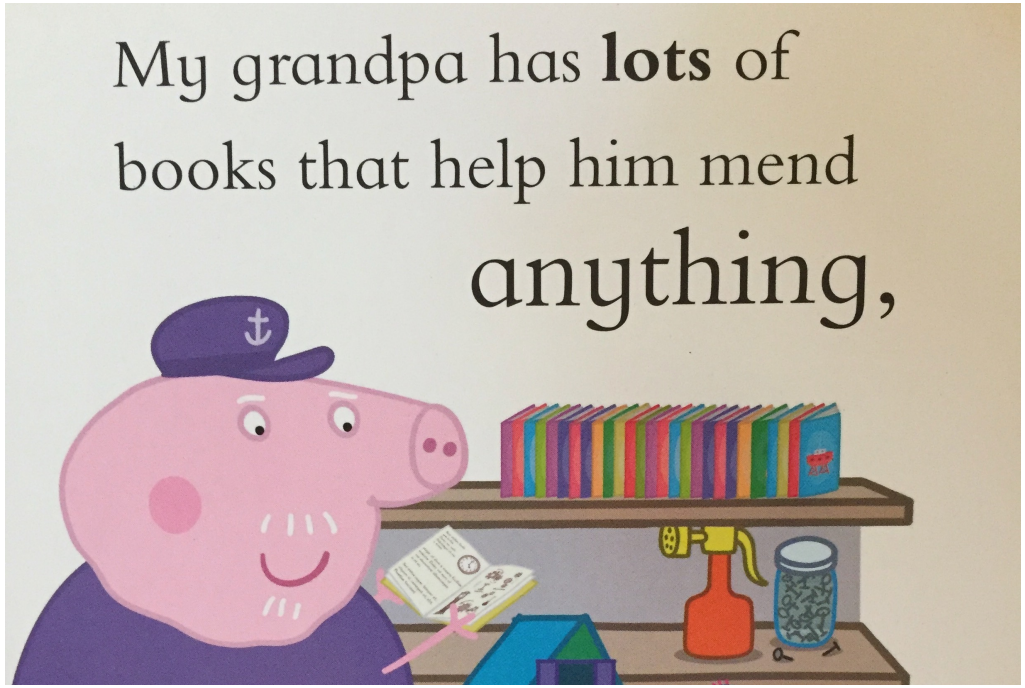
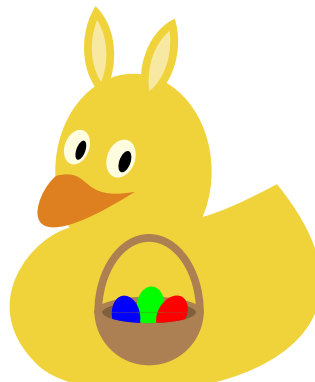
Progress of fontspec and unicode-math

Will Robertson

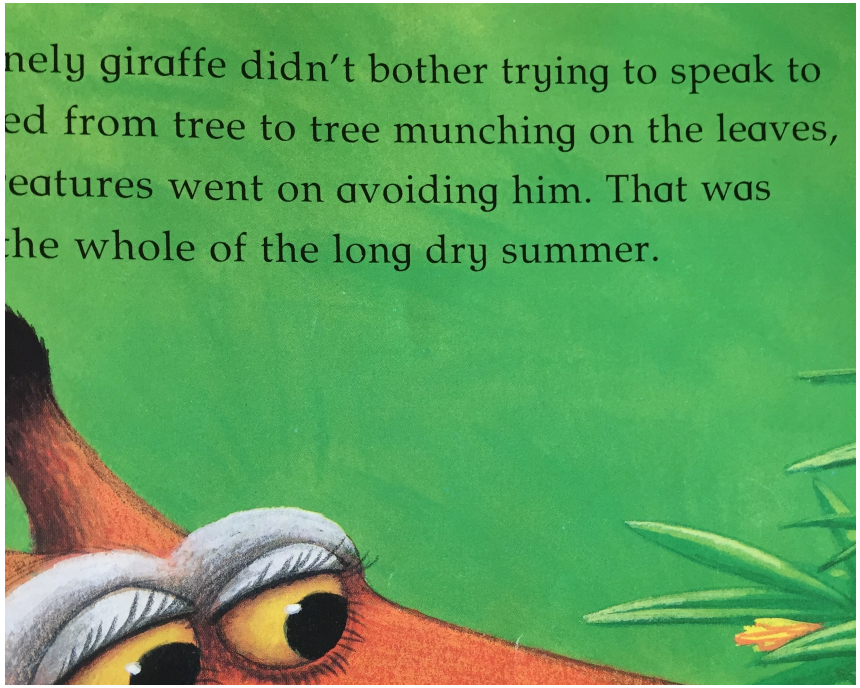
JULY 22, 2018

Setting the scene



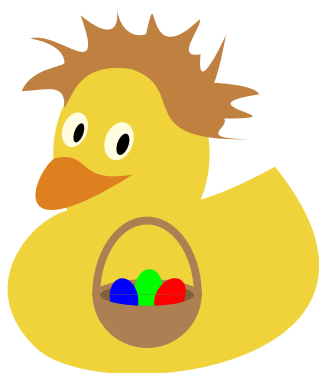


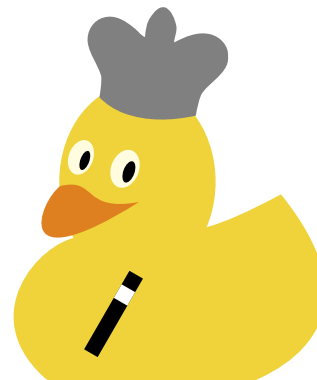
Setting the scene



How long has it been??

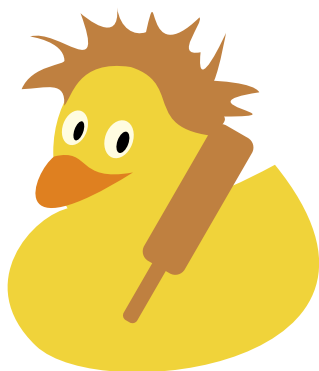
```
[XeTeX] Package for font loading

Will Robertson will at guerilla.net.au
Fri Oct 15 12:04:24 CEST 2004

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 a 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!
```





Evolution

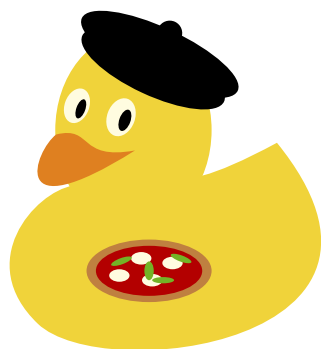
- LaTeX was my introduction to software engineering — scarily enough
- fontspec and unicode-math were initially released pre-expl3
- expl3 was needed to advance them beyond crude data-structures and algorithms
- Their programming styles evolved with expl3
- Has taken quite some time to become ‘respectable’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



To discuss

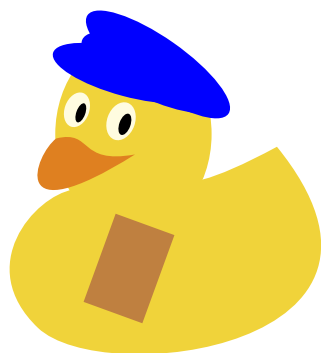
- fontspec – selecting fonts
- unicode-math – learning from my mistakes in best practices for TEX software development

(*Live demo to re-introduce the package.*)



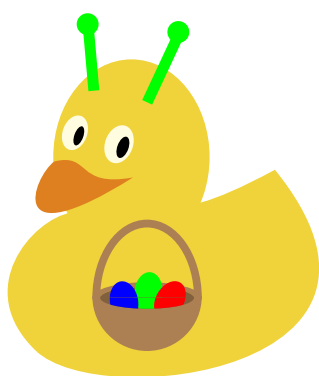
Code structure

- Once upon a time: monolithic DTX files
- Now: (thanks Joseph!)
  - ⟨*pkg*⟩`.dtx`: provide metadata



Code structure

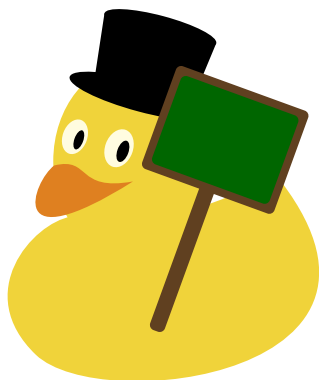
- Once upon a time: monolithic DTX files
- Now: (thanks Joseph!)
  - ⟨*pkg*⟩`.dtx`: provide metadata
  - ⟨*pkg*⟩`-code-`⟨*module*⟩`.dtx`



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

- 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

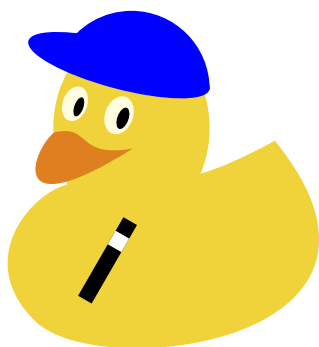


Code structure

- 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

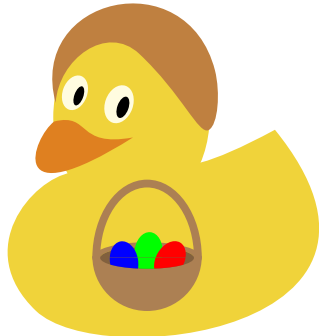
- Once upon a time: monolithic DTX files
- Now: (thanks Joseph!)
  - ⟨*pkg*⟩`.dtx`: provide metadata
  - ⟨*pkg*⟩`-code-`⟨*module*⟩`.dtx`
  - ⟨*pkg*⟩`.ins`: the standard Docstrip driver
  - ⟨*pkg*⟩`-code.ltx`: typeset code implementation
  - ⟨*pkg*⟩`.ltx`: typeset user documentation
  - ⟨*pkg*⟩`-doc-`⟨*chapter*⟩`.tex`

CHANGES file

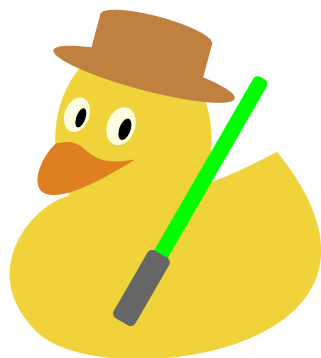


CHANGES file

```
CHANGE HISTORY
==============

## v0.8m ()

  * Restore behaviour of legacy syntax `x_\mathrm{x}` (i.e
    While strictly ‘incorrectly’, this usage is widely usec

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

  * Issue an error message if `\setmathfont{...}[range=...]
    declaration inherently implies a subset, so a ‘main’ ma
  * Fix issue when nesting `\mathXX` and `\symZZ` commands
  * ...
```



expl3 conventions

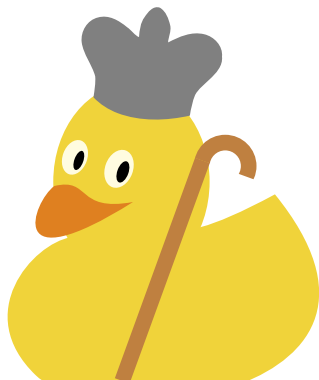


expl3 conventions

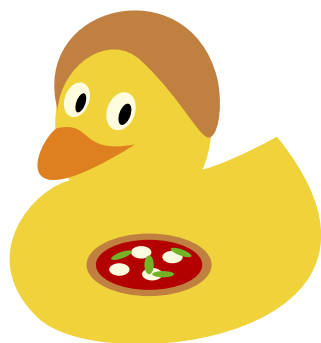
- ‘Auto-checking’:

```
\usepackage[enable-debug]{expl3}
\ExplSyntaxOn
\debug_on:n {
  check-declarations,
  check-expressions,
  deprecation
}
\ExplSyntaxOff
```

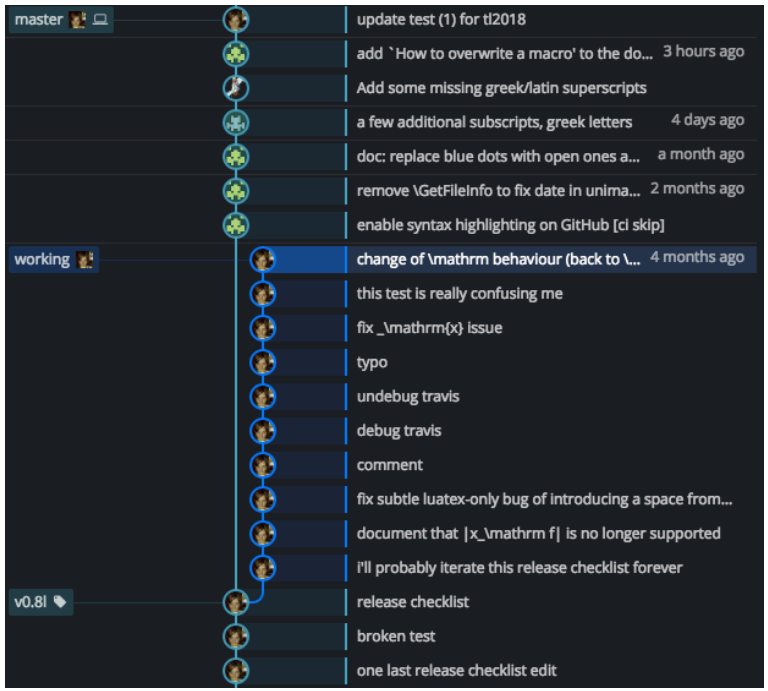
- Indentation
- Variables defined up front
- Separation between internal and user-facing commands
- …

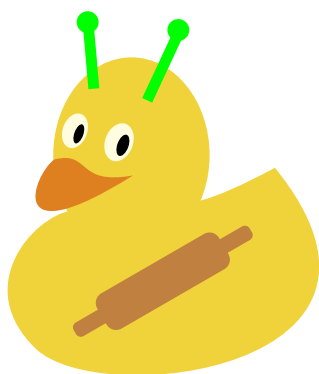


Git branches



Branches


master
update test (1) for tl2018
add `How to overwrite a macro' to the do... 3 hours ago
Add some missing greek/latin superscripts
a few additional subscripts, greek letters 4 days ago
doc: replace blue dots with open ones a... a month ago
remove \GetFileInfo to fix date in unima... 2 months ago
enable syntax highlighting on GitHub [ci skip]
working
change of \mathrm behaviour (back to \... 4 months ago
this test is really confusing me
fix _\mathrm{x} issue
typo
undebug travis
debug travis
comment
fix subtle luatex-only bug of introducing a space from...
document that |x_\mathrm f| is no longer supported
i'll probably iterate this release checklist forever
v0.8l
release checklist
broken test
one last release checklist edit



Early days of the test suite:

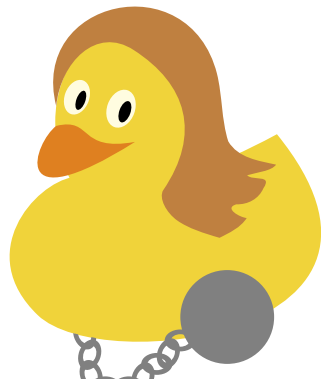
- XeLaTeX → PDF → PNG → ImageMagick
- A horribly-fragile and hard-to-read Makefile
- Pixel by pixel comparisons
- Slow, lots of false negatives
- Nonetheless, a large number of tests produced

l3build brought automated unit testing to the masses:

- Just wrap `\loggingoutout` around everything is fine?

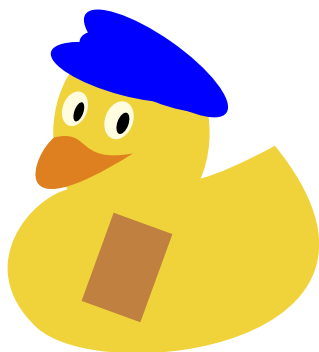
l3build brought automated unit testing to the masses:

- Just wrap `\loggingoutout` around everything is fine?
- It is really not fine.



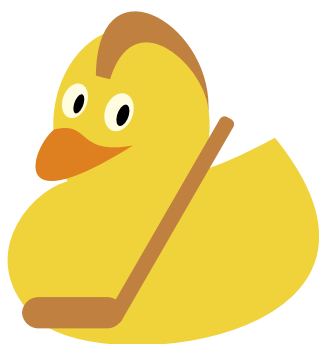
l3build brought automated unit testing to the masses:

- Just wrap `\loggingoutout` around everything is fine?
- It is really not fine.
- Slowly re-write all tests with custom, minimal, logging.



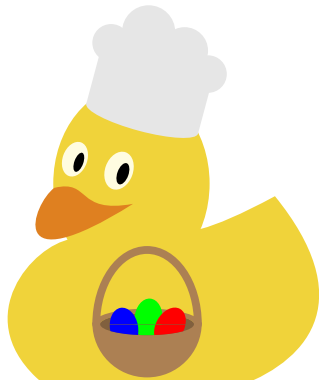
Example test input

```
\input{umtest-preamble}

\usepackage{fontspec}
\setmathsf{texgyrecursor-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
l. ...\TESTBOX{$\mathsf{X}=X$}
```

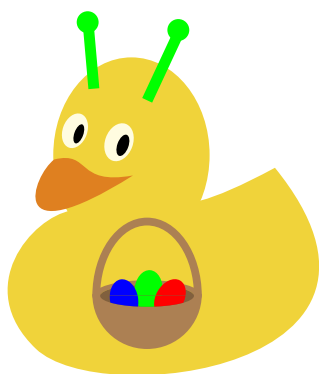


Release checklist

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'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



How do I load a font in fontspec?

- `\fontspec`
- `\setmainfont`
- `\newfontfamily`
- `\defaultfontfeatures`

As the package has grown it is probably less than clear!

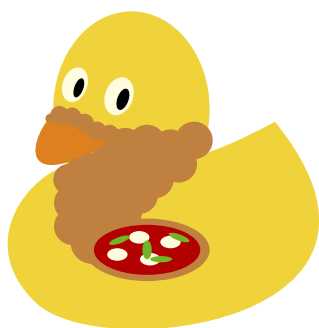
Font names

XeTeX was originally written to load fonts from the OS:

```
\setmainfont{Hoefler Text}% -- `just works'
```

luaotfload followed, and now:

```
\setmainfont{TeX Gyre Pagella}% -- `just works'
```

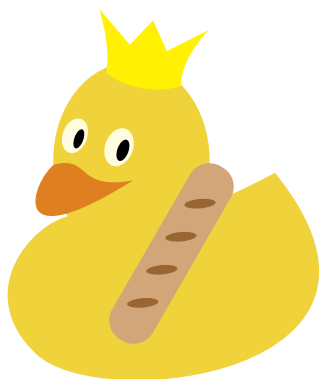


File names

But also:

```
\setmainfont{texgyrepagella-regular.otf}[

  ItalicFont      = texgyrepagella-italic.otf      ,
  BoldFont        = texgyrepagella-bold.otf        ,
  BoldItalicFont  = texgyrepagella-bolditalic.otf  ,

]
```

(or)

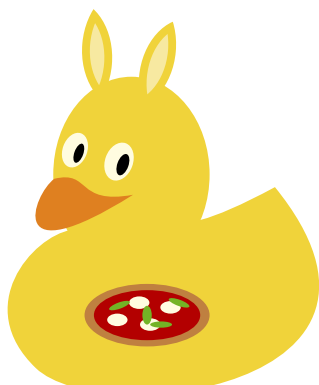


File names

But also:

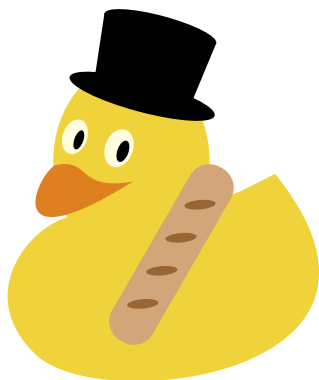
```
\setmainfont{texgyrepagella}[
  Extension       = .otf          ,
  UprightFont     = *-regular     ,
  ItalicFont      = *-italic      ,
  BoldFont        = *-bold        ,
  BoldItalicFont  = *-bolditalic  ,
]
```

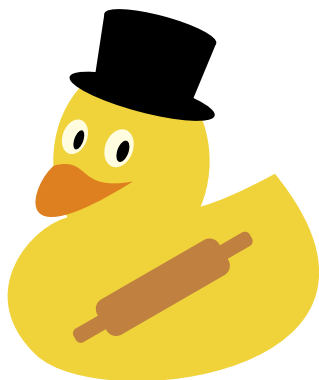
Which is better? This is the approach I now recommend.



The case against font names

1. Edge cases
   - Sometimes the correct italic/bold shape isn't picked up
2. Document portability
   - XeTeX/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



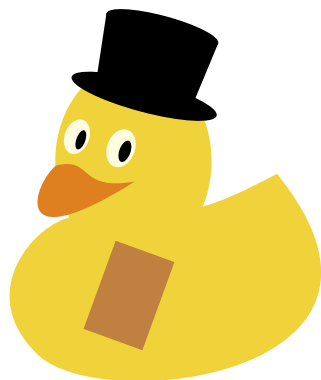


```
\font\x="[EBGaramond12-Regular.otf]"
```

Theory 123

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

Theory 123

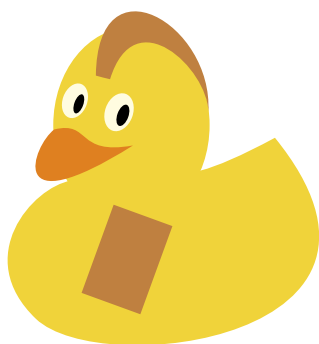


```
\fontspec{EBGaramond12-Regular.otf}
```

Theory 123

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

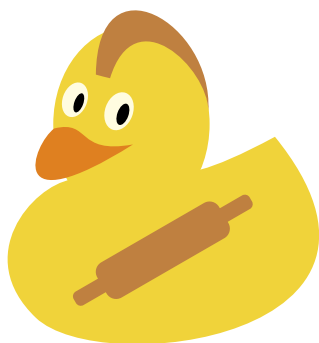
Theory 123



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`?



‘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**

Typical example

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

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   ,
}
```

- 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}`.

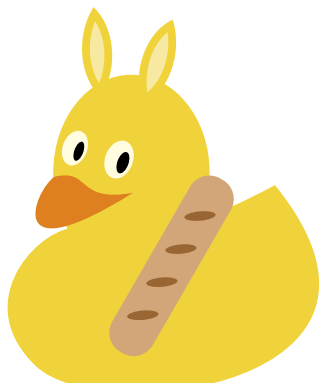
Typical example

- 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   ,
}
```

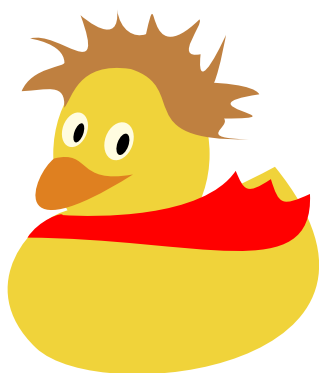
- Now I can write `\setmainfont{gill-sans-nova}`.
- Or, semantically,

```
\newfontfamily\captionfont{gill-sans-nova}
```



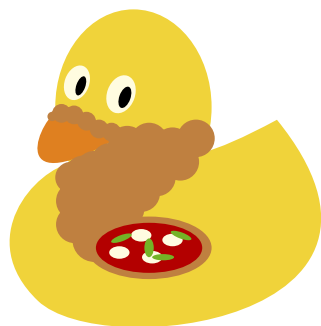
The full power of the NFSS is supported:

```
\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  } ,
}
```



Emphasis and inner emphasis

- LaTeX 2ε supports `\eminnershape` for markup with nested emphasis



Emphasis and inner emphasis

- LaTeX 2$_\varepsilon$ supports `\eminnershape` for markup with nested emphasis
- fontspec supports arbitrary nesting using (say)

```
\emfontdeclare{\itshape,\upshape\scshape,\itshape}
```

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.:

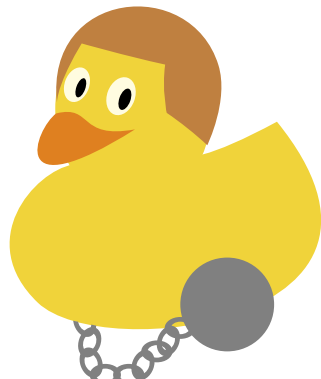
Rm `\emph{`*Aaa* `\emph{`Bbb `\EMPH{`*Iii*`}}}`

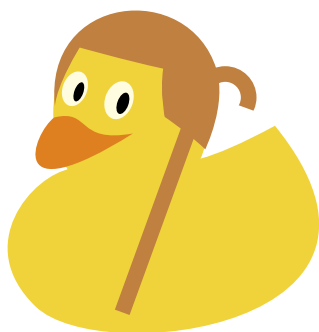
Strong and inner strong

- And more recently…`\strong`!

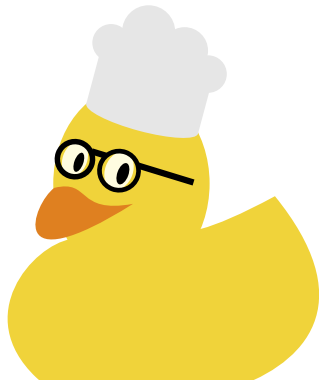
Strong and inner strong

- And more recently…\strong!
- ```
  \strongfontdeclare{
    \bfseries,
    \fontseries{hh}\selectfont,
    \fontseries{xx}\selectfont,
  }
  ```


  ```

Strong and inner strong

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

Abc \strong{**Abc** \strong{**Abc** \strong{**Abc**}}}


  ```

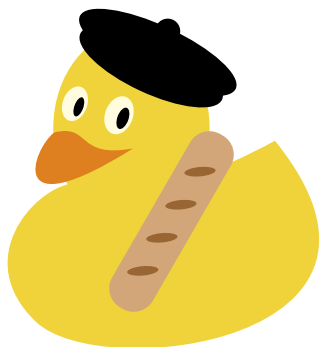


- Although everything is now Unicode, LaTeX's idea of 'encodings' is still useful



```
\newfontfamily\sanskitfont{charis}

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

KALITAṂ



```
\newfontfamily\oopsfont {posterama}

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

KALITAM.

```
\newfontfamily\titlefont{posterama}[
  NFSSEncoding=fakedotaccent
]

...{\titlefont   KALITA\d M}...  % <- uses fake accent
```

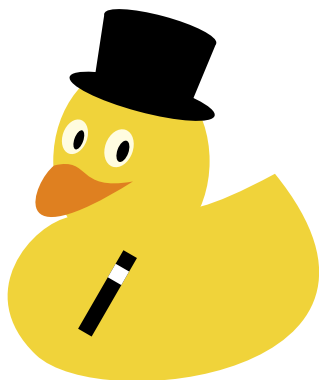
KALITAṂ

In the preamble:

```
\DeclareUnicodeEncoding{fakedotaccent}{
  \input{tuenc.def}
  \EncodingCommand{\d}[1]{%
    \hmode@bgroup
      \o@lign{\relax#1\crcr\hidewidth
        \ltx@sh@ft{-1ex}.\hidewidth}%
    \egroup
  }
}
```

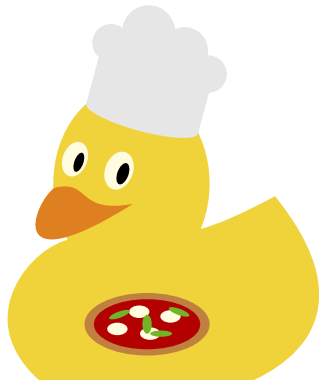
KALITAṂ

(1901)



KALITAM

(1913)



KALITAṂ

(1919)

KALITAṂ

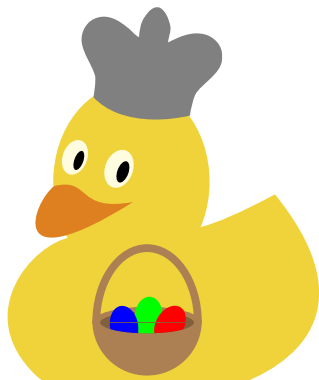
(1927)

KALITAṂ

(1933)

KALITAṂ

(1945)

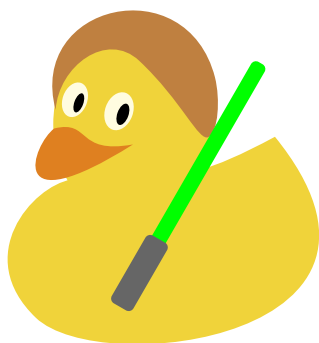


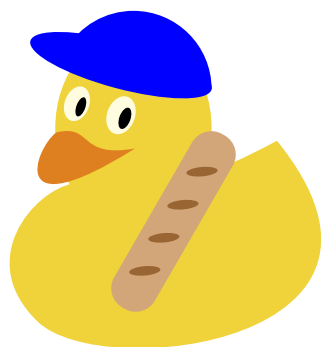
KALITAṂ

(1984)

KALITAṂ

(2001)





Conclusion

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

- Obrigado