Progress of fontspec and unicode-math

Will Robertson

JULY 22, 2018
Setting the scene

My grandpa has **lots** of books that help him mend **anything**,.
Gently giraffe didn’t bother trying to speak to them from tree to tree munching on the leaves, creatures went on avoiding him. That was the whole of the long dry summer.
Hi all

I've got a first release ready of a package for XeLaTeX that allows dynamic font loading, supporting all of the rich font features in AAT. It doesn't yet support OpenType, but that's coming.

It allows you to use commands like
\typespec[NumberCase=OldStyle, NumberSpacing=Monospaced]{Hoefler Text}
or
\typespec[Variant=5]{Zapfino}
to select a very broad selection of fonts.

It's only new, so it will definitely be improved!
Introduction

unicode-math – modern expl3 package development
  Code structure
  CHANGES file
  expl3 conventions
  Git branches
  Test suite
  Release checklist

fontspec – selecting fonts
  Font loading
  The interface for font features
  Typical example
  ‘Strong’ emphasis
  Custom encodings

Conclusion
Evolution

• \LaTeX \text{ was my introduction to software engineering — scarily enough}
• fontspec and unicode-math were initially released pre-expl3
• expl3 was needed to advance them beyond crude data-structures and algorithms
• Their programming styles evolved with expl3
• Has taken quite some time to become ‘respectable’!
Introduction

To discuss

- fontspect – selecting fonts
- unicode-math – learning from my mistakes in best practices for \TeX{} software development
Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file
expl3 conventions
Git branches
Test suite
Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
(Live demo to re-introduce the package.)
unicode-math – modern expl3 package development

Code structure

Introduction

unicode-math – modern expl3 package development

Code structure

CHANGES file
expl3 conventions
Git branches
Test suite
Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
Code structure

• Once upon a time: monolithic \texttt{dtx} files
• Now: (thanks Joseph!)
  • \texttt{⟨pkg⟩.dtx}: provide metadata
Once upon a time: monolithic \texttt{dtx} files

Now: (thanks Joseph!)
  \begin{itemize}
    \item \texttt{⟨pkg⟩.dtx}: provide metadata
    \item \texttt{⟨pkg⟩-code-⟨module⟩.dtx}
  \end{itemize}
Code structure

- Once upon a time: monolithic \texttt{dtx} files
- Now: (thanks Joseph!)
  - \texttt{\langle pkg\rangle.dtx}: provide metadata
  - \texttt{\langle pkg\rangle-code-\langle module\rangle.dtx}
  - \texttt{\langle pkg\rangle.ins}: the standard Docstrip driver
unicode-math – modern expl3 package development

Code structure

- Once upon a time: monolithic \texttt{dtx} files
- Now: (thanks Joseph!)
  - \texttt{⟨pkg⟩.dtx}: provide metadata
  - \texttt{⟨pkg⟩-code-⟨module⟩.dtx}
  - \texttt{⟨pkg⟩.ins}: the standard Docstrip driver
  - \texttt{⟨pkg⟩-code.1tx}: typeset code implementation
• Once upon a time: monolithic \texttt{dtx} files
• Now: (thanks Joseph!)
  • \texttt{⟨pkg⟩.dtx}: provide metadata
  • \texttt{⟨pkg⟩-code-⟨module⟩.dtx}
  • \texttt{⟨pkg⟩.ins}: the standard Docstrip driver
  • \texttt{⟨pkg⟩-code.ltx}: typeset code implementation
  • \texttt{⟨pkg⟩.ltx}: typeset user documentation
Code structure

- Once upon a time: monolithic `dtx` files
- Now: (thanks Joseph!)
  - `<pkg>.dtx`: provide metadata
  - `<pkg>-code-<module>.dtx`
  - `<pkg>.ins`: the standard Docstrip driver
  - `<pkg>-code.ltx`: typeset code implementation
  - `<pkg>.ltx`: typeset user documentation
  - `<pkg>-doc-<chapter>.tex`
Introduction
unicode-math – modern expl3 package development
Code structure
CHANGES file
expl3 conventions
Git branches
Test suite
Release checklist
fontspec – selecting fonts
Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings
Conclusion
unicode-math – modern expl3 package development

CHANGES file

CHANGE HISTORY
==============

## v0.8m ()

* Restore behaviour of legacy syntax `x_\mathrm{x}` (i.e., with no braces).
  While strictly ‘incorrectly’, this usage is widely used.

## v0.8l (2018/02/02)

* Issue an error message if `\setmathfont{...}[range=...]}` declaration inherently implies a subset, so a ‘main’ math font needs to be set up first.
* Fix issue when nesting `\mathXX` and `\symZZ` commands.
* ...

### Reference

1. [Change History](#)

---

**Note:**

This document is a changelog for the `unicode-math` package, focusing on version updates and feature enhancements. It provides a detailed history of changes, including bug fixes and new functionalities, aimed at improving the package's compatibility and usability in mathematical typesetting.
unicode-math – modern expl3 package development

expl3 conventions

Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file

expl3 conventions

Git branches
Test suite
Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
‘Auto-checking’:
\usepackage[enable-debug]{expl3}
\ExplSyntaxOn
\debug_on:n { check-declarations, check-expressions, deprecation }
\ExplSyntaxOff

• Indentation
• Variables defined up front
• Separation between internal and user-facing commands
• ...
unicode-math – modern expl3 package development

Git branches

Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file
expl3 conventions

Git branches

Test suite
Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong‘ emphasis
Custom encodings

Conclusion
unicode-math – modern expl3 package development

Git branches

Branches
Test suite

Introduction

unicode-math – modern expl3 package development

Code structure

CHANGES file

expl3 conventions

Git branches

Test suite

Release checklist

fontspec – selecting fonts

Font loading

The interface for font features

Typical example

‘Strong’ emphasis

Custom encodings

Conclusion
Early days of the test suite:

- \TeX \rightarrow \text{PDF} \rightarrow \text{PNG} \rightarrow \text{ImageMagick}
- A horribly-fragile and hard-to-read Makefile
- Pixel by pixel comparisons
- Slow, lots of false negatives
- Nonetheless, a large number of tests produced
\texttt{l3build} brought automated unit testing to the masses:

- Just wrap \texttt{\loggingout\out\out\out} around everything is fine?
\texttt{l3build} brought automated unit testing to the masses:

- Just wrap \texttt{\log}ging output around everything is fine?
- It is really not fine.
\texttt{l3build} brought automated unit testing to the masses:

- Just wrap \texttt{\textbackslash loggingoutput} around everything is fine?
- It is really not fine.
- Slowly re-write all tests with custom, minimal, logging.
Example test input

\input{umtest-preamble}

\usepackage{fontspec}
\setmathsf{texgyre-recurser-regular.otf}
\usepackage{unicode-math}

\begin{document}
\START
\TESTBOX{$\mathsf{X}=X$}
\end{document}
Example test output

This is a generated file for the l3build validation system.
Don't change this file in any respect.

> \box...=
\hbox(0.0+0.0)x0.0
  \mathon
  .\TU/texgyrecursor-regular.otf(0)/m/n/10 glyph#116
  .\glue(\thickmuskip) 2.77779 plus 2.77779
  .\TU/latinmodern-math.otf(1)/m/n/10 glyph#30
  .\glue(\thickmuskip) 2.77779 plus 2.77779
  .\TU/latinmodern-math.otf(1)/m/n/10 glyph#1293
  .\kern0.51
  .\mathoff
! OK.
<to be read again>
  \relax
1. ...
\TESTBOX{$\mathsf{X}=X$}
unicode-math – modern expl3 package development

Release checklist

Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file
expl3 conventions
Git branches
Test suite

Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
unicode-math – modern expl3 package development

Release checklist

1. Finish final changes on working branch
2. Ensure CHANGES.md is up-to-date with a new version number
3. build setversion
4. Update local distro fully with tlmgr
5. Run build check locally
6. git push
7. Check Travis build status
8. Install prerelease versions of fontspec and latex3 and re-check
9. git checkout master; git rebase working
10. build ctan
11. Upload to CTAN
12. texlua tagrelease.lua to tag release with version number, annotated with changes
13. git push — assuming tags are pushed by default (might need a local gitconfig)
14. git checkout working
15. build install
16. Check latex3/contrib/testfiles/unicode-math001.lvt and update if necessary
17. build uninstall
fontspec – selecting fonts

Introduction
unicode-math – modern expl3 package development
  Code structure
  CHANGES file
  expl3 conventions
  Git branches
  Test suite
  Release checklist
fontspec – selecting fonts
  Font loading
  The interface for font features
  Typical example
  ‘Strong’ emphasis
  Custom encodings

Conclusion
fontspec – selecting fonts

fontspec’s interface

• Originally very simple
• Based around AAT font features, not OpenType!
• The Graphite font renderer needs more attention
• A rewrite probably won’t happen, but a slimmed-down ‘\LaTeX3’ version might
Font loading

Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file
expl3 conventions
Git branches
Test suite
Release checklist

fontspec – selecting fonts

Font loading
The interface for font features
Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
fontspec – selecting fonts
Font loading

How do I load a font in fontspec?

• \fontspec
• \setmainfont
• \newfontfamily
• \defaultfontfeatures

As the package has grown it is probably less than clear!
\texttt{Xe\TeX} was originally written to load fonts from the OS:
\verbatim{\setmainfont{Hoefler Text}}\% -- `just works'
luaotfload followed, and now:
\verbatim{\setmainfont{TeX Gyre Pagella}}\% -- `just works'
fontspec – selecting fonts
Font loading

File names

But also:
\setmainfont{texgyrepagella-regular.otf}[

\texttt{ItalicFont} = \texttt{texgyrepagella-italic.otf},
\texttt{BoldFont} = \texttt{texgyrepagella-bold.otf},
\texttt{BoldItalicFont} = \texttt{texgyrepagella-bolditalic.otf},
]
(or)
File names

But also:

\setmainfont{texgyrepagella}[
  Extension = .otf ,
  UprightFont = *-regular ,
  ItalicFont = *-italic ,
  BoldFont = *-bold ,
  BoldItalicFont = *-bolditalic ,
]

Which is better? This is the approach I now recommend.
The case against font names

1. Edge cases
   - Sometimes the correct italic/bold shape isn’t picked up

2. Document portability
   - \( \LaTeX \)/luaotfload differences
   - Replicating font installation across computers
   - Differences in software/font versions → different names

3. Speed
   - Generating the font database is slow
   - Installing 100s of fonts in a system directory can be slow
fontspec – selecting fonts
The interface for font features

Introduction

unicode-math – modern expl3 package development

Code structure
CHANGES file
expl3 conventions
Git branches
Test suite
Release checklist

fontspec – selecting fonts

Font loading

The interface for font features

Typical example
‘Strong’ emphasis
Custom encodings

Conclusion
\font\x="[EBGaramond12-Regular.\texttt{otf}]"

Theory 123

\font\x="[EBGaramond12-Regular.\texttt{otf}]:+lnum;+dlig"

Theory 123
fontspec – selecting fonts
The interface for font features

\fontspec{EBGaramond12-Regular.otf}

Theory 123

\fontspec{EBGaramond12-Regular.otf}[
  Numbers = Lining ,
  Ligatures = Discretionary ,
]

Theory 123
fontspec – selecting fonts

Typical example

Introduction

unicode-math – modern expl3 package development

Code structure

CHANGES file

expl3 conventions

Git branches

Test suite

Release checklist

fontspec – selecting fonts

Font loading

The interface for font features

Typical example

‘Strong’ emphasis

Custom encodings

Conclusion
Consider Gill Sans Nova. Weights:

- UltraLight
- Light
- Book
- Medium
- Semibold
- Bold
- Heavy
- ExtraBold
- UltraBold

Do people want to control these with commands like \textbolder and \textlighter?
Typical example

‘Weight’ is relatively simple. Also have:

- CnUltraLight
- CnLight
- CnBook
- CnMedium
- CnSemibold
- CnBold
- CnHeavy
- CnExtraBold
- CnUltraBold
And then the variants:

- **Deco-Regular**
- **Shadowed-Light**
- **Shadowed-Medium**
- **SHADOWED-OUTLN**
- **INLINE-COND**

- **INLINE-EXTRALT**
- **INLINE-LIGHT**
- **INLINE-REGULAR**
- **INLINE-BOLD**
fontspec – selecting fonts

Typical example

• Most of the time I’m just after a ‘normal’ and a ‘bold’.
fontspec – selecting fonts

Typical example

- Most of the time I’m just after a ‘normal’ and a ‘bold’.

- Create `gill-sans-nova.fontspec`:
  
  ```latex
  \defaultfontfeatures[gill-sans-nova]{
    \UprightFont = GillSansNova-Medium.otf ,
    \ItalicFont  = GillSansNova-MediumItalic.otf ,
    \BoldFont    = GillSansNova-Bold.otf ,
    \BoldItalicFont = GillSansNova-BoldItalic.otf ,
  }
  ```
fontspec – selecting fonts

Typical example

• Most of the time I’m just after a ‘normal’ and a ‘bold’.
• Create gill-sans-nova.fontspec:

\defaultfontfeatures[gill-sans-nova]{
  UprightFont = GillSansNova-Medium.otf ,
  ItalicFont = GillSansNova-MediumItalic.otf ,
  BoldFont = GillSansNova-Bold.otf ,
  BoldItalicFont = GillSansNova-BoldItalic.otf ,
}

• Now I can write \setmainfont{gill-sans-nova}.
fontspec – selecting fonts

Typical example

• Most of the time I’m just after a ‘normal’ and a ‘bold’.
• Create gill-sans-nova.fontspec:

\defaultfontfeatures[gill-sans-nova]{
  UprightFont = GillSansNova-Medium.otf,
  ItalicFont = GillSansNova-MediumItalic.otf,
  BoldFont = GillSansNova-Bold.otf,
  BoldItalicFont = GillSansNova-BoldItalic.otf,
}

• Now I can write \setmainfont{gill-sans-nova}.
• Or, semantically,

\newfontfamily\captionfont{gill-sans-nova}
The full power of the \texttt{nfss} is supported:
\begin{verbatim}
\defaultfontfeature+[gill-sans-nova]{
  FontFace = \{uu\}\{m\}\{ GillSansNova-UltraLight.otf \} ,
  FontFace = \{ll\}\{m\}\{ GillSansNova-Light.otf \} ,
  FontFace = \{hh\}\{m\}\{ GillSansNova-Heavy.otf \} ,
  FontFace = \{xx\}\{m\}\{ GillSansNova-ExtraBold.otf \} ,
}\end{verbatim}
Introduction

unicode-math – modern expl3 package development

- Code structure
- CHANGES file
- expl3 conventions
- Git branches
- Test suite
- Release checklist

fontspec – selecting fonts

- Font loading
- The interface for font features
- Typical example
  - ‘Strong’ emphasis
- Custom encodings

Conclusion
fontspec – selecting fonts
‘Strong’ emphasis

Emphasis and inner emphasis

- \LaTeX 2\epsilon supports \eminnershape for markup with nested emphasis
fontspec – selecting fonts
‘Strong’ emphasis

Emphasis and inner emphasis

- \LaTeX{} 2\epsilon{} supports \eminnershape{} for markup with nested emphasis
- fontspec supports arbitrary nesting using (say) \emfontdeclare{}{\itshape,\upshape\scshape,\itshape}
fontspec – selecting fonts
‘Strong’ emphasis

Emphasis and inner emphasis

• \LaTeX{} 2ε supports \eminnershape for markup with nested emphasis
• fontspec supports arbitrary nesting using (say) \emfontdeclare{\itshape,\upshape\scshape,\itshape}
• Ex.:

\begin{verbatim}
\textit{Rm \textbf{AAA} \textbf{BBB} \textbf{III}}
\end{verbatim}
fontspec – selecting fonts

‘Strong’ emphasis

Strong and inner strong

• And more recently...\texttt{strong}!
fontspec – selecting fonts
‘Strong’ emphasis

Strong and inner strong

• And more recently...
• \strongfontdeclare{
  \bfseries,
  \fontseries{hh}\selectfont,
  \fontseries{xx}\selectfont,
}

Ex.: Abc \strong {Abc \strong {Abc \strong {Abc}}}

fontspec – selecting fonts

‘Strong’ emphasis

Strong and inner strong

- And more recently...\texttt{\textbf{strong}}!
- \texttt{\textbf{strongfontdeclare}{\texttt{\textbf{bfseries,}}}}
  \texttt{\texttt{\fontseries{hh}\selectfont,}}
  \texttt{\fontseries{xx}\selectfont,}
}
- Ex.:

\texttt{Abc \textbf{Abc} \textbf{Abc} \textbf{Abc} \textbf{Abc}}
fontspec – selecting fonts
Custom encodings

Introduction

unicode-math – modern expl3 package development
  Code structure
  CHANGES file
  expl3 conventions
  Git branches
  Test suite
  Release checklist

fontspec – selecting fonts
  Font loading
  The interface for font features
  Typical example
  ‘Strong’ emphasis

Custom encodings

Conclusion
• Although everything is now Unicode, \LaTeX’s idea of ‘encodings’ is still useful
\newfontfamily\sanskitfont{charis}

...{\sanskitfont KALITA\d M}... \% <- uses real accent

KALITAM Hã
\newfontfamily\oopsfont {posterama}

...{\oopsfont KALITA\d M}... % <- uses real accent

KALITAM.
\newfontfamily\titlefont{posterama}[
  NFSSEncoding=fakedotaccent
]
...
\titlefont KALITA\d M}...  \% <- uses fake accent

KALITAM
In the preamble:
\DeclareUnicodeEncoding{fakedotaccent}{
  \input{tuenc.def}
  \EncodingCommand{\d}[1]{%
    \hmode@bgroup
    \o@lign{\relax#1\cr\hidewidth
      \ltx@sh@ft{-1ex}.\hidewidth}%
    \egroup
  }
}
KALITAM.

(1901)
fontspec – selecting fonts
Custom encodings

KALITAM

(1913)
KALITAM.

(1919)
KALITAM.

(1927)
fontspec – selecting fonts
Custom encodings

KALITAM.

(1933)
fontspec – selecting fonts
Custom encodings

KALITAM

(1945)
KALITAM

(1984)
fontspec – selecting fonts

Custom encodings

KALITAM

(2001)
Conclusion

Introduction
unicode-math – modern expl3 package development
  Code structure
  CHANGES file
  expl3 conventions
  Git branches
  Test suite
  Release checklist
fontspec – selecting fonts
  Font loading
  The interface for font features
  Typical example
  ‘Strong’ emphasis
  Custom encodings

Conclusion
Conclusion

• Thanks to everyone
  (too many to count but especially [redacted])
• Thanks for patience
• Obrigado