Einführung in das Textsatzsystem LaTeX

Eigene Klassen und Pakete schreiben

Maximilian Jalea
latexkurs@mntl.de

28. Januar 2019

Übersicht

Begrüßung

\NeedsTeXFormat{⟨*Format*⟩}[⟨*Datum*⟩]
\ProvidesPackage{⟨*Name*⟩}[⟨*Datum*⟩ ⟨*weitere Infos*⟩]
\ProvidesClass{⟨*Name*⟩}[⟨*Datum*⟩ ⟨*weitere Infos*⟩]

```
\NeedsTeXFormat{LaTeX2e}[1996/12/01]
\ProvidesPackage{meinpaket}[2015/02/05 v0.1 Dolles Paket]
```

Pakete Laden

```
\RequirePackage[⟨Paketoptionen⟩]{⟨Paket⟩}[⟨Datum⟩]
\LoadClass[⟨Klassenoptionen⟩]{⟨Klasse⟩}[⟨Datum⟩]
```

Pakete Laden

```
\RequirePackage[⟨Paketoptionen⟩]{⟨Paket⟩}[⟨Datum⟩]
\LoadClass[⟨Klassenoptionen⟩]{⟨Klasse⟩}[⟨Datum⟩]
```

Eigene Optionen an geladenes Paket weitergeben:

```
\RequirePackageWithOptions{⟨Paket⟩}[⟨Datum⟩]
\LoadClassWithOptions{⟨Klasse⟩}[⟨Datum⟩]
```

Eine Klasse kann nur (einmal) von einer Klasse geladen werden

```
\RequirePackage[hmargin=3cm]{geometry}
```

Optionen

- Option definieren:
  `\DeclareOption{`⟨*Option*⟩`}{`⟨*Code*⟩`}`
- nicht definierte Optionen verwenden:
  `\DeclareOption*{`⟨*Code*⟩`}`
- Optionen verarbeiten:
  `\ProcessOptions`
- innerhalb von `\DeclareOprtion*`:
  `\CurrentOption`
  `\OptionNotUsed`

```
\DeclareOption{a4paper}{%
  \setlength{\paperheight}{297mm}%
  \setlength{\paperwidth}{210mm}%
}
\DeclareOption*{\OptionNotUsed}
\ProcessOptions
```

key=value-Optionen

Klassen/Paketoptionen mit Key-Value-Syntax lassen sich zum Beispiel mit `kvoptions` realisieren.

```
\SetupKeyvalOptions{
  family=meinpaket,
  prefix=mypkg@
}
\DeclareStringOption[default]{mystring}
\DeclareBoolOption{mybool}
\ProcessKeyvalOptions{mypkg}
```

Makros Definieren

- Befehl definieren:
  `\newcommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Befehl umdefinieren:
  `\renewcommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Befehl nur definieren, falls er nicht existiert:
  `\providecommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Testen ob ein Befehl (genau so) definiert ist:
  `\CheckCommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`

Makros Definieren

- Befehl definieren:
  `\newcommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Befehl umdefinieren:
  `\renewcommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Befehl nur definieren, falls er nicht existiert:
  `\providecommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`
- Testen ob ein Befehl (genau so) definiert ist:
  `\CheckCommand{`⟨*Befehl*⟩`}[`⟨*Anzahl*⟩`][`⟨*Default*⟩`]{`⟨*Definition*⟩`}`

Oder mit LaTeX3-Syntax (`expl3`, siehe letzte Vorlesung)

Nützliche Befehle

Code zu verschiedenen Zeitpunkten ausführen

`\AtBeginDocument{`⟨*Code*⟩`}`
`\AtEndDocument{`⟨*Code*⟩`}`

`\AtEndOfPackage{`⟨*Code*⟩`}`
`\AtEndOfClass{`⟨*Code*⟩`}`

Mit dem Nutzer sprechen

\typeout{⟨*Nachricht*⟩}
\PackageInfo{⟨*Paket*⟩}{⟨*Nachricht*⟩}
\PackageWarning{⟨*Paket*⟩}{⟨*Nachricht*⟩}
\PackageWarningNoLine{⟨*Paket*⟩}{⟨*Nachricht*⟩}
\PackageError{⟨*Paket*⟩}{⟨*Nachricht*⟩}{⟨*Hilfetext*⟩}

```
\PackageInfo{meinpaket}{Dies ist eine Info.}
\PackageWarning{meinpaket}{Dies ist eine Warnung.}
\PackageError{meinpaket}{Dies ist ein Fehler.}{Fehler lässt sich nicht beheben.}
```

Paket einbinden

Im Dokument: `\usepackage{meinpaket}`
`meinpaket.sty` muss im selben Ordner liegen

Paket einbinden

Im Dokument: `\usepackage{meinpaket}`
`meinpaket.sty` muss im selben Ordner liegen

Alternative: TEX durchsucht alle Ordner des TDS-Baums
Lokale Pakete können in `$TEXMFHOME` abgelegt werdern

Pakete ausliefern

Programm `DocStrip` kann aus einer Datei verschiedene Ausgabe-Dokumente erstellen.

1. Lösche alle Zeilen, die mit `%` anfangen → `sty` oder `cls`
2. Lösche alle `%` die am Anfang der Zeile stehen → `pdf`

In Overleaf ausprobieren:

http://qn3.de/tex1400

Beispiel dtx 1/4

```
% \iffalse meta-comment
% Copyright (C) 2015 by Lieschen Müller
% \fi \iffalse
%<driver>\ProvidesFile{meinpaket.dtx}
%<package>\NeedsTeXFormat{LaTeX2e}[2007/07/20]
%<package>\ProvidesPackage{meinpaket}[2015/02/05 v0.1 Dolles Paket]
%<*batchfile>
\begingroup
\input{docstrip.tex}
\preamble
Copyright (C) 2015 by Lieschen Müller
\endpreamble
\askforoverwritefalse
\generate{\file{meinpaket.sty}{\from{meinpaket.dtx}{package}}}
\endgroup
%</batchfile>
```

Beispiel dtx 2/4

```
%<*driver>
\documentclass{ltxdoc}
\usepackage[ngerman,english]{babel}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\begin{document}
\DocInput{meinpaket.dtx}
\end{document}
%</driver>
% \fi
% \CheckSum{0}
%
% \changes{v0.1}{2015/02/05}{Initial version}
%
% \GetFileInfo{meinpaket.dtx}
%
```

Beispiel dtx 3/4

```
% \title{Mein Paket\thanks{Diese Anleitung bezieht sich auf Version \fileversion}}
% \author{Lieschen Müller}
% \date{\filedate}
% \maketitle
%
% \begin{abstract}
%   \noindent Dieses tolle Paket tut tolle Dinge.
% \end{abstract}
%
% \tableofcontents
%
% \section{Anleitung}
% So funktioniert mein tolles Paket ...
%
```

Beispiel dtx 4/4

```
% \section{Implementierung}
% So habe ich mein Paket implementiert:
%
% \iffalse
%<*package>
% \fi
%    \begin{macrocode}
\providecommand{\meinbefehl}{Hier steht der eigentliche Inhalt des Pakets}
%    \end{macrocode}
% \iffalse
%</package>
% \fi
%
\endinput
```

Weiterführende Literatur

The LaTeX3 Project.
„LaTeX 2ε for class and package writers“.
`texdoc clsguide`

Scott Pakin.
„How to Package Your LaTeX Package“.
`texdoc dtxtut`

Frank Mittelbach u. a.
„The DocStrip program“.
`texdoc docstrip`