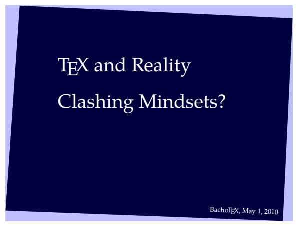
# Presentations

This is a selection of three sets of presentations at meetings and a bunch of examples for some styles that come with **ConTEXt**. In **MkIV** some of the older styles have been dropped. There are more presentations, starting mid 1990's: some are lost, others make no sense showing (or have articles instead) and I have no time anyway to collect all of them.

We show the first four pages but you can find the complete files in the **ConTEXt** distribution. This file is generated automatically so it's also a snapshot. Maybe some day I find the time to add some more.

## bachotex-2010-clash



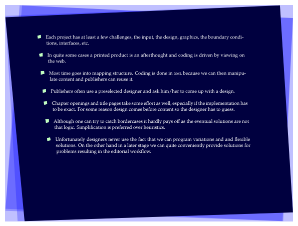
TEX and Reality

Clashing Mindsets?

BachoTEX, May 1, 2010

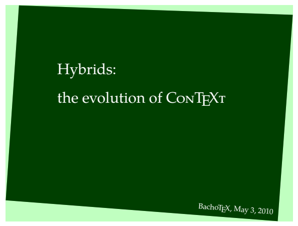
- There can be several reasons for using TEX. Some are subjective.
- You like the way it works: you edit a document using a simple editor, add a couple of directives and delay rendering. It's the content and structure that matter.
- You need it for instance because you have to typeset math and you believe that no other tool can do a better job on that.
- You found out that it can save time because it is programmable and after all, programming is a nice distraction from writing.
- You don't want to change a 20 year old habit and why quit using something that you know well by now.
- You like an occasional fight with a batch oriented system and updating (sometimes to the extend of compiling) can be done while watching a movie.
- You dislike learning a new program every 5 years. Of course it would be different if we'd live for 500 years.

- I've always used TEX and can do what I need to do with it. I like to focus on what can be done instead of what can't.
- I don't like disposable tools and am quite lucky that TEX still can adapt to my needs.
- I like my job but only when using the current tools and cooking up reuseable solutions.
- I need it for rendering (often educational) content and also use it for fun.
- In the process I need to implement styles based on designs provided by designers, most probably only know click and point tools but some of them can think outside that box.



- Each project has at least a few challenges, the input, the design, graphics, the boundary conditions, interfaces, etc.
- In quite some cases a printed product is an afterthought and coding is driven by viewing on the web.
- Most time goes into mapping structure. Coding is done in XML because we can then manipulate content and publishers can reuse it.
- Publishers often use a preselected designer and ask him/her to come up with a design.
- Chapter openings and title pages take some effort as well, especially if the implementation has to be exact. For some reason design comes before content so the designer has to guess.
- Although one can try to catch bordercases it hardly pays off as the eventual solutions are not that logic. Simplification is preferred over heuristics.
- Unfortunately designers never use the fact that we can program variations and and flexible solutions. On the other hand in a later stage we can quite conveniently provide solutions for problems resulting in the editorial workflow.

## bachotex-2010-move

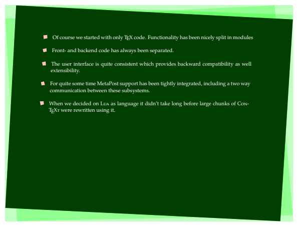


Hybrids:

the evolution of ConTEXt

BachoTEX, May 3, 2010

- Coding in TEX is quite natural and given a proper macro set the overhead is not that large.
- Coding in XML makes sense when you have to manipulate or reuse your data and when TEX is just the renderer.
- For non-artistic graphics MetaPost provides a convenient input language. It also plays well with TEX.
- Some problems can more conveniently be solved in a procedural programming language and Lua perfectly fits in there.

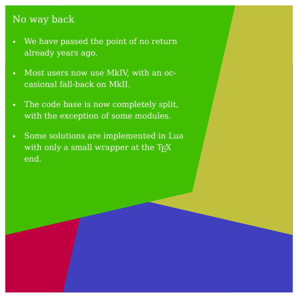


- Of course we started with only TEX code. Functionality has been nicely split in modules
- Front- and backend code has always been separated.
- The user interface is quite consistent which provides backward compatibility as well extensibility.
- For quite some time MetaPost support has been tightly integrated, including a two way communication between these subsystems.
- When we decided on Lua as language it didn't take long before large chunks of ConTEXt were rewritten using it.

- Most font handling takes place in Lua and as usual with TEX we can do more than fonts provide.
- Other subsystems, like languages, input encoding, file io and xml also were among the first to be supported by Lua.
- Lots of information is now carried around, especially related to structure. This will permit users more freedom.
- Notes, descriptions and enumerations also rely on Lua.
- Graphics (including MetaPost) is all dealt with in Lua. Float management is currently on the agenda.
- The backend code is completely rewritten in Lua. We've disabled the low level primitives so that third party modules can not spoil the game (this was already the case in MkII).
- Eventually most management tasks will move from TEX to Lua, but we keep in pace with LuaTEX development and don't push things to the limit.

## bachotex-2011-cld-and-mkvi

**Finding the balance**

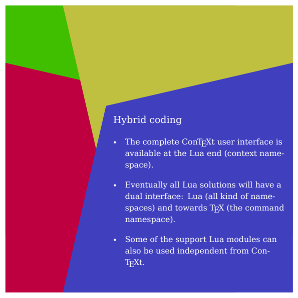


No way back

- We have passed the point of no return already years ago.
- Most users now use MkIV, with an occasional fall-back on MkII.
- The code base is now completely split, with the exception of some modules.
- Some solutions are implemented in Lua with only a small wrapper at the TEX end.

To get an idea

- structure: sectioning, notes, descriptions, registers, synonyms
- typesetting: sectioning, notes, descriptions,



Hybrid coding

- The complete ConTEXt user interface is available at the Lua end (context namespace).
- Eventually all Lua solutions will have a dual interface: Lua (all kind of namespaces) and towards TEX (the command namespace).
- Some of the support Lua modules can also be used independent from ConTEXt.

BachoTEX

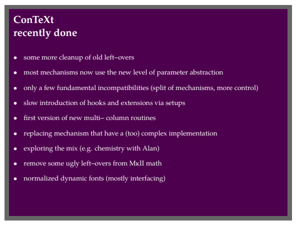
## bachotex-2013-bits

Bits and pieces:

ConTeXt
MetaPost
Lua and more

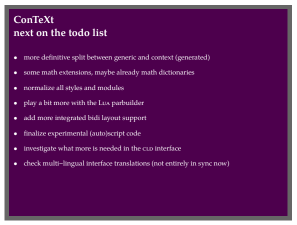
Hans Hagen
EuroBachoTEX
May 2013

| | | |
|---|---|---|
| ConTeXt<br>recently done | ConTeXt<br>next on the todo list | MetaPost<br>recently done |
| MetaPost<br>next on the todo list | Lua<br>recently done | Lua<br>next on the todo list |
| LuaTEX<br>recently done | LuaTEX<br>next on the todo list | Fonts<br>next on the todo list |
| Manuals | Scripts | Speed |



ConTeXt
recently done

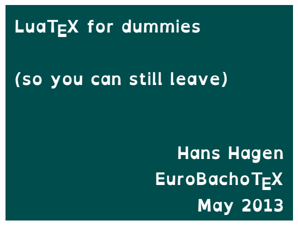
- some more cleanup of old left–overs
- most mechanisms now use the new level of parameter abstraction
- only a few fundamental incompatibilities (split of mechanisms, more control)
- slow introduction of hooks and extensions via setups
- first version of new multi– column routines
- replacing mechanism that have a (too) complex implementation
- exploring the mix (e.g. chemistry with Alan)
- remove some ugly left–overs from MkII math
- normalized dynamic fonts (mostly interfacing)



ConTeXt
next on the todo list

- more definitive split between generic and context (generated)
- some math extensions, maybe already math dictionaries
- normalize all styles and modules
- play a bit more with the Lua parbuilder
- add more integrated bidi layout support
- finalize experimental (auto)script code
- investigate what more is needed in the CLD interface
- check multi–lingual interface translations (not entirely in sync now)

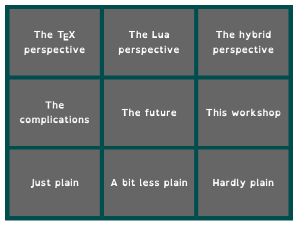
## bachotex-2013-luatex



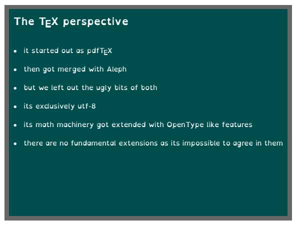
LuaTEX for dummies

(so you can still leave)

Hans Hagen
EuroBachoTEX
May 2013

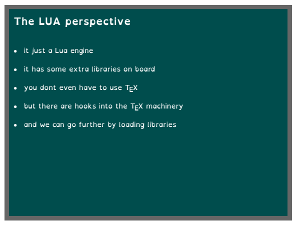


| | | |
|---|---|---|
| The TEX perspective | The Lua perspective | The hybrid perspective |
| The complications | The future | This workshop |
| Just plain | A bit less plain | Hardly plain |



The TEX perspective

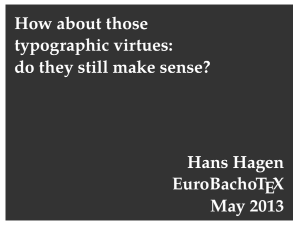
- it started out as pdfTEX
- then got merged with Aleph
- but we left out the ugly bits of both
- its exclusively utf-8
- its math machinery got extended with OpenType like features
- there are no fundamental extensions as its impossible to agree in them



The LUA perspective

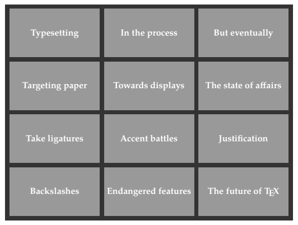
- it just a Lua engine
- it has some extra libraries on board
- you dont even have to use TEX
- but there are hooks into the TEX machinery
- and we can go further by loading libraries

## bachotex-2013-sense

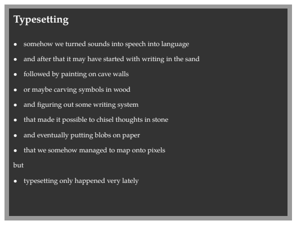


How about those
typographic virtues:
do they still make sense?

Hans Hagen
EuroBachoTEX
May 2013



| | | |
|---|---|---|
| Typesetting | In the process | But eventually |
| Targeting paper | Towards displays | The state of affairs |
| Take ligatures | Accent battles | Justification |
| Backslashes | Endangered features | The future of TEX |

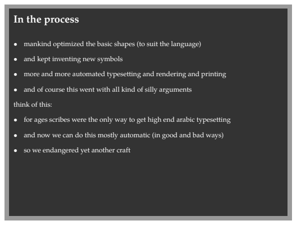


Typesetting

- somehow we turned sounds into speech into language
- and after that it may have started with writing in the sand
- followed by painting on cave walls
- or maybe carving symbols in wood
- and figuring out some writing system
- that made it possible to chisel thoughts in stone
- and eventually putting blobs on paper
- that we somehow managed to map onto pixels

but

- typesetting only happened very lately



In the process

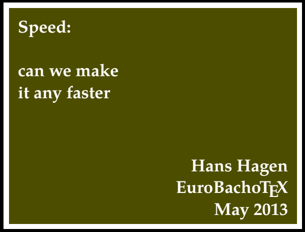
- mankind optimized the basic shapes (to suit the language)
- and kept inventing new symbols
- more and more automated typesetting and rendering and printing
- and of course this went with all kind of silly arguments

think of this:

- for ages scribes were the only way to get high end arabic typesetting
- and now we can do this mostly automatic (in good and bad ways)
- so we endangered yet another craft

BachoTEX

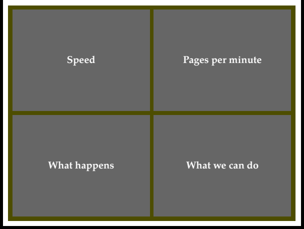
## bachotex-2013-speed



Speed:

can we make
it any faster

Hans Hagen
EuroBachoTEX
May 2013

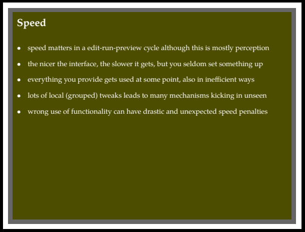


Speed

Pages per minute

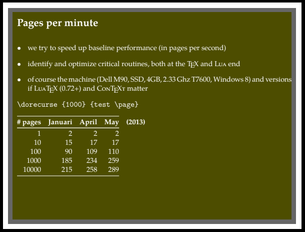
What happens

What we can do



Speed

- speed matters in a edit-run-preview cycle although this is mostly perception
- the nicer the interface, the slower it gets, but you seldom set something up
- everything you provide gets used at some point, also in inefficient ways
- lots of local (grouped) tweaks leads to many mechanisms kicking in unseen
- wrong use of functionality can have drastic and unexpected speed penalties



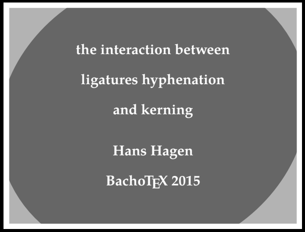
Pages per minute

- we try to speed up baseline performance (in pages per second)
- identify and optimize critical routines, both at the TEX and Lua end
- of course the machine (Dell M90, SSD, 4GB, 2.33 Ghz T7600, Windows 8) and versions if LuaTEX (0.72+) and ConTEXt matter

```
\dorecurse {1000} {test \page}
```

| # pages | Januari | April | May | (2013) |
|---|---|---|---|---|
| 1 | 2 | 2 | 2 | |
| 10 | 15 | 17 | 17 | |
| 100 | 90 | 109 | 110 | |
| 1000 | 185 | 234 | 259 | |
| 10000 | 215 | 258 | 289 | |

## bachotex-2015-ligatures



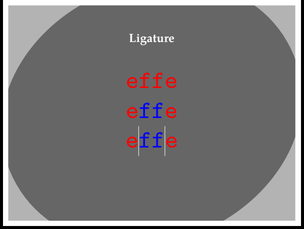
the interaction between

ligatures hyphenation

and kerning

Hans Hagen

BachoTEX 2015

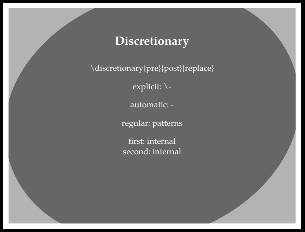


Ligature

effe

effe

effe



Discretionary

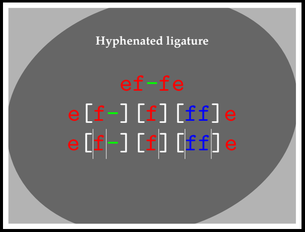
\discretionary{pre}{post}{replace}

explicit: \-

automatic: -

regular: patterns

first: internal
second: internal



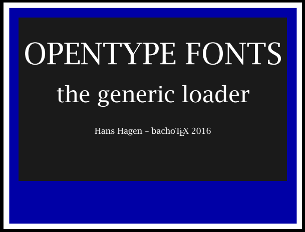
Hyphenated ligature

ef-fe

e[f-] [f] [ff]e

e[f-] [f] [ff]e

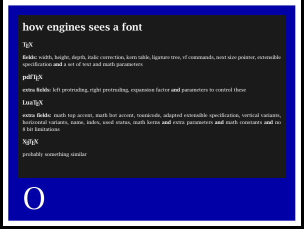
## bachotex-2016-opentype



OPENTYPE FONTS

the generic loader

Hans Hagen – bachoTEX 2016



how engines sees a font

TEX

**fields:** width, height, depth, italic correction, kern table, ligature tree, vf commands, next size pointer, extensible specification **and** a set of text and math parameters

pdfTEX

**extra fields:** left protruding, right protruding, expansion factor **and** parameters to control these

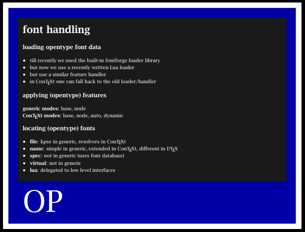
LuaTEX

**extra fields:** math top accent, math bot accent, tounicode, adapted extensible specification, vertical variants, horizontal variants, name, index, used status, math kerns **and** extra parameters **and** math constants **and** no 8 bit limitations

XeTEX

probably something similar

O



font handling

**loading opentype font data**

- till recently we used the built-in fontforge loader library
- but now we use a recently written Lua loader
- but use a similar feature handler
- in ConTEXt one can fall back to the old loader/handler

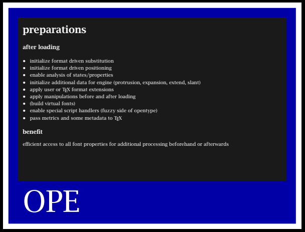
**applying (opentype) features**

**generic modes:** base, node
**ConTEXt modes:** base, node, auto, dynamic

**locating (opentype) fonts**

- **file**: kpse in generic, resolvers in ConTEXt
- **name**: simple in generic, extended in ConTEXt, different in LATEX
- **spec**: not in generic (uses font database)
- **virtual**: not in generic
- **lua**: delegated to low level interfaces

OP



preparations

**after loading**

- initialize format driven substitution
- initialize format driven positioning
- enable analysis of states/properties
- initialize additional data for engine (protrusion, expansion, extend, slant)
- apply user or TEX format extensions
- apply manipulations before and after loading
- (build virtual fonts)
- enable special script handlers (fuzzy side of opentype)
- pass metrics and some metadata to TEX

**benefit**

efficient access to all font properties for additional processing beforehand or afterwards

OPE

# bachotex-2016-toolbox

## Grandpa’s toolbox

(making closets)

## Dad’s hobby

(cleaning closets)

## Uncle’s friends

(talking closets)

bachoTEX 2016

− +

- there’s closets and closets
- take the ones you put stuff in ... e.g. that you buy at ikea
- you have to assemble them so you get out your toolbox
- this kid sits next to you wondering what that tool is
- what is natural for you to use might not be so for them
- but grandpa likes to carry over his knowledge and experiences

− **closets** +

- but . . . kids get their information from the internet, not from you
- (they watch discovery channel or national geographic and know a lot)
- or they look at vloggers (no bloggers) trying to learn something
- and they keep moving on . . . and on
- do gp’s live long tools really make sense to them?

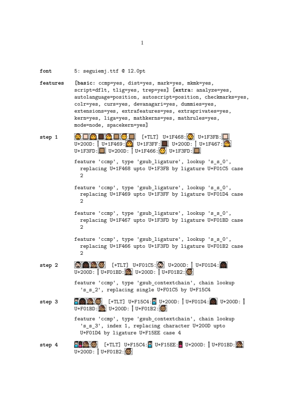
− **interest** +

- grandpa also has a vlogger, he is called knuth
- gp is as locked into tex as the kids are into apps
- he’s a do-it-him-self kind of person
- will his grandkids love watching him crafting
- and hear his stories about meetings and journals

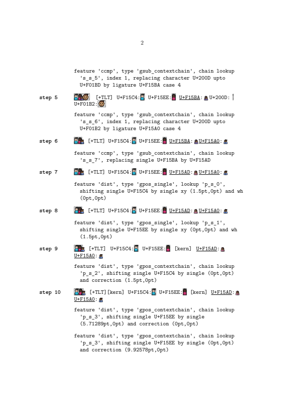
− **grandpas** +

BachoTEX

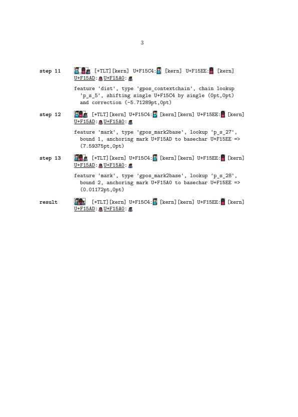
# bachotex-2017-emoji-demo



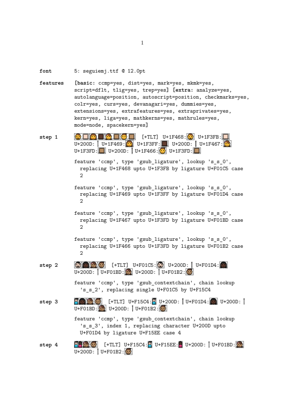
```
1

font       5: seguiemj.ttf @ 12.0pt

features   [basic: ccmp=yes, dist=yes, mark=yes, mkmk=yes,
           script=dflt, tlig=yes, trep=yes] [extra: analyze=yes,
           autolanguage=position, autoscript=position, checkmarks=yes,
           colr=yes, curs=yes, devanagari=yes, dummies=yes,
           extensions=yes, extrafeatures=yes, extraprivates=yes,
           kern=yes, liga=yes, mathkerns=yes, mathrules=yes,
           mode=node, spacekern=yes]

step 1       [+TLT] U+1F468:  U+1F3FB:
           U+200D:  U+1F469:  U+1F3FF:  U+200D:  U+1F467:
           U+1F3FD:  U+200D:  U+1F466:  U+1F3FD:

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F468 upto U+1F3FB by ligature U+F01C5 case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F469 upto U+1F3FF by ligature U+F01D4 case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F467 upto U+1F3FD by ligature U+F01BD case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F466 upto U+1F3FD by ligature U+F01B2 case
             2

step 2       [+TLT] U+F01C5:  U+200D:  U+F01D4:
           U+200D:  U+F01BD:  U+200D:  U+F01B2:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_2', replacing single U+F01C5 by U+F15C4

step 3       [+TLT] U+F15C4:  U+200D:  U+F01D4:  U+200D:
           U+F01BD:  U+200D:  U+F01B2:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_3', index 1, replacing character U+200D upto
             U+F01D4 by ligature U+F15EE case 4

step 4       [+TLT] U+F15C4:  U+F15EE:  U+200D:  U+F01BD:
           U+200D:  U+F01B2:
```



```
2

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_5', index 1, replacing character U+200D upto
             U+F01BD by ligature U+F15BA case 4

step 5       [+TLT] U+F15C4:  U+F15EE:  U+F15BA:  U+200D:
           U+F01B2:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_6', index 1, replacing character U+200D upto
             U+F01B2 by ligature U+F15A0 case 4

step 6       [+TLT] U+F15C4:  U+F15EE:  U+F15BA:  U+F15A0:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_7', replacing single U+F15BA by U+F15AD

step 7       [+TLT] U+F15C4:  U+F15EE:  U+F15AD:  U+F15A0:

           feature 'dist', type 'gpos_single', lookup 'p_s_0',
             shifting single U+F15C4 by single xy (1.5pt,0pt) and wh
             (0pt,0pt)

step 8       [+TLT] U+F15C4:  U+F15EE:  U+F15AD:  U+F15A0:

           feature 'dist', type 'gpos_single', lookup 'p_s_1',
             shifting single U+F15EE by single xy (0pt,0pt) and wh
             (1.5pt,0pt)

step 9       [+TLT] U+F15C4:  U+F15EE:  [kern] U+F15AD:
           U+F15A0:

           feature 'dist', type 'gpos_contextchain', chain lookup
             'p_s_2', shifting single U+F15C4 by single (0pt,0pt)
             and correction (1.5pt,0pt)

step 10      [+TLT][kern] U+F15C4:  U+F15EE:  [kern] U+F15AD:
           U+F15A0:

           feature 'dist', type 'gpos_contextchain', chain lookup
             'p_s_3', shifting single U+F15EE by single
             (5.71289pt,0pt) and correction (0pt,0pt)

           feature 'dist', type 'gpos_contextchain', chain lookup
             'p_s_3', shifting single U+F15EE by single (0pt,0pt)
             and correction (9.92578pt,0pt)
```

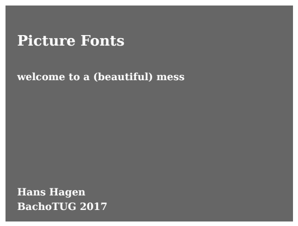


```
3

step 11      [+TLT][kern] U+F15C4:  [kern] U+F15EE:  [kern]
           U+F15AD:  U+F15A0:

           feature 'dist', type 'gpos_contextchain', chain lookup
             'p_s_5', shifting single U+F15C4 by single (0pt,0pt)
             and correction (-5.71289pt,0pt)

step 12      [+TLT][kern] U+F15C4:  [kern][kern] U+F15EE:  [kern]
           U+F15AD:  U+F15A0:

           feature 'mark', type 'gpos_mark2base', lookup 'p_s_27',
             bound 1, anchoring mark U+F15AD to basechar U+F15EE =>
             (7.59375pt,0pt)

step 13      [+TLT][kern] U+F15C4:  [kern][kern] U+F15EE:  [kern]
           U+F15AD:  U+F15A0:

           feature 'mark', type 'gpos_mark2base', lookup 'p_s_28',
             bound 2, anchoring mark U+F15A0 to basechar U+F15EE =>
             (0.01172pt,0pt)

result       [+TLT][kern] U+F15C4:  [kern][kern] U+F15EE:  [kern]
           U+F15AD:  U+F15A0:
```



```
1

font       5: seguiemj.ttf @ 12.0pt

features   [basic: ccmp=yes, dist=yes, mark=yes, mkmk=yes,
           script=dflt, tlig=yes, trep=yes] [extra: analyze=yes,
           autolanguage=position, autoscript=position, checkmarks=yes,
           colr=yes, curs=yes, devanagari=yes, dummies=yes,
           extensions=yes, extrafeatures=yes, extraprivates=yes,
           kern=yes, liga=yes, mathkerns=yes, mathrules=yes,
           mode=node, spacekern=yes]

step 1       [+TLT] U+1F468:  U+1F3FB:
           U+200D:  U+1F469:  U+1F3FF:  U+200D:  U+1F467:
           U+1F3FD:  U+200D:  U+1F466:  U+1F3FD:

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F468 upto U+1F3FB by ligature U+F01C5 case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F469 upto U+1F3FF by ligature U+F01D4 case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F467 upto U+1F3FD by ligature U+F01BD case
             2

           feature 'ccmp', type 'gsub_ligature', lookup 's_s_0',
             replacing U+1F466 upto U+1F3FD by ligature U+F01B2 case
             2

step 2       [+TLT] U+F01C5:  U+200D:  U+F01D4:
           U+200D:  U+F01BD:  U+200D:  U+F01B2:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_2', replacing single U+F01C5 by U+F15C4

step 3       [+TLT] U+F15C4:  U+200D:  U+F01D4:  U+200D:
           U+F01BD:  U+200D:  U+F01B2:

           feature 'ccmp', type 'gsub_contextchain', chain lookup
             's_s_3', index 1, replacing character U+200D upto
             U+F01D4 by ligature U+F15EE case 4

step 4       [+TLT] U+F15C4:  U+F15EE:  U+200D:  U+F01BD:
           U+200D:  U+F01B2:
```

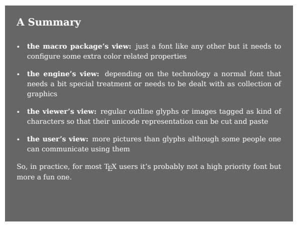
# bachotex-2017-emoji



## Picture Fonts

**welcome to a (beautiful) mess**

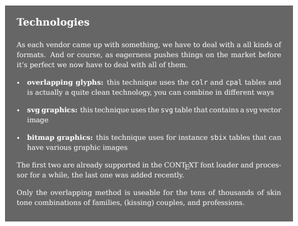
**Hans Hagen**
**BachoTUG 2017**



## A Summary

- **the macro package's view:** just a font like any other but it needs to configure some extra color related properties
- **the engine's view:** depending on the technology a normal font that needs a bit special treatment or needs to be dealt with as collection of graphics
- **the viewer's view:** regular outline glyphs or images tagged as kind of characters so that their unicode representation can be cut and paste
- **the user's view:** more pictures than glyphs although some people one can communicate using them

So, in practice, for most TeX users it's probably not a high priority font but more a fun one.



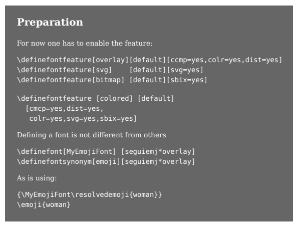
## Technologies

As each vendor came up with something, we have to deal with a all kinds of formats. And or course, as eagerness pushes things on the market before it's perfect we now have to deal with all of them.

- **overlapping glyphs:** this technique uses the `colr` and `cpal` tables and is actually a quite clean technology, you can combine in different ways
- **svg graphics:** this technique uses the `svg` table that contains a svg vector image
- **bitmap graphics:** this technique uses for instance `sbix` tables that can have various graphic images

The first two are already supported in the ConTeXt font loader and processor for a while, the last one was added recently.

Only the overlapping method is useable for the tens of thousands of skin tone combinations of families, (kissing) couples, and professions.



## Preparation

For now one has to enable the feature:

```
\definefontfeature[overlay][default][ccmp=yes,colr=yes,dist=yes]
\definefontfeature[svg]    [default][svg=yes]
\definefontfeature[bitmap] [default][sbix=yes]

\definefontfeature [colored] [default]
  [cmcp=yes,dist=yes,
   colr=yes,svg=yes,sbix=yes]
```

Defining a font is not different from others

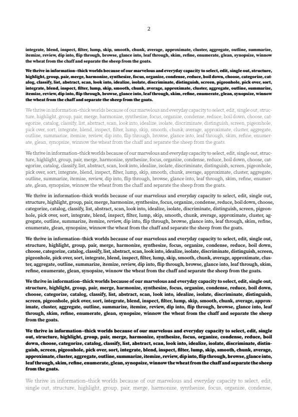
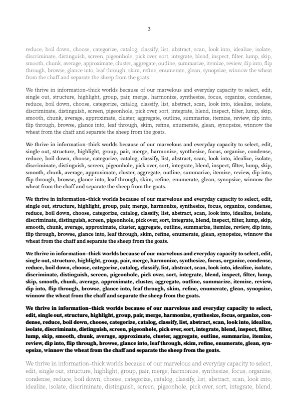
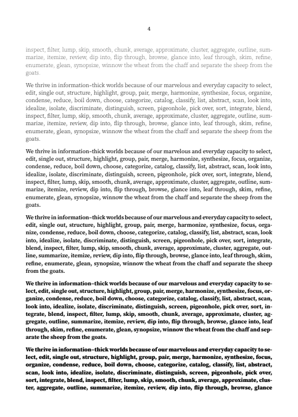
```
\definefont[MyEmojiFont] [seguiemj*overlay]
\definefontsynonym[emoji][seguiemj*overlay]
```

As is using:

```
{\MyEmojiFont\resolvedemoji{woman}}
\emoji{woman}
```

BachoTeX

## bachotex-2017-variablefonts-demo

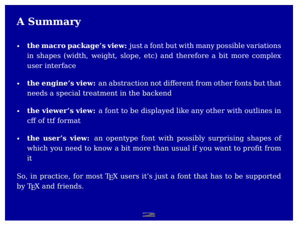




## bachotex-2017-variablefonts

### Variable Fonts

**we're ready for them**

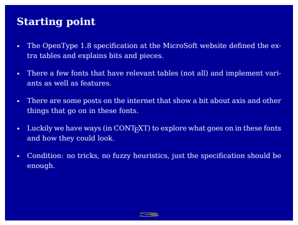
**Hans Hagen**
**BachoTUG 2017**



### A Summary

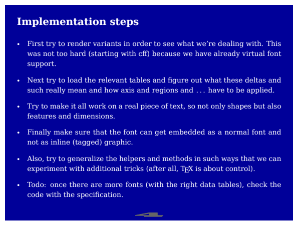
- **the macro package's view:** just a font but with many possible variations in shapes (width, weight, slope, etc) and therefore a bit more complex user interface
- **the engine's view:** an abstraction not different from other fonts but that needs a special treatment in the backend
- **the viewer's view:** a font to be displayed like any other with outlines in cff of ttf format
- **the user's view:** an opentype font with possibly surprising shapes of which you need to know a bit more than usual if you want to profit from it

So, in practice, for most TEX users it's just a font that has to be supported by TEX and friends.



### Starting point

- The OpenType 1.8 specification at the MicroSoft website defined the extra tables and explains bits and pieces.
- There a few fonts that have relevant tables (not all) and implement variants as well as features.
- There are some posts on the internet that show a bit about axis and other things that go on in these fonts.
- Luckily we have ways (in CONTEXT) to explore what goes on in these fonts and how they could look.
- Condition: no tricks, no fuzzy heuristics, just the specification should be enough.



### Implementation steps

- First try to render variants in order to see what we're dealing with. This was not too hard (starting with cff) because we have already virtual font support.
- Next try to load the relevant tables and figure out what these deltas and such really mean and how axis and regions and . . . have to be applied.
- Try to make it all work on a real piece of text, so not only shapes but also features and dimensions.
- Finally make sure that the font can get embedded as a normal font and not as inline (tagged) graphic.
- Also, try to generalize the helpers and methods in such ways that we can experiment with additional tricks (after all, TEX is about control).
- Todo: once there are more fonts (with the right data tables), check the code with the specification.

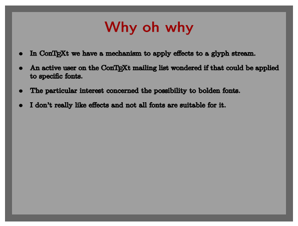
BachoTEX

# bachotex-2018-fonteffects

Modern

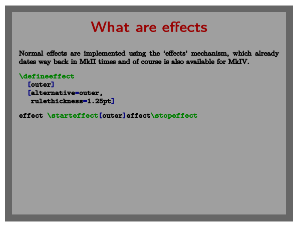
Latin

BachoTEX 2018 — Hans Hagen



## Why oh why

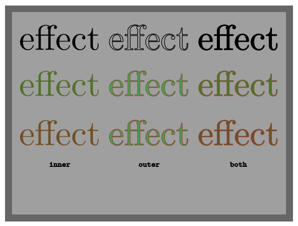
- In ConTEXt we have a mechanism to apply effects to a glyph stream.
- An active user on the ConTEXt mailing list wondered if that could be applied to specific fonts.
- The particular interest concerned the possibility to bolden fonts.
- I don't really like effects and not all fonts are suitable for it.



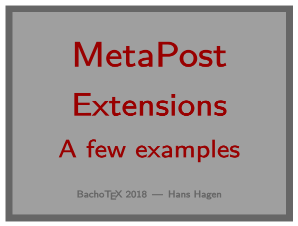
## What are effects

Normal effects are implemented using the 'effects' mechanism, which already dates way back in MkII times and of course is also available for MkIV.

```
\defineeffect
  [outer]
  [alternative=outer,
   rulethickness=1.25pt]

effect \starteffect[outer]effect\stopeffect
```



# bachotex-2018-mp

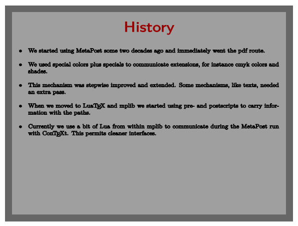


MetaPost

Extensions

A few examples

BachoTEX 2018 — Hans Hagen

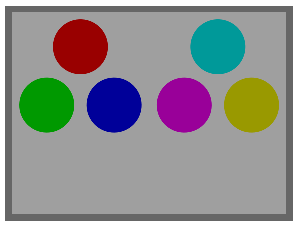


## History

- We started using MetaPost some two decades ago and immediately went the pdf route.
- We used special colors plus specials to communicate extensions, for instance cmyk colors and shades.
- This mechanism was stepwise improved and extended. Some mechanisms, like texts, needed an extra pass.
- When we moved to LuaTEX and mplib we started using pre- and postscripts to carry information with the paths.
- Currently we use a bit of Lua from within mplib to communicate during the MetaPost run with ConTEXt. This permits cleaner interfaces.

## Colors

```
\startMPcode
  draw image (
    draw image (
      fill unitcircle rotated  45 withcolor "red" ;
      fill unitcircle rotated 165 withcolor "green" ;
      fill unitcircle rotated 285 withcolor "blue" ;
    ) shifted (-1.25,0)  ;
    draw image (
      fill unitcircle rotated  45 withcolor "cyan" ;
      fill unitcircle rotated 165 withcolor "magenta" ;
      fill unitcircle rotated 285 withcolor "yellow" ;
    ) shifted ( 1.25,0) ;
  ) xsized TextWidth;
\stopMPcode
```

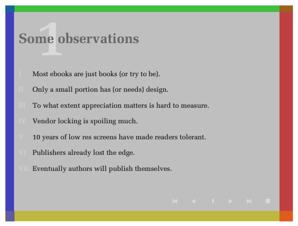


## context-2011-ebook-export

# e-books

old wine in new bottles

ConTEXt Meeting 2011



### 1 Some observations

I Most ebooks are just books (or try to be).

II Only a small portion has (or needs) design.

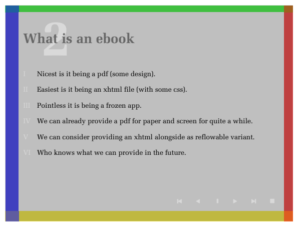
III To what extent appreciation matters is hard to measure.

IV Vendor locking is spoiling much.

V 10 years of low res screens have made readers tolerant.

VI Publishers already lost the edge.

VII Eventually authors will publish themselves.



### 2 What is an ebook

I Nicest is it being a pdf (some design).

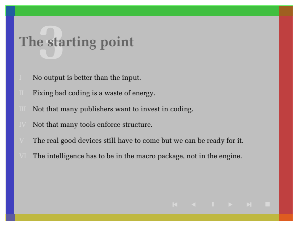
II Easiest is it being an xhtml file (with some css).

III Pointless it is being a frozen app.

IV We can already provide a pdf for paper and screen for quite a while.

V We can consider providing an xhtml alongside as reflowable variant.

VI Who knows what we can provide in the future.



### 3 The starting point

I No output is better than the input.

II Fixing bad coding is a waste of energy.

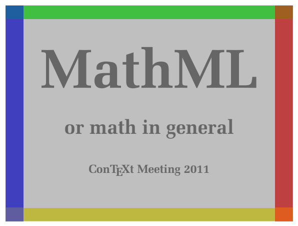
III Not that many publishers want to invest in coding.

IV Not that many tools enforce structure.

V The real good devices still have to come but we can be ready for it.

VI The intelligence has to be in the macro package, not in the engine.

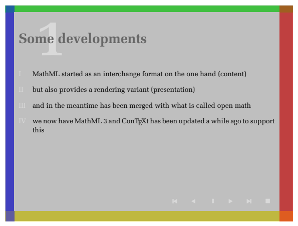
## context-2011-mathml-update



# MathML

or math in general

ConTEXt Meeting 2011



### 1 Some developments

I MathML started as an interchange format on the one hand (content)

II but also provides a rendering variant (presentation)

III and in the meantime has been merged with what is called open math

IV we now have MathML 3 and ConTEXt has been updated a while ago to support this

### 2 Some history

I we supported MathML right from the start

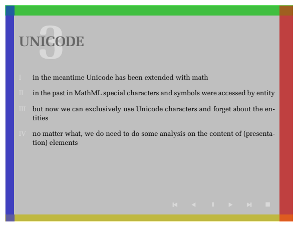
II in MkII quite some data juggling takes place because we need to do some analysis

III the MkII code has been upgraded a few times but is now frozen

IV in MkIV we have rewritten all code using the first version of the new xml parser

V it currently is a mixture of Lua, TEX and MetaPost

VI there will probably be a partial rewrite some day in the future



### 3 UNICODE

I in the meantime Unicode has been extended with math

II in the past in MathML special characters and symbols were accessed by entity

III but now we can exclusively use Unicode characters and forget about the entities

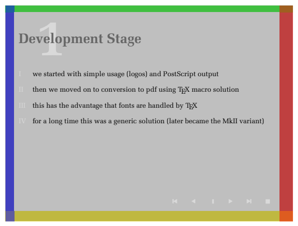
IV no matter what, we do need to do some analysis on the content of (presentation) elements

## context-2011-metapost-how-we-adapt

# MetaPost

how we adapt

ConTEXt Meeting 2011



### 1 Development Stage

I we started with simple usage (logos) and PostScript output

II then we moved on to conversion to pdf using TEX macro solution

III this has the advantage that fonts are handled by TEX

IV for a long time this was a generic solution (later became the MkII variant)

### 2 Development Stage

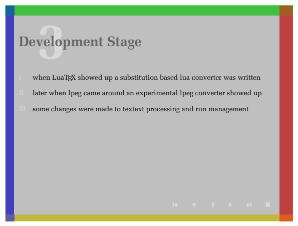
I we added some extensions (transparency, cmyk, etc) and MetaFun showed up

II that extension mechanism uses special colors as signals

III we always collected btex ... etex in order to speed up processing

IV in addition we added textext and similar features

V communication between MetaFun and ConTEXt became more advanced over time



### 3 Development Stage

I when LuaTEX showed up a substitution based lua converter was written

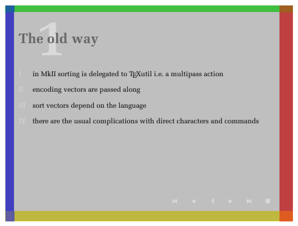
II later when lpeg came around an experimental lpeg converter showed up

III some changes were made to textext processing and run management

# context-2011-sorting-registers

# Sorting registers

ConTEXt Meeting 2011



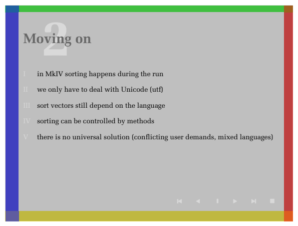
## 1 The old way

I in MkII sorting is delegated to TEXutil i.e. a multipass action

II encoding vectors are passed along

III sort vectors depend on the language

IV there are the usual complications with direct characters and commands



## 2 Moving on

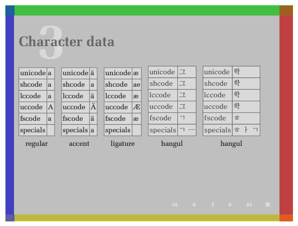
I in MkIV sorting happens during the run

II we only have to deal with Unicode (utf)

III sort vectors still depend on the language

IV sorting can be controlled by methods

V there is no universal solution (conflicting user demands, mixed languages)



## 3 Character data

| | |
|---|---|
| unicode | a |
| shcode | a |
| lccode | a |
| uccode | A |
| fscode | a |
| specials | |

regular

| | |
|---|---|
| unicode | ä |
| shcode | a |
| lccode | ä |
| uccode | Ä |
| fscode | ä |
| specials | a |

accent

| | |
|---|---|
| unicode | æ |
| shcode | ae |
| lccode | æ |
| uccode | Æ |
| fscode | æ |
| specials | |

ligature

| | |
|---|---|
| unicode | 그 |
| shcode | 그 |
| lccode | 그 |
| uccode | 그 |
| fscode | ㄱ |
| specials | ㄱ ㅡ |

hangul

| | |
|---|---|
| unicode | 학 |
| shcode | 학 |
| lccode | 학 |
| uccode | 학 |
| fscode | ㅎ |
| specials | ㅎ ㅏ ㄱ |

hangul

# context-2012-after-the-cleanup

Breskens 2012 After the cleanup

### The update

- The move to MKIV is more than supporting an engine.
- It is a complete rewrite (pruning, extending, cleaning).
- Although somewhat crippled by the fact that we want to remain compatible.
- But sometimes we sacrifice compatibility by getting rid of old stuff.

### The current state

- A lot of work, more than I had thought, so it takes longer.
- Most TEX code is done (some structure and column code left).
- New namespaces and helpers mostly done, but will be checked for constency.

Breskens 2012 After the cleanup

### What is there todo

- Some code might become generalized (also depends on others).
- Layer and positioning code might get a more extensive LUA and XML interface.
- Structure related code will support setups (some already in place).
- New page builder variants will be explored (anyway more column support and floats).
- Math domains cq. dictionaries (basics already in place, just needs time).
- Math list optimization (pet project).
- Generate dependecy trees (easier now) and more consistent code loading order.
- All error messages needs checking (some gone, some not yet translated).
- Update all xml definitions (work in progress, also relates to wiki).
- Optimize positioning system (a bit more powerful now, but also more resources).
- More support for css like styling (makes it easier to share code).
- Modules (especially those for tracing) need to be normalized.
- Some styles (mostly private presentation styles) needs to be fixed.
- Pick up the 'lean and mean' CONTEXT variant project.
- Now that we have more code isolated, we can define an api.
- Some manuals need to be updated (most still applies).

Breskens 2012 After the cleanup

### What I have to keep in mind

- What is handy for me is not always handy for all users.

### But nevertheless there will be new things

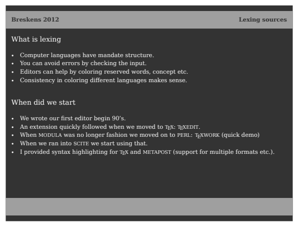
- Elements of our processing framework will show up in the distribution.
- It's just more convenient to have one installation for related things.
- This is also why support for databases has been added recently.
- Running (blocking) TEX jobs needs special treatment (ticket management).
- It makes sense to use the well developed TDS infrastructure.

Breskens 2012 After the cleanup

### Keep an eye on updates

- Rewriting the code base leads to bugs but these are often resolved quickly (indeed by Wolfgang).
- Following the mailing list helps and nowadays the wiki is adapted close to realtime (coordinated by Sietse).
- Changes in standards and related tools are supported and followed by those who depend on them (ask Peter).
- Sometimes users have demands and these end up as extensions to existing mechanisms (Aditya's elastic modules).
- Issues with platforms are often quickly dealt with (if Luigi doesn't know it . . . ).
- And of course I add new things driven by projects, challenges (and an occasional stack of new CD's).
- New releases (and betas) are checked against a growing set of test files (Lukas mails a report after each update).

# context-2012-lexing-sources



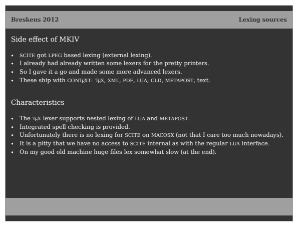
Breskens 2012 Lexing sources

### What is lexing

- Computer languages have mandate structure.
- You can avoid errors by checking the input.
- Editors can help by coloring reserved words, concept etc.
- Consistency in coloring different languages makes sense.

### When did we start

- We wrote our first editor begin 90's.
- An extension quickly followed when we moved to TEX: TEXEDIT.
- When MODULA was no longer fashion we moved on to PERL: TEXWORK (quick demo)
- When we ran into SCITE we start using that.
- I provided syntax highlighting for TEX and METAPOST (support for multiple formats etc.).



Breskens 2012 Lexing sources

### Side effect of MKIV

- SCITE got LPEG based lexing (external lexing).
- I already had already written some lexers for the pretty printers.
- So I gave it a go and made some more advanced lexers.
- These ship with CONTEXT: TEX, XML, PDF, LUA, CLD, METAPOST, text.

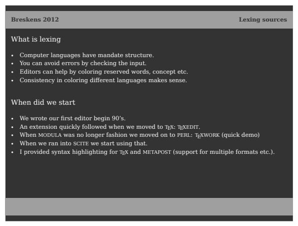
### Characteristics

- The TEX lexer supports nested lexing of LUA and METAPOST.
- Integrated spell checking is provided.
- Unfortunately there is no lexing for SCITE on MACOSX (not that I care too much nowadays).
- It is a pitty that we have no access to SCITE internal as with the regular LUA interface.
- On my good old machine huge files lex somewhat slow (at the end).

Breskens 2012 Lexing sources

### The future

- I will improve the current lexers.
- An SQL lexer will be added at some point
- I might make an HTML/CSS variant that supports nexted LMX.



Breskens 2012 Lexing sources

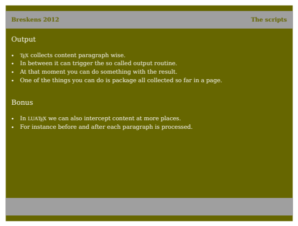
What is lexing

- Computer languages have mandate structure.
- You can avoid errors by checking the input.
- Editors can help by coloring reserved words, concept etc.
- Consistency in coloring different languages makes sense.

When did we start

- We wrote our first editor begin 90's.
- An extension quickly followed when we moved to TEX: TEXEDIT.
- When MODULA was no longer fashion we moved on to PERL: TEXWORK (quick demo)
- When we ran into SCITE we start using that.
- I provided syntax highlighting for TEX and METAPOST (support for multiple formats etc.).

# context-2012-mixed-columns

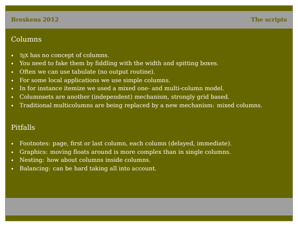

Breskens 2012 The scripts

Output

- TEX collects content paragraph wise.
- In between it can trigger the so called output routine.
- At that moment you can do something with the result.
- One of the things you can do is package all collected so far in a page.

Bonus

- In LUATEX we can also intercept content at more places.
- For instance before and after each paragraph is processed.

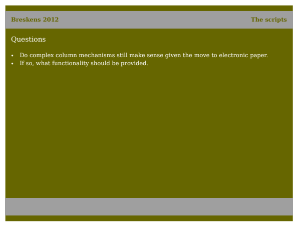

Breskens 2012 The scripts

Columns

- TEX has no concept of columns.
- You need to fake them by fiddling with the width and spitting boxes.
- Often we can use tabulate (no output routine).
- For some local applications we use simple columns.
- In for instance itemize we used a mixed one- and multi-column model.
- Columnsets are another (independent) mechanism, strongly grid based.
- Traditional multicolumns are being replaced by a new mechanism: mixed columns.

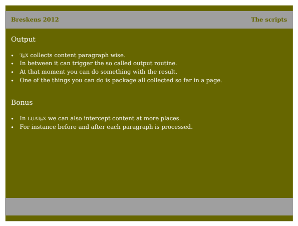
Pitfalls

- Footnotes: page, first or last column, each column (delayed, immediate).
- Graphics: moving floats around is more complex than in single columns.
- Nesting: how about columns inside columns.
- Balancing: can be hard taking all into account.


Breskens 2012 The scripts

Questions

- Do complex column mechanisms still make sense given the move to electronic paper.
- If so, what functionality should be provided.


Breskens 2012 The scripts

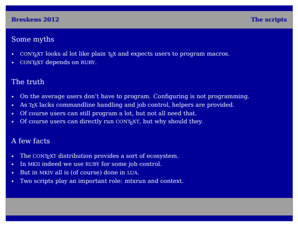
Output

- TEX collects content paragraph wise.
- In between it can trigger the so called output routine.
- At that moment you can do something with the result.
- One of the things you can do is package all collected so far in a page.

Bonus

- In LUATEX we can also intercept content at more places.
- For instance before and after each paragraph is processed.

# context-2012-the-script


Breskens 2012 The scripts

Some myths

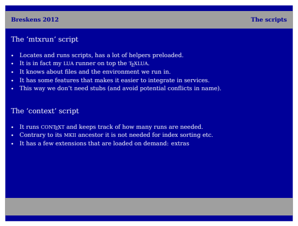
- CONTEXT looks al lot like plain TEX and expects users to program macros.
- CONTEXT depends on RUBY.

The truth

- On the average users don't have to program. Configuring is not programming.
- As TEX lacks commandline handling and job control, helpers are provided.
- Of course users can still program a lot, but not all need that.
- Of course users can directly run CONTEXT, but why should they.

A few facts

- The CONTEXT distribution provides a sort of ecosystem.
- In MKII indeed we use RUBY for some job control.
- But in MKIV all is (of course) done in LUA.
- Two scripts play an important role: mtxrun and context.

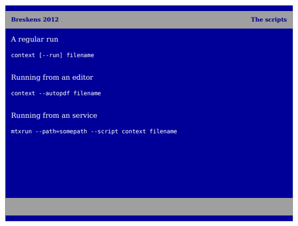

Breskens 2012 The scripts

The 'mtxrun' script

- Locates and runs scripts, has a lot of helpers preloaded.
- It is in fact my LUA runner on top the TEXLUA.
- It knows about files and the environment we run in.
- It has some features that makes it easier to integrate in services.
- This way we don't need stubs (and avoid potential conflicts in name).

The 'context' script

- It runs CONTEXT and keeps track of how many runs are needed.
- Contrary to its MKII ancestor it is not needed for index sorting etc.
- It has a few extensions that are loaded on demand: extras


Breskens 2012 The scripts

A regular run

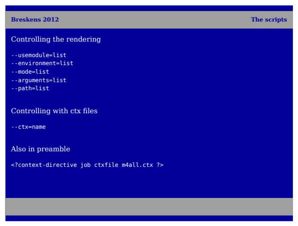
```
context [--run] filename
```

Running from an editor

```
context --autopdf filename
```

Running from an service

```
mtxrun --path=somepath --script context filename
```


Breskens 2012 The scripts

Controlling the rendering

```
--usemodule=list
--environment=list
--mode=list
--arguments=list
--path=list
```

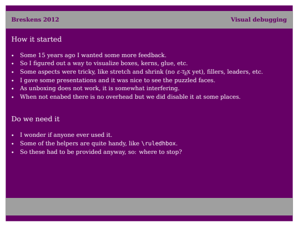
Controlling with ctx files

```
--ctx=name
```

Also in preamble

```
<?context-directive job ctxfile m4all.ctx ?>
```

# context-2012-visual-debugging

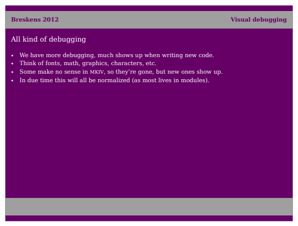

Breskens 2012 Visual debugging

How it started

- Some 15 years ago I wanted some more feedback.
- So I figured out a way to visualize boxes, kerns, glue, etc.
- Some aspects were tricky, like stretch and shrink (no ε-TEX yet), fillers, leaders, etc.
- I gave some presentations and it was nice to see the puzzled faces.
- As unboxing does not work, it is somewhat interfering.
- When not enabed there is no overhead but we did disable it at some places.

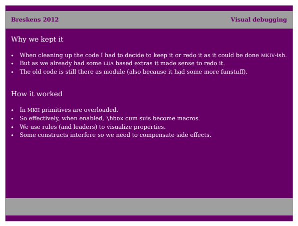
Do we need it

- I wonder if anyone ever used it.
- Some of the helpers are quite handy, like `\ruledhbox`.
- So these had to be provided anyway, so: where to stop?


Breskens 2012 Visual debugging

All kind of debugging

- We have more debugging, much shows up when writing new code.
- Think of fonts, math, graphics, characters, etc.
- Some make no sense in MKIV, so they're gone, but new ones show up.
- In due time this will all be normalized (as most lives in modules).

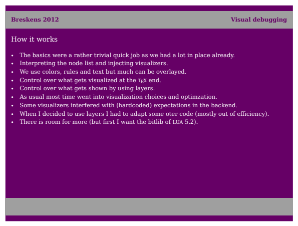

Breskens 2012 Visual debugging

Why we kept it

- When cleaning up the code I had to decide to keep it or redo it as it could be done MKIV-ish.
- But as we already had some LUA based extras it made sense to redo it.
- The old code is still there as module (also because it had some more funstuff).

How it worked

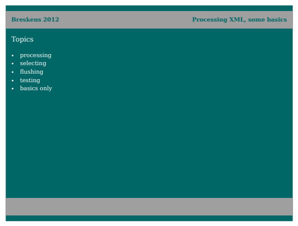
- In MKII primitives are overloaded.
- So effectively, when enabled, `\hbox` cum suis become macros.
- We use rules (and leaders) to visualize properties.
- Some constructs interfere so we need to compensate side effects.


Breskens 2012 Visual debugging

How it works

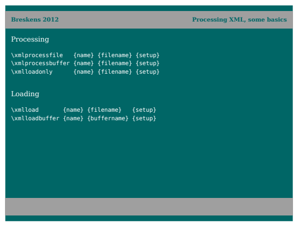
- The basics were a rather trivial quick job as we had a lot in place already.
- Interpreting the node list and injecting visualizers.
- We use colors, rules and text but much can be overlayed.
- Control over what gets visualized at the TEX end.
- Control over what gets shown by using layers.
- As usual most time went into visualization choices and optimzation.
- Some visualizers interfered with (hardcoded) expectations in the backend.
- When I decided to use layers I had to adapt some oter code (mostly out of efficiency).
- There is room for more (but first I want the bitlib of LUA 5.2).

# context-2012-xml-news


Breskens 2012 Processing XML, some basics

Topics

- processing
- selecting
- flushing
- testing
- basics only

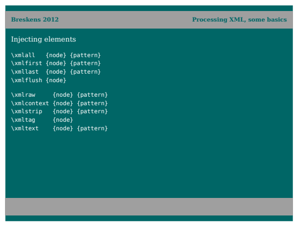

Breskens 2012 Processing XML, some basics

Processing

```
\xmlprocessfile   {name} {filename} {setup}
\xmlprocessbuffer {name} {filename} {setup}
\xmlloadonly      {name} {filename} {setup}
```

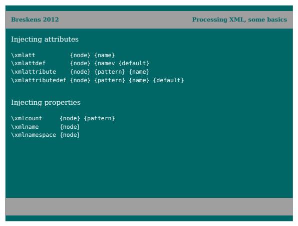
Loading

```
\xmlload       {name} {filename}   {setup}
\xmlloadbuffer {name} {buffername} {setup}
```


Breskens 2012 Processing XML, some basics

Injecting elements

```
\xmlall   {node} {pattern}
\xmlfirst {node} {pattern}
\xmllast  {node} {pattern}
\xmlflush {node}

\xmlraw     {node} {pattern}
\xmlcontext {node} {pattern}
\xmlstrip   {node} {pattern}
\xmltag     {node}
\xmltext    {node} {pattern}
```


Breskens 2012 Processing XML, some basics

Injecting attributes

```
\xmlatt          {node} {name}
\xmlattdef       {node} {namev {default}
\xmlattribute    {node} {pattern} {name}
\xmlattributedef {node} {pattern} {name} {default}
```

Injecting properties

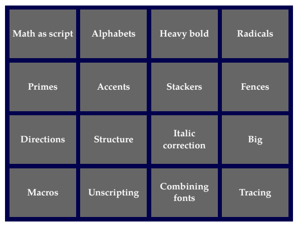
```
\xmlcount     {node} {pattern}
\xmlname      {node}
\xmlnamespace {node}
```

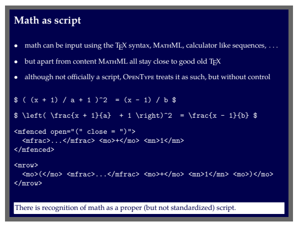
# context-2013-math

Math:

progress or standing still

Hans Hagen
ConTEXt Meeting
September 2013

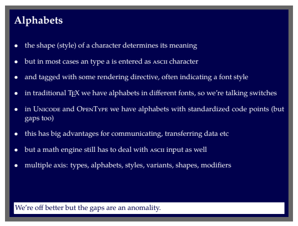

Math as script
Alphabets
Heavy bold
Radicals
Primes
Accents
Stackers
Fences
Directions
Structure
Italic correction
Big
Macros
Unscripting
Combining fonts
Tracing


Math as script

- math can be input using the TEX syntax, MATHML, calculator like sequences, . . .
- but apart from content MATHML all stay close to good old TEX
- although not officially a script, OPENTYPE treats it as such, but without control

```
$ ( (x + 1) / a + 1 )^2  = (x - 1) / b $

$ \left( \frac{x + 1}{a}  + 1 \right)^2  = \frac{x - 1}{b} $

<mfenced open="(" close = ")">
  <mfrac>...</mfrac> <mo>+</mo> <mn>1</mn>
</mfenced>

<mrow>
  <mo>(</mo> <mfrac>...</mfrac> <mo>+</mo> <mn>1</mn> <mo>)</mo>
</mrow>
```

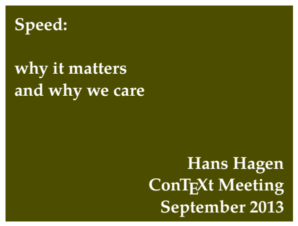
There is recognition of math as a proper (but not standardized) script.


Alphabets

- the shape (style) of a character determines its meaning
- but in most cases an type a is entered as ASCII character
- and tagged with some rendering directive, often indicating a font style
- in traditional TEX we have alphabets in different fonts, so we're talking switches
- in UNICODE and OPENTYPE we have alphabets with standardized code points (but gaps too)
- this has big advantages for communicating, transferring data etc
- but a math engine still has to deal with ASCII input as well
- multiple axis: types, alphabets, styles, variants, shapes, modifiers

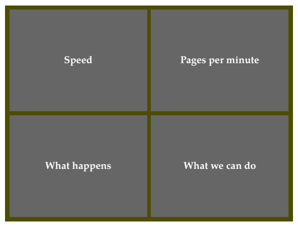
We're off better but the gaps are an anomality.

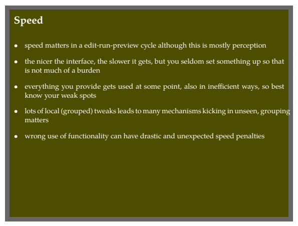
# context-2013-speed


Speed:

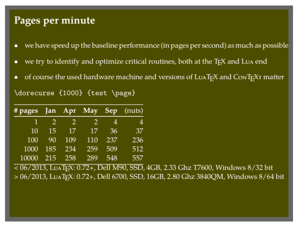
why it matters
and why we care

Hans Hagen
ConTEXt Meeting
September 2013


Speed
Pages per minute
What happens
What we can do


Speed

- speed matters in a edit-run-preview cycle although this is mostly perception
- the nicer the interface, the slower it gets, but you seldom set something up so that is not much of a burden
- everything you provide gets used at some point, also in inefficient ways, so best know your weak spots
- lots of local (grouped) tweaks leads to many mechanisms kicking in unseen, grouping matters
- wrong use of functionality can have drastic and unexpected speed penalties


Pages per minute

- we have speed up the baseline performance (in pages per second) as much as possible
- we try to identify and optimize critical routines, both at the TEX and LUA end
- of course the used hardware machine and versions of LUATEX and CONTEXT matter

```
\dorecurse {1000} {test \page}
```

| # pages | Jan | Apr | May | Sep | (nuts) |
|---|---|---|---|---|---|
| 1 | 2 | 2 | 2 | 4 | 4 |
| 10 | 15 | 17 | 17 | 36 | 37 |
| 100 | 90 | 109 | 110 | 237 | 236 |
| 1000 | 185 | 234 | 259 | 509 | 512 |
| 10000 | 215 | 258 | 289 | 548 | 557 |

< 06/2013, LUATEX: 0.72+, Dell M90, SSD, 4GB, 2.33 Ghz T7600, Windows 8/32 bit
> 06/2013, LUATEX: 0.72+, Dell 6700, SSD, 16GB, 2.80 Ghz 3840QM, Windows 8/64 bit

## context-2015-status

ConTEXt

2015

**fonts**

new loader

stable interfaces

related mechanisms can be cleaned up

extensions possible

**hyphenation**

experimental

normalization considered

**spacing**

functional stable

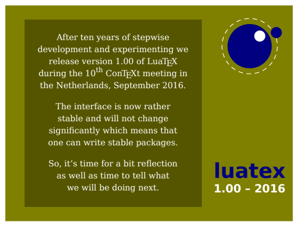
maybe some cleanup needed

## context-2016-luatex

**LuaTEX**

**Version 1.00**

ConTeXt meeting — September 2016


After ten years of stepwise development and experimenting we release version 1.00 of LuaTEX during the 10th ConTEXt meeting in the Netherlands, September 2016.

The interface is now rather stable and will not change significantly which means that one can write stable packages.

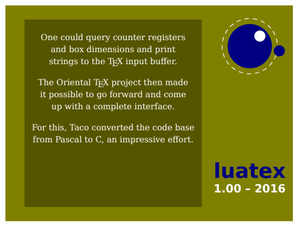
So, it's time for a bit reflection as well as time to tell what we will be doing next.

**luatex**
**1.00 - 2016**

Around 2005, after we talked a bit about it, Hartmut added the Lua scripting language to pdfTEX as an experiment.

This add-on was inspired by the Lua extension to the Scite editor that I (still) use.

**luatex**
**1.00 - 2016**

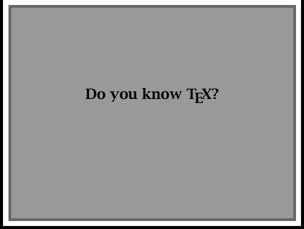

One could query counter registers and box dimensions and print strings to the TEX input buffer.

The Oriental TEX project then made it possible to go forward and come up with a complete interface.

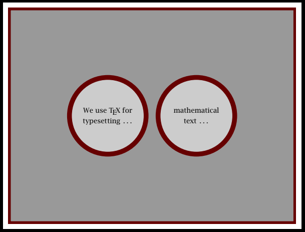
For this, Taco converted the code base from Pascal to C, an impressive effort.

**luatex**
**1.00 - 2016**

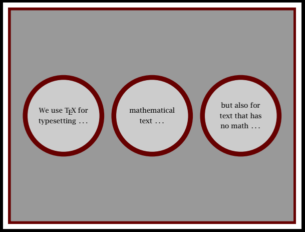
## present-balls-001


Do you know TEX?

We use TEX for typesetting …


We use TEX for typesetting …

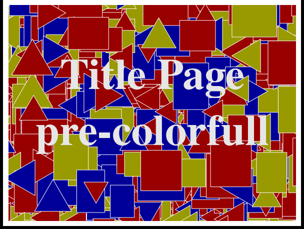
mathematical text …


We use TEX for typesetting …

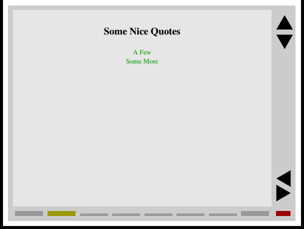
mathematical text …

but also for text that has no math …

## present-colorful-001

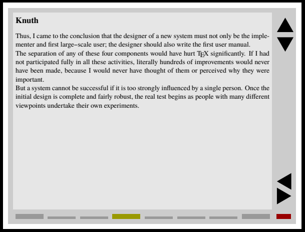

Title Page

pre-colorfull


Some Nice Quotes

A Few
Some More

A Few


Knuth

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

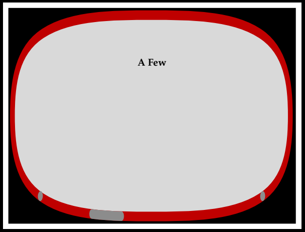
## present-funny-001

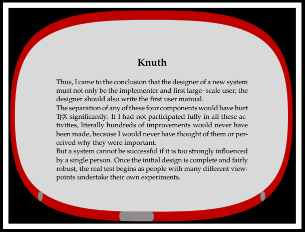
Title Page

pre-funny

Some Nice Quotes

A Few
Knuth
Tufte
Some More
Zapf
Bryson

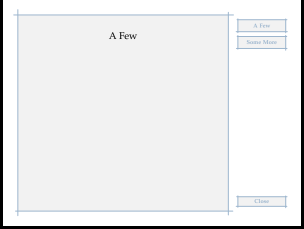

A Few

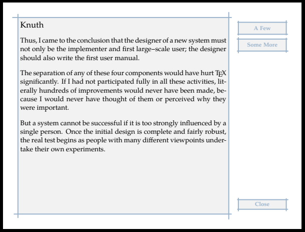

Knuth

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

# Examples

## present-fuzzy-001



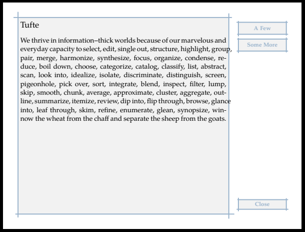


Knuth

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

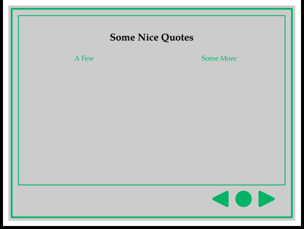
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

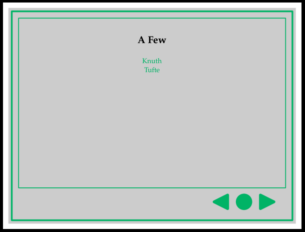


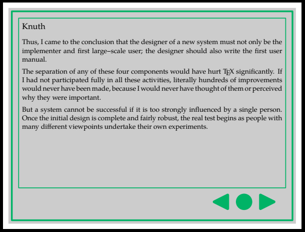
Tufte

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.

## present-green-001







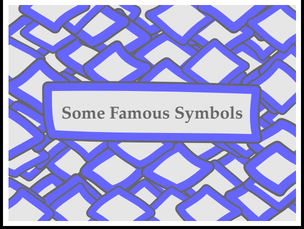
Knuth

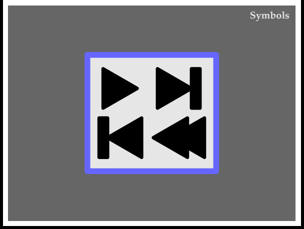
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

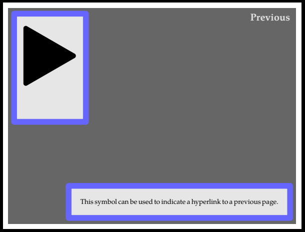
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

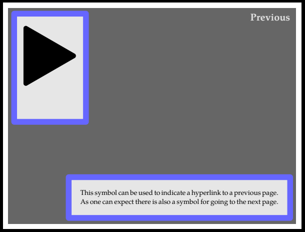
## present-grow-001







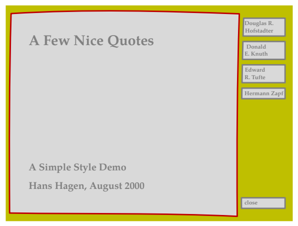
This symbol can be used to indicate a hyperlink to a previous page.



This symbol can be used to indicate a hyperlink to a previous page.
As one can expect there is also a symbol for going to the next page.

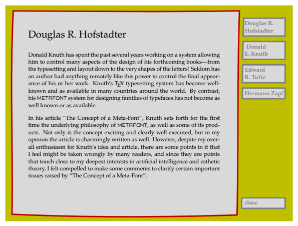
# Examples

## present-organic-001


A Few Nice Quotes

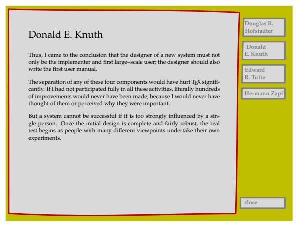
A Simple Style Demo

Hans Hagen, August 2000


Douglas R. Hofstadter

Donald Knuth has spent the past several years working on a system allowing him to control many aspects of the design of his forthcoming books—from the typesetting and layout down to the very shapes of the letters! Seldom has an author had anything remotely like this power to control the final appearance of his or her work. Knuth's TEX typesetting system has become well-known and as available in many countries around the world. By contrast, his METAFONT system for designing families of typefaces has not become as well known or as available.

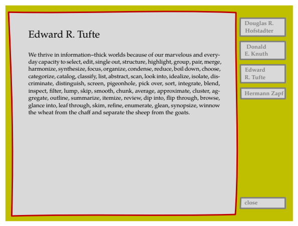
In his article "The Concept of a Meta-Font", Knuth sets forth for the first time the underlying philosophy of METAFONT, as well as some of its products. Not only is the concept exciting and clearly well executed, but in my opinion the article is charmingly written as well. However, despite my overall enthusiasm for Knuth's idea and article, there are some points in it that I feel might be taken wrongly by many readers, and since they are points that touch close to my deepest interests in artificial intelligence and esthetic theory, I felt compelled to make some comments to clarify certain important issues raised by "The Concept of a Meta-Font".


Donald E. Knuth

Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

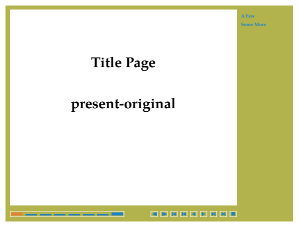
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

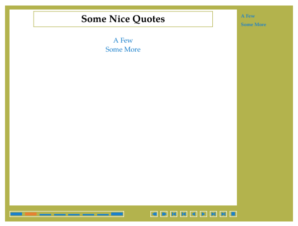

Edward R. Tufte

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.

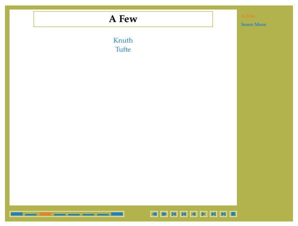
## present-original-001


Title Page

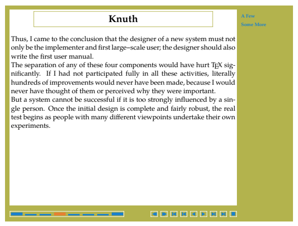
present-original


Some Nice Quotes

A Few
Some More

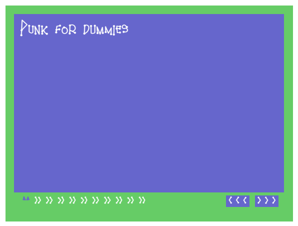

A Few

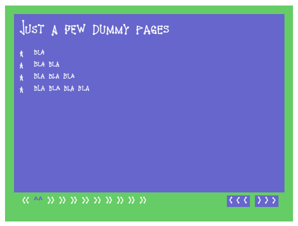
Knuth
Tufte


Knuth

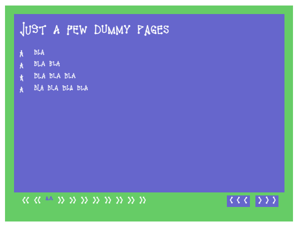
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

## present-punk-001

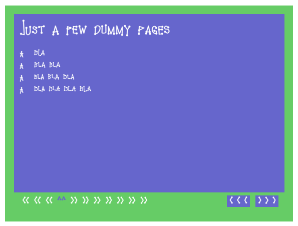

Punk for Dummies


Just a few dummy pages

- bla
- bla bla
- bla bla bla
- bla bla bla bla


Just a few dummy pages

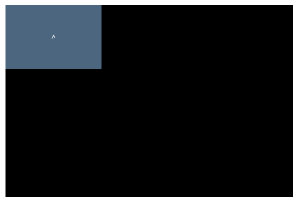
- bla
- bla bla
- bla bla bla
- bla bla bla bla

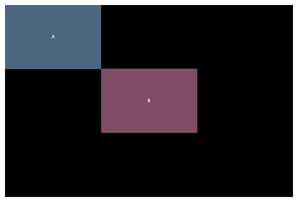

Just a few dummy pages

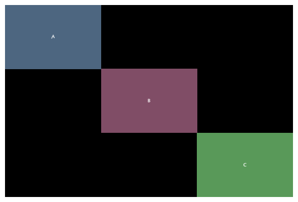
- bla
- bla bla
- bla bla bla
- bla bla bla bla

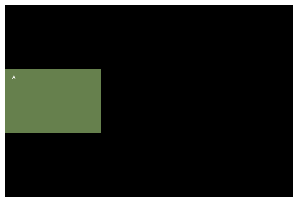
Examples

## present-random-001

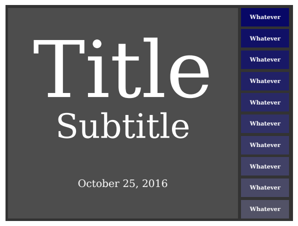








## present-shaded-001

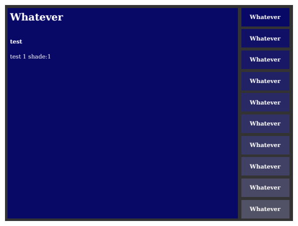


Title

Subtitle

October 25, 2016

Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever

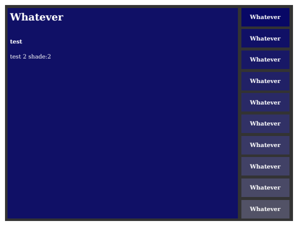


**Whatever**

**test**

test 1 shade:1

Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever

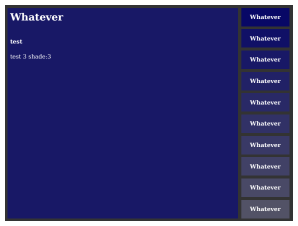


**Whatever**

**test**

test 2 shade:2

Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever



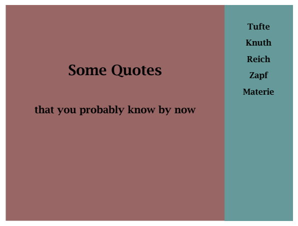
**Whatever**

**test**

test 3 shade:3

Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever
Whatever

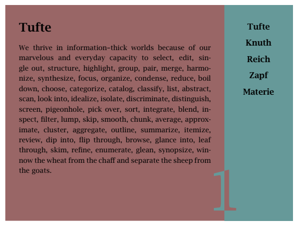
## present-split-001



Some Quotes

that you probably know by now

Tufte
Knuth
Reich
Zapf
Materie

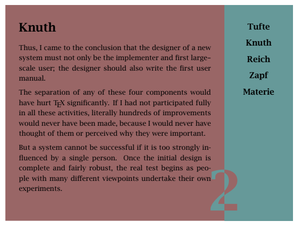


**Tufte**

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.

1

Tufte
Knuth
Reich
Zapf
Materie



**Knuth**

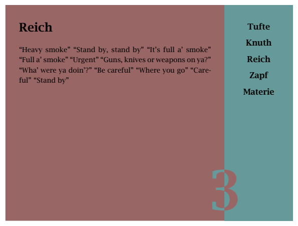
Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.

The separation of any of these four components would have hurt TeX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.

But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

2

Tufte
Knuth
Reich
Zapf
Materie



**Reich**

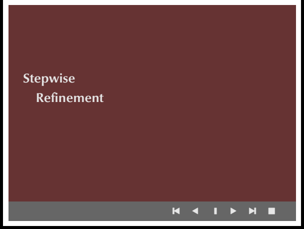
“Heavy smoke” “Stand by, stand by” “It’s full a’ smoke” “Full a’ smoke” “Urgent” “Guns, knives or weapons on ya?” “Wha’ were ya doin’?” “Be careful” “Where you go” “Careful” “Stand by”

3

Tufte
Knuth
Reich
Zapf
Materie

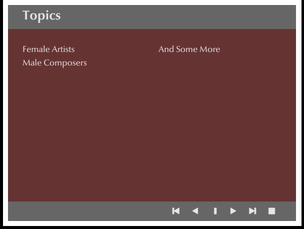
Examples

## present-stepper-001



Stepwise

Refinement

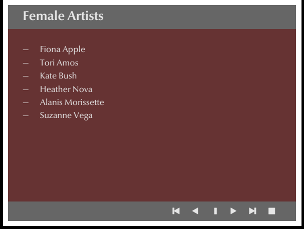


Topics

Female Artists

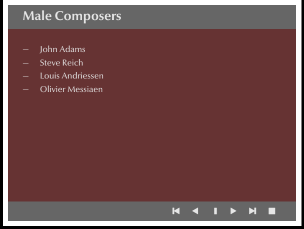
Male Composers

And Some More



Female Artists

- Fiona Apple
- Tori Amos
- Kate Bush
- Heather Nova
- Alanis Morissette
- Suzanne Vega



Male Composers

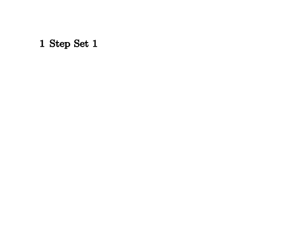
- John Adams
- Steve Reich
- Louis Andriessen
- Olivier Messiaen

## present-steps-001

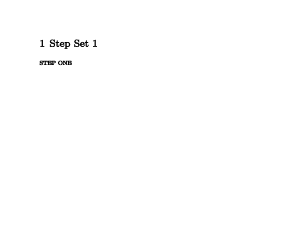
Contents

1 Step Set 1 2

2 Step Set 2 7

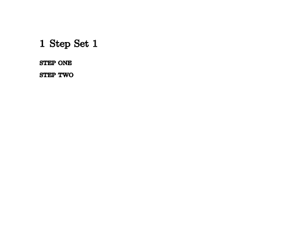


1 Step Set 1



1 Step Set 1

STEP ONE



1 Step Set 1

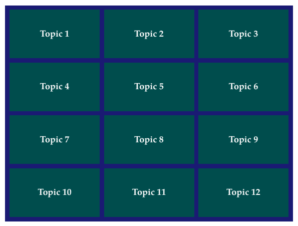
STEP ONE

STEP TWO

## present-tiles-001

Whatever We
Want Here

Whatever We
Want There

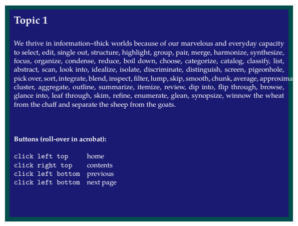


Topic 1 Topic 2 Topic 3

Topic 4 Topic 5 Topic 6

Topic 7 Topic 8 Topic 9

Topic 10 Topic 11 Topic 12

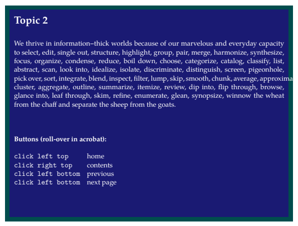


Topic 1

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, approxima 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.

**Buttons (roll-over in acrobat):**

```
click left top     home
click right top    contents
click left bottom  previous
click left bottom  next page
```



Topic 2

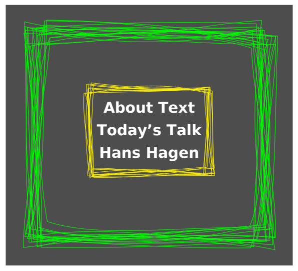
We thrive in information–thick worlds because of our marvelous and everyday capacity to select, edit, single out, structure, highlight, group, pair, merge, harmonize, synthesize, focus, organize, condense, reduce, boil down, choose, categorize, catalog, classify, list, abstract, scan, look into, idealize, isolate, discriminate, distinguish, screen, pigeonhole, pick over, sort, integrate, blend, inspect, filter, lump, skip, smooth, chunk, average, approxima 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.

**Buttons (roll-over in acrobat):**

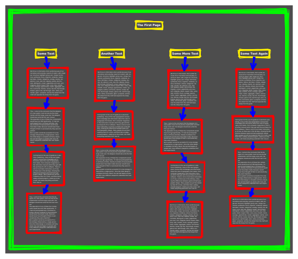
```
click left top     home
click right top    contents
click left bottom  previous
click left bottom  next page
```

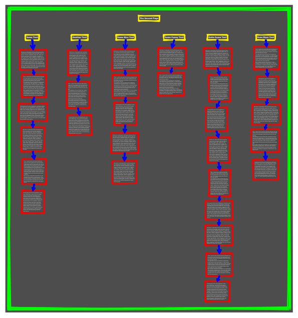
Examples

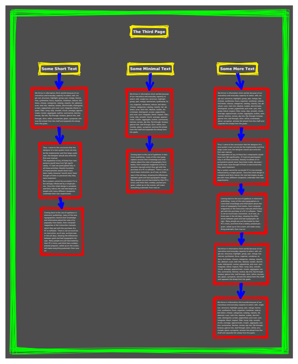
## present-weird-001



About Text
Today’s Talk
Hans Hagen

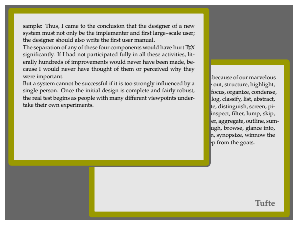






## present-windows-001

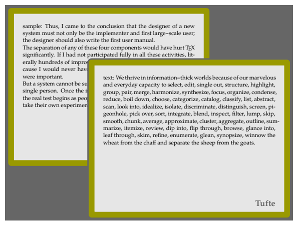
Quotes, Quotes
and more quotes



sample: Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, literally hundreds of improvements would never have been made, because I would never have thought of them or perceived why they were important.
But a system cannot be successful if it is too strongly influenced by a single person. Once the initial design is complete and fairly robust, the real test begins as people with many different viewpoints undertake their own experiments.

because of our marvelous
e out, structure, highlight,
focus, organize, condense,
log, classify, list, abstract,
te, distinguish, screen, pi-
inspect, filter, lump, skip,
er, aggregate, outline, sum-
ugh, browse, glance into,
n, synopsize, winnow the
p from the goats.

Tufte



sample: Thus, I came to the conclusion that the designer of a new system must not only be the implementer and first large–scale user; the designer should also write the first user manual.
The separation of any of these four components would have hurt TEX significantly. If I had not participated fully in all these activities, lit-
erally hundreds of impro
cause I would never hav
were important.
But a system cannot be su
single person. Once the i
the real test begins as peo
take their own experimen

text: 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.

Tufte

sample: Zaagh, of Steek–zaagh. Domme–kraght. Mokers. Yzere Wiggen. Brandt–yzer. Hout–haak. Spaander–haak. Een Klaas Jacobzen. Tange. Nagel–hamer. Wigge. Oor–houten. Kluften. Hellen. Steven–haken. Hevels en Klein–touwen. Een Koe–voet. Schot–bouten. Yzere ram. Avegaar. Een houte Ram. Een groote Wigge daar toe. Teer–ketel. Slyp–steen. Een Mal. Rye. Hout–bok. Schraagh. Vlotten. Een Slee. Dwars–slee. Kaap–stander. Bytels. Klavaats–hamer. Rabat–yzer. Klavaats–yzer. Spyker–yzer. Werk–bytel. Duim–stok. Schraper. Een Moker. Spyker–hamertje. Een Roffel. Gerf–schaaf. Odief. Ploegen. Handt–zaagh. Klamp–spykers–boor, tien duims Boor. Dissel. Byl.

“It’s full a’ smoke” “Full
pons on ya?” “Wha’ were
Careful” “Stand by”

Examples