# \definefontfamily

ConTeXt Meeting 2021

Wolfgang Schuster

# How can we change fonts with ConTeXt?

ConTeXt support a large number of fonts out of the box and many of them are part of a normal installation.

The following list is a small sample of these fonts:

- Latin Modern Roman, Sans and Typewriter.
- TeX Gyre Pagella, Termers etc.
- DejaVu Serif, Sans and Sans Mono
- IBM Plex Serif, Sans and Mono.
- Lucida Bright Opentype.

## How can we add additional fonts?

To make additional fonts usable with ConTEXt a *typescript* is needed which maps the font files to corresponding *alternatives* (i.e. **\tf**, **\it** etc.).

```
\starttypescript [serif] [myfont]
    \definefontsynonym [Serif]       [file:myregularfont]
    \definefontsynonym [SerifItalic] [file:myitalicfont]
     . . .
\stoptypescript
```

Besides the *typescript* there is also a *typeface* definition needed to load the *typescript* with **\setupbodyfont**.

```
\definetypeface [mytypeface] [rm] [serif] [myfont] [default]
```

## Is the a simpler way?

In 2009 I wrote a short module called *simplefonts* to make font loading in a document easier.

The user interface was inspired by *fontspec* and immediately enabled the selected fonts.

```
\usemodule [simplefonts]

\setmainfont [TeX Gyre Pagella]
\setsansfont [TeX Gyre Heros]
\setmonofont [TeX Gyre Cursor]
```

## What happended with *simplefonts*?

The initial code for the module was LuaTEX only and while later basic support for XƎTEX was added.

With the help from Hans there was a way to extend the module to use informations from the font database.

```
\startluacode
dolookupfontbyspec{
    familyname = texgyrepagella,
    weight     = bold,
    style      = italic,
}
\stopluacode
```

While this new code provided many possibilities it was never used and the module remained in the existing form.

## A new approach

A few years later in 2013 I started on a new version of the interface which tried to fix many flaws of the original version.

- The commands should be closer to the existing font mechanism.
- It should be possible to use multiple *typeface* and not be limited to a single main font.
- The search mechanism to find files for a given font shouldn't rely only on the filename.

## The new interface

The new version of the *simplefonts* mechanism uses two commands to select a font for a document:

1. **`\definefontfamily`**
2. **`\definefallbackfamily`**

## The `\definefontfamily` command

The main command of the new system is **`\definefontfamily`** which takes three mandatory arguments.

```
\definefontfamily [.1.] [.2.] [.3.]
```

1. Name of the typeface, used to enable the font with **`\setupbodyfont`**.
2. Style of the font in short (e.g. *rm*) or long (e.g. *serif*) form.
3. Family name of the font (e.g. *Latin Modern Roman*).

## `\definefontfamily` example

The following example uses the font *Comic Neue* with the help of the **`\definefontfamily`** command.

The code

```
\definefontfamily [examplefont] [ss] [Comic Neue]

\switchtobodyfont [examplefont]

This example uses the {\em Comic Neue} font.
```

produces this output:

This example uses the *Comic Neue* font.

## How does `\definefontfamily` work?

- The **`\definefontfamily`** uses the Lua part of the previously shown database search mechanism to looks for entries with the given family name.
- The entries from the search are filtered to get the necessary file for all font *alternatives* in ConTEXt.
- When all *alternatives* are set the command creates a simplified *typescript* where the files are again mapped to *synonyms* in the font mechanism.
- To use the created *typescript* with **`\setupbodyfont`** in a document, the mechanism also create the necessary *typeface* setting.

## Example entry from the database

| | | |
|---|---|---|
| **familyname** | : | antykwapoltawskiego |
| **subfamilyname** | : | bolditalic |
| **family** | : | antpolt |
| **subfamily** | : | bolditalic |
| **fontname** | : | antpoltbolditalic |
| **rawname** | : | AntPolt-BoldItalic |
| **fullname** | : | antpoltbolditalic |
| **filename** | : | antpolt-bolditalic.otf |
| **cleanfilename** | : | antpoltbolditalic |
| **style** | : | italic |
| **weight** | : | bold |
| **width** | : | normal |
| **variant** | : | normal |
| **pfmwidth** | : | 5 |
| **pfmweight** | : | 700 |
| **designsize** | : | 100 |
| **minsize** | : | 95 |
| **maxsize** | : | 110 |
| **angle** | : | -10.0 |

## Mixing \definefontfamily and \definetypeface

Based on the fact that **\definefontfamily** just create a hidden *typescript*, it is possible to use the command together with **\definetypeface** to create a combined *typeface*.

```
\definefontfamily [examplefont] [ss] [Helvetica Neue LT Std]
\definetypeface   [examplefont] [tt] [mono] [modern] [default] [rscale=1.2]

\switchtobodyfont [examplefont]

This example use the \type{\definefontfamily} and \type{\definetypeface}
commands to create a example with the two font {\tt Helvetica Neue} and
{\tt Latin Modern Mono}.
```

This example use the `\definefontfamily` and `\definetypeface` commands to create a example with the two font `Helvetica Neue` and `Latin Modern Mono`.

## Options for `\definefontfamily`

Besides the three mandatory argument **`\definefontfamily`** provides a fourth optional to change values for the font family.

```
\definefontfamily [...] [...] [...] [..,..=..,..]
```

The values *rscale*, *features* and *designsize* are applies to all *alternatives* in a family.

Additional features for a certain *alternative* can be passed with the name of the *alternative* as key.

# Values for all fonts in a family

- **rscale:** Changes the relative size of the font to create a size math between different styles (e.g. serif and mono).
- **features:** Applies a previously create feature set to all font files. Monospace fonts require the argument `features=none` to prevent ligatures.
- **designsize:** Used to enable optical sizes for fonts which provide this feature.

## Values for a single font in a family

To pass values to a single font one use the name of the *alternative* as key (e.g. `bf={file:<filename>}`).

- **name:** Select a font file based on the *name* entry.
- **file:** Select a font file based on the *file* entry. This does not look for files in the working directory.
- **style:** Select a font file based on the *subfamily* entry or when no match is found on a complexer search method.
- **features:** Can be used to set a different feature set to a certain file. To pass the global value together with the local value `*` can be used (e.g. `features:{*,f:smallcaps}`).

## The `\definefallbackfamily` command

Besides **`\definefontfamily`** there is another command called **`\definefallbackfamily`** which it used to set fallbacks for the main font.

The command has the same structure with three mandatory arguments.

```
\definefallbackfamily [.1.] [.2.] [.3.]
```

1. Name of the typeface, has to be same name as the mainfont set by **`\definefontfamily`**.
2. Style of the font, can be given in short (e.g. *rm*) or long form (e.g. *serif*).
3. Family name of the font, can be the same font as the main font.

## Which additional options are available?

Like `\definefontfamily` the `\definefallbackfamily` command provides a fourth argument which accepts the same keys but in addition the command provides a few extra setup-options.

- **range:** Expects either the unicode value of the character to be replaced or named range.
- **check:** Replace the character only when it present in the fallback font.
- **force:** Can be set to *yes* to use the character from the fallback font even when it present in the main font.
- **preset:** Uses a predefined list of the above listed options.

# \definefallbackfamily example

The following example uses a *fallback* font to combine *TeX Gyre Pagella* with *New Athena Unicode* to typeset greek text.

The example

```
\definefallbackfamily [fallbackexample] [rm] [New Athena Unicode]
  [range={greekandcoptic,greekextended},force=yes]

\definefontfamily [fallbackexample] [rm] [TeX Gyre Pagella]

\switchtobodyfont [fallbackexample]

Greek alphabet \emdash\ Ελληνικό αλφάβητο
```

in result in

Greek alphabet — Ελληνικό αλφάβητο

## Does the order of both commands matter?

All `\definefallbackfamily` settings for a font have to be before the corresponding `\definefontfamily` line.

```
\definefallbackfamily [myfont] [ss] [Font A] [..,..=..,..]
\definefallbackfamily [myfont] [ss] [Font B] [..,..=..,..]

\definefontfamily [myfont] [ss] [Font C]
```

This is necessary because `\definefallbackfamily` saves the files for the given name in a table and only when `\definefontfamily` occurs all fallbacks are applied to the main font.

## Additional example 1

The following example uses the *spec* typescript to load a font without the need to write the rest of the necessary typescript.

```
\definetypeface [spec-example] [ss] [specsans] [Helvetica Neue LT Std] [default]

\setupbodyfont [spec-example]

\starttext

\tf Helvetica Neue\par
\it Helvetica Neue\par
\bf Helvetica Neue\par
\bi Helvetica Neue\par

\stoptext
```

## Additional example 2

The following example shows how one can apply different font files based on the style to **`\definefontfamily`**.

```
\definefontfamily [lato-1] [ss] [Lato]
\definefontfamily [lato-2] [ss] [Lato] [tf=style:thin,bf=style:regular]

\starttext

\start \switchtobodyfont [lato-1]
Lato regular \bf and bold
\stop

\start \switchtobodyfont [lato-2]
Lato thin \bf and regular
\stop

\stoptext
```

```
\definefallbackfamily
```