TEX without \TeX

or:

The hidden beauty of LuaTEX

Patrick Gundlach
gundlach@speedata.de

ConTEXt meeting 2010
Brejlov

Why?

- 100% automatic, unattended XML to PDF workflow
- Big product catlogs (> 1000 pages), complex data sheets, on demand typesetting
- High-speed processing
- Not particularly demanding details

Why use TEX?

- I can focus on implementing the “logic”
- TEX is very fast and reliable
- Excellent typography
- Unicode, OpenType, PDF reading / writing, ... for “free”

Why (not) use TEX?

- I can focus on implementing the “logic”
- TEX is very fast and reliable
- Excellent typography
- Unicode, OpenType, PDF reading / writing, ... for “free”
- But: TEX is hard to program (expansion, catcodes, funny things like \futurelet and \afterassignment)

The question

How can I take the advantages of (Lua)TEX without the need to program in TEX?

The answer is obvious

- With Lua TEX, I can program in Lua
- Very rich API
- Everything can be done, except for parsing TEX-input

The layout definition language

- A programming language that also the customer can use
- A Layer between TEX and “the user”
- Programs are written in XML
  (actually sort of fun if you have a good XML editor)
- Few building blocks: text, tables, images and a few other things

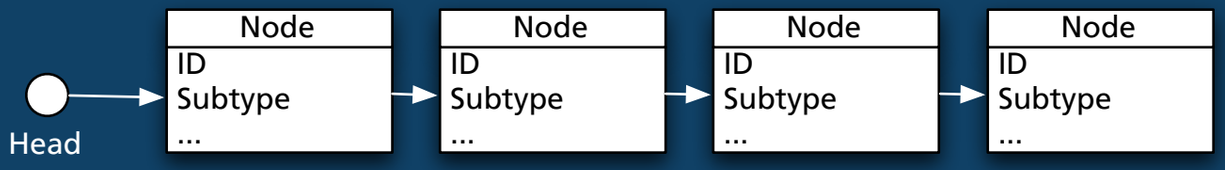
Nodelists

- To generate output from the Lua side, you need to create nodelists
- All texts, images, rules, tables are represented by nodelists
- All you need to do is to create the correct nodelists and send them to the output stream:

```
node.write(my_nodelist)
```

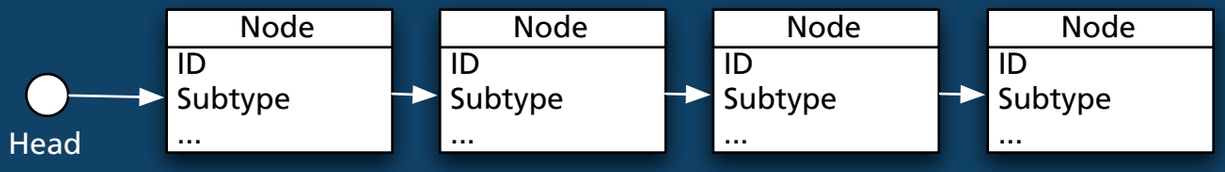
Nodelists

- Data structure: list of records (nodes)



Nodelists

- Data structure: list of records (nodes)



- Example: glyph node (the letter “H”)

Node	
ID	37
subtype	0
next	...
char	72
font	0
lang	1
...	...

Glyph node “H”

```
n = node.new(37)
n.char = 72
n.font = 0
n.lang = 1
```

How to make a text box

Create font instance

Create language instance

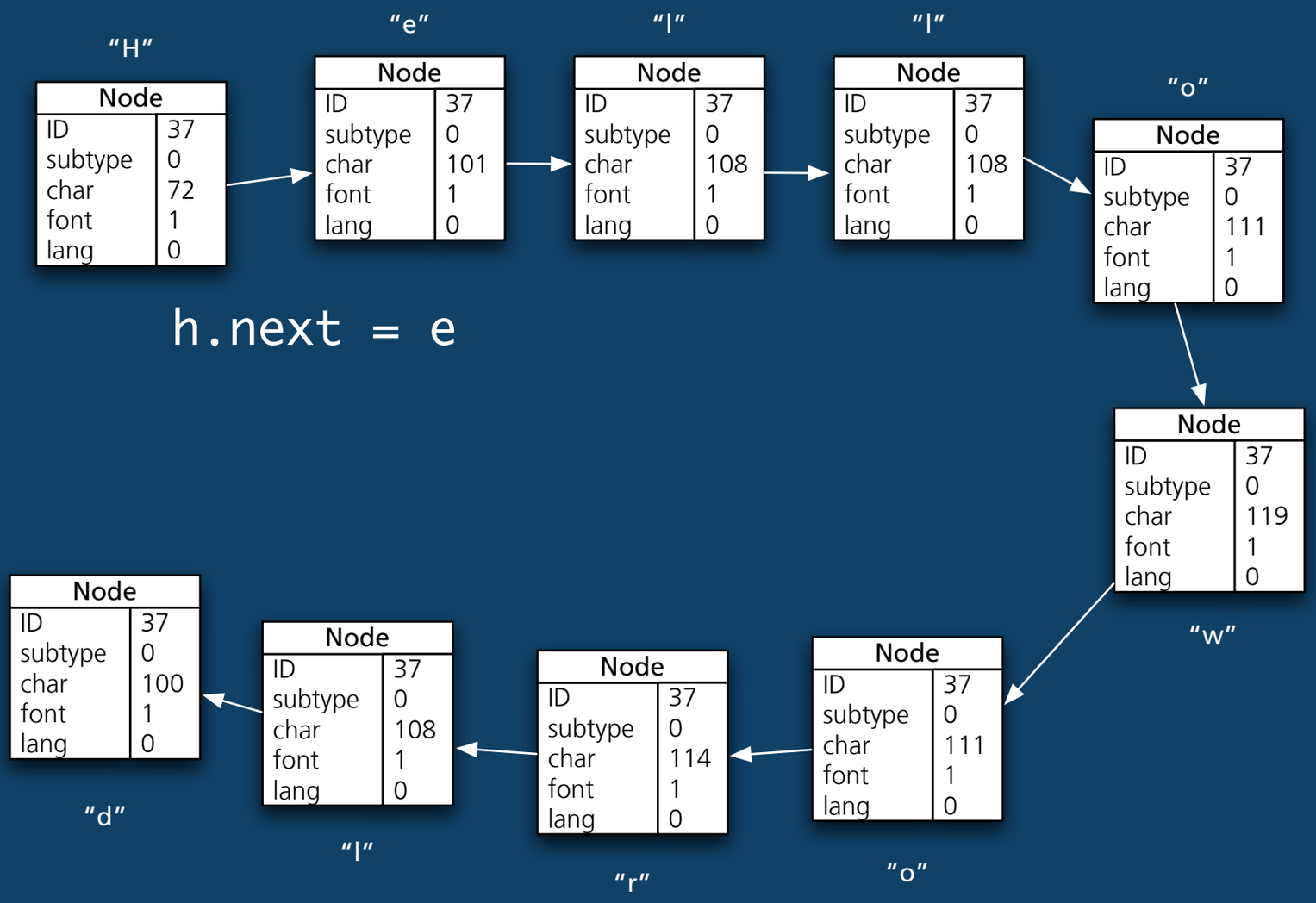
Create nodelist

Hyphenation / ligaturing / kerning

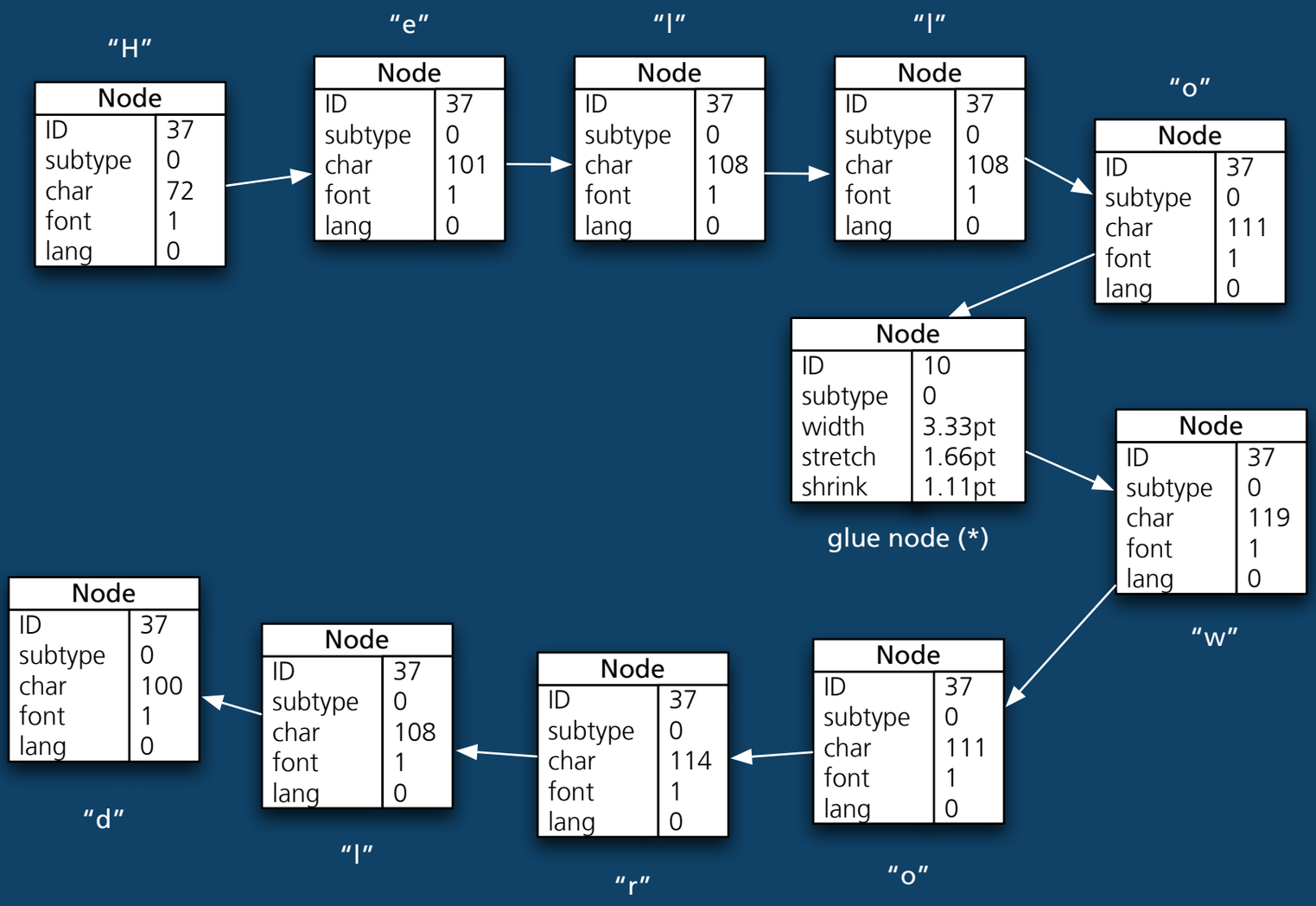
Breaking paragraphs into lines

Output nodelist

Hello world

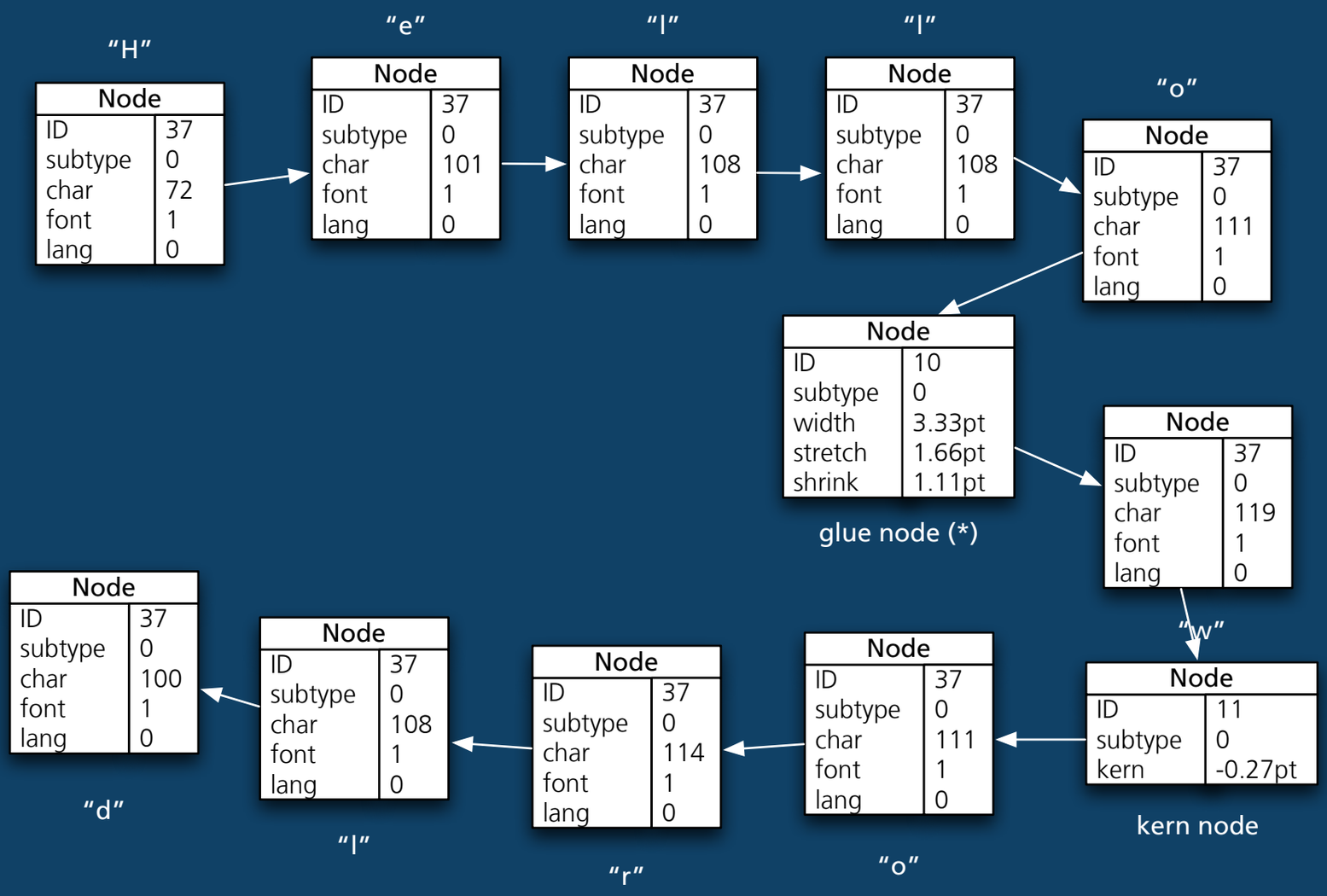

"H"
Node
ID 37
subtype 0
char 72
font 1
lang 0
"e"
Node
ID 37
subtype 0
char 101
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"o"
Node
ID 37
subtype 0
char 111
font 1
lang 0
h.next = e
Node
ID 37
subtype 0
char 119
font 1
lang 0
"w"
Node
ID 37
subtype 0
char 111
font 1
lang 0
"o"
Node
ID 37
subtype 0
char 114
font 1
lang 0
"r"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 100
font 1
lang 0
"d"

Hello world



(*) the glue node needs a glue_spec node, not displayed here

Hello world


"H"
Node
ID 37
subtype 0
char 72
font 1
lang 0
"e"
Node
ID 37
subtype 0
char 101
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"o"
Node
ID 37
subtype 0
char 111
font 1
lang 0
Node
ID 10
subtype 0
width 3.33pt
stretch 1.66pt
shrink 1.11pt
glue node (*)
Node
ID 37
subtype 0
char 119
font 1
lang 0
"w"
Node
ID 11
subtype 0
kern -0.27pt
kern node
Node
ID 37
subtype 0
char 111
font 1
lang 0
"o"
Node
ID 37
subtype 0
char 114
font 1
lang 0
"r"
Node
ID 37
subtype 0
char 108
font 1
lang 0
"l"
Node
ID 37
subtype 0
char 100
font 1
lang 0
"d"

How to make a text box

- Create font instance
- Create language instance
- ✓ Create nodelist
- Hyphenation / ligaturing / kerning
- Breaking paragraphs into lines
- Output nodelist

Hyphenation, ligaturing and kerning

```
success = lang.hyphenate(nodelist)
head,tail,success = node.kerning(nodelist)
head,tail,success = node.ligaturing(head)
```

How to make a text box

- Create font instance
- Create language instance
- ✓ Create nodelist
- ✓ Hyphenation / ligaturing / kerning
- Breaking paragraphs into lines
- Output nodelist

Breaking paragraphs into lines

- Add penalty 10000 at the end
- Add parfillskip glue at the end

```
head = tex.linebreak(nodelist,parameters)
node.vpack(head)
```

Breaking paragraphs into lines

- Add penalty 10000 at the end
- Add parfillskip glue at the end

```
head = tex.linebreak(nodelist,parameters)
node.vpack(head)
```

How to make a text box

- Create font instance
- Create language instance
- ✓ Create nodelist
- ✓ Hyphenation / ligaturing / kerning
- ✓ Breaking paragraphs into lines
- Output nodelist

Create a font instance

```
\font\myfont=cmr10
\myfont
```

- \TeX syntax again
- Only tfm fonts, no OpenType etc.

Create a font instance

```
\font\myfont=cmr10
\myfont
```

- \TeX syntax again
- Only tfm fonts, no OpenType etc.

```
-- assume that tbl has been created
-- with information form cmr10
num = font.define(tbl)
tex.definefont("myfont",num)
```

Create a font instance

```
\font\myfont=cmr10
\myfont
```

- \TeX syntax again
- Only tfm fonts, no OpenType etc

```
-- assume that tbl has been creat
-- with information form cmr10
num = font.define(tbl)
tex.definefont("myfont",num)
```

Node	
ID	37
subtype	0
next	…
char	72
font	0
lang	1
…	…

Glyph node “H”

Loading a font

(the table mentioned on the last slide)

Loading a font

(the table mentioned on the last slide)

Loading a font

name	TeXGyreHeros-Bold
designsize	789384
italicangle	0
upos	-52
parameters	(table)
size	789384
characters	(table)

(the table mentioned on the last slide)

Loading a font

name	TeXGyreHeros-Bold
designsize	789384
italicangle	0
upos	-52
parameters	(table)
size	789384
characters	(table)

(the table mentioned on the last slide)

slant	0
space	197346
space_stretch	236815
space_shrink	78938
x_height	315753
quad	789384
extra_space	0

Loading a font

name	TeXGyreHeros-Bold
designsize	789384
italicangle	0
upos	-52
parameters	(table)
size	789384
characters	(table)

slant	0
space	197346
space_stretch	236815
space_shrink	78938
x_height	315753
quad	789384
extra_space	0

(the table mentioned on the last slide)

...
92
93
94
...

height	433371
depth	7104
width	438897
kerns	(table)
ligatures	(table)

height	575460
depth	7104
width	482313
kerns	‹Tabelle›
ligatures	‹Tabelle›

Loading a for

name	TeXGyreHeros-Bold
designsize	789384
italicangle	0
upos	-52
parameters	(table)
size	789384
characters	(table)

slant	0
space	197346
space_stretch	236815
space_shrink	78938
x_height	315753
quad	789384
extra_space	0

(the table mentioned on the last slide)

...
92
93
94
...

height	433371
depth	7104
width	438897
kerns	(table)
ligatures	(table)

height	575460
depth	7104
width	482313
kerns	‹Tabelle›
ligatures	‹Tabelle›

How to make a text box

- ✓ Create font instance
- Create language instance
- ✓ Create nodelist
- ✓ Hyphenation / ligaturing / kerning
- ✓ Breaking paragraphs into lines
- Output nodelist

Create language instance

```
l=lang.new()
l:patterns(
"a1ab
aa2be
aa1c
aa2gr
aale5ne
...
z3zo
zzoll2")

-- now l:id() returns the language number
```

Create language instance

```
l=lang.new()
l:patterns(
"a1ab
aa2be
aa1c
aa2gr
aa1e5ne
...
z3zo
zzoll2")

-- now l:id() returns the language number
```

Node	
ID	37
subtype	0
next	...
char	72
font	0
lang	1
...	...

Glyph node “H”

How to make a text box

- ✓ Create font instance
- ✓ Create language instance
- ✓ Create nodelist
- ✓ Hyphenation / ligaturing / kerning
- ✓ Breaking paragraphs into lines
- Output nodelist

How to make a text box

- ✓ Create font instance
- ✓ Create language instance
- ✓ Create nodelist
- ✓ Hyphenation / ligaturing / kerning
- ✓ Breaking paragraphs into lines
- ✓ Output nodelist

```
node.write(nodelist)
```

Conclusion

- Creating output with Lua is easy and fun!
- Fast, complete control, no catcode problems
- Only works for non-TEX input (XML for example)
- Nodelist manipulation offers many new possibilities (attributes, callbacks)
- Perfectly suited for my needs