Handling fonts in ConTEXt

ConTEXt-Meeting 15th

2021

Willi Egger

Where to Place a New Font in ConTEXt

Where to Place a New Font in ConTEXt

- New fonts are best placed into the tex-tree:

```
.../tex/texmf-fonts/data
```

- If you have many fonts, then it is adviseable to make subdirectories:

```
.../tex/texmf-fonts/data/Lato
```

- After placing new files into the tex tree we need to update de **file database:**

```
mtxrun --generate
```

- Further the **font-index** has to be updated:

```
mtxrun --script font --reload
```

Using System Fonts

Using System Fonts

In order to tell ConTeXt where to find the system fonts we make use of the environment variable OSFONTDIR.

- In Windows issue the following command on the commandline:

  ```
  set OSFONTDIR=c:/windows/fonts
  ```

- On UNIX-like systems like MacOSX and Linux we have to issue the following command in the terminal:
  - Mac:
    ```
    export OSFONTDIR=/Library/fonts:/System/Library/Fonts:$HOME/Library/Fon
    ```
  - Linux:
    ```
    export OSFONTDIR=$HOME/.local/share/fonts:/usr/share/fonts
    ```

For these UNIX based systems we need to add these command line statements to the .bashrc, or the corresponding file for the shell you use, to make this permanent.

Using System Fonts

And now of course update the file database and the font-index!

Finding a Fontname

Finding a Fontname

ConTeXt comes with a series of tools to find out available fonts.
For finding font-names issue the following command in a terminal:

```
mtxrun --script fonts --list --all --pattern=*fontname*
```

The pattern for the font name can be with wildcards (*).

Example: LATO

Example: LATO

LATO is a free font, which can be downloaded from https://www.latofonts.com/lato-free-fonts/ or https://fonts.google.com/specimen/Lato#standard-styles. The latest version is 2.015.

The LATO font family was designed by Łukasz Dziedzić, Warsaw from 2010 to 2015.

Running in the terminal:

```
mtxrun --script font --list --file --pattern=*lato*
```

When we get output, we know that ConTEXt can find the font.

Example: LATO

This results in the following list:

familyn.	weight	style	width	variant	fontname	filename
lato	black	normal	normal	normal	latoblack	Lato-Black.ttf
lato	black	italic	normal	normal	latoblackitalic	Lato-BlackItalic.ttf
lato	bold	normal	normal	normal	latobold	Lato-Bold.ttf
lato	bold	italic	normal	normal	latobolditalic	Lato-BoldItalic.ttf
lato	light	normal	normal	normal	latohairline	Lato-Hairline.ttf
lato	light	italic	normal	normal	latohairlineitalic	Lato-HairlineItalic.ttf
lato	extrabold	normal	normal	normal	latoheavy	Lato-Heavy.ttf
lato	extrabold	italic	normal	normal	latoheavyitalic	Lato-HeavyItalic.ttf
lato	normal	italic	normal	normal	latoitalic	Lato-Italic.ttf
lato	light	normal	normal	normal	latolight	Lato-Light.ttf
lato	light	italic	normal	normal	latolightitalic	Lato-LightItalic.ttf
lato	medium	normal	normal	normal	latomedium	Lato-Medium.ttf
lato	medium	italic	normal	normal	latomediumitalic	Lato-MediumItalic.ttf
lato	normal	normal	normal	normal	latoregular	Lato-Regular.ttf
lato	semibold	normal	normal	normal	latosemibold	Lato-Semibold.ttf
lato	semibold	italic	normal	normal	latosemibolditalic	Lato-SemiboldItalic.ttf
lato	light	normal	thin	normal	latothin	Lato-Thin.ttf
lato	light	italic	thin	normal	latothinitalic	Lato-ThinItalic.ttf

Using a Font with `\definedfont`

Using a Font with `\definedfont`

Any font known to ConTEXt can be used with the command `\definedfont`

```
\definedfont[latoblack*default at 12pt]
```

“Have Fun with Fonts in ConTEXt”

This is latoblack at 12.0pt

Adding *default makes ConTEXt use the default feature set which includes (ligatures, kerning etc.)

This approach can be used to setup headings and alike or in a situation, where a piece of text must be typeset in a dedicated font.

Typescripts

Typescripts

It would be very tedious to type in all situations again and again the command `\definedfont[fontname*default at size]`.
For setting up the fonts to be used throughout a document one needs to setup the fonts for the body font environment. For this purpose there are **typescripts**.
Typescripts define a font for the use throughout a document and enable the use of the usual font switches `\tfa, \bf, \bi` etc.

Fonts in the Distribution

ConTeXt distribution comes with some 20 fonts, all of them can be used out of the box. All necessary setups (typescripts) are provided in the distribution and the fonts can simply be loaded by `\setupbodyfont[fontname, style, size]`.

Find the fonts in:

.../tex/texmf/fonts/opentype/public and .../tex/texmf/fonts/truetype/public

Typescripts

The list of fonts delivered with the ConTeXt distribution is:

Computer Modern:

modern	:	synonym modern-base
modernvariable	:	synonym modern-variable (a variable width typewriter font)

The TeX Gyre collection of fonts, cross-platform OpenType format:

pagella	:	synonym palatino	including math
termes	:	synonym times	including math
heros	:	synonym helvetica	
bonum	:	synonym bookman	including math
scholas	:	synonym schoolbook	including math
adventor	:	synonym avantgarde	
cursor	:	synonym courier	
chorus	:	synonym chancery	

The DejaVu font family : dejavu and dejavu-condensed

Typescripts

The IBM-plex family: IBM-plex sans, sans-condensed, serif, mono and sans-Hebrew, Devanagari and Thai.

6 additional fonts (covering serif, sans serif, and monospaced)

Gentium	:	gentium
Antykwa Poltawskiego	:	antykwa-poltawskiego
Antykwa Toruńska	:	antykwa-torunska
Kurier	:	kurier
Iwona	:	iwona
Libertinus	:	libertinus

3 additional math fonts:

Euler	:	eulernova
STIX2	:	stix
XITS	:	xits
DejaVu	:	dejavu

Typescripts

Creating a Typescript for a Third-Party Font

Typescripts provide a method to map font names onto the basic names inside ConTEXt. Because these symbolic names are used by the `\definebodyfont` command, one has to use the predefined names for each style and alternative.

	serif	**sans**	**mono**	**handwriting**	**calligraphy**
tf	Serif	Sans	Mono	Handwriting	Calligraphy
bf	SerifBold	SansBold	MonoBold		
it	SerifItalic	SansItalic	MonoItalic		
sl	SerifSlanted	SansSlanted	MonoSlanted		
bi	SerifBoldItalic	SansBoldItalic	MonoBoldItalic		
bs	SerifBoldSlanted	SansBoldSlanted	MonoBoldSlanted		
sc	SerifCaps	SansCaps	MonoCaps		

Typescripts

A typescript is a start-stop construct with two arguments

```
\starttypescript[sans][name]
...
\stoptypescript
```

The first argument is the style where the second argument is the name of the typescript. The predefined styles are:

- serif
- sans
- mono
- math
- calligraphy
- handwriting

Typescripts

Font Fallback System

It is possible, that a font does not provide all glyphs we might be using. For this reason ConTeXt has a font fallback system implemented. This makes it possible that a missing glyph is retrieved from the fallback font.
For each of the default styles exists a fallback setup:

- font:fallback:serif
- font:fallback:sans
- font:fallback:mono

The definitions of these fallback setups can be found in the file type-fbk.mkxl. As an example here are the setups for the sans-style.

```
\startsetups [font:fallback:sans]
 \definefontsynonym [Sans] [DefaultFont]
 \definefontsynonym [SansBold] [Sans]
 \definefontsynonym [SansItalic] [Sans]
 \definefontsynonym [SansSlanted][SansItalic]
```

Typescripts

```
 \definefontsynonym [SansBoldItalic] [Sans]
 \definefontsynonym [SansBoldSlanted][SansBoldItalic]
 \definefontsynonym [SansCaps] [Sans] [features=smallcaps]
\stopsetups
```

Setting up a Typescript

In order to end up with a useable font within ConTeXt, one proceeds in three steps involving three typescripts:

- First typescript: map a symbolic, human-readable name onto the filename by using `\definefontsynonym`
- Second typescript: map the internal basic name on to the human readable name by using `\definefontsynonym`
- Third typescript: create a typeface by using `\definetypeface`

Typescripts

Example Lato

First we map the human readable name onto the font file names:

```
\starttypescript[sans][lato]
 \definefontsynonym[latoregular]   [file:Lato-Regular]
 \definefontsynonym[latobold]      [file:Lato-Bold]
 \definefontsynonym[latoitalic]    [file:Lato-Italic]
 \definefontsynonym[latobolditalic][file:Lato-BoldItalic]
\stoptypescript
```

In a second typescript the internal basic names are linked to the just defined human readable names.

```
\starttypescript[sans][lato]
 \setups[font:fallback:sans]
 \definefontsynonym[Sans]          [latoregular]   [features=default]
 \definefontsynonym[SansBold]      [latobold]      [features=default]
 \definefontsynonym[SansItalic]    [latoitalic]    [features=default]
```

Typescripts

```
 \definefontsynonym[SansBoldItalic][latobolditalic][features=default]
\stoptypescript
```

In this step we see a third argument to the `\definefontsynonym` command. Here we can invoke features provided by the font.

In the third step the typeface is defined.

```
\starttypescript[MyLato]
 \definetypeface[MyLato][ss][sans][lato][default]
\stoptypescript
```

These three typescripts should be placed in a file with the name starting with **type-imp-** with added the font name e.g. type-imp-lato.mkxl.

Typescripts

Setting up the Lato-Font in a Document

In an actual document one loads this typescript-file, tells ConTeXt the name of the typescript to be used and sets up the body font.

```
\usetypescriptfile[type-imp-lato]
\usetypescript[MyLato]

\setupbodyfont[MyLato,ss,12pt]
\blank
{    \getbuffer[sampletext]}\crlf
{\it \getbuffer[sampletext]}\crlf
{\bf \getbuffer[sampletext]}\crlf
{\bi \getbuffer[sampletext]}\crlf
```

Typescripts

The output becomes:

“Have Fun with Fonts in ConTeXt”
“Have Fun with Fonts in ConTeXt”
“Have Fun with Fonts in ConTeXt”
“Have Fun with Fonts in ConTeXt”

This is Lato-Regular at 12.0pt

Typescripts

Font Features

A lot of information about font features can be found in the font-manual.
Modern fonts as OTF and TTF can have many features, which can be turned on and off. The list of possible features is long and ConTeXt provides even more features than fonts can provide. This is due to the fact, that the features in ConTeXt are considered more a concept.
Examples of font features are:

liga	general ligatures
tlig	TeX ligatures
trep	TeX ligatures replacements
kern	Kerning information
smcp	Small caps
onum	Oldstyle numbers
tnum	Table numbers
lnum	Line numbers
pnum	Proportional numbers
salt	Stylistic alternates

Typescripts

swash	Swash letters
sub	Subscript
sup	Superscript
ss01	Stylistic set 1

...

ConTEXt can show the features contained in a font:

```
mtxrun –script font –list –info –pattern=*lato*
```

The output of this run looks as follows:

```
mtx-fonts     | gpos features:
mtx-fonts     |
mtx-fonts     | feature script languages
mtx-fonts     |
mtx-fonts     | kern    cyrl   dflt
mtx-fonts     |         dflt   dflt
```

Typescripts

```
mtx-fonts      |         grek   dflt
mtx-fonts      |         latn   dflt
mtx-fonts      | mark    cyrl   dflt
mtx-fonts      |         dflt   dflt
mtx-fonts      |         grek   dflt
mtx-fonts      |         latn   dflt
mtx-fonts      |
mtx-fonts      | gsub features:
mtx-fonts      |
mtx-fonts      | feature script languages
mtx-fonts      |
mtx-fonts      | calt    cyrl   dflt srb
mtx-fonts      |         dflt   dflt
mtx-fonts      |         grek   dflt
mtx-fonts      |         latn   dflt rom trk
```

Typescripts

mtx-fonts	case	cyrl	dflt srb
mtx-fonts		dflt	dflt
mtx-fonts		grek	dflt
...			

Typescripts

This is the list of features provided by the Lato fonts in a compressed form:

gpos features:

kern
mark

gsub features:

calt
case
dlig
dnom
frac
liga
lnum
numr
onum
ordn
pnum
salt
sinf
ss01
ss02
ss03
ss04
subs
sups
tnum

For the configuration of a font one can define feature-sets. These definitions may best be included in the type-imp-'font-name' file.

Typescripts

It is advisable to make the new feature-set inherit from the default-feature set.
A font feature definition can look then like this:

```
\definefontfeature
 [mylatofeature]
 [default]
 [onum=yes,
  pnum=yes]
```

```
\definefontfeature
       [f:onum]
       [onum=yes]
```

Where to use feature-sets:

– in the typescripts where the human readable name is mapped onto the internal basic style name

– in connection with the `\definedfont` command.

Switching on and off Font Features

Font features can be switched on and off on the fly.

Typescripts

There are different variants of these commands:

```
\addfeature{f:onum}
\feature[+]{f:onum}
```

Of course there is also the possibility to subtract, switch off, a given font feature by

```
\subtractfeature{f:tnum}
\feature[-]{f:tnum}
```

By means of a comma separated list one can turn on and off more features at once.
It is also possible to use square brackets instead of curly braces for the second argument.
This mechanism is handy e.g. for switching on table numbers for making up at table with numbers, where in the rest of the text proportional numbers are used.
In the font-manual there are more feature commands described, see page 43 and further.

The \definebodyfont Command

All defined symbolic names use the information setup by the `\definebodyfont` command.

```
\starttypescript [sans] [default] [size]
\definebodyfont
 [4pt,5pt,6pt,7pt,8pt,9pt,10pt,11pt,12pt,14.4pt,17.3pt]
 [ss]
 [tf=Sans sa 1,
 bf=SansBold sa 1,
 it=SansItalic sa 1,
 sl=SerifSlanted sa 1,
 bi=SansBoldItalic sa 1,
 bs=SansfBoldSlanted sa 1,
 sc=SansCaps sa 1]
\stoptypescript
```

The \definebodyfont Command

Such typescripts also exist predefined for serif and mono. If we need to add a fontsize permanently we can easily add such a typescript e.g. in the typescriptfile:

```
\starttypescript[sans][default][size]
 \definebodyfont
  [24pt]
  [tf=Sans sa 1
   bf= SansBold sa 1
   ...]
\stoptypescript
```

The \definebodyfontenvironment Command

Another set of definitions is contained in the body font environments. These definitions are related to a specific size of the body font. They define script and scriptscript sizes for math as well as the font switches to x and xx (\tfx and \tfxx and alike).

```
\definebodyfontenvironment
  [12pt]
  [text=12pt,
   script=9pt,
   scriptscript=7pt,
   x=10pt,
   xx=8pt,
   big=12pt,
   small=10pt]
```

The first argument specifies the body font size to which the settings apply. All second parameters are specified in dimensions and tell us more about related sizes.

Defining a Typeface

Defining a Typeface

Now we have all elements for completing the typescript file.
We can define the typeface:
A basic typescript can consist out of a single `\definetypeface` command for the third party font.

```
\starttypescript[Mylato]
 \definetypeface[Mylato][sans][ss][lato][default]
\stoptypescript
```

This typescript tells, that the name of the typescript is Mylato. The typeface is called Mylato, that is the name to be used in the `\setupbodyfont` command. The typeface specifies a sansserif font, which is defined by the typescripts with the name lato.
However normally typefaces are setup as a group of serif, sansserif, mono and math fonts.

The Complete Typescript File

The Complete Typescript File

```
\loadtypescriptfile[dejavu]
\loadtypescriptfile[xits]

\definefontfeature
 [mylatofeature]
 [default]
 [onum=yes,
  pnum=yes]

\starttypescript[sans][lato]
 \definefontsynonym[latoregular]   [file:Lato-Regular]
 \definefontsynonym[latobold]      [file:Lato-Bold]
 \definefontsynonym[latoitalic]    [file:Lato-Italic]
 \definefontsynonym[latobolditalic][file:Lato-BoldItalic]
\stoptypescript
```

The Complete Typescript File

```
\starttypescript[sans][lato]
 \setups[font:fallback:sans]
 \definefontsynonym[Sans]          [latoregular]   [features=default]
 \definefontsynonym[SansBold]      [latobold]      [features=default]
 \definefontsynonym[SansItalic]    [latoitalic]    [features=default]
 \definefontsynonym[SansBoldItalic][latobolditalic][features=default]
\stoptypescript

\starttypescript[Mylato]
 \definetypeface [Mylato] [ss]  [sans]  [lato]   [default]
 \definetypeface [Mylato] [rm]  [serif] [dejavu] [default]
 \definetypeface [Mylato] [tt]  [mono]  [dejavu] [default]
 \definetypeface [Mylato] [mm]  [math]  [xits]   [default]
                          [rscale=1.2]
\stoptypescript
```

The Complete Typescript File

In a document the following lines are set into the preamble:

```
\usetypescriptfile[type-imp-lato.mkxl]
\usetypescript[Mylato]
\setupbodyfont[Mylato,ss,10pt]
```

The Lato Font in Use

The Lato Font in Use

The sample file Ward

The Earth, as a habitat for animal life, is in old age and has a fatal illness. Several, in fact. It would be happening whether humans had ever evolved or not. But our presence is like the effect of an old-age patient who smokes many packs of cigarettes per day—and we humans are the cigarettes.

The numbers in oldstyle: 1 2 3 4 5 6 7 8 9 0.
Inhabitants in Three European Cities

Town	**Number of Inhabitants**	
Amsterdam	1.149	Million
London	8.982	Million
Warsaw	1.69	Million

Lato-Regular at 12.0pt

Font Classes

Font Classes

- A font class is a definition of a name space
- Font classes allow the use of different bodyfont setups in the same document. e.g. you can use the pagella namespace next to the termes namespace
- Switching the name space make immediately the predefined styles available

The command \usebodyfont

When using many fonts in document it is wise to set a list of fonts to be used in the preamble

```
\usebodyfont[pagella]
\usebodyfont[termes]
\usebodyfont[lato,10pt]
```

This causes ConTEXt to preload all necessary setups/environments.

References

References

Public sources for in depth information:

- Font manual. Hans Hagen. In the distribution.
- https://wiki.contextgarden.net/Fonts
- https://wiki.contextgarden.net/Typescripts_examples
- https://wiki.contextgarden.net/Style_Alternatives

Thank you for your attention!