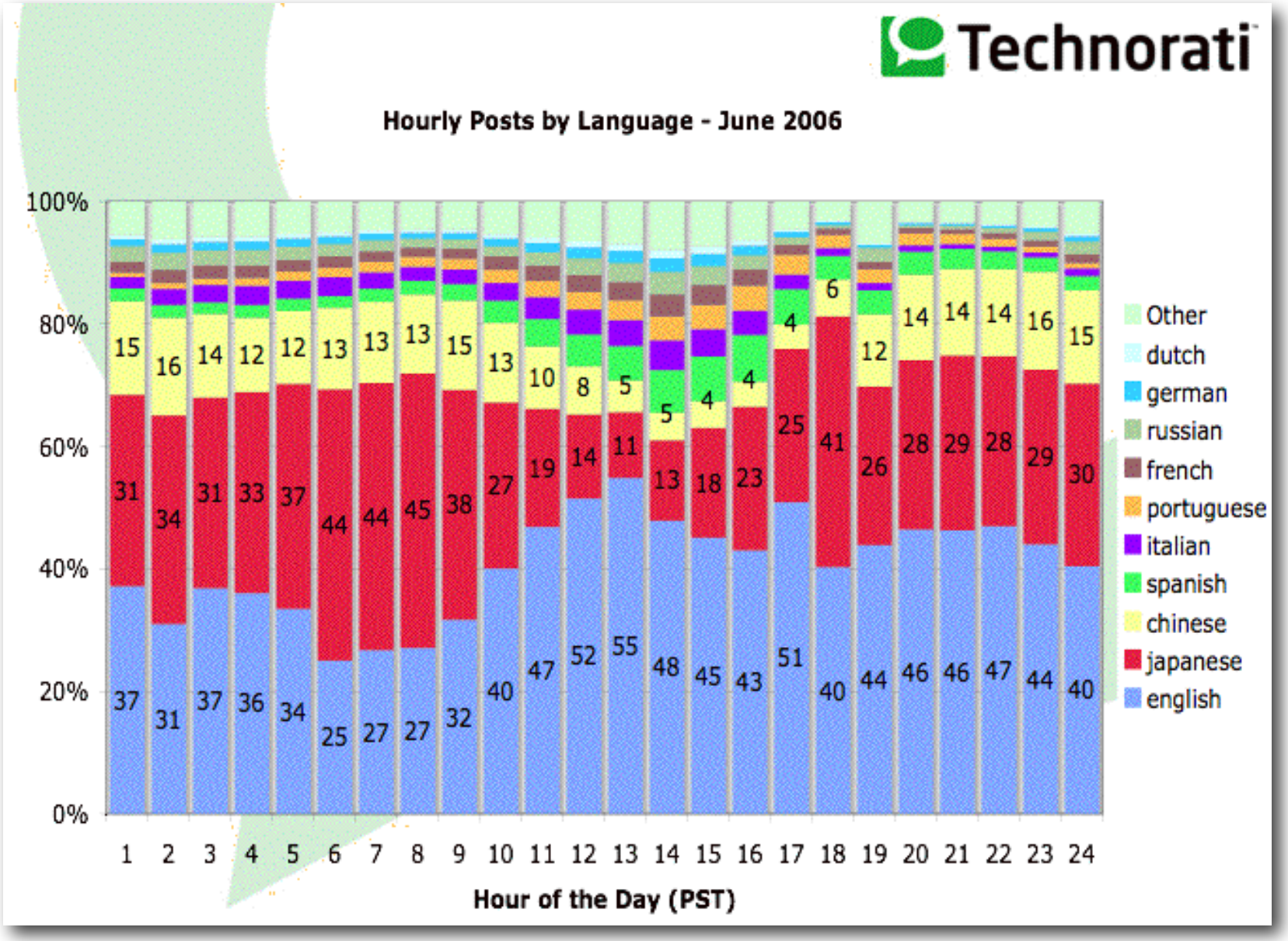
THE WORLD IN 30 MINUTES

Tim Bray
RubyKaigi 2007
Sun Microsystems

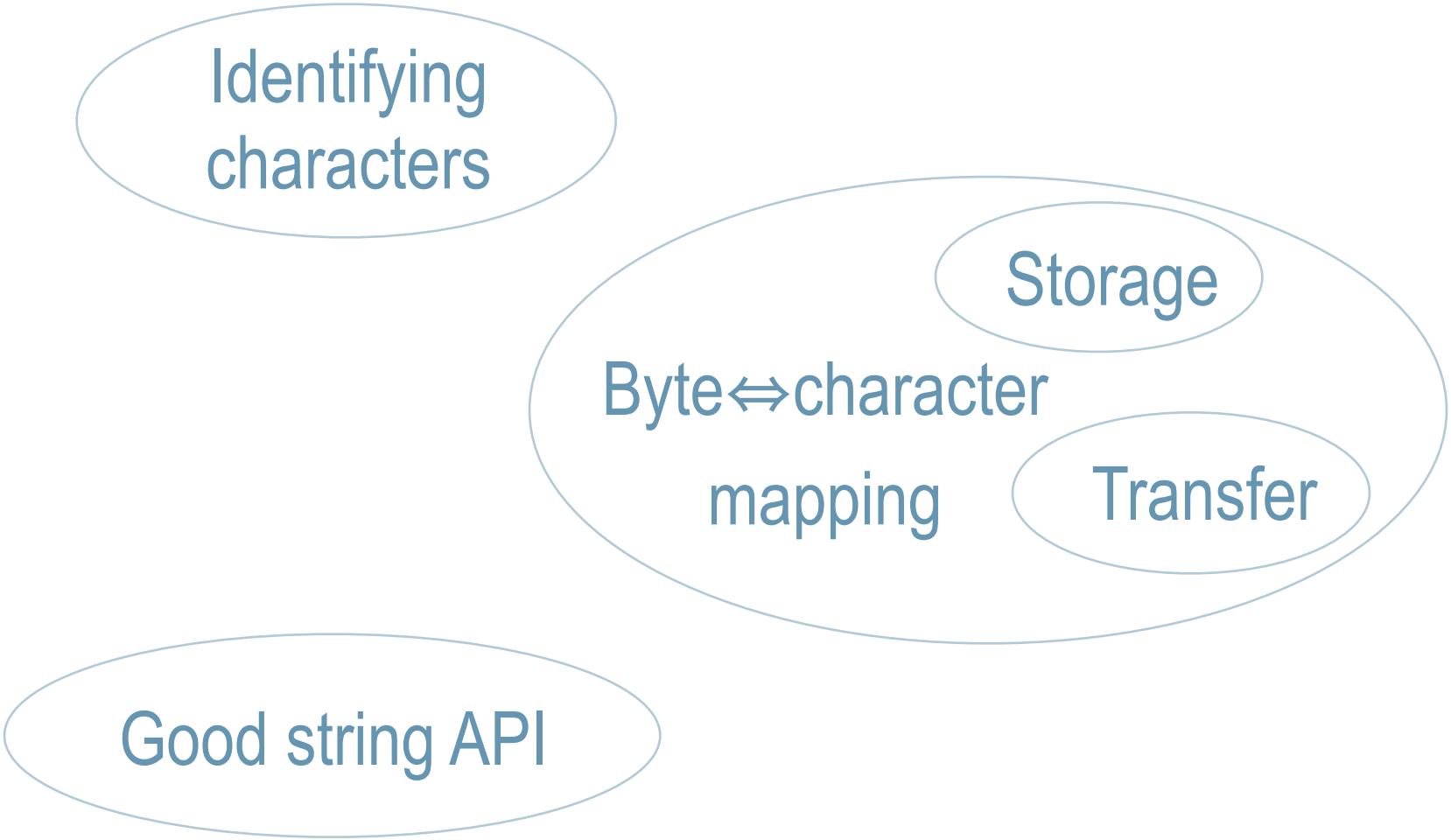

Technorati
Hourly Posts by Language - June 2006
100%
80%
60%
40%
20%
0%
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24
Hour of the Day (PST)
Other
dutch
german
russian
french
portuguese
italian
spanish
chinese
japanese
english
15 16 14 12 12 13 13 13 15 13 10 8 5 5 4 4 4 25 41 6 12 14 14 14 16 15
31 34 31 33 37 44 44 45 38 27 19 14 11 13 18 23 25 41 26 28 29 28 29 30
37 31 37 36 34 25 27 27 32 40 47 52 55 48 45 43 51 40 44 46 46 47 44 40

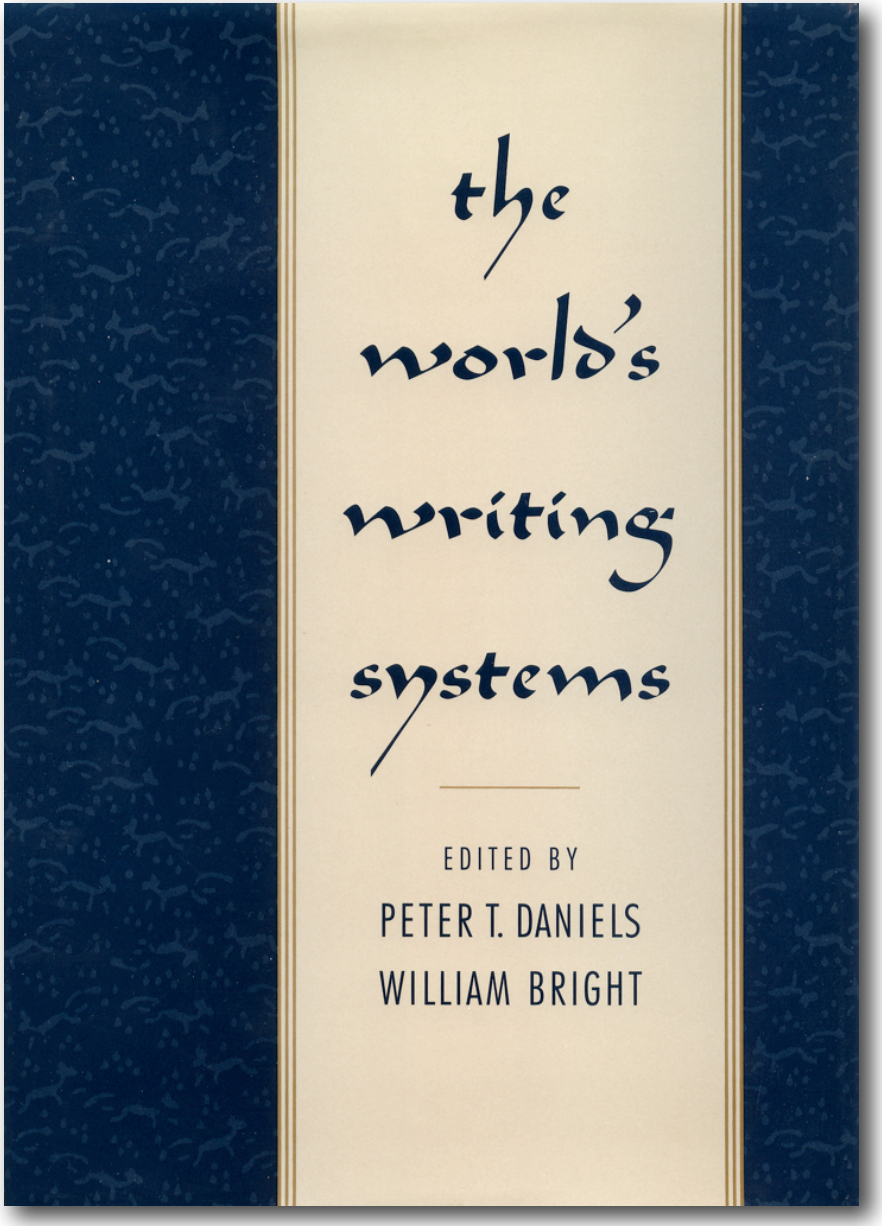
/[a-zA-Z]+/

This is probably a bug.

たぶんバグ

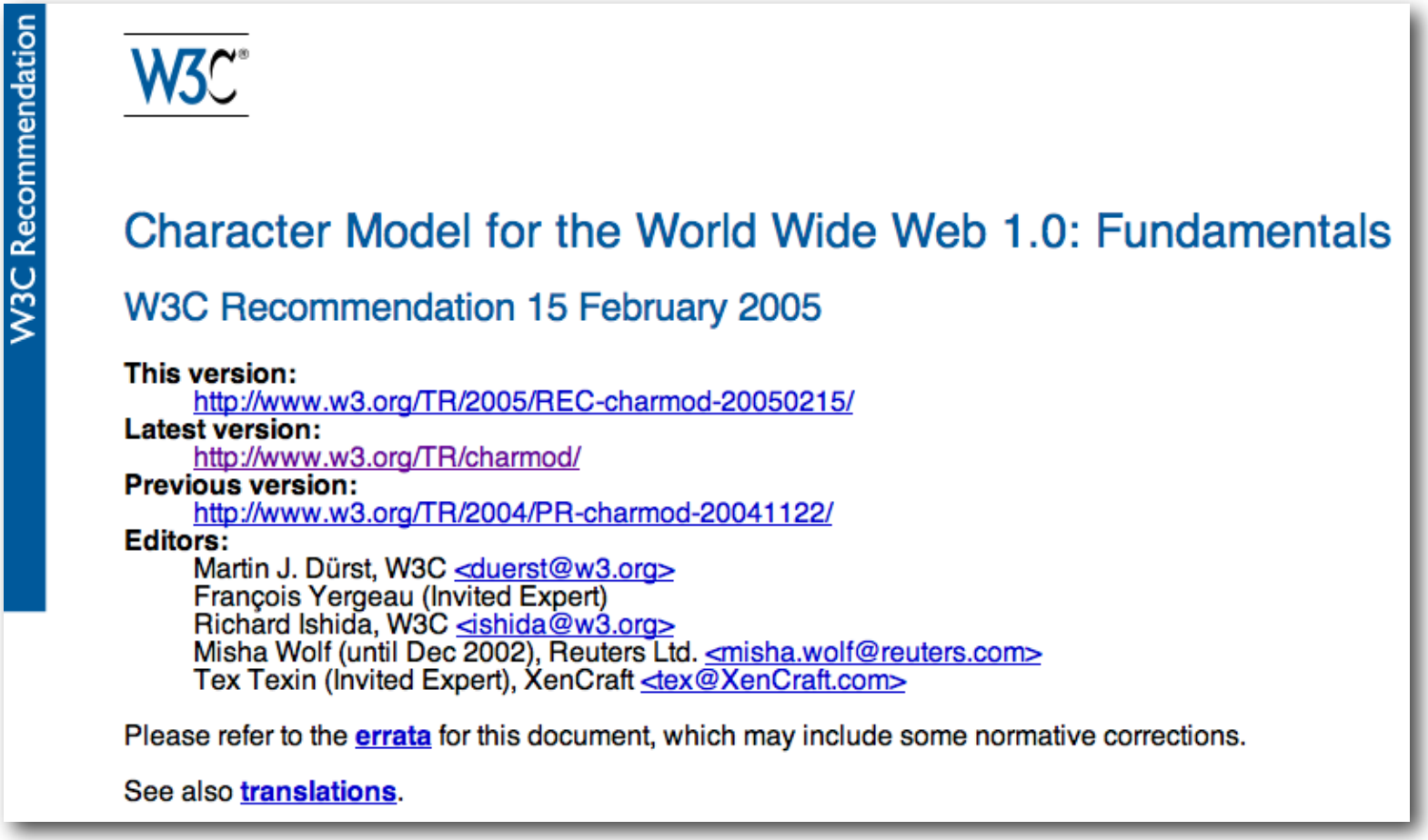
The Problems We Have To Solve
解決すべき問題





Published in 1996; it has 74 major sections, most of which discuss whole families of writing systems.

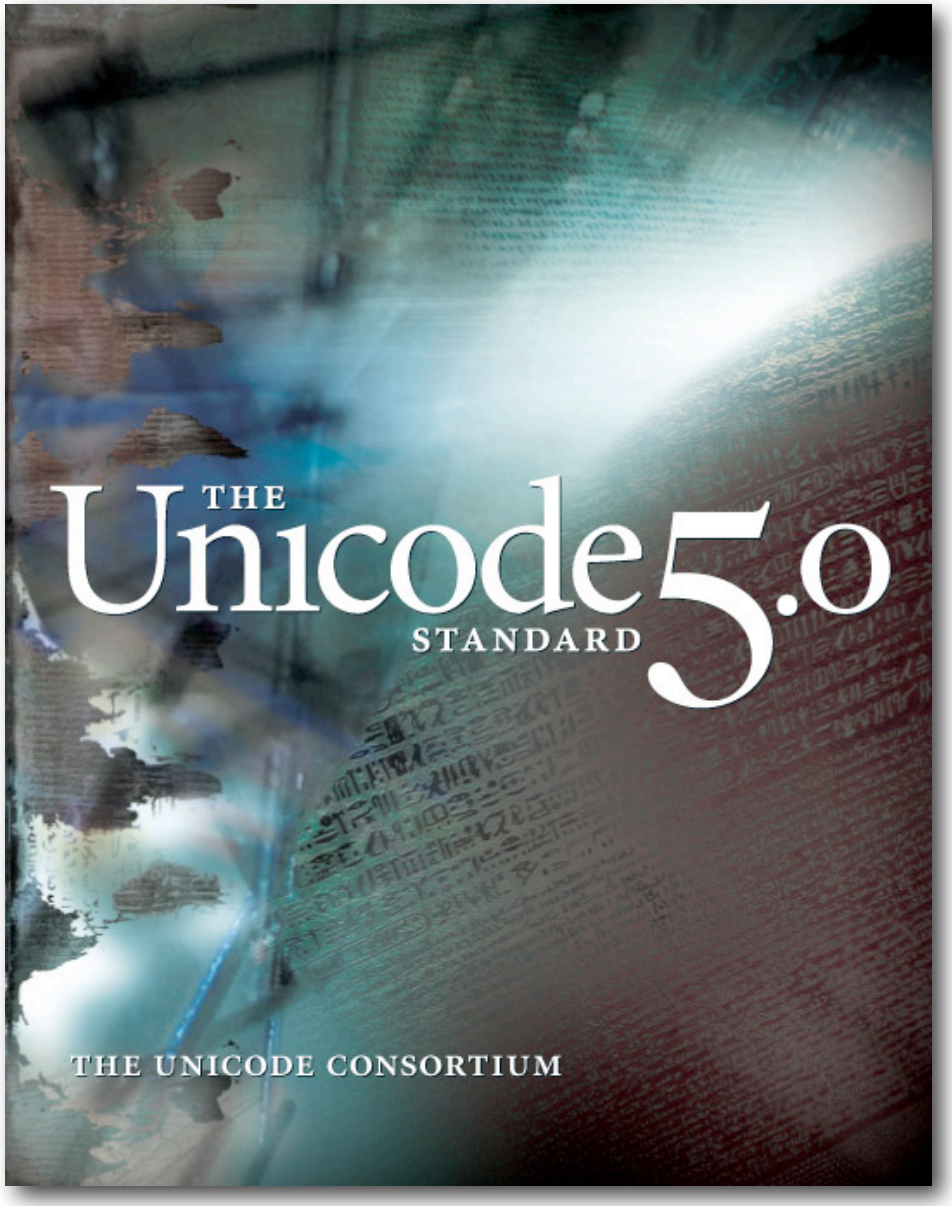
1996年刊;
世界の書記体系
についての書籍



www.w3.org/TR/charmod

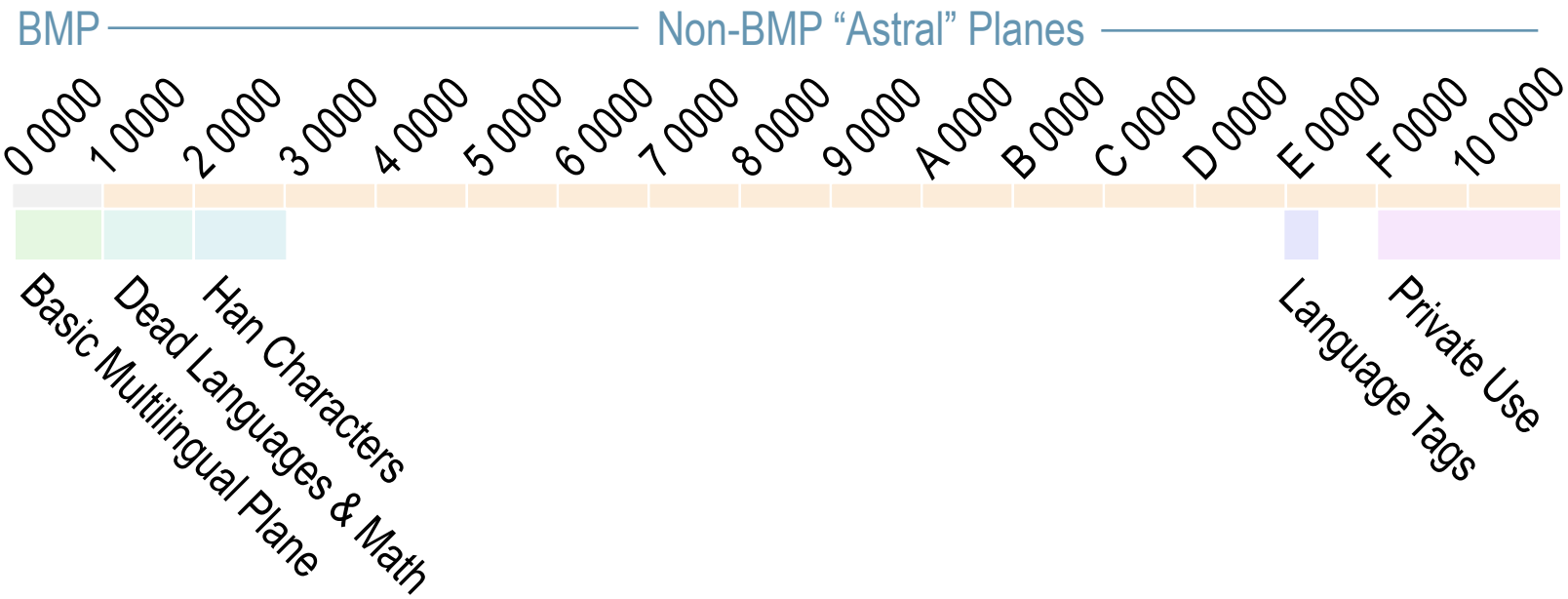
Identifying Characters

文字の識別



1,114,112 Unicode Code Points

17 “Planes” each with 64k code points: U+0000 – U+10FFFF

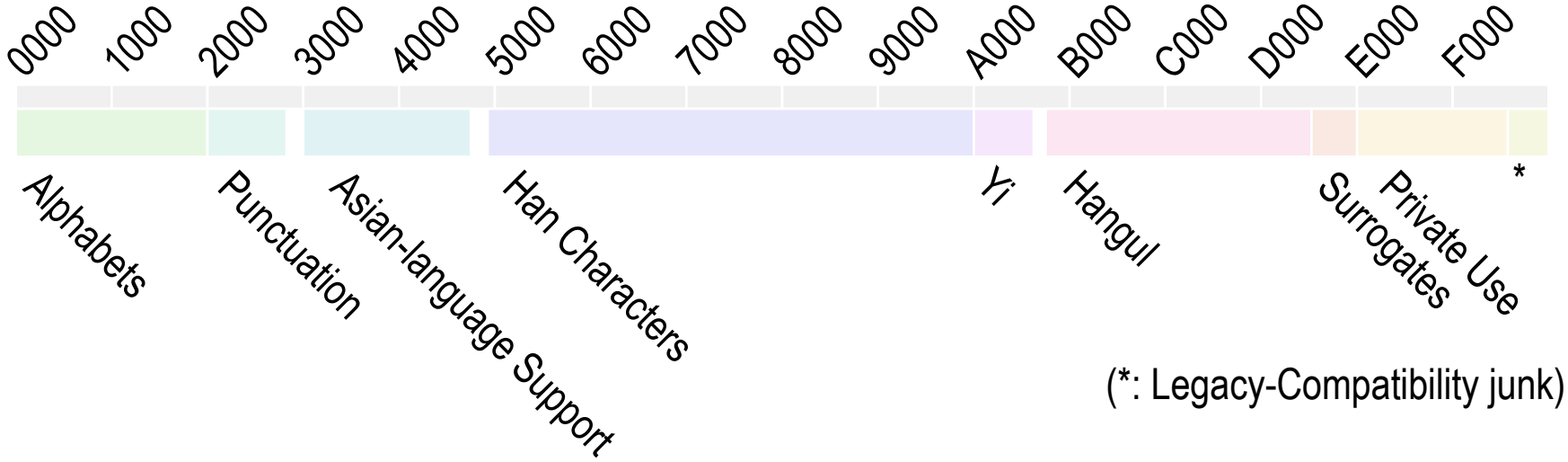


99,024 characters defined in Unicode 5.0

Unicode5.0は99,024文字を定義

The Basic Multilingual Plane (BMP)

U+0000 – U+FFFF



(*: Legacy-Compatibility junk)

Unicode Character Database

```
00C8;LATIN CAPITAL LETTER E WITH
GRAVE;Lu;0;L;0045 0300;;;;N;LATIN
CAPITAL LETTER E GRAVE;;;00E8;
```

"Character #200 is LATIN CAPITAL LETTER E WITH GRAVE, a lower-case letter, combining class 0, renders L-to-R, can be composed by U+0045/U+0300, had a different name in Unicode 1, isn't a number, lowercase is U+00E8."

www.unicode.org/Public/Unidata

U+0024 DOLLAR SIGN

U+017D LATIN CAPITAL LETTER Z WITH CARON

U+00AE REGISTERED SIGN

U+03AE GREEK SMALL LETTER ETA WITH TONOS

U+0416 CYRILLIC CAPITAL LETTER ZHE

U+05D0 HEBREW LETTER ALEF

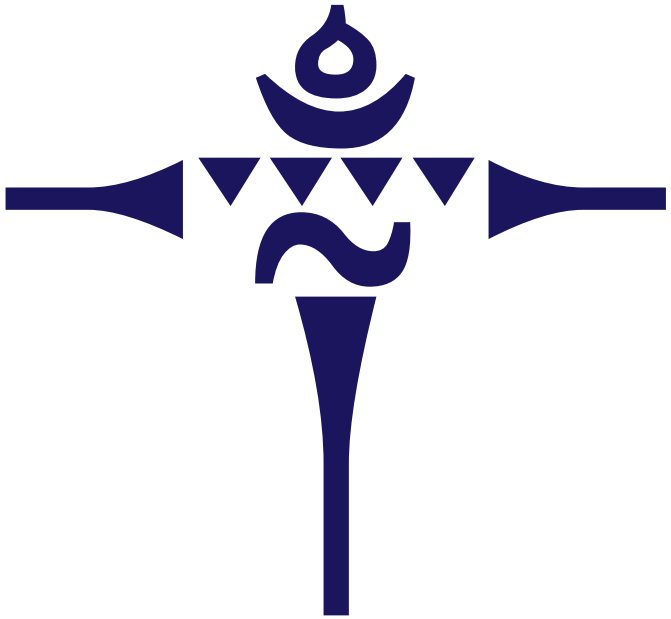
U+0638 ARABIC LETTER ZAH

ਗ

U+0A17 GURMUKHI LETTER GA

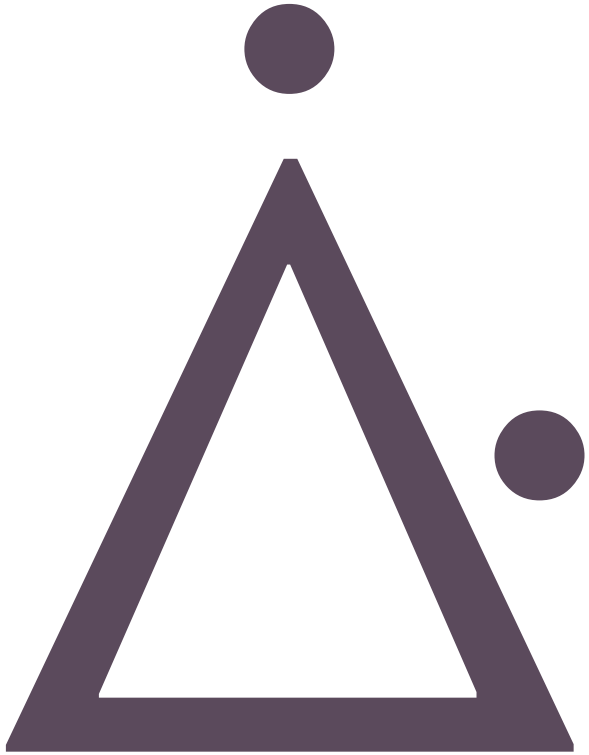
U+0A88 GUJARATI LETTER II

U+0E06 THAI CHARACTER KHO RAKHANG

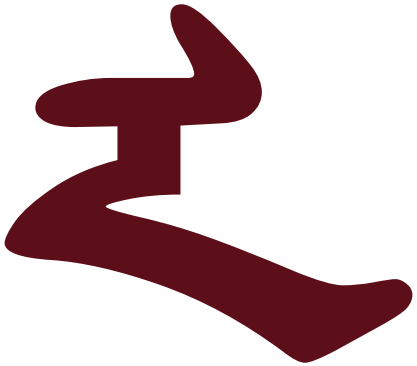


U+0F12 TIBETAN MARK RGYA GRAM SHAD

U+13BA CHEROKEE LETTER ME



U+1411 CANADIAN SYLLABICS WEST-CREE WII

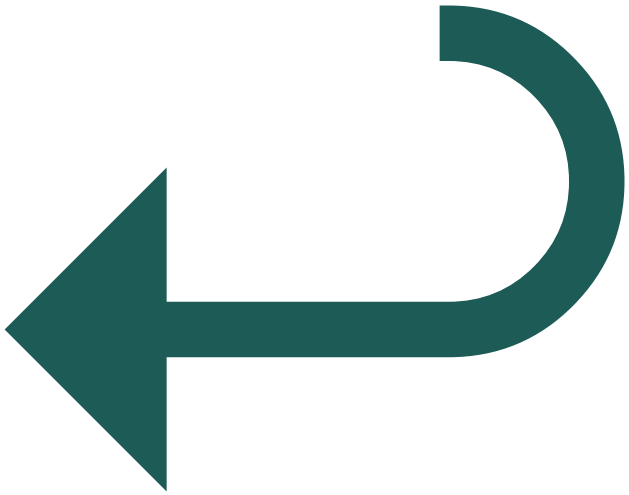


U+1820 MONGOLIAN LETTER ANG

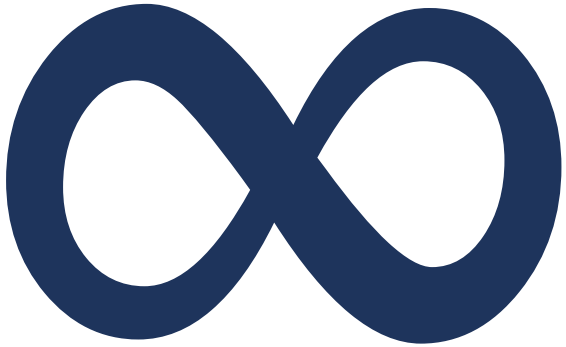
‰

U+2030 PER MILLE SIGN

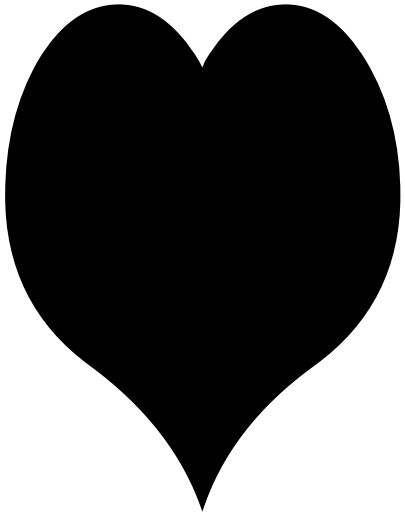
U+215D VULGAR FRACTION FIVE EIGHTHS



U+21A9 LEFTWARDS ARROW WITH HOOK



U+221E INFINITY



U+2764 HEAVY BLACK HEART

U+3055 HIRAGANA LETTER SA

U+30C0 KATAKANA LETTER DA

U+4E2D (Han character)

U+8A9E (Han character)

U+AC7A (Hangul syllabic)

U+1D12B (Non-BMP) Musical Symbol Double Flat

U+2004E (Non-BMP) (Han character)

Nice Things About Unicode
Unicodeの嬉しさ

Huge repertoire (豊富なレパートリー)

Room for growth (余裕がある)

Private use areas (私用領域)

Sane process (健全なプロセス)

Unicode character database
(ユニコード文字DB)

Ubiquitous standards/tools support
(どこでも使える標準/ツールのサポート)

Difficulties With Unicode
Unicodeの難点

Combining forms
文字合成

Awkward historical compromises
過去の負の遺産

Han unification
日中韓の漢字統合

Han Unification (漢字統合)

賛成: en.wikipedia.org/wiki/Han_Unification

反対: tronweb.super-nova.co.jp/characcodehist.html

中立: www.jbrowse.com/text/unij.html

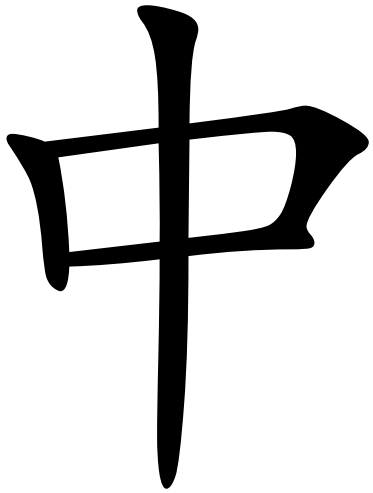
Alternatives (代替案)

For Japanese scholarly/historical work: 文字鏡,

www.mojikyo.org; also see Tron, GTCode.

Also see Wittern, *Embedding Glyph Identifiers in XML Documents*.

Byte⇔Character Mapping



U+4E2D (Han character)
How do I encode 0x4E2D in bytes for computer processing?

どうやってバイト列にしようか？

Storing Unicode in Bytes
バイト列のエンコード方式

公式: UTF-8, UTF-16, UTF-32

実用的: ASCII, EBCDIC, Shift-JIS, Big5,

GB18030, EUC-JP, EUC-KR, ISCII, KOI8,
Microsoft code pages, ISO-8859-*, and others.

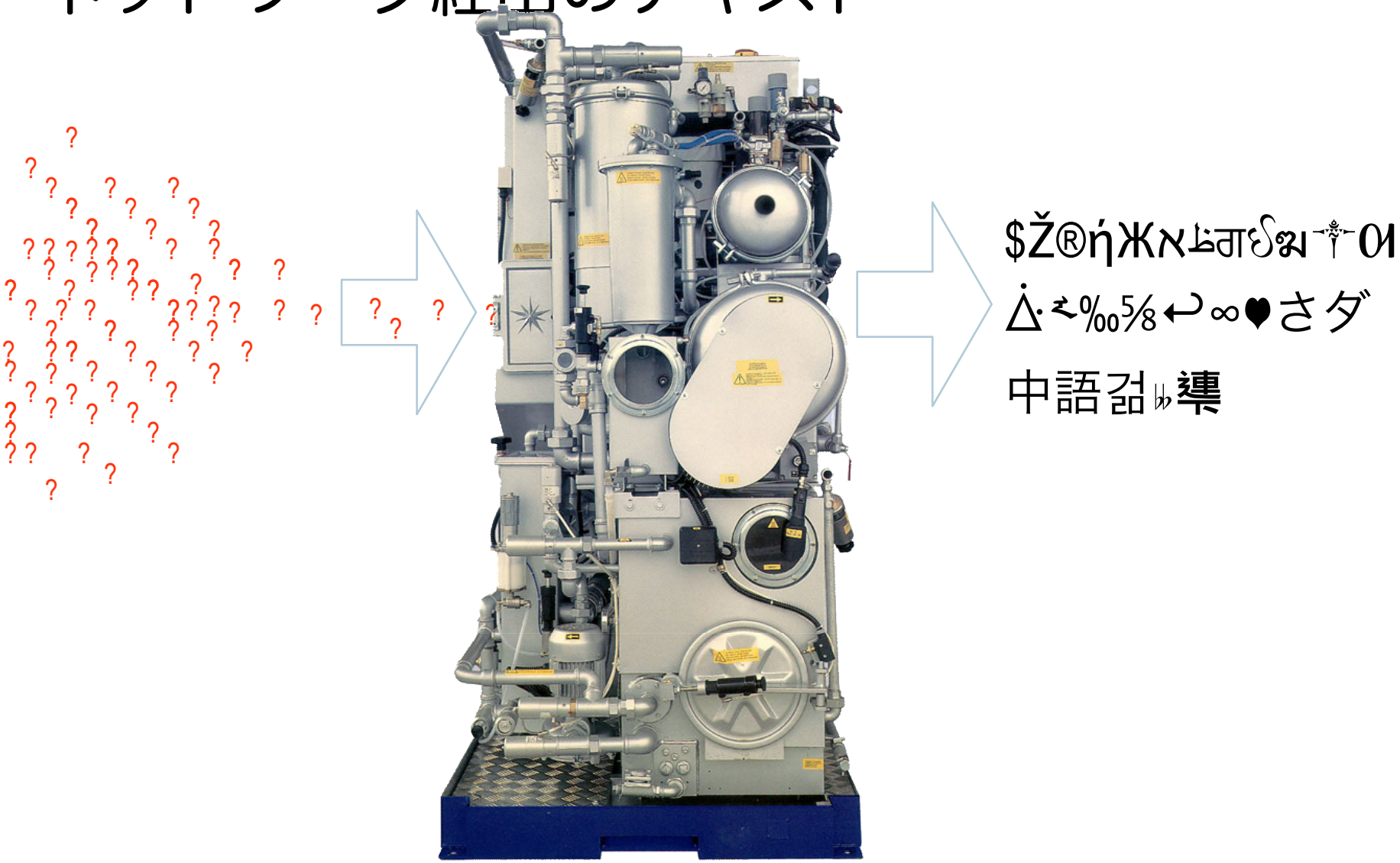
UTF-* Trade-offs

UTF-8: Most compact for Western languages, C-friendly, non-BMP processing is transparent.
UTF-16: Most compact for Eastern languages, Java/C#-friendly, C-unfriendly, non-BMP processing is horrible.
UTF-32: *wchar_t*, semi-C-friendly, 4 bytes/char.
Note: Video is 100MB/minute...

Web search: “characters vs. bytes”

続きはWebで: “characters vs. bytes”

Text Arriving Over the Network
ネットワーク経由のテキスト



“An XML document knows what encoding it’s in.”

- Larry Wall

“XMLは自分のエンコーディングを知っている”

Java の場合

Stringはユニコードだ。Javaの”char”は実際にはUTF-16のコードポイントなのでnon-BMPの扱いは微妙。Stringとbyteバッファは異なる: 符号なしバイトがない。実装は安定していて、早い。APIはちょっと無器用で、特別な正規表現がない。

Perl の場合

理論上、Perl 5はUnicodeに対応しているが、実際のアプリで、Web、ファイル、DBを使用する場合に、Unicodeを無傷で往復させるのはなかなか難しい。ただ、正規表現対応は素晴らしい。
Perl 6は、全てを解決するらしいが・・・

What Python 3000 Will Do

String型の再編

- strとunicodeではなく、bytesとstr(Guido's Slide)
  - bytesは変更可能（mutable）のint配列（range(256)）
  - encode/decode API? bytes(s, "Latin-1")?
  - bytesにstrっぽいメソッドがある (例. b1.find(b2))
  - 全部がそうだというわけじゃない (例. not b.upper())
- データはバイナリかテキスト
  - テキストデータは全てUnicodeで表現
  - I/O時に変換される
- バイナリとテキスト・ストリームのAPIが異なる
  - ファイルのエンコーディングは?(プラットフォームが決める)

(c) 2006 Python Software Foundation

Core Methods With I18n Issues
i18n問題のある組み込みメソッド

```
== =~ [] []= eql? gsub gsub! index length
lstrip lstrip! match rindex rstrip
rstrip! scan size slice slice! strip
strip! sub sub! tr tr!
```

Problem: Missing Character Iterator
問題: 文字単位のイテレータが無い

Maybe `String#each_char`

Maybe change `String#each`

Maybe `String#chars`

On Case-folding

Lower-case ‘I’: ‘i’ or ‘ı’?

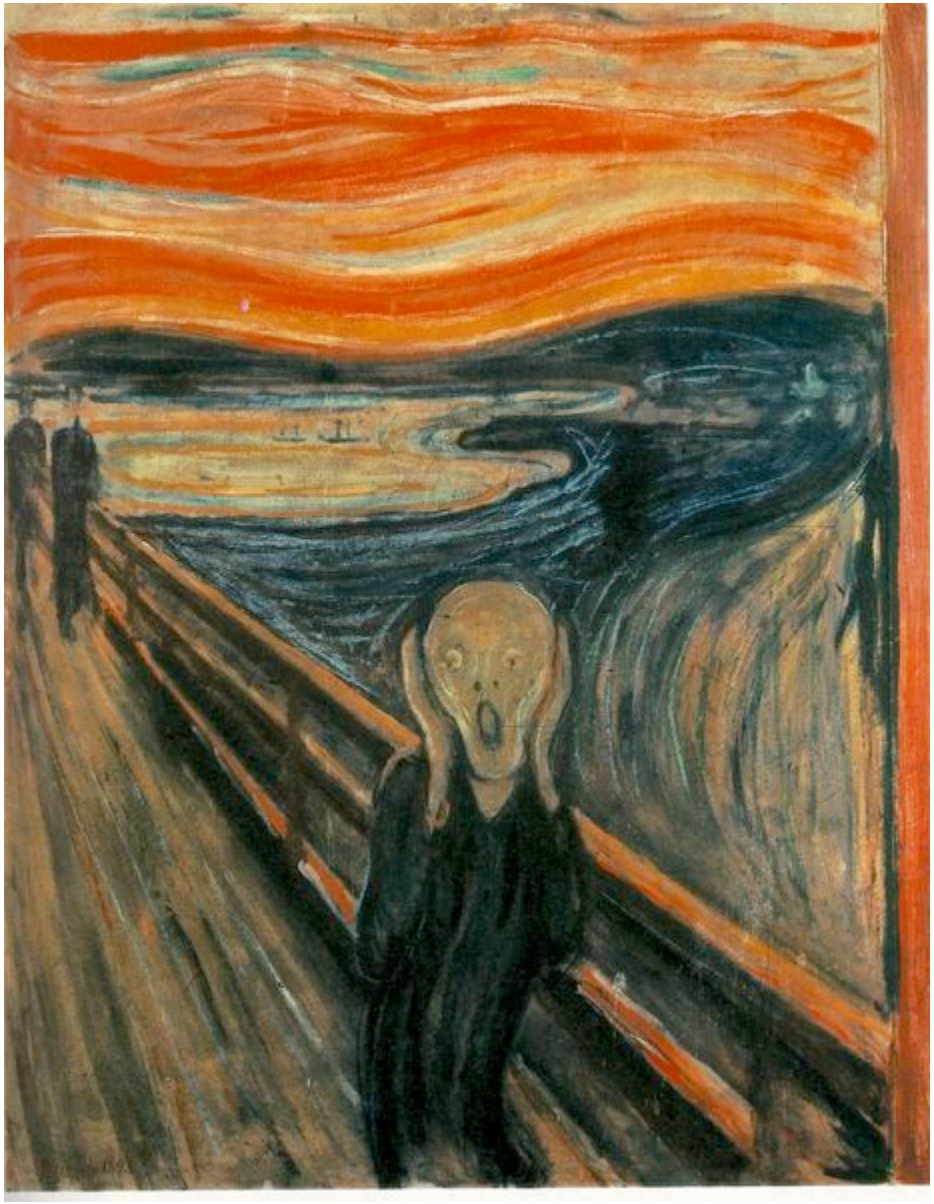
Upper-case ‘i’: ‘I’ or ‘İ’?

Upper-case ‘ß’?

Upper-case ‘é’?

Just Say No!

ほんとダメだってば!



Regexp and Unicode

Will Ruby 2.0 do these?

```
stag = "<[^/]([^>]*[^/>])?>"
etag = "</[^>]*>"
empty = "<[^>]*/>"

alnum = '\p{L}|\p{N}|' +
   '[\x{4e00}-\x{9fa5}]|' +
   '\x{3007}|[\x{3021}-\x{3029}]'
wordChars =
   '\p{L}|\p{N}|' + "[-._:']|" +
   '\x{2019}|[\x{4e00}-\x{9fa5}]|\x{3007}|' +
   '[\x{3021}-\x{3029}]'

word = "((#{alnum})((#{wordChars})*(#{alnum}))?)"
text = "(#{stag})|(#{etag})|(#{empty})|#{word}"
regex = /#{text}/
```

e.g. “won’t-go”

Referring to Characters
文字種別の参照

```
if in_euro_area?
  append 0x20ac # Euro
elsif in_japan?
  append 0xa5   # Yen
else
  append '$'
end
```

Common idiom while writing XML.
XMLを書くときのイディオム

Rubyはどうすべきか？

2007年現在、世のプログラマは、いまどきの言語ならデフォルトで文字列をUnicodeで扱えることを期待している。文字列APIがUnicodeをまともにかつ効率的に処理できることも。でなければ、自分の文化や言語への軽蔑と捉えられかねない。人文系の学問ではUnicode以外の文字も必要になるが、それ以外ではほとんど使われない。

Who's Working on the Problem?
この問題に取り組んでいるのは?

Matz/Ko1: M17n for Ruby 2
Julik: ActiveSupport::MultiByte (in edge Rails)
Nikolai: Character encodings project (rubyforge.org/projects/char-encodings/)
JRuby guys: Ruby on a Unicode platform

Thank You!

Tim.Bray@sun.com

www.tbray.org/ongoing/

this talk: www.tbray.org/talks/rubykaigi2007.pdf