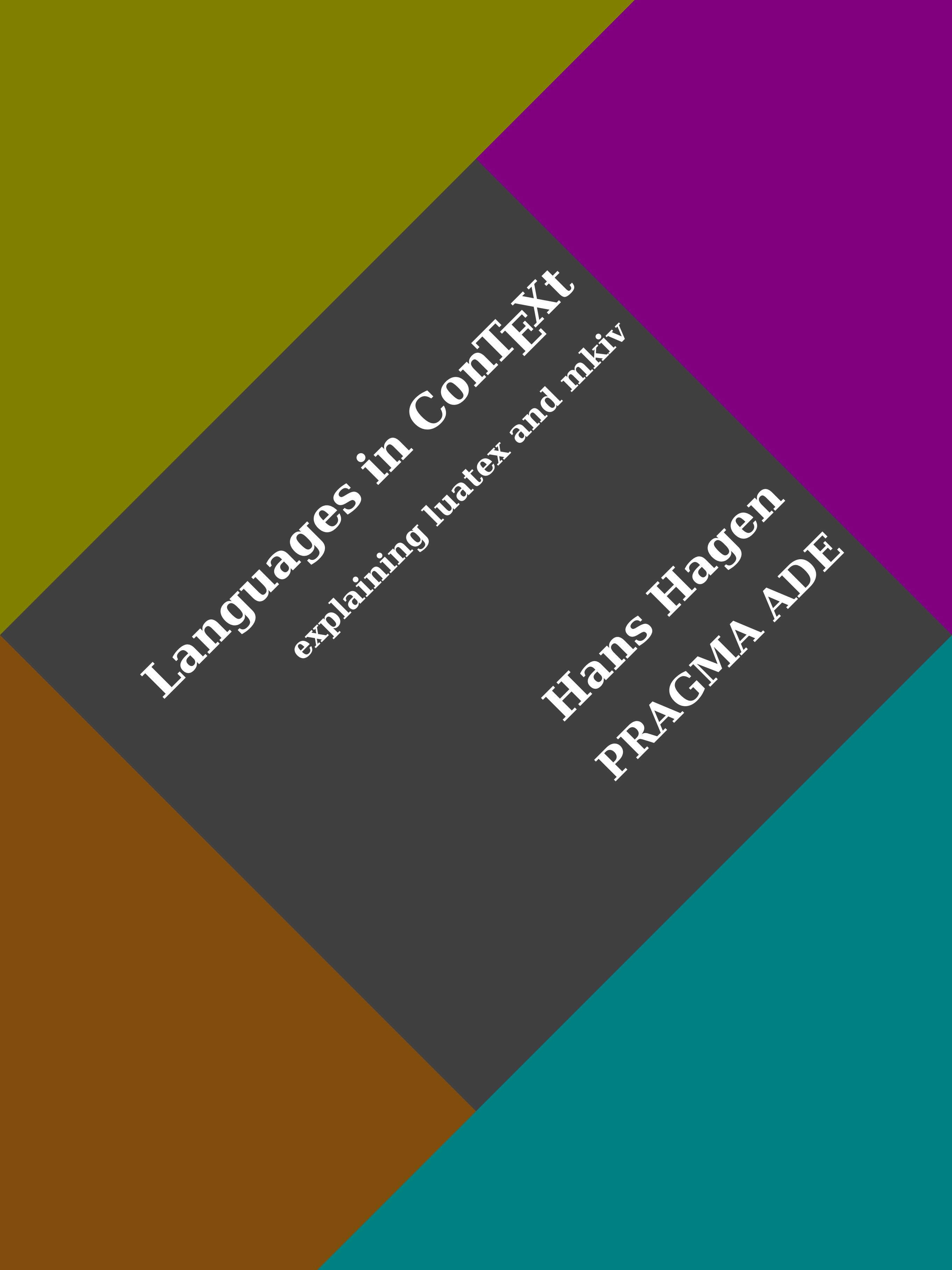


Languages in ConTeXt

explaining luatex and mkiv

Hans Hagen

PRAGMA ADE

Contents

Introduction

1 Some basics

2 Hyphenation

3 Labels

4 Numbering

5 Typesetting

Introduction

This document describes an important property of the TEX typesetting system and ConTEXt in particular: the ability to deal with different languages at the same time. With languages we refer to natural languages. So, we're not going to discuss the TEX language itself, not MetaPost, nor Lua.

The original application of TEX was English that uses the Latin script. The fonts that came with TEX were suitable for that usage. When lines became too long they could be hyphenated using so called hyphenation patterns. Due to the implementation for many years there was a close relationship between fonts and hyphenation. Although at some point many more languages and scripts were supported, it was only when the Unicode aware variants showed up that hyphenation and fonts were decoupled. This makes it much more easier to mix languages that use different scripts. Although Greek, Cyrillic, Arabic, Chinese, Japanese, Korean and other languages have been supported for a while using (sometimes dirty) tricks, we now have cleaner implementations.

We can hyphenate words in all languages (and scripts) that have a need for it, that is, split it at the end of a line and add a symbol before and/or after the break. The way words are broken into parts is called hyphenation and so called patterns are used to achieve that goal. The way these patterns are constructed and applied was part of the research related to TEX development. The method used is also applied in other programs and is probably one of the few popular ways to deal with hyphenation. There have been ideas about extensions that cover the demands of certain languages but so far nothing better has shown up. In the end TEX does a pretty decent job and more advanced tricks don't necessarily lead to better results.

Hyphenation is driven by a language number and that's about it. This means that one cannot claim that TEX in its raw form supports languages, other than that it can hyphenate and use fonts that provide the glyphs. It's upto a macro package to wrap this into a mechanism that provides the user an interface. So, when we speak about language support, hyphenation is only one aspect. Labels, like the `figure` in *figure 1.2* need to adapt to the main document language. When dates are shown they can be language specific. Scientific units and math function names can also be subjected to translation. Registers and other lists have to be sorted according to specific rules. Spacing dan differ per language.

In this manual we will cover some of functionality in ConTEXt MkIV that relates to languages (and scripts). This manual is a compliment to other manuals, articles and documentation. Here we mostly focus on the language aspects. Some of the content (or maybe most) might looks alien and complex to you. This is because one purpose of this manual is to provide a place to wrap up some aspects of ConTEXt. If you're not interested in that, just stick to the more general manuals that also cover language aspects.

This document is still under construction. The functionality discussed here will stay and more might show up. Of course there are errors, and they're all mine. The text is not checked for spelling errors. Feel free to let me know what should get added.

Hans Hagen
PRAGMA ADE, Hasselt NL
2013 — 2016

1 Some basics

1.1 Introduction

In this chapter we will see how we can toggle between languages. A first introduction to patterns will be given. Some details of how to control the hyphenation with specific patterns will be given in a later chapter.

1.2 Available languages

When you use the English version of ConTEXt you will default to US English as main language. This means that hyphenation will be US specific, which by the way is different from the rules in GB. All labels that are generated by the system are also in English. Languages can often be accessed by names like `english` or `dutch` although it is quite common to use the short tags like `en` and `nl`. Because we want to be as compatible as possible with MkII, there are quite some synonyms. The following table lists the languages that for which support is built-in.[1]

tag	n	parent	file	synonyms	patterns	characters
af	6		af	afrikaans		
agr	60	gr	agr	ancientgreek		
ala	66	la	ala			
ar	32		ar	arabic		
ar-ae	34	ar	ar			
ar-bh	35	ar	ar			
ar-dz	51	ar	ar			
ar-eg	36	ar	ar			
ar-in	37	ar	ar			
ar-iq	48	ar-sy	ar			
ar-ir	46	pe	pe			
ar-jo	49	ar-sy	ar			
ar-kw	38	ar	ar			
ar-lb	50	ar-sy	ar			
ar-ly	39	ar	ar			
ar-ma	52	ar	ar			
ar-om	40	ar	ar			
ar-qa	41	ar	ar			
ar-sa	42	ar	ar			
ar-sd	43	ar	ar			
ar-sy	47	ar	ar			

---

[1] More languages can be defined. It is up to users to provide the information.

ar-tn	44	ar	ar	
ar-ye	45	ar	ar	
be	22		be	belarussian
bg	25		bg	bulgarian
ca	63		ca	catalan
cn	56		cn	chinese
cs	16		cs	cz czech
da	4		da	danish
de	3		de	deu german
de-at	11	de	de	
de-ch	12	de	de	
de-de	10	de	de	
deo	9	de	deo	
en	1		us	eng english
en-gb	13	en	gb	uk ukenglish
en-us	14	en	us	us usenglish
eo	54	esperanto	eo	
es	62		es	sp spanish
es-es	69	es	es	
es-la	70	es	es	
esperanto	53		eo	
et	27	en	et	estonian
farsi	33		farsi	fa pe persian
fi	26		fi	finnish
fr	61		fr	fra french
gr	59		agr	greek
he	31		he	hebrew yi
hr	18		hr	croatian
hu	28		hu	hungarian
it	64		it	italian
ja	57		ja	japanese
kr	58		kr	korean
la	65		la	latin
lt	55		lt	lithuanian

mk	24		mk	macedonian
ml	74		ml	malayalam
nb	7		nb	bokmal no norwegian
nl	2		nl	dutch nld
nn	8		nn	nynorsk
null	0			
pl	15		pl	polish
pt	67		pt	portuguese
pt-br	68	pt	pt	
ro	71		ro	romanian
ru	20		ru	russian
sk	17		sk	slovak
sl	19		sl	slovene slovenian
sr	23		sr	serbian sr-cyrl sr-latn
sv	5		sv	swedish
th	73		th	thai
tk	30		tk	turkmen
tr	29		tr	turkish
ua	21		uk	ukrainian
vi	72		vi	vietnamese

You can call up such a table with the following commands:

```
\usemodule[languages-system]

\loadinstalledlanguages
\showinstalledlanguages
```

Instead you can run `context --global languages-system.mkiv`.

As you can see, many languages have hyphenation patterns but for Japanese, Korean, Chinese as well as Arabic languages they make no sense. The patterns are loaded on demand. The number is the internal number that is used in the engine; a user never has to use that number. Numbers < 1 are used to disable hyphenation. The file tag is used to locate and load a specification. Such files have names like type lang-nl.lua.

Some languages share the same hyphenation patterns but can have demands that differ, like labels or quotes. The characters shown in the table are those found in the pattern files. The number of patterns differs a lot between languages. This relates to the systematic behind them. Some languages use word stems, others base their hyphenation on syllables. Some language have inflections which adds to the complexity while others can combine words in ways that demand special care for word boundaries. Of course a low or high number can signal a low quality as well, but most pattern collections are assembled over many years and updated when for instance spelling rules change. I think that we can safely say that most patterns are quite stable and of good quality.

1.3 Switching

The document language is set with

```
\mainlanguage[en]
```

but when you want to apply the proper hyphenation rules to an embedded language you can use:

```
\language[en]
```

or just:

```
\en
```

The main language determines what labels show up, how numbering happens, in what way dates get formatted, etc. Normally the `\mainlanguage` command comes before the `\starttext` command.

1.4 Hyphenation

In LuaTeX each character that gets typeset not only carries a font id and character code, but also a language number. You can switch language whenever you want and the change will be carried with the characters. Switching within a word doesn't make sense but it is permitted:

```
1  \de incrediblykompliziert       incredi-b-ly-kom-pli-ziert
2  \en incrediblykompliziert       in-cred-i-blykom-plìziert
3  \en incredibly\de kompliziert   in-cred-i-blykom-pli-ziert
4  \en incredibly\de \-kompliziert  incredibly-kompliziert
5  \en incredibly\de -kompliziert   in-cred-i-bly-kom-pli-ziert
```

In the line 4 we have a `\-` between the two words, and in the last line just a `-`. If you look closely you will notice that the snippets can be quite small. If we typeset a word with a 1mm text width we get this:

in-
cred-
i-
bly

If you are familiar with the details of hyphenation, you know that the number of characters at the end and beginning of a word is controlled by the two variables `\lefthyphenmin` and `\righthyphenmin`. However, these only influence the hyphenation process. What bits and pieces eventually end up on a line is determined by the par builder and there the `\hsize` matters. In practice you will not run into these situations, unless you have extreme long words and a narrow column.

Hyphenation normally is limited to regular characters that make up the alphabet of a language. It is insensitive for capitalization as the following text shows:

> This time the mu-si-cal dis-trac-tion while de-vel-op-ing code came from watch-ing youtube per-for-mances of Cory Henry (also known from Snarky Puppy, a con-glom-er-ate of ex-cel-lent play-ers). Just search the web for his name with ‘Ste-vie Won-der and Michael Jack-son Trib-ute’. There is no key-board he can't play. An-other in-ter-est-ing key-board player is Sun Rai (a short name for Rai Thistleth-wayte, just google for ‘The Bea-t-les, Come To-gether, Live Pi-ano Acoustic with Loop Pedal’, or do a com-bined search with ‘Matt Cham-ber-lain’. Okay, and talk-ing of key-boards, let's not for-get Vika Yer-molyeva (vk-goeswild) as she's one of a kind too on the web. And then there is Ja-cob Col-lier, in one word: in-cred-i-ble (or hy-phen-ated the Dutch way in-cre-di-ble, let me re-peat that in French in-cre-dible).[2]

Of course, names are often short and don't need to be hyphenated (or the left and right settings prohibit it). Another complication with names is that they can come from another language so we either need to switch language temporarily or we need to add an exception (more about that later).

1.5 Primitives

In traditional TEX the language is not a property of a character but is triggered by a signal in the (so called) list. Think of:

```
<language 1>this is <language 2>nederlands<language 1> mixed with english
```

This number is set by the primitive `\language`. Language triggers are injected into the list depending on the value of this number. There is also a `\setlanguage` primitive that can inject triggers without setting the `\language` number. Because in LuaTEX the state is kept with the character you don't need to worry about the subtle differences here.

In ConTEXt the `\language` and `\setlanguage` commands are overloaded by a more advanced switch macro. You cannot assume that they work as explained in general manuals about TEX. Currently you can still assign a number but that might change. Just consider the language to be an abstraction and don't mess with this number. Both com-

---

[2] Get me right, there are of course many more fantastic musicians.

mands not only change the current language but also do specific initializations when needed.

What characters get involved in hyhenation is historically determines by the so called `\lccode` values. Each character can have such a value which maps an uppercase to a lowercase character. This concept has been extended in ε-TEX where it binds to a pattern set (language). However, in ConTEXt the user never has to worry about such details.

In traditional hyphenation there will not be hyphenated if the sum of `\lefthyphenmin` and `\righthyphenmin` exceeds 62. This limitation is not present in the to be presented Lua variant of this routine as there is no good reason for this limitation other than implementation constraints.

1.6 Control

We already mentioned `\lefthyphenmin` and `\righthyphenmin`. These two variables control the area in a word that is subjected to hyphenation. Setting these values is a matter of taste but making them too small can result in bad hyphenation when the patterns are made with the assumptions that certain minima are used. Using a `\lefthyphenmin` of 2 while the patterns are made with a value of 3 in mind is a bad idea.

\lefthyphenmin	**\righthyphenmin**				
	1	2	3	4	5
1	in-ter-est-ing	in-ter-est-ing	in-ter-est-ing	in-ter-esting	in-ter-esting
2	in-ter-est-ing	in-ter-est-ing	in-ter-est-ing	in-ter-esting	in-ter-esting
3	inter-est-ing	inter-est-ing	inter-est-ing	inter-esting	inter-esting
4	inter-est-ing	inter-est-ing	inter-est-ing	inter-esting	inter-esting
5	inter-est-ing	inter-est-ing	inter-est-ing	inter-esting	inter-esting

When TEX breaks a paragraph into lines it will try do so without hyphenation. When that fails (read: when the badness becomes too high) a next effort will take hyphenation into account.[3] When the badness is still too high, an optional emergency pass can be made but only when the tolerances are set to permit this. In ConTEXt you can try these settings when you get too many over- or underfull boxes reported on the console.

```
\setupalign[tolerant]
\setupalign[verytolerant]
\setupalign[verytolerant,stretch]
```

Personally I tend to use the last setting, especially in automated flows. After all, TEX will

---

[3] Because in LuaTEX we always hyphenate there is no real gain in trying not to hyphenate. Because in traditional TEX hyphenation happens on the fly a pass without hyphenating makes more sense.

not apply stretch unless it's really needed.

The two `\*hyphenmin` parameters can be set any time and the current value is stored with each character. They can also be set with the language which we will see later.

When TEX hyphenates words it has to decide where a word starts and ends. In traditional TEX the words starts normally at a character that falls within the scope of the hyphenator. It ends at when a box (hlist or vlist) is seen, but also at a rule, discretionary, accent (forget about this in ConTEXt) or math. An example will be given in the chapter that discussed the Lua alternative.

1.7 Installing

todo

1.8 Modes

Languages are one of the mechanisms where you can access the current state. There are for instance two (official) macros that contain the current (main) language:

macro	**value**
`\currentmainlanguage`	`en`
`\currentlanguage`	`en`

When we have set `\language[nl]` we get this:

macro	**value**
`\currentmainlanguage`	`en`
`\currentlanguage`	`nl`

If you write a style that needs to adapt to a language you can use modes. There are several ways to do this:

```
\language[nl]

\startmode[**en]
    \color[darkred]{main english}
\stopmode

\startmode[*en]
    \color[darkred]{local english}
\stopmode

\startmode[**nl]
```

```
    \color[darkblue]{main dutch}
\stopmode

\startmode[*nl]
    \color[darkblue]{local dutch}
\stopmode

\startmodeset
    [*en] {\color[darkgreen]{english set}}
    [*nl] {\color[darkgreen]{dutch set}}
\stopmodeset
```

This typesets:

main english
local dutch
dutch set

When you use setups you can use the following trick:

```
\language[nl]

\startsetups language:en
    \color[darkorange]{something english}
\stopsetups

\startsetups language:nl
    \color[darkorange]{something dutch}
\stopsetups

\setups[language:\currentlanguage]
```

As expected we get:

something dutch

2 Hyphenation

2.1 How it works

Proper hyphenation is one of the strong points of TEX. Hyphenation in TEX is done using so called hyphenation patterns. Making these patterns is an art and most users (including me) happily use whatever is available. Patterns can be created automatically using `patgen` but often manual tweaking is needed too. A pattern looks as follows:

```
pat1tern
```

This means as much as: you can split the word `pattern` in two pieces, with a hyphen between the two `t`'s. Actually it will also split the word `patterns` because the hyphenation mechanism looks at substrings. When no number between characters in a pattern is given, a zero is assumed. This means as much as *undefined*. An even number inhibits hyphenation, an odd number permits it. The larger the number (weight), the more influence it has. A more restricted pattern is:

```
.pat1tern.
```

Here the periods set the word boundaries. The pattern dictionary for us english has smaller patterns and the next trace shows how these are applied.

```
   . p a t t e r n .  . p a t t e r n .
1   1p0a0              1 0 0 0 0 0 0 0
2     0a0t5t0e0        1 0 0 5 0 0 0 0
3       4t3t2          1 0 4 5 2 0 0 0
   .1p0a4t5t2e0r0n0.  . p a t-t e r n .
```

The effective hyphenation of a word is determined by several factors:

- the current language, each language can have different patterns
- the characters, as some characters might block hyphenation
- the settings of `\lefthyphenmin` and `\righthyphenmin`

A place where a word can be hyphenated is called a discretionary. When TEX analyzes a stream, it will inject discretionary nodes into that stream.

```
pat\discretionary{-}{}{}tern.
```

In traditional TEX hyphenation, ligature building and kerning are tightly interwoven which is quite effective. However, there was also a strong relationship between the current font and hyphenation. This is a side effect of traditional TEX having at most 256 characters in a font and the fact that the used character is fact a reference to a slot in a font. There a character in the input initially ends up as a character node and eventually becomes a glyph node. For instance two characters `fi` can become a ligature glyph representing this combination.

In LuaTEX the hyphenation, ligature building and kerning stages are separated and can be overloaded. In ConTEXt all three can be replaced by code written in Lua. Because normally hyphenation happens before font logic is applied, there is no relationship with font encoding. I wrote the first Lua version of the hyohenator on a rainy weekend and the result was not that bad so it was presented at the 2014 ConTEXt meeting. After some polishing I decided to add this routine to the standard MkIV repertoire which then involved some proper interfacing.

You can enable the Lua variant with the following command:

```
\setuphyphenation[method=traditional]
```

We call this method `traditional` because in principle we can have many more methods and this one is (supposed to be) mostly compatible to the built-in method. This is a global setting. You can switch back with:

```
\setuphyphenation[method=default]
```

In the next sections we will see how we can provide alternatives within the traditional method. These alternatives can be set local and therefore can operate over a limited range of characters.

One complication in interfacing is that TEX has grouping (which permits local settings) and we want to limit some of the above functionality using groups. At the same time hyphenation is a paragraph related action so we need to enable the hyphenation related code at a global level (or at least make sure that it gets exercised by forcing a `\par`). That means that the alternative hyphenator has to be quite compatible so that we could just enable it for a whole document. This can have an impact on performance but in practice that can be neglected. In LuaTEX the Lua variant is 4 times slower than the built-in one, in LuajitTEX it's 3 times slower. But the good news is that the amount of time spent in the hyphenator is relatively small compared to other manipulations and macro expansion. The additional time needed for loading and preparing the patterns into a more Lua specific format can be neglected.

You can check how words get hyphenated using the patterns management script:

```
>mtxrun --script patterns --hyphenate language

hyphenator      |
hyphenator      | . l a n g u a g e .   . l a n g u a g e .
hyphenator      |    0a2n0               0 0 2 0 0 0 0 0 0
hyphenator      |    2a0n0g0             0 2 2 0 0 0 0 0 0
hyphenator      |      0n1g0u0           0 2 2 1 0 0 0 0 0
hyphenator      |        0g0u4a0         0 2 2 1 0 4 0 0 0
hyphenator      |              2g0e0.0   0 2 2 1 0 4 2 0 0
hyphenator      | .0l2a2n1g0u4a2g0e0.   . l a n-g u a g e .
hyphenator      |
mtx-patterns    | us 3 3 : language : lan-guage
```

2.2 The last words

Mid 2014 we had to upgrade a style for a pdf assembly service: chapters from (technical) school books are combined into arbitrary new books. There are some nasty aspects with this flow: for instance, all section numbers in a chapter are replaced by new numbers and this also involves figure and table prefixes. It boils down to splitting up books, analyzing the typeset content and preparing it for replacements. The structure is described in xml files so that we can generate tables of contents. The reason for not generating from xml sources is that the publisher doesn't have a xml workflow and that books already were available. Also, books from several series are combined and even within a series structure (and rendering) differs.

What has this to do with hyphenation? Writing a style for such a flow always results in a more complex one that estimated and as usual it's in the details. The original style was written in MkII and used some box juggling to achieve reasonable results but in MkIV we can do better.

Each chapter has a title and books get titles and subtitles as well. The titles are typeset each time a new book is composed. This happens within some layout constraints. Think of constraints like these:

- the title goes on top of a shape that doesn't permit much overflow
- there can be very long words (not uncommon in Dutch or German)
- a short word or hyphenated part should not end up on the last line
- the left and right hyphenation minima are at least four

The last requirement is a compromise because in most cases publishers seem to want ragged right not hyphenated rendering (at least in Dutch schoolbooks). The arguments for this are quite weak and probably originate in fear of bad rendering given past experiences. It's this kind of situations that drive the development of the more obscure features that ship with ConTeXt and a (partial) solution for this specific case will be given later.

If you look at thousands of titles and turn these into (small) paragraphs TeX does a pretty good job. It's the few exceptions that we need to catch. The next examples demonstrate such an extreme case.

1	a verylongword and then anevenlonger- word	a verylong- word and then anevenlongerword

2	a verylongword and then aneven- longerword	a verylong- word and then anevenlongerword

3	a verylongword and then aneven- longerword	a verylong- word and then anevenlongerword

4	a verylongword and then aneven- longerword	a verylong- word and then anevenlongerword

5	a verylong- word and then anevenlonger- word	a verylong- word and then anevenlongerword

Of course in practice there need to be some reasonable width and when we pose these limits the longest possible word should fit into the allocated space. In these examples the rule shows the width. In the right columns we see a red colored word and that one will not get hyphenated.

2.3 Explicit hyphens

Another special case that we needed to handle were (compound) words with explicit hyphens. Because often data comes from xml files we can not really control the typesetting as in a TEX document where the author sees what gets done. So here we need a way to turn these hyphens into proper hyphenation directives and at the same time permit the words to be hyphenated.

1	a very-long-word and then an-even-longer-word	a very-long-word and then an- even-longer-word

2	a very-long-word and then an-even-longer-word	a very-long-word and then an- even-longer-word

3	a very-long-word and then an-even-longer-word	a very-long-word and then an- even-longer-word

4	a very-long-word and then an-even-longer-word	a very-long-word and then an- even-longer-word

5	a very-long-word and then an-even-longer-word	a very- long-word and then an- even-longer-word

2.4 Extended patterns

As with more opened up mechanisms, in MkIV we can extend functionality. As an example I have implemented the extensions discussed in the article by László Németh in the Proceedings of EuroTEX 2006: *Hyphenation in OpenOffice.org* (TUGboat, Volume 27, 2006). The syntax for these extension is somewhat ugly and involves optional offsets and ranges.[4]

```
\registerhyphenationpattern[nl][e1ë/e=e]
\registerhyphenationpattern[nl][a9atje./a=t,1,3]
\registerhyphenationpattern[en][eigh1tee/t=t,5,1]
\registerhyphenationpattern[de][c1k/k=k]
\registerhyphenationpattern[de][schif1f/ff=f,5,2]
```

These patterns result in the following hyphenations:

reëel	re-eel
omaatje	oma-tje
eighteen	eight-teen
Zucker	Zuk-ker
Schiffahrt	Schiff-fahrt

In a specification, the `.` indicates a word boundary and numbers indicate the weight of a breakpoint. The optional extended specification comes after the `/`. The values separated by a `=` are the pre and post sequences: these end up at the end of the current line and

---

[4] I'm not sure if there were ever patterns released that used this syntax.

beginning of the next one. The optional numbers are the start position and length. These default to 1 and 2, so in the first example they identify `eë` (the weights don't count).

There is a pitfall here. When the language already has patterns that for instance prohibit a hyphen between `e` and type ë, like `e2ë`, we need to make sure that we give our new one a higher priority, which is why we used a `e9ë`.

This feature is somewhat experimental and can be improved. Here is a more Lua-ish way of setting such patterns:

```
local registerpattern =
    languages.hyphenators.traditional.registerpattern

registerpattern("nl","e1ë", {
    start  = 1,
    length = 2,
    before = "e",
    after  = "e",
} )

registerpattern("nl","a9atje./a=t,1,3")
```

Just adding extra patterns to an existing set without much testing is not wise. For instance we could add these to the dutch dictionary:

```
\registerhyphenationpattern[nl][e3ë/e=e]
\registerhyphenationpattern[nl][o3ë/o=e]
\registerhyphenationpattern[nl][e3ï/e=i]
\registerhyphenationpattern[nl][i3ë/i=e]
\registerhyphenationpattern[nl][a5atje./a=t,1,3]
\registerhyphenationpattern[nl][toma8at5je]
```

That would work oke well for words like

```
coëfficiënt
geïntroduceerd
copiëren
omaatje
tomaatje
```

However, the last word only goes right because we explicitly added a pattern for it. One reason is that the existing patterns already contain rules to prevent weird hyphenations. The same is true for the accented characters. So, consider these examples and coordinate additional patterns with other users so that errors can be identified.

2.5 Exceptions

We have a variant on the TEX primitive `\hyphenation`, the official way to register a specific way to hyphenate a word.

```
\registerhyphenationexception[aaaaa-bbbbb]
aaaaabbbbb \par
```

This code is self explaining and results in:

aaaaa-
bbbbb

There can be multiple hyphens and even multiple words in such a specification:

```
\registerhyphenationexception[aaaaa-bbbbb cc-ccc-ddd-dd]
aaaaabbbbb \par
cccccddddd \par
```

We get:

aaaaa-
bbbbb

cc-
ccc-
ddd-
dd

2.6 Boundaries

A box, rule, math or discretionary will end a word and prohibit hyphenation of that word. Take this example:

```
whatever \par
whatever\hbox{!} \par
\vl whatever\vl \par
whatever$x$ \par
whatever-whatever \par
```

These lines will hyphenate differently and in traditional TEX you need to insert penalties and/or glue to get around it unless you instruct LuaTEX to be more. In the Lua variant we can enable that limitation.

```
\definehyphenationfeatures
  [strict]
  [rightedge=tex]
```

Here we show the three variants: traditional TEX and Lua with and without strict settings.

default	**traditional**	**traditional strict**
what- ever what- ever! what- ever what- ever*x* what- ever- what- ever	what- ever what- ever! what- ever what- ever*x* what- ever-what- ever	what- ever what- ever! what- ever what- ever*x* whatever- whatever

By default ConTEXt is configured to hyphenate words that start with an uppercase character. This behaviour is controlled in TEX by the `\uchyph` variable. A positive value will enable this and a negative one disables it.

default 0	**default 1**	**traditional 0**	**traditional 1**
TEX- i- fied	TEX- i- fied	TEX- i- fied	TEX- i- fied

The Lua variants behaves the same as the built-in implementation (that of course remains the reference).

2.7 Plug-ins

The default hyphenator is similar to the built-in one, with a couple of extensions as mentioned. However, you can plug in your own code, given that it does return a proper hyphenation result. One reason for providing this plug is that there are users who want to play with hyphenators based on a different logic. In ConTEXt we already have some methods to deal with languages that (for instance) have no spaces but split on words or syllables. A more tight integration with the hyphenator can have advantages so I will explore these options when there is demand.

A result table indicates where we can break a word. If we have a four character word and can break after the second character, the result looks like this:

```
result = { false, true, false, false }
```

Instead of `true` we can also have a table that has entries like the extensions discussed in a previous section. Let's give an example of a plug-in.

```
\startluacode
    local subset = {
        a = true,
        e = true,
        i = true,
        o = true,
        u = true,
        y = true,
    }

    languages.hyphenators.traditional.installmethod("test",
        function(dictionary,word,n)
            local t = { }
            for i=1,#word do
                local w = word[i]
                if subset[w] then
                    t[i] = {
                        before = "<" .. w,
                        after  = w .. ">",
                        left   = false,
                        right  = false,
                    }
                else
                    t[i] = false
                end
            end
            return t
        end
    )
\stopluacode
```

Here we hyphenate on vowels and surround them by angle brackets when split over lines. This alternative is installed as follows:

```
\definehyphenationfeatures
  [demo]
  [alternative=test]
```

We can now use it as follows:

```
\setuphyphenation[method=traditional]
\sethyphenationfeatures[demo]
```

When applied to one the tufte example we get:

> We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, h<a a>monize, synthesize, focus, organize, condense, reduce, boil down, choose, c<a a>egorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discr<i i>inate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, o<u u>line, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

A more realistic (but not perfect) example is the following:

```
\startluacode
    local packslashes = false

    local specials = {
        ["!"]  = "before", ["?"]  = "before",
        ['"']  = "before", ["'"]  = "before",
        ["/"]  = "before", ["\\"] = "before",
        ["#"]  = "before",
        ["$"]  = "before",
        ["%"]  = "before",
        ["&"]  = "before",
        ["*"]  = "before",
        ["+"]  = "before", ["-"]  = "before",
        [","]  = "before", ["."]  = "before",
        [":"]  = "before", [";"]  = "before",
        ["<"]  = "before", [">"]  = "before",
        ["="]  = "before",
        ["@"]  = "before",
        ["("]  = "before",
        ["["]  = "before",
        ["{"]  = "before",
        ["^"]  = "before", ["_"]  = "before",
        ["`"]  = "before",
        ["|"]  = "before",
        ["~"]  = "before",
        --
        [")"]  = "after",
        ["]"]  = "after",
        ["}"]  = "after",
    }
    languages.hyphenators.traditional.installmethod("url",
        function(dictionary,word,n)
            local t = { }
            local p = nil
            for i=1,#word do
                local w = word[i]
                local s = specials[w]
                if s == "after" then
                    s = {
                        start  = 1,
                        length = 1,
                        after  = w,
                        left   = false,
                        right  = false,
                    }
                    specials[w] = s
                elseif s == "before" then
                    s = {
                        start  = 1,
                        length = 1,
                        before = w,
                        left   = false,
                        right  = false,
                    }
                    specials[w] = s
                end
                if not s then
                    s = false
                elseif w == p and w == "/" then
                    t[i-1] = false
                end
                t[i] = s
                if packslashes then
                    p = w
                end
            end
            return t
        end
    )
\stopluacode
```

Again we define a plug:

```
\definehyphenationfeatures
  [url]
  [characters=all,
   alternative=url]
```

So, we only break a line after symbols.

http://www.pragma-ade.nl

A quick test can look as follows:

```
\starthyphenation[traditional]
    \sethyphenationfeatures[url]
    \tt
    \dontcomplain
    \hsize 1mm
    http://www.pragma-ade.nl
\stophyphenation
```

Or:

```
http:
/
/
www.
pragma-
ade.nl
```

2.8 Blocking ligatures

Yet another predefined feature is the ability to block a ligature. In traditional TeX this can be done by putting a `{}` between the characters, although that effect can get lost when the text is manipulated. The natural way to do this in a Unicode environment is to use the special characters `zwj` and `zwnj`.

We use the following example lines:

```
supereffective \blank
superef\zwnj fective
```

and define two featuresets:

```
\definehyphenationfeatures
  [demo-1]
  [characters=\zwnj\zwj,
   joiners=yes]

\definehyphenationfeatures
```

```
  [demo-2]
  [joiners=no]
```

We limit the width to 1mm and get:

`method=default`	`method=traditional`	`method=traditional`
		`featureset=demo-1`
			`method=traditional`
			`featureset=demo-2`

super- ef- fec- tive	super- ef- fec- tive	super- ef- fec- tive	super- ef- fec- tive
supereffec- tive	supereffec- tive	super- ef- fec- tive	supereffec- tive

2.9 Special characters

The `characters` example can be used (to some extend) to do the same as the breakpoints mechanism (compounds).

```
\definehyphenationfeatures
  [demo-3]
  [characters={()[]}]

\starthyphenation[traditional]
    \sethyphenationfeatures[demo-3]
    \dontcomplain
    \hsize 1mm
    we use (super)special(ized) patterns
\stophyphenation
```

we
use
(su-
per)spe-
cial(ized)
pat-
terns

We can make this more clever by adding patterns:

```
\registerhyphenationpattern[en][)9]
```

```
\registerhyphenationpattern[en][9(]
```

This gives:

we
use
(su-
per)spe-
cial(ized)
pat-
terns

A detailed trace shows that these patterns get applied:

```
    . ( s u p e r ) s p e c i a l ( i z e d ) .  . ( s u p e r ) s p e c i a l ( i z e d ) .
 1   9(0                                          9 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 2     1s0u0                                      9 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 3       0u1p0e0                                  9 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 4               0)9                              9 1 0 1 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0
 5                   0p0e2c0                      9 1 0 1 0 0 0 9 0 0 2 0 0 0 0 0 0 0 0 0 0
 6                     0e0c2i0                    9 1 0 1 0 0 0 9 0 0 2 2 0 0 0 0 0 0 0 0 0
 7                       1c0i0                    9 1 0 1 0 0 0 9 0 0 2 2 0 0 0 0 0 0 0 0 0
 8                       3c0i0a0                  9 1 0 1 0 0 0 9 0 0 3 2 0 0 0 0 0 0 0 0 0
 9                         2i1a0                  9 1 0 1 0 0 0 9 0 0 3 2 1 0 0 0 0 0 0 0 0
10                         0i2a0l0                9 1 0 1 0 0 0 9 0 0 3 2 2 0 0 0 0 0 0 0 0
11                               9(0              9 1 0 1 0 0 0 9 0 0 3 2 2 0 9 0 0 0 0 0 0
12                                   2z0e0        9 1 0 1 0 0 0 9 0 0 3 2 2 0 9 0 2 0 0 0 0
13                                         0)     9 1 0 1 0 0 0 9 0 0 3 2 2 0 9 0 2 0 0 0 9
    .9(1s0u1p0e0r0)9s0p0e3c2i2a0l9(0i2z0e0d0)9.  . (-s u-p e r )-s p e-c i a l-( i z e d )-.
```

The somewhat weird hyphens at the edges will in practice not show up because there is always one regular character there.

2.10 Counting

There is not much you can do about patterns. It's a craft to make them and so they are shipped with the distribution. In order to hyphenate well, TEX looks at some character properties. In ConTEXt only the characters used in the patterns of a language get tagged as valid in a word.

The following example illustrates that there can be corner cases. In fact, this example might render differently depending on the patterns available. First we define an extra language, based on French.

```
\installlanguage[frf][default=fr,patterns=fr,factor=yes]
```

Here we set the `factor` parameter which tells the loader that it should look at the characters used in a special way: some count for none, and some count for more than one when determining the min values used to determine if and where hyphenation is to be applied.

```
\startmixedcolumns[n=3,balance=yes]
    \hsize 1mm \dontcomplain
    \language[fr]  aesop oedipus æsop œdipus \column
    \hsize 1mm \dontcomplain
    \language[frf] aesop oedipus æsop œdipus \column
    \startexceptions æ-sop \stopexceptions
    \hsize 1mm \dontcomplain
    \language[frf] aesop oedipus æsop œdipus
\stopmixedcolumns
```

We get three (when writing this manual) different columns:

ae-
sop
oe-
di-
pus
æsop
œdi-
pus

ae-
sop
oe-
di-
pus
æsop
œ-
di-
pus

ae-
sop
oe-
di-
pus
æ-
sop
œ-
di-
pus

The trick is in the `factor`: when set to `yes` an æ is counted as two characters. Combining marks count as zero but you will not find them being used as we already resolve them in an earlier stage.

U+000C6	Æ	2	letter	U+001C9	ǉ	2	letter
U+000DF	ß	2	letter	U+001CA	Ǌ	2	letter
U+000E6	æ	2	letter	U+001CC	ǌ	2	letter
U+00132	Ĳ	2	dubious	U+01E9E	ẞ	2	letter
U+00133	ĳ	2	dubious	U+0FB01	ﬁ	2	ligature
U+00152	Œ	2	dubious	U+0FB02	ﬂ	2	ligature
U+00153	œ	2	dubious	U+0FB03	ﬃ	3	ligature
U+001C7	Ǉ	2	letter	U+0FB04	ﬄ	3	ligature
U+001C8	ǈ	2	letter	U+0FB06	ﬆ	2	ligature

It is very unlikely to find an in the input and even an ĳ is rare. The æ is marked as character and the œ a ligatyure in Unicode. Maybe all the characters here are dubious but al least we provide a way to experiment with them.

2.11 Tracing

Among the tracing options (low level trackers) there is one for pattern developers:

```
\usemodule[s-languages-hyphenation]

\startcomparepatterns[de,nl,en,fr]
    \input zapf \quad (\showcomparepatternslegend)
\stopcomparepatterns
```

The different hyphenation points are shown with colored bars. Some valid points might not be shown because the font engine can collapse successive discretionaries.

Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on. (de nl en fr)

3 Labels

3.1 Introduction

When we started using TeX, I naturally started with plain TeX. But it didn't take long before we tried LaTeX. Because our documents were in Dutch one of the first fights with this package was to get rid of the english labels. Because rather soon we decided to cook up an alternative package, a decent label mechanism was one of the first things to show up. And as soon as multiple language typesetting gets into view, such a mechanism becomes one of those language dependent features. In this chapter the basics will be covered.

3.2 Defining labels

Before we define a label we need to define a label class. You probably seldom need that but this is how it's done:

```
\definelabelclass [mylabel]
```

There are some classes predefined:

head	(complete) titles like and
label	in–text labels like and Figure
mathlabel	function names like sin and cos
taglabel	labels used for tagging purposed in the backend
btxlabel	labels used in typesetting bibliographic items

The physical units mechanism also uses labels: unit, operator, prefix and suffix. All these labels are defined per language with a fall back on english.

Given that we have defined class `mylabel`, a label itself is set like this:

```
\setupmylabeltext
  [en]
  [first={<after first},
   second={{before second>},{<after second}}]
```

The first argument (the language) is optional. In the next section we will see how these labels are used. A lot of labels are predefined, in MkIV this happens in the file `lang-txt.lua`. There is no need to adapt this file as you can always add labels run time.

3.3 Using labels

How a label is called depends on the way it needs to be used. In any case the main language set determines the language of the label. So, when in an Dutch text we temporary switch to German, the Dutch labels are used.

command	**first**	**second**
`\leftmylabeltext{tag}`	<after first	before second>
`\rightmylabeltext{tag}`		<after second
`\mylabeltext{tag}`	<after first	before second>
`\mylabeltexts{tag}{text}`	<after firsttext	before second>text<after second

3.4 Hooks

Some mechanisms have label support built in, most noticeably sections heads and numbered items, like figure captions.

```
\definehead
  [myhead]
  [subsection]

\setuphead
  [myhead]
  [bodypartlabel=bodypartmyhead]

\setuplabeltext
  [en]
  [bodypartmyhead=My Head: ]

\myhead{Welcome}
```

My Head: 3.4.1 Welcome

The head text label class can be used as follows:

```
\setupheadtext
  [SomeHead=Just A Title]

\subsection
  [title=\headtext{SomeHead}]
```

3.4.2 Just A Title

A label will obey the style settings, as in:

```
\definehead
  [MyFancyHead]
  [subsection]
  [style={\bs\setcharactercasing[Words]}]

\setupheadtext
  [SomeHead=just another title]

\MyFancyHead
  [title=\headtext{SomeHead}]
```

3.4.3 Just Another Title

Labels

4 Numbering

4.1 Introduction

Numbering is complex and in ConTEXt it's not easy either. This is because we not only have 1, 2, 3 … but also sub numbers like 1a, 1b, 1ic … or 1.a, 1.b, 1.c … There can be many levels, different separators, final symbols. As we're talking languages we only discuss conversion here: the mechanism that turns a number in for instance a letter. It happens that the mapping from a number onto a letter is language dependent. The next lines show how English, Spanish and Slovenian numbers:

a b c d e f g h i j k l m n o p q r s t u v w x y z aa ab
a b c d e f g h i j k l m n ñ o p q r s t u v w x y z aa
a b c č d e f g h i j k l m n o p r s š t u v z ž aa ab ac

You convert a number into a letter with:

```
\convertnumber{alphabetic}{15}
```

There is also `\uconvertnumber` which does not expand unless typesetting is going on. Normally you don't need to bother about this.

The `alphabetic` converter adapts to the current main language. When a language has no special alphabet, the regular 26 characters are used.

A converter can also convert to a roman numeral, a language specific ordered list, a day or month, an ordinal string and again there can be a language specific conversion. The general conversion macro takes a conversion name and a number. When a conversion can be set (for instance in an itemized list, or in section numbering) you can use these names. You can define additional converters if needed, as long as the converter can handle a number.

```
\defineconversion [alphabetic] [\alphabeticnumerals]
```

Here `\alphabeticnumerals` is a converter. If you look into the source of ConTEXt you will see that many converters are calling out to Lua, where we have implemented those specific conversions. The following table has long and short names. The short one are historic.

month	`\monthlong`
month:mnem	`\monthshort`
character	`\character`
Character	`\Character`
characters	`\characters`
Characters	`\Characters`

AK	`\smallcappedcharacters`
KA	`\smallcappedcharacters`
alphabetic a	`\alphabeticnumerals`
Alphabetic A	`\Alphabeticnumerals`
number numbers n	`\numbers`
Numbers N	`\Numbers`
mediaeval m	`\mediaeval`
word words	`\verbosenumber`
Word Words	`\VerboseNumber`
ordinal	`\ordinalnumber`
Ordinal	`\Ordinalnumber`
romannumerals i r	`\romannumerals`
Romannumerals I R	`\Romannumerals`
o	`\oldstylenumerals`
O	`\oldstylenumerals`
or	`\oldstyleromannumerals`
KR	`\smallcappedromannumerals`
RK	`\smallcappedromannumerals`
greek g	`\greeknumerals`
Greek G	`\Greeknumerals`
mathgreek	`\mathgreek`
abjadnumerals	`\abjadnumerals`
abjadnodotnumerals	`\abjadnodotnumerals`
abjadnaivenumerals	`\abjadnaivenumerals`
thainumerals	`\thainumerals`
devanagarinumerals	`\devanagarinumerals`
gurmurkhinumerals	`\gurmurkhinumerals`
gujaratinumerals	`\gujaratinumerals`
tibetannumerals	`\tibetannumerals`
greeknumerals	`\greeknumerals`
Greeknumerals	`\Greeknumerals`
arabicnumerals	`\arabicnumerals`
persiannumerals	`\persiannumerals`
arabicexnumerals	`\arabicexnumerals`
arabicdecimals	`\arabicdecimals`
persiandecimals	`\persiandecimals`
koreannumerals kr	`\koreannumerals`
koreanparenthesisnumerals kr-p	`\koreanparenthesisnumerals`
koreancirclenumerals kr-c	`\koreancirclenumerals`

chinesenumerals cn	`\chinesenumerals`
chinesecapnumerals cn-c	`\chinesecapnumerals`
chineseallnumerals cn-a	`\chineseallnumerals`
sloveniannumerals	`\sloveniannumerals`
slovenianNumerals	`\slovenianNumerals`
spanishnumerals	`\spanishnumerals`
spanishNumerals	`\spanishNumerals`

The `alphabetic` and `Alphabetic` converters adapt to slovenian and spanish as do their small capped alternatives. There are more general helpers for it too:

```
\languagecharacters{number}
\languageCharacters{number}
```

Also language related is the `\continuednumber` macro. Here we see an application:

```
1 \continuednumber{1}
1, 2 \continuednumber{2}
1, 2, 3 \continuednumber{3}
```

What renders as:

1
1, 2 (continued)
1, 2, 3 (continued)

Such a macro is typically used in combination with counters ant it just typesets a label text depending on the valu ebeing non-zero.

```
\setuplabeltext[en][continued={and so on}]
1, 2, 3 \continuednumber{3}
1, 2, 3 \convertnumber{continued}{3}
```

This gives:

1, 2, 3 and so on
1, 2, 3 and so on

In the rare case that you want to check if a conversion is defined you can use

```
\doifelseconversiondefined{name}{true}{false}
```

So,

```
\doifelseconversiondefined{characters}{we can convert}{forget about it}
```

Gives:

we can convert

There are also some non language related converters that we mention here for completeness:

`set 0`: • – ⋆ ▷ ∘ ● ⬤ □ √ • – ⋆ ▷ ∘ ● ⬤ □ √ • –

`set 1`: ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡‡‡ ∗ ∗∗ ∗∗∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡‡‡ ∗ ∗∗ ∗∗∗ ⋆ ⋆⋆

`set 2`: ∗ † ‡ ∗∗ †† ‡‡ ∗∗∗ ††† ‡‡‡ ∗∗∗∗ †††† ‡‡‡‡ ∗ † ‡ ∗∗ †† ‡‡ ∗∗∗ †††

`set 3`: ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡ ‡‡‡ ¶ ¶¶ ¶¶¶ § §§ §§§ ∗ ∗∗ ∗∗∗ ⋆ ⋆⋆ ⋆ ⋆ ⋆ ‡ ‡‡

When a set overruns we start again at the first element.

The ordinal converter produces output like 123rd and 654th. The corresponding string renderer is `\highordinalstr`.

4.2 Dates

Dates are also language dependent. The following macros take a number and return the name of the month or day.

`\monthlong`	October
`\monthshort`	oct
`\MONTH`	OCTOBER
`\MONTHLONG`	OCTOBER
`\MONTHSHORT`	OCT
`\weekday`	Thursday
`\WEEKDAY`	THURSDAY

The current date can be typeset with `\currentdate` and a specific date with `\date`, for instance:

```
\currentdate[weekday,day,month,year]
\currentdate[WEEKDAY,day,MONTH,year]
\date[d=12,m=12,y=1998][weekday]
\date[d=12,m=12,y=1998]
```

Saturday 24 July 2021
SATURDAY 24 JULY 2021
Saturday
December 12, 1998

Possible elements of the specification are:

+ ord	ordinal suffix
++ highord	high ordinal suffix
mnem:	mnemonic prefix
Y y year	year 4 digits
yy	year 2 digits
M	month 1 or 2 digits
mm	month 2 digits
D	day 1 or 2 digits
dd	day 2 digits
W	1 digit
month m	language dependent (can be mnemonic)
day d	language dependent
weekday w	language dependent
MONTH	month uppercased
WEEKDAY	weekday uppercased
referral	YYYMMDD
space	space
<word>	word

There are also some converters built in (more can be added), for instance:

```
The current {\em gregorian} date \currentdate [month, day, {, }, year] is
in {\em jalali} \currentdate [jalali:to, month, day, {, }, year] but we
can also as a specific one, so {\em jalali} \date [y=1395, m=4, d=18]
[month, day, {, }, year] is {\em gregorian} \date [y=1395, m=4, d=18]
[jalali:from, month, day, {, }, year].
```

> The current *gregorian* date July 24, 2021 is in *jalali* May 2, 1400 but we can also as a specific one, so *jalali* April 18, 1395 is *gregorian* July 8, 2016.

For time we have `\currenttime` and here the specification is just an `h`, `m` and whatever connects them. Both date and time are pre-configured in the language definition file `lang-def`.

5 Typesetting

5.1 Introduction

In this chapter we will discuss a few settings and mechanisms that deal with typesetting from the perspective of languages. We will not go into details about the often obscure demands that a language puts on a system like ConTEXt. Often these are rooted in tradition, limitations of past rendering (brushed, written, mechanical or electronic), subjective decisions made by committees, contradicting opinions of typographers, etc. The most we can do is provide the mechanism that make it possible to honour most of these demands and provide a reasonable set of defaults. It's good to mention here that wasting energy on discussing language specific issues only makes sense when a similar amount of energy is spent on getting the rest of the document rendering right. It really makes no sense to whine about a lost (or bad) ligature, or a missed hyphenation point, or a loose paragraph when vertical spacing is sloppy, the use of color messy, the choice of fonts debatable, etc. The worst discussions I ran into are those involving inter-character spacing as way to optimize look, feel and readability while at the same time the choice of fonts and rest of the layout were not that attractive anyway.

5.2 Spacing

The look and feel of a paragraph is determined by several factors and language is undeniable one of them. Dutch and German have compound words that can be quite long, English and French use short words, some with only one character, Czech, Polish and many other languages have diacritics.

Interword spacing makes the text lighter, and the more short words there are, the more spacing shows up. Although, if the hyphenation patterns are suboptimal, spaces can stretch which can become annoying. Many uppercase characters (as in German) and accented characters make the text darker.

The user has not much influence on this, apart from rewriting the text. When a narrow column is used, that can be a bit of a challenge, as too narrow columns can just look bad. In the next example we see a sample text (`tufte.tex`) typeset with the align options `normal`.

We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

We see one word sticking into the margin, and there is not much that TeX can do about it, given its constraints. We can be a bit more tolerant if we also add the option `tolerant`. This sample text is always good for lots of successive hyphenation I must admit that I never make a big deal about that if only because trying to avoid it often gives worse results.

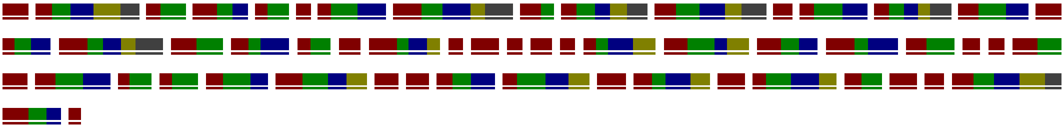
We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

Normally `normal,tolerant` is good enough for a document, but if you really want to play safe you can better also permit some stretch.

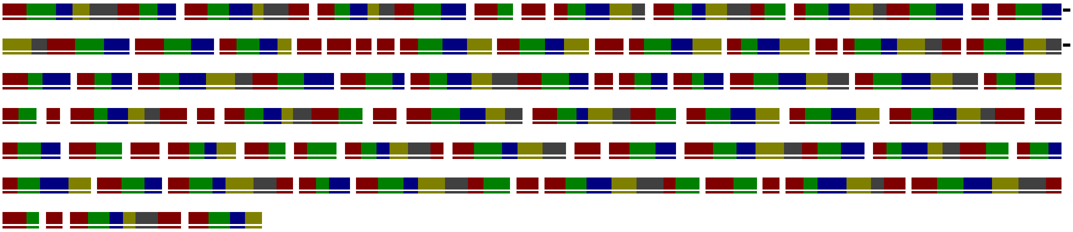
We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approximate, cluster, aggregate, outline, summarize, itemize, review, dip into, flip through, browse, glance into, leaf through, skim, refine, enumerate, glean, synopsize, winnow the wheat from the chaff and separate the sheep from the goats.

So, `normal,tolerant,stretch` or `normal,verytolerant,stretch` gives TeX enough degrees of freedom to produce good results. When we typeset Dutch documents we always use that alignment setting.

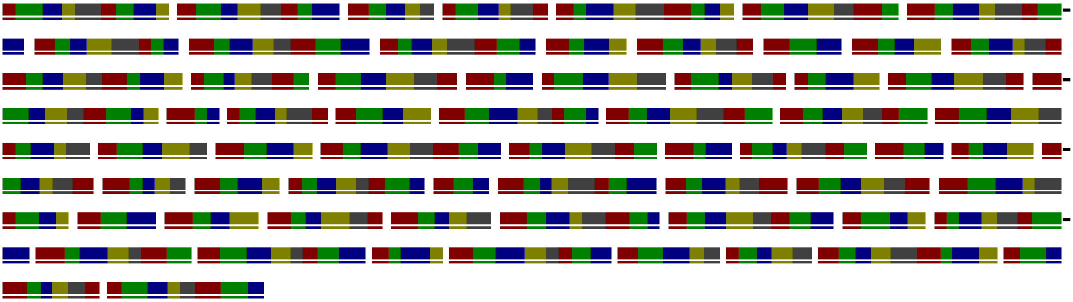
Let's simulate a langaueg with words of an average length. We just randomize the length, permit some hyphnetation.



If we assume more longer words we get:

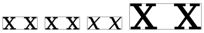


and if we have less short words we get:

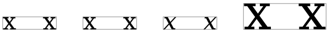


Even in the last case TEX can still quite well make a good paragraph, thanks to the large number of possible breakpoints (each color is a unit within a word). If you look careful you will see that the amount of whitespace differs per line.

A space in a text stream becomes so called glue: a horizontal skip with optional stretch and/or shrink. The values come from the font or are derived from spacing related properties of the font. This value can be overloaded by setting the `\spaceskip` register. In the next line we show the font driven spacing of some different fonts:



When we set `\spaceskip` to the `10pt` we get:



When we use the font related spacing typesettign the Zapf quote gives:

> Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction,

> as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

but when we use a fixed `\spaceskip` of `10pt` we get the following. This demonstrates that it really makes sense to have some stretch in the skip specificaton.

> Coming back to the use of typefaces in electronic publishing: many of the new typographers receive their knowledge and information about the rules of typography from books, from computer magazines or the instruction manuals which they get with the purchase of a PC or software. There is not so much basic instruction, as of now, as there was in the old days, showing the differences between good and bad typographic design. Many people are just fascinated by their PC's tricks, and think that a widely–praised program, called up on the screen, will make everything automatic from now on.

The French add spaces before punctuation but no extra space after punctuation. Traditional TeX can only handle additional spacing after characters at the end of a word. The term ‘frenchspacing’ is used for the feature that disables such extra spacing and `nonfrenchspacing` enables it. The extra space can differ per character and is specified by a factor. That factor (devided by 1000) is applied to the `\xspaceskip`. Spacing before such a final character is supported by ConTeXt but not a generic TeX feature. Frenchspacing looks like:

> foo: 1! foo: 2! foo: 3! foo: 4! foo: 5! foo: 6! foo: 7! foo: 8! foo: 9! foo: 10! foo: 11! foo: 12! foo: 13! foo: 14! foo: 15! foo: 16! foo: 17! foo: 18! foo: 19! foo: 20! foo: 21! foo: 22! foo: 23! foo: 24! foo: 25! foo: 26! foo: 27! foo: 28! foo: 29! foo: 30!

Contrary to:

> foo: 1! foo: 2! foo: 3! foo: 4! foo: 5! foo: 6! foo: 7! foo: 8! foo: 9! foo: 10! foo: 11! foo: 12! foo: 13! foo: 14! foo: 15! foo: 16! foo: 17! foo: 18! foo: 19! foo: 20! foo: 21! foo: 22! foo: 23! foo: 24! foo: 25! foo: 26! foo: 27! foo: 28! foo: 29! foo: 30!

The logic is as follows. When the `\spaceskip` is zero, then the spacing after a character is the one related to the font. Otherwise the `\spaceskip` is added. However, when a character has an `\sfcode` other that 1000, the stretch and shrink component of that glue are multiplied by sfcode/1000. When the `\sfcode` exceeds 2000 the `\xspaceskip` parameter is used. So, what frenchspacing actually does, is resetting those space related codes.

	`broad`	fixed	packed
.	`3000`	1000	1050
,	`1250`	1000	1050
?	`3000`	1000	1050

!	3000	1000	1050
:	2000	1000	1050
;	1500	1000	1050

The fixed variant is the french spacing as known for ages and discussed in the TEX book. You switch the model with:

```
\setupspacing[fixed] % french spacing
```

Additional models can be installed with `\installspacingmethod` and in the source code you can see how we did that for the above.

As mentioned, in TEX a space character `U+32`) is turned into glue. When you input some text with macros, you sometimes need to get a space into the stream. Take:

```
I am a \TEX user.
```

Here scanning the control sequence `\TEX` will gobble the space, so we need to do something:

```
I am a {\TEX} user, a happy \TEX\ user, yes,
a proud \TEX{} user, forever a \TEX\space user!
```

All these cases will inject a space after the logo. The second solution, the `\ ` originally was different because it introduces a space with an explicit `\sfcode` of 1000 (so a factor 1.0) but in ConTEXt we now just let that be a regular space in text mode and the original (primitive) maning in math mode.

Another special case is ~ which is a nobreak space with a fixed width and in some cases (like in tables) a space width the same width as a digit.

fixed	`xx xx\ X`	xx xx X	`xx dr.\ X`	xx dr. X
broad	`xx xx\ X`	xx xx X	`xx dr.\ X`	xx dr. X
fixed	`xx xx X`	xx xx X	`xx dr. X`	xx dr. X
broad	`xx xx X`	xx xx X	`xx dr. X`	xx dr. X
fixed	`xx xx~X`	xx xx X	`xx dr.~X`	xx dr. X
broad	`xx xx~X`	xx xx X	`xx dr.~X`	xx dr. X

These subtle details probably seldom get noticed. For instance, most languages use the `packed` variant while English, Turkish and Arabic are configured to use `broad`. To some extend you can say that the European continent uses the same spacing setup. You can adapt spacing for a language with:

```
\setuplanguage[en][spacing=packed]
```

In addition to a regular space there are many other spacing directives but these always concern fixed width spaces. When possible these spaces travel through the system as Unicode characters which also means that you can use such characters directly.

▮	`\nobreakspace \nbsp`	`U+00A0`	space
▮	`\ideographicspace`	`U+2000`	quad/2
■	`\ideographichalffillspace`	`U+2001`	quad
▮	`\twoperemspace`	`U+2002`	quad/2
■	`\quad`	`U+2003`	quad (emwidth)
▮	`\threeperemspace`	`U+2004`	quad/3
▮	`\fourperemspace`	`U+2005`	quad/4
❘	`\sixperemspace`	`U+2006`	quad/6
▮	`\figurespace`	`U+2007`	width of `0` (zero)
▮	`\punctuationspace`	`U+2008`	width of `.` (period)
❘	`\breakablethinspace`	`U+2009`	quad/8
❘	`\hairspace`	`U+200A`	quad/8
	`\zerowidthspace`	`U+200B`	
	`\zerowidthnonjoiner \zwnj`	`U+200C`	
	`\zerowidthjoiner \zwj`	`U+200D`	
❘	`\narrownobreakspace`	`U+202F`	quad/8
	`\zerowidthnobreakspace`	`U+FEFF`	

The `\nospacing` macro makes spaces disappear, not even a zero glue will be injected. Of course this macro will only be used grouped and in special cases.

5.3 Frequencies

Right from when ConTeXt became multilingual there have been users submitting language specific settings for their language. Some changed over time which indicates that there can be different views, which is not that surprising because many ‘typographic rules’ are just formalizations of ‘this is the way we did it for ages’. I often wonder what rules came from limitations in the systems used to get thing son paper: pens, pensils, letter by letter in wood or lead, line based printing, the size of paper, the reading direction, the quality of ink and paper, and so on. I've been present at debates about how high an accent should be placed on a character, depending on language, referring to some standard well known font that did it this or that way.

The dimensions of a page, the text area, the size and weight of characters, spacing, (the often neglectable or even debatable positive influence of) protrusion into the margin of certain characters, expansion of glyphs to give a better grayness . . . all this can lead to hefty discussions. They don't make a bad looking design or text with all kind of textual elements (not all documents are novels) look better.

The optimal width of a text column is one of these properties can opinions can vary on. Long lines in a small font or short lines in a big one are often not pleasant to read and if you have to turn the page every ten lines you might loose track of the content. But what does it say to have for instance 65 characters on a line. This is quite language dependent. How do spaces count? Anyway, in ConTEXt we have a variable that can help you decide what line length (text width) is acceptable. You can decide yourself what looks better:

322.7436pt
english
dejavu
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.

327.63428pt
dutch
dejavu
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.

326.45996pt
german
dejavu
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.

322.7436pt
french
dejavu
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.

282.60063pt
english
pagella
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.

286.75438pt
dutch
pagella
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.

284.63387pt
german
pagella
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.

282.60063pt
french
pagella
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 differences are not that large but at least you can play with it. The relevant helpers are:

```
\averagecharwidth
\languagecharwidth{language}
```

These are used like:

```
\hsize=65\averagecharwidth
\hsize=65\languagecharwidth{nl}
```

or when a width is asked for

```
\setupsomething[width=65\averagecharwidth]
\setupsomething[width=65\languagecharwidth{de}]
```

Keep in mind that these are font dependent so you might want to use

```
\freezemeasure[MyWidth][65\averagecharwidth]
```

and then use `\measure{MyWidth}` when needed. The frequences themselves are stored in tables like `lang-frq-nl.lua`:

```
return {
    language    = "nl",
    source      = "http://www.onzetaal.nl/advies/letterfreq.html",
    frequencies = {
        [0x61] = 7.47, [0x62] = 1.58,  [0x63] = 1.24, [0x64] =  5.93, [0x65] = 18.91,
        [0x66] = 0.81, [0x67] = 3.40,  [0x68] = 2.38, [0x69] =  6.50, [0x6A] =  1.46,
        [0x6B] = 2.25, [0x6C] = 3.57,  [0x6D] = 2.21, [0x6E] = 10.03, [0x6F] =  6.06,
        [0x70] = 1.57, [0x71] = 0.009, [0x72] = 6.41, [0x73] =  3.73, [0x74] =  6.79,
        [0x75] = 1.99, [0x76] = 2.85,  [0x77] = 1.52, [0x78] =  0.04, [0x79] =  0.035,
        [0x7A] = 1.39,
    }
}
```

As we only have a few such files, feel free to submit ones that suit your language.

You can show the frequencies in a nice table by loading a module (there is no need to have this in the core):

```
\usemodule[s-languages-frequencies]

\startcolumns[balance=yes] \en \showfrequencies \stopcolumns
\startcolumns[balance=yes] \de \showfrequencies \stopcolumns
\startcolumns[balance=yes] \nl \showfrequencies \stopcolumns
```

We get three tables:

`en: 4.96529pt`

U+00061	a	8.040
U+00062	b	1.540
U+00063	c	3.060
U+00064	d	3.990
U+00065	e	12.510
U+00066	f	2.300
U+00067	g	1.960
U+00068	h	5.490
U+00069	i	7.260
U+0006A	j	0.160
U+0006B	k	0.670
U+0006C	l	4.140
U+0006D	m	2.530
U+0006E	n	7.090
U+0006F	o	7.600
U+00070	p	2.000
U+00071	q	0.110
U+00072	r	6.120
U+00073	s	6.540
U+00074	t	9.250
U+00075	u	2.710
U+00076	v	0.990
U+00077	w	1.920
U+00078	x	0.190
U+00079	y	1.730
U+0007A	z	0.090

`de: 5.02247pt`

U+00061	a	6.470
U+00062	b	1.930
U+00063	c	2.680
U+00064	d	4.830
U+00065	e	17.480
U+00066	f	1.650
U+00067	g	3.060
U+00068	h	4.230
U+00069	i	7.730
U+0006A	j	0.270
U+0006B	k	1.460
U+0006C	l	3.490
U+0006D	m	2.580
U+0006E	n	9.840
U+0006F	o	2.980
U+00070	p	0.960
U+00071	q	0.020
U+00072	r	7.540
U+00073	s	6.830
U+00074	t	6.130
U+00075	u	4.170
U+00076	v	0.940
U+00077	w	1.480
U+00078	x	0.040
U+00079	y	0.080
U+0007A	z	1.140

`nl: 5.04054pt`

U+00061	a	7.470
U+00062	b	1.580
U+00063	c	1.240
U+00064	d	5.930
U+00065	e	18.910
U+00066	f	0.810
U+00067	g	3.400
U+00068	h	2.380
U+00069	i	6.500
U+0006A	j	1.460
U+0006B	k	2.250
U+0006C	l	3.570
U+0006D	m	2.210
U+0006E	n	10.030
U+0006F	o	6.060
U+00070	p	1.570
U+00071	q	0.009
U+00072	r	6.410
U+00073	s	3.730
U+00074	t	6.790
U+00075	u	1.990
U+00076	v	2.850
U+00077	w	1.520
U+00078	x	0.040
U+00079	y	0.035
U+0007A	z	1.390

5.4 Quotes

The limited support in first generation text editors has made some of these cultural aspects of typesetting disappear or at least it made users sloppy and a new generation forget about them. Quotes are an example and the default rendering on ascii keyboards hasn't helped either. For instance nowadays the Dutch double quotation marks, let's call them lower and upper nine quotes according to their shape, seems to have been replaced by upper double six quotes at the left and upper right double nine quotes on the right. Of course the real names are different.

`U+00022`	"	`quotedbl`	quotation mark
`U+000AB`	«	`leftguillemot`	left-pointing double angle quotation mark
`U+000BB`	»	`rightguillemot`	right-pointing double angle quotation mark
`U+02018`	‘	`quoteleft`	left single quotation mark
`U+02019`	’	`quoteright`	right single quotation mark
`U+0201A`	‚	`quotesinglebase`	single low-0x0009 quotation mark
`U+0201B`	‛		single high-reversed-0x0009 quotation mark
`U+0201C`	“	`quotedblleft`	left double quotation mark
`U+0201D`	”	`quotedblright`	right double quotation mark
`U+0201E`	„	`quotedblbase`	double low-0x0009 quotation mark
`U+0201F`	‟		double high-reversed-0x0009 quotation mark
`U+02039`	‹	`guilsingleleft`	single left-pointing angle quotation mark
`U+0203A`	›	`guilsingleright`	single right-pointing angle quotation mark
`U+02358`	⍘		apl functional symbol quote underbar
`U+0235E`	⍞		apl functional symbol quote quad
`U+0275B`	❛		heavy single turned comma quotation mark ornament
`U+0275C`	❜		heavy single comma quotation mark ornament
`U+0275D`	❝		heavy double turned comma quotation mark ornament
`U+0275E`	❞		heavy double comma quotation mark ornament
`U+0275F`			heavy low single comma quotation mark ornament
`U+02760`			heavy low double comma quotation mark ornament
`U+0276E`	❮		heavy left-pointing angle quotation mark ornament
`U+0276F`	❯		heavy right-pointing angle quotation mark ornament
`U+02E42`			double low-reversed-9 quotation mark
`U+0301D`			reversed double prime quotation mark
`U+0301E`			double prime quotation mark
`U+0301F`			low double prime quotation mark
`U+0A404`			yi syllable quot
`U+0FF02`			fullwidth quotation mark
`U+1F676`			sans-serif heavy double turned comma quotation mark ornament
`U+1F677`			sans-serif heavy double comma quotation mark ornament
`U+1F678`			sans-serif heavy low double comma quotation mark ornament
`U+E0022`			tag quotation mark

In the language definition file (`lang-def.mkiv`) we ue these names; they date from the

MkII times:

`\lowerleftsingleninequote`	`\quotesinglebase`
`\lowerleftdoubleninequote`	`\quotedblbase`
`\lowerrightsingleninequote`	`\quotesinglebase`
`\lowerrightdoubleninequote`	`\quotedblbase`
`\upperleftsingleninequote`	`\quoteright`
`\upperleftdoubleninequote`	`\quotedblright`
`\upperrightsingleninequote`	`\quoteright`
`\upperrightdoubleninequote`	`\quotedblright`
`\upperleftsinglesixquote`	`\quoteleft`
`\upperleftdoublesixquote`	`\quotedblleft`
`\upperrightsinglesixquote`	`\quoteleft`
`\upperrightdoublesixquote`	`\quotedblleft`
`\leftsubguillemot`	`\guilsingleleft`
`\rightsubguillemot`	`\guilsingleright`

In traditional TeX fonts a `''` and ``` `` ``` were implemented as ligatures but in ConTeXt we never supported that (and we won't). In fact, even using explicit quotes is not advised. When you use `\quotation`, `\quote` and friends the quotes will be used according the current main language.

Quotes are normally applied in quotations. The double quotes are used when we quote a person and single ones often when we refer to something. Each language does it different although computer usage manages to let most of us forget what is is the right tradition. Kids don't seem to care much so in the end it might evolve into all of us using the (US) English traditions.

```
\usemodule[s-languages-frequencies]

\startcolumns[balance=yes] \en \showfrequencies \stopcolumns
\startcolumns[balance=yes] \de \showfrequencies \stopcolumns
\startcolumns[balance=yes] \nl \showfrequencies \stopcolumns
```

By default, we obey the current language:

en: 4.96529pt

U+00061	a	8.040
U+00062	b	1.540
U+00063	c	3.060
U+00064	d	3.990
U+00065	e	12.510
U+00066	f	2.300
U+00067	g	1.960
U+00068	h	5.490
U+00069	i	7.260
U+0006A	j	0.160
U+0006B	k	0.670
U+0006C	l	4.140
U+0006D	m	2.530
U+0006E	n	7.090
U+0006F	o	7.600

U+00070	p	2.000
U+00071	q	0.110
U+00072	r	6.120
U+00073	s	6.540
U+00074	t	9.250
U+00075	u	2.710
U+00076	v	0.990
U+00077	w	1.920
U+00078	x	0.190
U+00079	y	1.730
U+0007A	z	0.090

`de: 5.02247pt`

U+00061	a	6.470
U+00062	b	1.930
U+00063	c	2.680
U+00064	d	4.830
U+00065	e	17.480
U+00066	f	1.650
U+00067	g	3.060
U+00068	h	4.230
U+00069	i	7.730
U+0006A	j	0.270
U+0006B	k	1.460
U+0006C	l	3.490
U+0006D	m	2.580
U+0006E	n	9.840
U+0006F	o	2.980
U+00070	p	0.960
U+00071	q	0.020
U+00072	r	7.540
U+00073	s	6.830
U+00074	t	6.130
U+00075	u	4.170
U+00076	v	0.940
U+00077	w	1.480
U+00078	x	0.040
U+00079	y	0.080
U+0007A	z	1.140

`nl: 5.04054pt`

U+00061	a	7.470
U+00062	b	1.580
U+00063	c	1.240
U+00064	d	5.930
U+00065	e	18.910
U+00066	f	0.810
U+00067	g	3.400
U+00068	h	2.380
U+00069	i	6.500
U+0006A	j	1.460
U+0006B	k	2.250
U+0006C	l	3.570
U+0006D	m	2.210
U+0006E	n	10.030
U+0006F	o	6.060
U+00070	p	1.570
U+00071	q	0.009
U+00072	r	6.410
U+00073	s	3.730
U+00074	t	6.790
U+00075	u	1.990
U+00076	v	2.850
U+00077	w	1.520
U+00078	x	0.040
U+00079	y	0.035
U+0007A	z	1.390

But we can force the main language:

Instead of `global` one can set a known language, nothing or `local`. Anyway, `global` gives us:

5.5 Sentences

Another language specific property is sub-sentences (or asides). We just demonstrate some rendering here. Again this is configured in the language definition file.

```
test \aside {test \aside {test} test} test |<|test test|>| test
```

For English, German, Dutch and French this looks as follows. As with quotations a nested instance can render differently.

test —test —test— test— test —test test— test
test – test – test – test – test – test test – test
test —test —test— test— test —test test— test
test — test — test — test — test — test test — test

5.6 Local control

Many ConTeXt commands can be controlled by the `align` parameter that accepts a list of directives. In addition to the justification directives the following ones are used to control the par builder.

option	**effect**
`tolerant`	accept suboptimal hyphenation i.e. give less warnings
`verytolerant`	accept even less optimal hyphenation
`stretch`	permit stretch between words to satisfy the hyphenation demands
`nothyphenated`	don't hyphenate at all

These directives are in fact just controllers for the following variables:

`variable`	**effect**
`\pretolerance`	when a paragraph is not hyphenated (first pass) and the result stays within this tolerance TeX doesn't try further
`\tolerance`	when a paragraph is hyphenated (second pass) and the result stays within this tolerance TeX doesn't try further
`\emergencystretch`	permit in a third pass this extra stretch in a line before complaining

You need to keep in mind that the left and right hyphenmin variables (per language or locally) also influence the way a paragraph is broken into lines. If you want a paragraph to have more lines than TeX want to give it, you can set the `\looseness` variable. Its value is forgotten when the paragraph is typeset so you don't need to reset it yourself. This mechanism will only kick in when there are three passes and a large enough `\emergencystretch` is set.

> 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 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.

For this text the results looks quite bad. Personally I never use(d) this command.

There are two commands that can be used to increment or decrement the current values of `\lefthyphenmin` and `\righthyphenmin`:

```
\lesshyphens
\morehyphens
```

These can come in handy in for instance titles.

6 Goodies

6.1 Introduction

There are some features that will only be used in rare cases. They were often implemented as experiment but found useful enough to keep around.

6.2 Spell checking

There are some means to check the spelling of words in your document but get it right: ConTEXt is not a spell-checker. These features were added in order to be able to do some quick checking of documents written by multiple authors. There are currently three options and we only show a simple examples.

First you need to load word lists. These are either text files with just words separated by spacing.

```
foobar  foo-bar  foo=bar  foo{}{}{}bar  foo{}{}{bar}
```

All these words become `foobar` which means that one can use words with discretionary specifications. A text list is loaded with:

```
\loadspellchecklist[en][t:/manuals/lua/words-en.txt]
```

Instead you can load a Lua file with words. Here we use the same structure that we use for the spell checker provided for SciTE:

```
return {
    max   = 9,
    min   = 6,
    n     = 2,
    words = {
        ["barfoo"]    = "Barfoo"
        ["foobarred"] = "foobarred",
    }
}
```

We use the same load command (you can also load bytecode files with suffix `luc` this way):

```
\loadspellchecklist[nl][t:/scite/data/context/lexers/data/spell-nl.lua]
```

Usage boils down to enabling the checker. If needed we can add more methods. The first method colors the known and unknown colors. Words shorter then the threshold of 4 will be skipped.

```
\setupspellchecking[state=start,method=1]
\en Is this written right or is this wromg?\par % m -> n error
\nl Is dit goed geschreven of niet?\par
\setupspellchecking[state=stop]
```

Is this written right or is this wromg?
Is dit goed geschreven of niet?

You can change the colors:

```
\definecolor[word:yes]    [g=.75]
\definecolor[word:no]     [r=.75]
```

The second method doesn't show anything but produces a file `jobname.words`) with used words. The `found` value of `list` is used as key in the produced table.

```
\setupspellchecking[state=start,method=2,list=found]
\en Is this written right or is this wrong?\par
\nl Is dit goed geschreven of niet?\par
\setupspellchecking[state=stop]
```

Is this written right or is this wrong?
Is dit goed geschreven of niet?

The produced table is:

```
return {
 ["categories"]={
  ["found"]={
   ["languages"]={
    ["en"]={
     ["list"]={
      ["right"]=1,
      ["this"]=2,
      ["written"]=1,
      ["wrong"]=1,
     },
     ["number"]={
      ["clean"]="function: 000000000048fa92",
      ["clearhyphenation"]="function: 00000000004901d1",
      ["clearpatterns"]="function: 00000000004902e9",
      ["current"]="function: 000000000048f790",
      ["gethjcode"]="function: 000000000048ffe4",
      ["has_language"]="function: 000000000048f910",
      ["hyphenate"]="function: 000000000048f876",
      ["hyphenation"]="function: 000000000048fd1c",
      ["hyphenationmin"]="function: 000000000048fe2d",
      ["id"]="function: 0000000000490401",
      ["new"]="function: 000000000048f7b5",
      ["patterns"]="function: 000000000048ff2b",
      ["postexhyphenchar"]="function: 0000000000490051",
      ["posthyphenchar"]="function: 0000000000490151",
      ["preexhyphenchar"]="function: 00000000004900d1",
      ["prehyphenchar"]="function: 000000000048fead",
      ["sethjcode"]="function: 000000000048fc6f",
      ["setwordhandler"]="function: 000000000048fb80",
     },
     ["parent"]="",
     ["patterns"]="us",
     ["tag"]="en",
     ["total"]=5,
     ["unique"]=4,
    },
    ["nl"]={
     ["list"]={
      ["geschreven"]=1,
      ["goed"]=1,
      ["niet"]=1,
     },
     ["number"]={
      ["clean"]="function: 000000000048fa92",
      ["clearhyphenation"]="function: 00000000004901d1",
      ["clearpatterns"]="function: 00000000004902e9",
      ["current"]="function: 000000000048f790",
      ["gethjcode"]="function: 000000000048ffe4",
      ["has_language"]="function: 000000000048f910",
      ["hyphenate"]="function: 000000000048f876",
      ["hyphenation"]="function: 000000000048fd1c",
      ["hyphenationmin"]="function: 000000000048fe2d",
      ["id"]="function: 0000000000490401",
      ["new"]="function: 000000000048f7b5",
      ["patterns"]="function: 000000000048ff2b",
      ["postexhyphenchar"]="function: 0000000000490051",
      ["posthyphenchar"]="function: 0000000000490151",
      ["preexhyphenchar"]="function: 00000000004900d1",
      ["prehyphenchar"]="function: 000000000048fead",
      ["sethjcode"]="function: 000000000048fc6f",
      ["setwordhandler"]="function: 000000000048fb80",
     },
     ["parent"]="",
     ["patterns"]="nl",
     ["tag"]="nl",
     ["total"]=3,
     ["unique"]=3,
    },
   },
   ["total"]=8,
  },
 },
 ["threshold"]=4,
 ["total"]=8,
 ["version"]=0x1p+0,
}
```

The result can be traced with a module:

```
\usemodule[s-languages-words]

\showwords
```

This shows up as:

category: found, language: en, total: 5, unique: 4: right (1) this (2) written (1) wrong (1)

category: found, language: nl, total: 3, unique: 3: geschreven (1) goed (1) niet (1)

The third mechanism colors languages differently. We only defined a few colors:

```
\definecolor[word:en]     [b=.75]
\definecolor[word:de]     [r=.75]
\definecolor[word:nl]     [g=.75]
\definecolor[word:unknown][r=.75,g=.75]
```

but you can of course define a color for your favourite language in a similar way.

```
\setupspellchecking[state=start,method=3]
\en Is this written right or is this wrong?\par
\nl Is dit goed geschreven of niet?\par
\setupspellchecking[state=stop]
```

Is this written right or is this wrong?
Is dit goed geschreven of niet?

7 Sorting

7.1 Introduction

Sorting is complex, not so much for English, Dutch, German, etc. only texts but there are languages and scripts that are more demanding. There are several complications:

- There can be characters that have accents, like à, á, â, ã, ä . . . that have a base shape a and in an index these often end up close to each other. The order can differ per language.
- There are upper and lowercase words and there can be different expectations to them being mixed or separated.
- Some scripts have characters that are combinations, like Æ, and one might want to see them as one character or two, in which the second one obeys the sorting order. The shape can dominate here.
- Some scripts, like Japanese, are a combination of several scripts and sorting then depends on normalization.
- When there are many glyphs, like in Chinese, the order can depend on the complexity of the glyph and when we're lucky that order is reflected in the numeric character order.

Often the rules are somewhat strict and one can doubt of the same rules would have been imposed if computers had been developed earlier. Given discussions one can doubt if the rules are really consistent or just there because someone (or a group) with influence set the standard (not so much different from grammar). So, if we deal with sorting, we do that in such a way that users can (to some extend) influence the outcome. After all, one important aspect of typesetting and organizing content is that the users gets the feeling of control and a diversion from a standard can be part of that. The reader will often not notice these details. In the next sections we will explore the way sorting is done in ConTEXt. The method evolved over a few decades. In MkII sorting happened between runs and it was just part of the processing of a document that users never really saw in action. Sorting just happened and few users will have noticed that we moved from a Modula program to a Perl script and ended up with a Ruby script. In fact, there is a Lua replacement but it never got tested well because we moved in to MkIV. There all happens inside the engine using Lua. Some principles stayed the same but we are more flexible now.

7.2 How it works

How does sorting work out? Take these words:

abracadabra
abräcàdábra
àbracádabrä
ábracadàbra
äbrácadabrà

As long as they end up in an order where the reader can find it, we're okay. After all we're pretty good in pattern recognition.

There are probably many ways to implement a sorter but the one we uses is more or less a follow up on the one we had for over a decade and was the result of an evolution based on user demand. It boils down to cleaning up the string in such a way that it can be split into meaningful characters. One can argue that we should use some kd of standardized sorting method but the problem is that we always have to deal with for instance embedded tex commands and mixed content, for instance numbers. And users using the same language can have different opinions about the rules too.

A word (or sequence of words) is split into characters. Because there can be TeX commands in there some cleanup happens beforehand. After that we create several lists with numbers that will be compared when sorting two entries.

We can best demonstrate this with a few examples. As usual an English language example is trivial.

	en	abracadabra
ch	raw character	a b r a c a d a b r a
uc	unicode	x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
zc	lowercase	x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
mc	lowercase - 1	x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
pc	lowercase + 1	x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
zm	zero mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
mm	minus mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
pm	plus mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02

When we add an uppercase character we get a slightly different outcome:

	en	Abracadabra
ch	raw character	A b r a c a d a b r a
uc	unicode	x41 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
zc	lowercase	x61 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
mc	lowercase - 1	x60 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
pc	lowercase + 1	x62 x62 x72 x61 x63 x61 x64 x61 x62 x72 x61
zm	zero mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
mm	minus mapping	x01 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
pm	plus mapping	x03 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02

Some characters will be split, like æ:

	en	æsop
ch	raw character	æ s o p
uc	unicode	xE6 x73 x6F x70
zc	lowercase	xE6 x73 x6F x70
mc	lowercase - 1	xE6 x73 x6F x70
pc	lowercase + 1	xE6 x73 x6F x70
zm	zero mapping	x02 x0A x26 x1E x20
mm	minus mapping	x02 x0A x26 x1E x20
pm	plus mapping	x02 x0A x26 x1E x20

It gets more complex when langiage specific demands kick in. Compare an English, German and Austrian split:

	en	Abräcàdábra
ch	raw character	A b r ä c à d á b r a
uc	unicode	x41 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
zc	lowercase	x61 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
mc	lowercase - 1	x60 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
pc	lowercase + 1	x62 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
zm	zero mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
mm	minus mapping	x01 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
pm	plus mapping	x03 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02

	de	Abräcàdábra
ch	raw character	A b r a e c à d á b r a
uc	unicode	x41 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61
zc	lowercase	x61 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61
mc	lowercase - 1	x60 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61
pc	lowercase + 1	x62 x62 x72 x61 x65 x63 xE0 x64 xE1 x62 x72 x61
zm	zero mapping	x02 x04 x24 x02 x0A x06 x02 x08 x02 x04 x24 x02
mm	minus mapping	x01 x04 x24 x02 x0A x06 x02 x08 x02 x04 x24 x02
pm	plus mapping	x03 x04 x24 x02 x0A x06 x02 x08 x02 x04 x24 x02

	de-at	Abräcàdábra
ch	raw character	A b r ä c à d á b r a
uc	unicode	x41 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
zc	lowercase	x61 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
mc	lowercase - 1	x60 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
pc	lowercase + 1	x62 x62 x72 xE4 x63 xE0 x64 xE1 x62 x72 x61
zm	zero mapping	x02 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02

mm	minus mapping	x01 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02
pm	plus mapping	x03 x04 x24 x02 x06 x02 x08 x02 x04 x24 x02

The way a character gets replaced, like ä into ae, is defined in `sort-lan.lua` using Lua tables. We will not explain all the obscure details here; most of the work is already done, so users are not bothered by these definitions. And new ones can often be made by copying and adapting an existing one.

The sorting itself is specified by a sequence:

default	zc,pc,zm,pm,uc
before	mm,mc,uc
after	pm,mc,uc
first	pc,mm,uc
last	mc,mm,uc

The raw character is what we get after the (language specific) replacement has been applied and the unicodes are used when comparing. Lowercasing is done using the Unicode lowercase code, but one can define language specific ones too. The plus and minus variants can be used to force lowercase before or after uppercase. The mapping is based on an alphabet specification so this can differ per language and again we also provide plus and minus values that depend on case. When a character has no case we use shapes instead. For instance, the shape of à is a. Digits are treated special and currently get an offset so that they end up last in the sort order.

```
ぱあ \jindex{ぱあ}
ぱー \jindex{ぱー}
ぱぁ \jindex{ぱぁ}
```

This three entry index should be sorted in the order: ぱー ぱぁ ぱあ.

ぱ

ぱあ	*60*
ぱぁ	*60*
ぱー	*60*

ぱ

ぱあ	*60*
ぱぁ	*60*
ぱー	*60*

	jp	ぱあ
ch	raw character	ぱ あ
uc	unicode	x3071 x3042
zc	lowercase	x3071 x3042
mc	lowercase - 1	x3071 x3042
pc	lowercase + 1	x3071 x3042

zm	zero mapping	x34 x02
mm	minus mapping	x34 x02
pm	plus mapping	x34 x02

	jp	ぱー
ch	raw character	ぱ ー
uc	unicode	x3071 x30FC
zc	lowercase	x3071 x30FC
mc	lowercase - 1	x3071 x30FC
pc	lowercase + 1	x3071 x30FC
zm	zero mapping	x34 x315C
mm	minus mapping	x34 x315C
pm	plus mapping	x34 x315C

	jp	ぱぁ
ch	raw character	ぱ あ
uc	unicode	x3071 x3042
zc	lowercase	x3071 x3042
mc	lowercase - 1	x3071 x3042
pc	lowercase + 1	x3071 x3042
zm	zero mapping	x34 x02
mm	minus mapping	x34 x02
pm	plus mapping	x34 x02

To be continued!

7.3 Special usage

The following example demonstrates how you can trick the sorter into doing other things:[5]

```
\startluacode
    local list = {
        -- old testament
        "Genesis", "Exodus", "Leviticus", "Numbers", "Deuteronomy", "Joshua",
        "Judges", "Ruth", "1 Samuel", "2 Samuel", "1 Kings", "2 Kings",
        "1 Chronicles", "2 Chronicles", "Ezra", "Nehemiah", "Esther", "Job",
        "Psalms", "Proverbs", "Ecclesiastes", "Canticles", "Isaiah", "Jeremiah",
        "Lamentations", "Ezekiel", "Daniel", "Hosea", "Joel", "Amos", "Obadiah",
        "Jonah", "Micah", "Nahum", "Habakkuk", "Zephaniah", "Haggai",
        "Zechariah", "Malachi",
```

---

[5] The `replacementlist` helper is the result of a request by John Grasty on the mailing list.

```
        -- new testament
        "Matthew", "Mark", "Luke", "John", "Acts", "Romans", "1 Corinthians",
        "2 Corinthians", "Galatians", "Ephesians", "Philippians", "Colossians",
        "1 Thessalonians", "2 Thessalonians", "1 Timothy", "2 Timothy", "Titus",
        "Philemon", "Hebrews", "James", "1 Peter", "2 Peter", "1 John", "2
John",
        "3 John", "Jude", "Revelation",
    }

    sorters.definitions["bible"] = {
        replacements = sorters.replacementlist(list),
    }
\stopluacode

\defineregister
  [booksort]
  [language=bible,
   n=3,
   criterium=text,
   indicator=no]
```

We use this as follows:

```
One   \booksort{Genesis+5.2}
Two   \booksort{Exodus+2}
Three \booksort{Genesis+45}
Four  \booksort{Philemon+2}
Five  \booksort{John+45}
Six   \booksort{1 John 1+45}
Seven \booksort{2 John 2+45}

\placeregister
  [booksort]
  [language=bible]
```

which gives:

One Two Three Four Five Six Seven

Genesis
  5.2 *62*
  45 *62*

Exodus
  2 *62*

John
  45 *62*

Philemon
  2 *62*

1 John 1
  45 *62*

2 John 2
  45 *62*

8 Options

8.1 Introduction

Hyphenation of words is controlled by so called patterns. They take a word and try to match parts with a pattern that describes where a hyphen can be injected. Preferred and discouraged injection points accumulate to a score that in the end determine where so called discretionary nodes gets injected in the list of glyphs that make a word. The patterns are language specific.

This mechanism is agnostic when it comes to the characters involved: they are just numbers. However, when in a next step font features like ligature building and kerning are applied we also have to deal with language specific properties (and meanings). Often a ligature at the boundary of a composed word can make reading confusing and has to be avoided. Some of that can be controlled by the font when it implements language specific features but because that approach is not based on a dictionary it is more about playing safe and prevention than about quality.

In the next sections a mechanism is discussed that also uses patterns. This time it is about controlling fonts as well as how hyphenation patterns are applied. This process kicks in before hyphenation is applied but it definitely has to be seen as part of that same process. It is integrated in hyphenation machinery and acts as preprocessor with the possibility to feedback and move forward. The implementation is such that when it's not used there is no performance penalty.[6]

There are several predefined operations that are characterized by keywords and shortcuts and collected in an option list that is part of a language goodie file. Examples can be found in the distribution in files with the suffix `llg` (Lua language goodie). The framework of such a file is:

```
return {
    name       = "whatever",
    version    = "1.00",
    comment    = "Goodies for experiments and demo.",
    author     = "Hans Hagen",
    copyright  = "ConTeXt development team",
    options    = {
        { ... },
        ........
        { ... },
```

---

[6] There are by now plenty of alternative approaches to these problems but after some discussion about the pro's and cons of each this new mechanism was made. I admit that the fun factor played a role. It is also one of the things we can do in LuaMetaTEX without worrying about a possible negative impact on LuaTEX users other than ConTEXt.

```
    }
}
```

These options will eventually result in patterns that are bound to words, think of:

`effe`	`foo|bar`	`..|..`	inhibit ligature
`foobar`	`foo=bar`	`...=...`	inhibit kerning
`somemore`	`some+more`	`....+....`	compound word

The whole repertoire is:

`a|b`	`a:norightligature, b:noleftligature`
`a=b`	`a:norightkern, b:noleftkern`
`a<b`	`b:noleftkern`
`a>b`	`a:norightkern`
`a+b`	`a:compound:b`

Later we will see how some can be combined. An option can be defined using entries in a subtable:

`patterns`	hash	`[snippet] = "replacement pattern"`
`words`	string	string of words, separated by whitespace
`prefixes`	string	snippets that combine with words (at the start)
`suffixes`	string	snippets that combine with words (at the end)
`matches`	array or number	a number or table indicating which match matters
`actions`	hash	`[character] = "action(s)"`
`characters`	string	permitted characters (additional hjcodes)
`return`	integer	what to do next

The default return value is 2 but there are some more:

`0`	go to the next (valid) word
`1`	restart
`2`	exceptions and after that patterns
`3`	patterns

There are some safeguards built in that force a restart. For instance when a word is replaced a restart is enforces unless we skip the word. A restart will not permit a second replacement (after all we need to avoid endless loops).

In a multi-line word list, lines that start with a comment trigger: Lua's double dash or the usual TeX percent sign.

8.2 Inhibiting

The next definition replaces `ff` by `f|f` in the words given and eventually block a ligature.

```
{
    patterns = {
        ff  = "f|f",
    },
    words = [[
        effe
    ]],
}
```

Some fonts provide the `ij` ligature or do some special kerning between these characters (something Dutch). Because it depends on the font logic if a dedicated replacement or kerning is used this is an example where we do this:

```
{
    patterns = {
        ij = "i|j",
    },
    actions = {
        ["|"] = "nokern noligature",
    },
    words = [[
        ijverig
     -- fijn -- to ligature fi or ij, that's the question
    ]],
}
```

A more extensive definition is the following. Here we explicitly define that only the first match in a word get treated. Here we not only block ligatures but also kerns.

```
{
    patterns = {
        ff  = "f|f",
    },
    matches = { 1 },
    actions = {
        ["|"] = "noligature nokern"
    },
    words = [[
        effe
        effeffe
    ]],
}
```

You can also omit the pattern when you inject specifiers yourself:

```
{
    actions = {
        ["|"] = "noligature nokern"
    },
    words = [[
        ef|fe
        ef|fef|fe
    ]],
}
```

You can also use different shortcuts:

```
{
    actions = {
        ["1"] = "noligature"
        ["2"] = "nokern"
    },
    words = [[
        ef1fe
        ef1fef2fe
    ]],
}
```

Although I cannot come up with a nice example, there can be reasons for inhibiting kerns. Here we inhibit kerns left of the upcoming character:

```
{
    patterns = {
        fo = "f<o",
        rm = "r<m",
    },
    words = [[
        information
    ]],
}
```

And here we inhibit kerns left of the previous and upcoming character:

```
{
    patterns = {
        th = "t=h",
    },
    words = [[
        thrive
    ]],
}
```

Just look in the files in the distribution for realistic examples, like

```
{
    patterns = {
        fi = "f|i",
    },
    words = [[
        deafish dwarfish elfish oafish selfish
    ]],
    suffixes = [[
        ness ly
    ]]
}
```

where we block ligatures in 15 words. There's also a `prefixes` key.

8.3 Replacements

Replacements are probably not used that much but here is one for German. Not only is the uppercase variant of ß seldom used, many fonts don't provide it so we can best replace it:

```
{
    characters = "", -- uppercase ß, not visible in all verbatim fonts
    patterns   = {
        [""] = "SS", -- key is uppercase ß
    },
}
```

Here we define that character as valid, something that normally is done with the patterns but patterns don't have them. If we do not specify it here, the hyphenator will skip this word. For the record: this can also be done with a font feature that decomposes the character.

8.4 Compound words

You might want to suppress ligatures and maybe even kerning when compound words are involved.

```
{
    patterns = {
        ff = "f+f",
    },
    words = [[
        aaaaffaaaa
        bbffbb
    ]],
}
```

Again you can also say:

```
{
    words = [[
        aaaaf|faaaa
        bbf|fbb
    ]],
}
```

But patterns make sense when you have a large list (that might come from some other source than yourself).

The next specification will turn two times three `bla`'s into a compound word but also make sure that we have at least 4 characters left and right of a potential break.

```
    {
        left  = 4,
        right = 4,
        words = [[
            blablabla+blablabla
        ]],
    }
```

8.5 Performance

Although these mechanisms introduce overhead, the performance hit in LMTX is not that large. This is because the number of words in a document is limited and Lua is fast enough.

8.6 Plugins

This interface is preliminary but for the record I put an example here anyway.

```
local n = 0
function document.myhack(original)
    n = n + 1
    print(n,original)
    return original
end

languages.installhandler("de","document.myhack")
```

One can manipulate a text as in:

```
function document.myhack(original)
    local t = utf.split(original)
    local t = table.reverse(t)
    local f = t[#t]
    local l = t[1]
    if characters.upper(f) == f then
        t[1]  = characters.upper()
        t[#t] = characters.lower(f)
    end
    local original = table.concat(t)
    return original
end

languages.installhandler("en","document.myhack")
```

The text will fed again into the hyphenator and treated in the normal way. There are some safeguards against the text being processed twice.

8.7 Tracing

You can also embed definitions in the source file:

```
\startlanguageoptions[de]
    Zapf|innovation
\stoplanguageoptions
```

8.8 Exceptions

When you set exceptions in a goodie file, it will use the plugin mechanism to check for them. This is a bit more efficient than using the internal checkerm which actually also goes via aLua hash.

```
{
    exceptions = [[
        a-very{-}{-}{w}eird{1}{2}{3}(w)ord
    ]],
}
```

Watch out: when you specify a discretionary replacement three braced valued are passed: the pre, post and replace text. The replace text is used in the lookup, unless you add a string between parentheses, which then will be used instead. A digit between bracket will apply a penalty according to the following logic (in the engine): A zero digit results in `\hyphenpenalty`, otherwise the digits 1 upto 9 will be used as multiplier for `\exceptionpenalty` when that value is larger than 100000, otherwise `\exceptionpenalty` is used.

8.9 Tracing

The following tracker can be used:

```
\enabletrackers[languages.goodies]
```

In addition the style `languages-goodies` implements some tracing options. You can just run that one to see what it does.

The engine itself has also a tracing option: `\tracinghyphenation`. When set to zero nothing is shown, when set to one redundant patterns will be reported. A value of two reports what words get fed into the hyphenator and if they got hyphenated. A value of three gives more detail: when a word gets hyphenated the relevant (resulting) part of the node list is shown. You need to set `\tracingonline` to a value larger than zero to get this reported to the console. Expects lots of extra output to the console for large documents but it can be revealing.

A Appendix

A.1 The language files

Todo.

A.2 The `mtx-patterns` script

Todo.

A.3 Installed sorters

```
\usemodule[s-languages-sorting]

\showinstalledsorting
```

language	DIN 5007-1
parent	default
method	mm,mc,uc
replacements	ß=ss
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	DIN 5007-2
parent	default
method	mm,mc,uc
replacements	ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=ss
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	Duden
parent	default
method	mm,mc,uc
replacements	ß=s
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	be
parent	default

method	mm,mc,uc
replacements	none
order	а б в г д е ё ж з і й к л м н о п р с т у ў ф х ц ч ш ы ь э ю я
entries	а=а б=б в=в г=г д=д е=е ж=ж з=з й=й к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш ы=ы ь=ь э=э ю=ю я=я ё=е і=і ў=ў

language	bg
parent	default
method	mm,mc,uc
replacements	none
order	а б в г д е ж з и й к а л а м н о п р с т у ф х ц ч ш щ ъ ь ю я
entries	а=а а=а б=б в=в г=г д=д е=е ж=ж з=з и=и й=й к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш щ=щ ъ=ъ ь=ь ю=ю я=я

language	bible
parent	default
method	mm,mc,uc
replacements	Genesis=`0x10001` Exodus=`0x10002` Leviticus=`0x10003` Numbers=`0x10004` Deuteronomy=`0x10005` Joshua=`0x10006` Judges=`0x10007` Ruth=`0x10008` 1 Samuel=`0x10009` 2 Samuel=`0x1000A` 1 Kings=`0x1000B` 2 Kings=`0x1000C` 1 Chronicles=`0x1000D` 2 Chronicles=`0x1000E` Ezra=`0x1000F` Nehemiah=`0x10010` Esther=`0x10011` Job=`0x10012` Psalms=`0x10013` Proverbs=`0x10014` Ecclesiastes=`0x10015` Canticles=`0x10016` Isaiah=`0x10017` Jeremiah=`0x10018` Lamentations=`0x10019` Ezekiel=`0x1001A` Daniel=`0x1001B` Hosea=`0x1001C` Joel=`0x1001D` Amos=`0x1001E` Obadiah=`0x1001F` Jonah=`0x10020` Micah=`0x10021` Nahum=`0x10022` Habakkuk=`0x10023` Zephaniah=`0x10024` Haggai=`0x10025` Zechariah=`0x10026` Malachi=`0x10027` Matthew=`0x10028` Mark=`0x10029` Luke=`0x1002A` John=`0x1002B` Acts=`0x1002C` Romans=`0x1002D` 1 Corinthians=`0x1002E` 2 Corinthians=`0x1002F` Galatians=`0x10030` Ephesians=`0x10031` Philippians=`0x10032` Colossians=`0x10033` 1 Thessalonians=`0x10034` 2 Thessalonians=`0x10035` 1 Timothy=`0x10036` 2 Timothy=`0x10037` Titus=`0x10038` Philemon=`0x10039` Hebrews=`0x1003A` James=`0x1003B` 1 Peter=`0x1003C` 2 Peter=`0x1003D` 1 John=`0x1003E` 2 John=`0x1003F` 3 John=`0x10040` Jude=`0x10041` Revelation=`0x10042`
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	cs
parent	cz
method	mm,mc,uc

replacements	ch=0x10001 Ch=0x10001 CH=0x10001
order	a á b c č d ď e é ě f g h 0x10001 i í j k l m n ň o ó p q r ř s š t ť u ú ů v w x y ý z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ó=o ú=u ý=y č=č ď=d ě=e ň=n ř=ř š=š ť=t ů=u ž=ž 0x10001=ch

language	cu
parent	default
method	mm,mc,uc
replacements	оу=0x10001 ОУ=0x1000B
order	а б в г д є ж ѕ 0x0A643 з 0x0A641 и і ї ћ к л м н о п р с т у 0x00479 0x0A64B 0x10001 ф х 0x00461 0x0047F 0x0047D 0x0A64D ц ч ш щ ъ ы ꙑ ь ѣ ю ꙗ ѥ 0x00467 0x00469 ѫ ѭ 0x0046F ѱ ѳ ѵ ѷ
entries	а=а б=б в=в г=г д=д ж=ж з=з и=и к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш щ=щ ъ=ъ ы=ы ь=ь ю=ю є=є ѕ=ѕ і=и ї=и ћ=ћ 0x00461=0x00461 ѣ=ѣ ѥ=ѥ 0x00467=0x00467 0x00469=0x00469 ѫ=ѫ ѭ=ѭ 0x0046F=0x0046F ѱ=ѱ ѳ=ѳ ѵ=ѵ ѷ=ѵ 0x00479=у 0x0047D=0x00461 0x0047F=0x00461 0x0A641=з 0x0A643=ѕ 0x0A64B=у 0x0A64D=0x00461 ꙑ=ы ꙗ=ꙗ 0x10001=у

language	cz
parent	default
method	mm,mc,uc
replacements	ch=0x10001 Ch=0x10001 CH=0x10001
order	a á b c č d ď e é ě f g h 0x10001 i í j k l m n ň o ó p q r ř s š t ť u ú ů v w x y ý z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ó=o ú=u ý=y č=č ď=d ě=e ň=n ř=ř š=š ť=t ů=u ž=ž 0x10001=ch

language	da
parent	no
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z æ ø å
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z å=å æ=æ ø=ø

language	de
parent	default
method	mm,mc,uc
replacements	ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=s

order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	de-AT
parent	default
method	mm,mc,uc
replacements	none
order	a ä b c d e f g h i j k l m n o ö p q r s t u ü v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ä=ä ö=ö ü=ü

language	de-CH
parent	de
method	mm,mc,uc
replacements	ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=s
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	de-DE
parent	de
method	mm,mc,uc
replacements	ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=s
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	default
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	deo
parent	de
method	mm,mc,uc
replacements	ä=ae Ä=Ae ö=oe Ö=Oe ü=ue Ü=Ue ß=s
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	en
parent	default

method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	es
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n ñ o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ñ=ñ ó=o ú=u ü=u

language	et
parent	default
method	mm,mc,uc
replacements	none
order	a b c č d e f g h i j k l m n o p q r s š z ž t u v w õ ä ö ü x y
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ä=ä õ=õ ö=ö ü=ü č=č š=š ž=ž

language	fi
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z å ä ö
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ä=ä å=å ö=ö

language	fr
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	gr
parent	default
method	mm,mc,uc
replacements	α=αa ά=αb ὰ=αc ὰ=αd ᾳ=αe ἀ=αf ἁ=αg ἄ=αh ἂ=αi ἆ=αj ἁ=αk ἅ=αl ἃ=αm ἇ=αn ᾁ=αo ᾴ=αp ᾲ=αq ᾷ=αr ᾄ=αs ὰ=αt ᾅ=αu ᾃ=αv ᾆ=αw ᾇ=αx ε=εa έ=εb ὲ=εc ἐ=εd ἔ=εe ἒ=εf ἑ=εg ἕ=εh ἓ=εi η=ηa

	η=ηb ή=ηc ὴ=ηd ῆ=ηe ῃ=ηf ἠ=ηg ἤ=ηh ἢ=ηi ἦ=ηj ᾐ=ηk ἡ=ηl ἥ=ηm ἣ=ηn ἧ=ηo ᾑ=ηp ῄ=ηq ῂ=ηr ῇ=ηs ᾔ=ηt ᾒ=ηu ᾕ=ηv ᾓ=ηw ᾖ=ηx ᾗ=ηy ι=ιa ί=ιb ὶ=ιc ῖ=ιd ἰ=ιe ἴ=ιf ἲ=ιg ἶ=ιh ἱ=ιi ἵ=ιj ἳ=ιk ἷ=ιl ϊ=ιm ΐ=ιn ῒ=ιo ῗ=ιp ο=οa ό=οb ὸ=οc ὀ=οd ὄ=οe ὂ=οf ὁ=οg ὅ=οh ὃ=οi ρ=ρa ῤ=ῤb ῥ=ῥc υ=υa ύ=υb ὺ=υc ῦ=υd ὐ=υe ὔ=υf ὒ=υg ὖ=υh ὑ=υi ὕ=υj ὓ=υk ὗ=υl ϋ=υm ΰ=υn ῢ=υo ω=ωa ώ=ωb ὼ=ωc ῶ=ωd ῳ=ωe ὠ=ωf ὤ=ωg ὢ=ωh ὦ=ωi ᾠ=ωj ὡ=ωk ὥ=ωl ὣ=ωm ὧ=ωn ᾡ=ωo ῴ=ωp ῲ=ωq ῷ=ωr ᾤ=ωs ᾢ=ωt ᾥ=ωu ᾣ=ωv ᾦ=ωw ᾧ=ωx
order	α β γ δ ε ζ η θ ι κ λ μ ν ξ ο π ρ σ ς τ υ φ χ ψ ω
entries	ΐ=ι ά=α έ=ε ή=η ί=ι ΰ=υ α=α β=β γ=γ δ=δ ε=ε ζ=ζ η=η θ=θ ι=ι κ=κ λ=λ μ=μ ν=ν ξ=ξ ο=ο π=π ρ=ρ ς=ς σ=σ τ=τ υ=υ φ=φ χ=χ ψ=ψ ω=ω ϊ=ι ϋ=υ ό=ο ύ=υ ώ=ω ἀ=α ἁ=α ἂ=α ἃ=α ἄ=α ἅ=α ἆ=α ἇ=α ἐ=ε ἑ=ε ἒ=ε ἓ=ε ἔ=ε ἕ=ε ἠ=η ἡ=η ἢ=η ἣ=η ἤ=η ἥ=η ἦ=η ἧ=η ἰ=ι ἱ=ι ἲ=ι ἳ=ι ἴ=ι ἵ=ι ἶ=ι ἷ=ι ὀ=ο ὁ=ο ὂ=ο ὃ=ο ὄ=ο ὅ=ο ὐ=υ ὑ=υ ὒ=υ ὓ=υ ὔ=υ ὕ=υ ὖ=υ ὗ=υ ὠ=ω ὡ=ω ὢ=ω ὣ=ω ὤ=ω ὥ=ω ὦ=ω ὧ=ω ὰ=α ὲ=ε ὴ=η ὶ=ι ὸ=ο ὺ=υ ὼ=ω ᾀ=α ᾂ=α ᾃ=α ᾄ=α ᾅ=α ᾆ=α ᾇ=α ᾐ=η ᾑ=η ᾒ=η ᾓ=η ᾔ=η ᾕ=η ᾖ=η ᾗ=η ᾠ=ω ᾡ=ω ᾢ=ω ᾣ=ω ᾤ=ω ᾥ=ω ᾦ=ω ᾧ=ω ᾲ=α ᾳ=α ᾴ=α ᾶ=α ᾷ=α ῂ=η ῃ=η ῄ=η ῆ=η ῇ=η ῒ=ι ῖ=ι ῗ=ι ῢ=υ ῤ=ῤ ῥ=ῥ ῦ=υ ῧ=υ ῲ=ω ῳ=ω ῴ=ω ῶ=ω ῷ=ω

language	he
parent	default
method	mm,mc,uc
replacements	none
order	`0x005D0` `0x005D1` `0x005D2` `0x005D3` `0x005D4` `0x005D5` `0x005D6` `0x005D7` `0x005D8` `0x005D9` `0x005DB` `0x005DC` `0x005DE` `0x005E0` `0x005E1` `0x005E2` `0x005E4` `0x005E6` `0x005E7` `0x005E8` `0x005E9` `0x005EA`
entries	`0x005D0`=`0x005D0` `0x005D1`=`0x005D1` `0x005D2`=`0x005D2` `0x005D3`=`0x005D3` `0x005D4`=`0x005D4` `0x005D5`=`0x005D5` `0x005D6`=`0x005D6` `0x005D7`=`0x005D7` `0x005D8`=`0x005D8` `0x005D9`=`0x005D9` `0x005DB`=`0x005DB` `0x005DC`=`0x005DC` `0x005DE`=`0x005DE` `0x005E0`=`0x005E0` `0x005E1`=`0x005E1` `0x005E2`=`0x005E2` `0x005E4`=`0x005E4` `0x005E6`=`0x005E6` `0x005E7`=`0x005E7` `0x005E8`=`0x005E8` `0x005E9`=`0x005E9` `0x005EA`=`0x005EA`

language	hr
parent	default
method	mm,mc,uc
replacements	dž=`0x10001` DŽ=`0x1000B` lj=`0x10002` LJ=`0x1000C` nj=`0x10003` NJ=`0x1000D`
order	a b c č ć d `0x10001` đ e f g h i j k l `0x10002` m n `0x10003` o p r s š t u v z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v z=z ć=ć č=č đ=đ š=š ž=ž `0x10001`=dž `0x10002`=lj `0x10003`=nj

language	hu
parent	default
method	mm,mc,uc
replacements	cs=0x10001 CS=0x1000B dz=0x10002 DZ=0x1000C dzs=0x10003 DZS=0x1000D gy=0x10004 GY=0x1000E ly=0x10005 LY=0x1000F ny=0x10006 NY=0x10010 sz=0x10007 SZ=0x10011 ty=0x10008 TY=0x10012 zs=0x10009 ZS=0x10013
order	a á b c 0x10001 d 0x10002 0x10003 e é f g 0x10004 h i í j k l 0x10005 m n 0x10006 o ó ö ő p q r s 0x10007 t 0x10008 u ú ü ű v w x y z 0x10009
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a é=e í=i ó=o ö=ö ú=u ü=ü ő=ö ű=ü 0x10001=cs 0x10002=dz 0x10003=dzs 0x10004=gy 0x10005=ly 0x10006=ny 0x10007=sz 0x10008=ty 0x10009=zs

language	is
parent	default
method	mm,mc,uc
replacements	none
order	a á b d ð e é f g h i í j k l m n o ó p r s t u ú v x y ý þ æ ö
entries	a=a b=b d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v x=x y=y á=a æ=æ é=e í=i ð=ð ó=o ö=ö ú=u ý=y þ=þ

language	it
parent	default
method	mm,mc,uc
replacements	none
order	a á b c d e é è f g h i í ì j k l m n o ó ò p q r s t u ú ù v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a è=e é=e ì=i í=i ò=o ó=o ù=u ú=u

language	jp
parent	default
method	zm
replacements	0x03041=0x03042 0x03043=0x03044 0x03045=0x03046 0x03047=0x03048 0x03049=0x0304A 0x03063=0x03064 0x03083=0x03084 0x03085=0x03086 0x03087=0x03088
order	0x03042 0x03044 0x03046 0x03048 0x0304A 0x0304B 0x0304D 0x0304F 0x03051 0x03053 0x03055 0x03057 0x03059 0x0305B 0x0305D 0x0305F 0x03061 0x03064 0x03066 0x03068 0x0306A 0x0306B 0x0306C 0x0306D 0x0306E 0x0306F 0x03072 0x03075 0x03078 0x0307B 0x0307E 0x0307F 0x03080 0x03081 0x03082

	0x03084 0x03086 0x03088 0x03089 0x0308A 0x0308B 0x0308C 0x0308D 0x0308F 0x03090 0x03091 0x03092 0x03093
entries	0x03042=0x03042 0x03044=0x03044 0x03046=0x03046 0x03048=0x03048 0x0304A=0x0304A 0x0304B=0x0304B 0x0304D=0x0304D 0x0304F=0x0304F 0x03051=0x03051 0x03053=0x03053 0x03055=0x03055 0x03057=0x03057 0x03059=0x03059 0x0305B=0x0305B 0x0305D=0x0305D 0x0305F=0x0305F 0x03061=0x03061 0x03064=0x03064 0x03066=0x03066 0x03068=0x03068 0x0306A=0x0306A 0x0306B=0x0306B 0x0306C=0x0306C 0x0306D=0x0306D 0x0306E=0x0306E 0x0306F=0x0306F 0x03072=0x03072 0x03075=0x03075 0x03078=0x03078 0x0307B=0x0307B 0x0307E=0x0307E 0x0307F=0x0307F 0x03080=0x03080 0x03081=0x03081 0x03082=0x03082 0x03084=0x03084 0x03086=0x03086 0x03088=0x03088 0x03089=0x03089 0x0308A=0x0308A 0x0308B=0x0308B 0x0308C=0x0308C 0x0308D=0x0308D 0x0308F=0x0308F 0x03090=0x03090 0x03091=0x03091 0x03092=0x03092 0x03093=0x03093

language	kr
parent	default
method	mm,mc,uc
replacements	none
order	0x03131 0x03134 0x03137 0x03139 0x03141 0x03142 0x03145 0x03147 0x03148 0x0314A 0x0314B 0x0314C 0x0314D 0x0314E a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	la
parent	default
method	mm,mc,uc
replacements	æ=ae Æ=AE
order	a ā ă b c d e ē ĕ f g h i ī ĭ j k l m n o ō ŏ p q r s t u ū ŭ v w x y ȳ y̆ z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=i k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=u w=w x=x y=y y̆=y z=z ā=a ă=a ē=e ĕ=e ī=i ĭ=i ō=o ŏ=o ū=u ŭ=u ȳ=y

language	lt
parent	default
method	mm,mc,uc
replacements	ch=0x10001 CH=0x1000B
order	a ą b c 0x10001 č d e ę ė f g h i į y j k l m n o p r s š t u ų ū v z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v y=i z=z ą=a č=č ė=e ę=e į=i š=š ū=u ų=u ž=ž 0x10001=c

language	lv
parent	default

method	mm,mc,uc
replacements	none
order	a ā b c č d e ē f g ģ h i ī j k ķ l ļ m n ņ o ō p r ŗ s š t u ū v z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v z=z ā=a č=č ē=e ģ=ģ ī=i ķ=ķ ļ=ļ ņ=ņ ō=o ŗ=ŗ š=š ū=u ž=ž

language	nl
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z

language	no
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z æ ø å
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z å=å æ=æ ø=ø

language	ocs-scn
parent	default
method	mm,mc,uc
replacements	ou=0x10001 OU=0x10015 g’=0x10002 G’=0x10016 št=0x10003 ŠT=0x10017 ju=0x10004 JU=0x10018 ja=0x10005 JA=0x10019 je=0x10006 JE=0x1001A ję=0x10007 JĘ=0x1001B jǫ=0x10008 JǪ=0x1001C ks=0x10009 KS=0x1001D ps=0x1000A PS=0x1001E th=0x1000B TH=0x1001F šč=0x1000C ŠČ=0x10020
order	a b v g d e ž ʒ z i ï 0x10002 k l m n o p r s t u f x o c č š 0x10003 0x1000C ъ y 0x10001 ь ě 0x10004 0x10005 0x10006 ę 0x10007 ǫ 0x10008 0x10009 0x1000A 0x1000B ü
entries	a=a b=b c=c d=d e=e f=f g=g i=i k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v x=x y=y z=z ï=ï ü=ü č=č ę=ę ě=ě š=š ž=ž ǫ=ǫ ʒ=ʒ ъ=ъ ь=ь 0x10001=y 0x10002=g’ 0x10003=št 0x10004=ju 0x10005=ja 0x10006=je 0x10007=ję 0x10008=jǫ 0x10009=ks 0x1000A=ps 0x1000B=th 0x1000C=šč

language	pl
parent	default
method	mm,mc,uc
replacements	none

order	a ą b c ć d e ę f g h i j k l ł m n ń o ó p q r s ś t u v w x y z ź ż
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ó=ó ą=ą ć=ć ę=ę ł=ł ń=ń ś=ś ź=ź ż=ż

language	pt
parent	default
method	mm,mc,uc
replacements	none
order	a á â ã à b c ç d e é ê f g h i í j k l m n o ó ô õ p q r s t u ú ü v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z à=a á=a â=a ã=a ç=c é=e ê=e í=i ó=o ô=o õ=o ú=u ü=u

language	ro
parent	default
method	mm,mc,uc
replacements	none
order	a ă â b c d e f g h i î j k l m n o p q r s ș t ț u v w x y z
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z â=â î=î ă=ă ș=ș ț=ț

language	ru
parent	default
method	mm,mc,uc
replacements	none
order	а б в г д е ё ж з и і й к л м н о п р с т у ф х ц ч ш щ ъ ы ь ѣ э ю я ѳ ѵ
entries	а=а б=б в=в г=г д=д е=е ж=ж з=з и=и й=й к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш щ=щ ъ=ъ ы=ы ь=ь э=э ю=ю я=я ё=е і=и ѣ=ѣ ѳ=ѳ ѵ=ѵ

language	ru-iso9
parent	default
method	mm,mc,uc
replacements	''=`0x10001`
order	a b v g d e ë ž z i ì j k l m n o p r s t u f h c č š ŝ ʺ `0x10001` y ʹ ' ě è û â û â
entries	'=ʹ a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p r=r s=s t=t u=u v=v y=y z=z â=â è=è ë=ë ì=ì û=û č=č ě=ě ŝ=ŝ š=š ž=ž ʹ=ʹ ʺ=ʺ `0x10001`=ʺ

language	sk
parent	default
method	mm,mc,uc

replacements	dz=0x10001 dz=0x1000B dž=0x10002 dž=0x1000C ch=0x10003 ch=0x1000D
order	a á ä b c č d ď 0x10001 0x10002 e é f g h 0x10003 i í j k l ĺ ľ m n ň o ó ô p q r ŕ s š t ť u ú v w x y ý z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z á=a ä=a é=e í=i ó=o ô=o ú=u ý=y č=č ď=d ĺ=l ľ=l ň=n ŕ=r š=š ť=t ž=ž 0x10001=dz 0x10002=dž 0x10003=ch

language	sl
parent	default
method	mm,mc,uc
replacements	none
order	a b c č ć d đ e f g h i j k l m n o p q r s š t u v w x y z ž
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ć=ć č=č đ=đ š=š ž=ž

language	sr
parent	default
method	mm,mc,uc
replacements	none
order	а б в г д ђ е ж з и ј к л љ м н њ о п р с т ћ у ф х ц ч џ ш
entries	а=а б=б в=в г=г д=д е=е ж=ж з=з и=и к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш ђ=ђ ј=ј љ=љ њ=њ ћ=ћ џ=џ

language	sv
parent	default
method	mm,mc,uc
replacements	none
order	a b c d e f g h i j k l m n o p q r s t u v w x y z å ä ö
entries	a=a b=b c=c d=d e=e f=f g=g h=h i=i j=j k=k l=l m=m n=n o=o p=p q=q r=r s=s t=t u=u v=v w=w x=x y=y z=z ä=ä å=å ö=ö

language	uk
parent	default
method	mm,mc,uc
replacements	none
order	а б в г ґ д е є ж з и і ї й к л м н о п р с т у ф х ц ч ш щ ь ю я
entries	а=а б=б в=в г=г д=д е=е ж=ж з=з и=и й=й к=к л=л м=м н=н о=о п=п р=р с=с т=т у=у ф=ф х=х ц=ц ч=ч ш=ш щ=щ ь=ь ю=ю я=я є=є і=і ї=ї ґ=ґ

language	yi
parent	he

method	mm,mc,uc
replacements	0x005D00x005B7=0x005D0 0x005D00x005B8=0x005D0 0x005D10x005BC=0x005D1 0x005D10x005BF=0x0FB4C 0x005D50x005BC=0x005D5 0x005F0=0x005D50x005D5 0x005F1=0x005D50x005D9 0x0FB1D=0x005D9 0x005F2=0x005D90x005D9 0x0FB1F=0x005D90x005D9 0x005DB0x005BC=0x0FB3B 0x005DA=0x005DB 0x005DD=0x005DE 0x005DF=0x005E0 0x005E40x005BC=0x0FB44 0x005E4=0x0FB44 0x005E40x005BF=0x0FB4E 0x005E3=0x0FB4E 0x005E5=0x005E6 0x0FB2A=0x005E9 0x005E90x005C2=0x0FB2B 0x005EA0x005BC=0x0FB4A
order	0x005D0 0x005D1 0x0FB4C 0x005D2 0x005D3 0x005D4 0x005D5 0x005D6 0x005D7 0x005D8 0x005D9 0x0FB3B 0x005DB 0x005DC 0x005DE 0x005E0 0x005E1 0x005E2 0x0FB44 0x0FB4E 0x005E6 0x005E7 0x005E8 0x005E9 0x0FB2B 0x0FB4A 0x005EA
entries	0x005D0=0x005D0 0x005D1=0x005D1 0x005D2=0x005D2 0x005D3=0x005D3 0x005D4=0x005D4 0x005D5=0x005D5 0x005D6=0x005D6 0x005D7=0x005D7 0x005D8=0x005D8 0x005D9=0x005D9 0x005DB=0x005DB 0x005DC=0x005DC 0x005DE=0x005DE 0x005E0=0x005E0 0x005E1=0x005E1 0x005E2=0x005E2 0x005E4=0x005E4 0x005E6=0x005E6 0x005E7=0x005E7 0x005E8=0x005E8 0x005E9=0x005E9 0x005EA=0x005EA 0x0FB2B=0x0FB2B 0x0FB3B=0x0FB3B 0x0FB4A=0x0FB4A 0x0FB4C=0x0FB4C 0x0FB4E=0x0FB4E

A.4 Verbose counters

```
\usemodule[s-languages-counters]

\showverbosecounters[language={en,es}]
```

en	es	number
zero		0
one	uno	1
two	dos	2
three	tres	3
four	cuatro	4
five	cinco	5
six	seis	6
seven	siete	7
eight	ocho	8
nine	nueve	9
ten	diez	10
eleven	once	11
twelve	doce	12
thirteen	trece	13

fourteen	catorce	14
fifteen	quince	15
sixteen	dieciséis	16
seventeen	diecisiete	17
eighteen	dieciocho	18
nineteen	diecinueve	19
twenty	veinte	20
	veintiuno	21
	veintidós	22
	veintitrés	23
	veinticuatro	24
	veinticinco	25
	veintiséis	26
	veintisiete	27
	veintiocho	28
	veintinueve	29
thirty	treinta	30
forty	cuarenta	40
fifty	cincuenta	50
sixty	sesenta	60
seventy	setenta	70
eighty	ochenta	80
ninety	noventa	90
hundred	ciento	100
	doscientos	200
	trescientos	300
	cuatrocientos	400
	quinientos	500
	seiscientos	600
	setecientos	700
	ochocientos	800
	novecientos	900
thousand	mil	1000
million	millón	1000000
billion	mil millones	1000000000
trillion	billón	1000000000000

This book explains how we support languages (and scripts) in ConTEXt MkIV and LuaTEX. Some of the mechanisms discussed are generic and not ConTEXt specific. We discuss the way languages are dealt with in the engine, hyphenation, standard features and additional goodies. Tracing and the extensibility of code are also discussed.

work in progress