegor256/ppt-slides

```
\pptToc
\plush{\pptChapter{About Me}}
...
\plush{\pptChapter[Idea]{My Idea Is Novel}}
...
\plush{\pptChapter[FAQ]{Discussion \& Questions}}
...
\end{document}
```

`\pptToc` will render the table of contents in an interactive "clickable" format. Thanks to the use of crumbs, there will be a navigation at the top left corner.

`\pptSection` Then, put sections inside chapters:

```
\begin{document}
\pptToc
\plush{\pptChapter{About Me}}
\plush{\pptSection[Student]{I'm a Student}}
...
\plush{\pptSection[Athlete]{Also, I'm an Athlete}}
...
\plush{\pptChapter[Idea]{My Idea Is Novel}}
\plush{\pptSection{Novelty}}
\plush{\pptSection{Impact}}
\end{document}
```

The `\pptChapter` and `\pptSection` commands used together will render nice two-level nagivation menu at the top left corner.

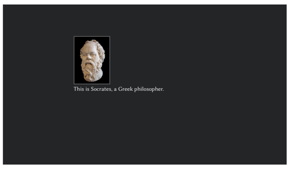
`\pptLeft` `\pptRight` You can place the title of the presentation and your name at the bottom of each slide, on the left and on the right respectively:

```
\documentclass{article}
\usepackage{clicks}
\usepackage[template,scheme=light]{ppt-slides}
\pptLeft{How Did I Spend Last Summer}
\pptRight{Yegor Bugayenko}
\begin{document}
...
\end{document}
```

If you don't use the `\pptLeft` or `\pptRight` commands, nothing will be printed at the bottom.

3 Commands

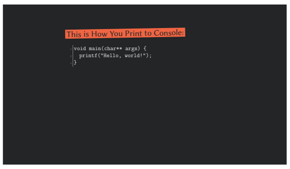
`\pptPic` You can add an image to the slide (the first argument is the width of the image in relation to `\textwidth`, while the second one is the path of it):



This is Socrates, a Greek philosopher.

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\pptPic{0.2}{socrates.jpg} \\
This is Socrates, a Greek philosopher.
\end{document}
```

\pptHeader You can add a piece of code to the slide (we recommend using ffcode):

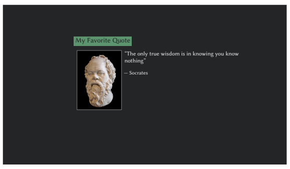

This is How You Print to Console:

```
void main(char** args) {
  printf("Hello, world!");
}
```

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\usepackage{ffcode}
\begin{document}
\pptHeader{This is How You Print to Console:}
\begin{ffcode}
void main(char** args) {
  printf("Hello, world!");
}
\end{ffcode}
\end{document}
```

Don't use \plick or \plush, they won't work with code snippets. The \pptHeader command prints a header similar to what the \pptSection command prints, but doesn't start a new section.

\pptQuote \pptBanner You can put a quote next to the image, by using the \pptQuote command:

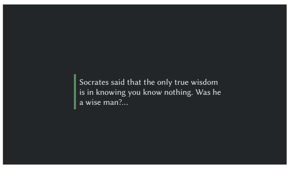

My Favorite Quote

"The only true wisdom is in knowing you know nothing"

— Socrates

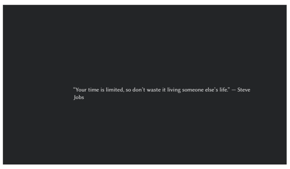
```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\pptBanner{My Favorite Quote}
\pptQuote{socrates.jpg}{The only true wisdom
is in knowing you know nothing}{Socrates}
\end{document}
```

\pptThought For an interesting thought in the middle of the slide, use \pptThought:


Socrates said that the only true wisdom is in knowing you know nothing. Was he a wise man?...

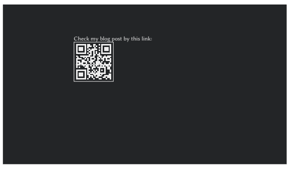
```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\pptThought{Socrates said that the only true
wisdom is in knowing you know nothing. Was he
a wise man?...}
\end{document}
```

pptMiddle You can put your content vertically in the middle of the page, using the pptMiddle environment:


"Your time is limited, so don't waste it living someone else's life." — Steve Jobs

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\begin{pptMiddle}
``Your time is limited, so don't waste it
living someone else's life.'' --- Steve Jobs
\end{pptMiddle}
\end{document}
```

\pptQR Sometimes it's convenient to show a QR code to your audience instead of a URL, since it's easier to use — they can scan it:



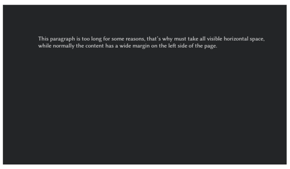
Check my blog post by this link:

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
Check my blog post by this link: \\
\pptQR{https://www.yegor256.com}
\end{document}
```

The code will look like this, thanks to qrcode package that we use behind the scene.

pptWide
pptWideOne

Sometimes you need your slide content to take all visible horizontal space:



This paragraph is too long for some reasons, that's why must take all visible horizontal space, while normally the content has a wide margin on the left side of the page.

```
\documentclass{article}
\usepackage[template,scheme=dark]{ppt-slides}
\begin{document}
\begin{pptWideOne}
This paragraph is too long for some reasons,
that's why must take all visible horizontal
space, while normally the content has a wide
margin on the left side of the page.
\end{pptWideOne}
\end{document}
```

You can also use `\begin{pptWide}{X}`, where `X` is the number of columns to render.

4 Package Options

template

You start with a template for your slide deck:

```
\documentclass{article}
\usepackage[template=9x6]{ppt-slides}
\begin{document}
...
\end{document}
```

There is only one template, which is used by default: `9x6`. If you don't specify the name, it will be used. If you don't use the `template` option at all, the default `article` will be rendered, which is not what you want.

scheme

You can choose a color scheme for your slides, using the `scheme` option of the package:

```
\usepackage[template,scheme=light]{ppt-slides}
```

There are a few out-of-the-box schemes available: `light`, `dark`, `light-mono`, and `dark-mono`. You can design your own, using `ppt-light.tex` file as an example:

```
\usepackage[template,scheme=/usr/local/my-colors.tex]{ppt-slides}
```

directory

You can change the location of the templates and schemes by means of the `directory` package option:

```
\usepackage[directory=/tmp,template=foo]{ppt-slides}
```

nominutes

By default, if you render your slide deck in non-static mode (the `static` option for the "clicks" package), there will be minutes tracking in the right top corner of each slide. You can turn this off by using the `nominutes` option of the package:

```
\usepackage[nominutes]{ppt-slides}
```

5 Implementation

First, we parse the package options:

```
\RequirePackage{pgfopts}
\makeatletter
\pgfkeys{
  /ppt-slides/.is family,
  /ppt-slides,
  template/.estore in = \ppt@template,
  template/.default = 9x6,
  scheme/.estore in = \ppt@scheme,
  directory/.estore in = \ppt@directory,
  directory/.default = ,
  nominutes/.estore in = \ppt@nominutes,
  directory
}
\ProcessPgfPackageOptions{/ppt-slides}
\makeatother
```

Then, we include the template:

```
\makeatletter\ifdefined\ppt@template
  \input{\ppt@directory ppt-templates/ppt-\ppt@template.tex}
\else
  \message{^^Jppt: No template is loaded, because
    the 'template' option is not specified}
\fi\makeatother
```

Then, we include the scheme:

```
\makeatletter\ifdefined\ppt@scheme
  \RequirePackage{xcolor}
  \input{\ppt@directory ppt-schemes/ppt-\ppt@scheme.tex}
\else
  \message{^^Jppt: No color scheme is loaded, because
    the 'scheme' option is not specified}
\fi\makeatother
```

Then, we include ifthen in order to enable `if/then/else` commands:

```
\RequirePackage{ifthen}
```

Then, we include href-ul in order to enable underlined hyperlinks:

```
\RequirePackage{href-ul}
```

Then, we include pagecolor in order to make it possible to change the background color of a page:

```
\RequirePackage{pagecolor}
```

Then, we include varwidth in order to change the width of the page, inside `pptWide`:

```
\RequirePackage{varwidth}
```

Then, we include qrcode in order to render QR codes:

```
\RequirePackage{qrcode}
```

Then, we include tikz in order to position elements on the page, especially in the Table of Contents:

```
\RequirePackage{tikz}
```

Then, we include tikzpagenodes in order to pin a block in `\pptPin`:

```
\RequirePackage{tikzpagenodes}
```

Then, we include enumitem in order to render inline lists in crumbs:

```
\RequirePackage[inline]{enumitem}
```

Then, we include crumbs in order to show top-page menus:

```
\RequirePackage{crumbs}
```

Then, we include calc in order to calculate positions of elements more precisely:

```
\RequirePackage{calc}
```

Then, we include tabularx in order to render `\pptQuote`:

```
\RequirePackage{tabularx}
```

Then, we include seqsplit in order to split long links:

```
\RequirePackage{seqsplit}
```

`\pptMiddle` Then, we define the `pptMiddle` environment:

```
\newenvironment{pptMiddle}
  {\vspace*{\fill}}
  {\vspace*{\fill}}
```

`\pptBanner` Then, we define the `\pptBanner` command:

```
\newcommand\pptBanner[2][green]{%
  \par%
  {\setlength{\fboxsep}{6pt}%
    \colorbox{#1}{\color{\thepagecolor}\large #2}}%
  \par%
}
```

`\pptChapter` Then, we define the `\pptChapter` and `\pptChapterLabel`commands:

```
\newcommand\pptChapterLabel{Chapter}
\newcommand\pptChapter[2][]{%
  \crumbection[#1]{#2}%
  \begin{pptMiddle}%
    {\large\ttfamily \pptChapterLabel{} \#\the\value{section}:}%
    \newline%
    \rotatebox{1}{\pptBanner[blue]{\LARGE#2}}%
  \end{pptMiddle}%
}
```

`\pptSection` Then, we define the `\pptSection` command:

```
\newcommand\pptSection[2][]{%
  \subcrumbection[#1]{#2}%
  \pptHeader{#2}%
}
```

`\pptHeader` Then, we define the `\pptHeader` command:

```
\newcommand\pptHeader[1]{%
  \vspace*{-0.75in}\hspace*{-0.5in}%
  \rotatebox{1}{\pptBanner[orange]{\Large #1}}\vspace*{0.15in}%
  \par%
}
```

\pptTitle

Then, we define the \pptTitle command, which prints the title and subtitle on the first page. The first argument is the title, the second is the subtitle. If the second argument is empty, it won't be printed:

```
\newcommand\pptTitle[2]{%
  {\pptBanner{\Huge #1}}%
  \def\param{#2}%
  \ifx\param\empty\else%
    {\pptBanner{\large #2}}%
  \fi%
}
```

\pptToc

Then, we define the \pptToc command:

```
\newcommand*\pptTOC{}
\newcommand\pptToc[1][]{%
  \renewcommand*\pptTOC{}%
  \print{%
    \def\param{#1}%
    \ifx\param\empty\else%
      \pptBanner{#1}%
    \fi%
    \tikz{%
      \node (z) {%
        \begin{varwidth}{\textwidth}%
          \ifx\pptTOC\empty\else%
            \begin{itemize}[label={}]
              \pptTOC
            \end{itemize}
          \fi%
        \end{varwidth}%
      };%
      \path [draw=blue,line width=4pt] (z.north west) -- (z.south west);%
    }%
  }%
  \begin{@empty}%
    \renewcommand\crumb[2]{%
      \ifx\pptTOC\empty\else\ifdefined\click\click[0]\fi\fi
      \gappto\pptTOC{\item ##2}
    }
    \crumbs
  \end{@empty}%
  \ifdefined\flush\flush[1]\fi%
}
```

\pptQuote

Then, we define the \pptQuote command, with one optional and three mandatory arguments:

```
\newcommand\pptQuote[4][]{%
  \begin{tabularx}{\columnwidth}{c>{\raggedright\arraybackslash}X}%
    \raisebox{\dimexpr-\height+\ht\strutbox}{
      \parbox{0.25\textwidth+2pt}{%
        \raggedright%
        \pptPic{0.25}{#2}%
        \def\person{#1}%
        \ifx\person\empty\else%
```

```
          \par%
          \small%
          \person%
        \fi%
      }%
    }%
    &%
    ‘‘#3{}’’%
    \def\param{#4}%
    \ifx\param\empty\else%
      \par\vspace*{1em}%
      \small%
      --- \param%
      \par%
    \fi%
    \\%
  \end{tabularx}%
}
```

\pptPic Then, we define the \pptPic command:

```
\newcommand\pptPic[2]{%
  {%
    \setlength{\fboxsep}{0pt}%
    \setlength{\fboxrule}{1pt}%
    \fcolorbox{gray}{white}{%
      \includegraphics[width=#1\columnwidth]{#2}%
    }%
  }%
}
```

\pptPin Then, we define the \pptPin command:

```
\newcommand\pptPin[2][right]{%
  \begin{tikzpicture}[remember picture,overlay]%
    \ifthenelse{\equal{#1}{left}}%
    {%
      \node[anchor=north east] at
        ($(current page text area.north west) - (24pt, 18pt)$) {%
        \begin{minipage}{0.25\textwidth}%
          \raggedleft #2%
        \end{minipage}%
      };%
    }%
    {%
      \node[anchor=north east] at (current page text area.north east) {%
        \begin{minipage}{0.25\textwidth}%
          \raggedright #2%
        \end{minipage}%
      };%
    }%
  \end{tikzpicture}%
}
```

\pptThought Then, we define the \pptThought command:

```
\newcommand\pptThought[1]{%
```

```
  \begin{pptMiddle}%
    \tikz{%
      \node [inner sep=18pt] (z) {%
        \begin{varwidth}{0.8\textwidth}%
          \raggedright\Large #1%
        \end{varwidth}%
      };
      \path [draw=green,line width=8pt] (z.north west) -- (z.south west);
    }%
  \end{pptMiddle}%
}
```

\pptSnippet Then, we define the \pptSnippet command:

```
\newcommand\pptSnippet[2][\small]{%
  \begin{samepage}%
    #1\verbatiminput{#2}%
  \end{samepage}%
}
```

\pptQR Then, we define the \pptQR command:

```
\newcommand\pptQR[2][2in]{%
  \tikz{%
    \node[draw=white]{%
      \href{#2}{%
        \qrcode[height=#1]{#2}%
      }%
    }%
  }%
}
```

\pptPinQR Then, we define the \pptPinQR command:

```
\newcommand\pptPinQR[2][right]{%
  \pptPin[#1]{%
    \begin{minipage}{1.4in}%
      \raggedleft%
      \pptQR[1in]{#2}%
      \\[3pt]
      \ttfamily\fontsize{8pt}{10pt}\selectfont%
      \seqsplit{#2} \href{#2}{$\rightarrow$}%
      \\
    \end{minipage}%
  }%
}
```

pptWide Finally, we define the pptWide and pptWideOne environments:

```
\newenvironment{pptWideOne}
  {\begin{adjustwidth}{-2in}{-1in}}
  {\end{adjustwidth}}
\newenvironment{pptWide}[1]
  {\begin{pptWideOne}\begin{multicols}{#1}}
  {\end{multicols}\end{pptWideOne}}
```

```
\endinput
```

Change History

0.0.1

General: First draft. 5

0.1.4

General: The `nominutes` package option added. 5

0.1.5

General: The default directory fixed. . . 5

0.2.0

General: We migrated to the `ltxdoc` format and `.dtx` file. 5

0.2.1

`\pptMiddle`: The `\pptMiddle` command turned into `pptMiddle` environment. 6

0.3.0

`\pptQuote`: The `\pptQuote` command has got an optional parameter, where the name of the person on the photo can be specified. 7

`\pptThought`: The `\pptThought` command aligns text to the left. . . . 8

0.3.1

General: The `calc` package added. . . . 6

0.4.0

`\pptChapter`: The `\pptChapterLabel` command introduced. 6

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.