Scripters UTF-8 Survival Guide

Harry Fuecks
webtuesday.ch
8th August 2006

Agenda

- Theory
- Surviving UTF-8...
  - ...in PHP
  - ...in MySQL
  - ...a glance and Perl and Python

Disclaimer

- “Live experience” from involvement in Dokuwiki
  - the rest is theory / research / reading
- Wrote PHP UTF-8 http://phputf8.sourceforge.net/
- Not a Unicode nerd

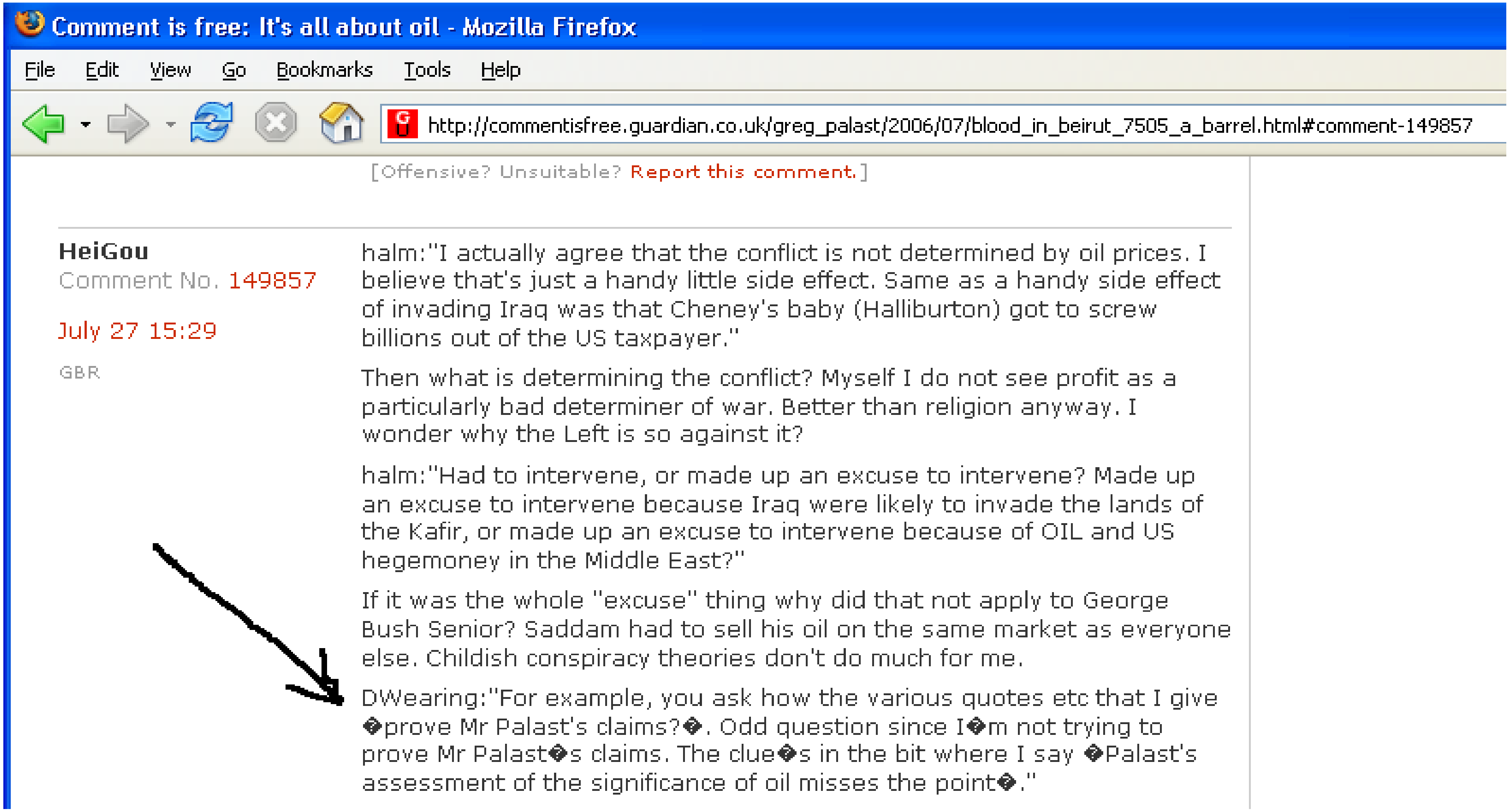
Impedance Mismatch

- Unicode first draft ~ 1990
- Most (web) developers still don't get it. Why?
  - “Not my problem” : US / UK / Western Europe
  - Learning Curve
  - Implementations (or lack of... PHP!)
  - WYSINWYG: bytes are invisible
  - Migration can hurt

Fear and Loathing

- What not getting it means...
  - Ugly output – what are those funny characters?

Fear and Loathing


Comment is free: It's all about oil - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://commentisfree.guardian.co.uk/greg_palast/2006/07/blood_in_beirut_7505_a_barrel.html#comment-149857

[Offensive? Unsuitable? Report this comment.]

HeiGou
Comment No. 149857

July 27 15:29

GBR

halm:"I actually agree that the conflict is not determined by oil prices. I believe that's just a handy little side effect. Same as a handy side effect of invading Iraq was that Cheney's baby (Halliburton) got to screw billions out of the US taxpayer."

Then what is determining the conflict? Myself I do not see profit as a particularly bad determiner of war. Better than religion anyway. I wonder why the Left is so against it?

halm:"Had to intervene, or made up an excuse to intervene? Made up an excuse to intervene because Iraq were likely to invade the lands of the Kafir, or made up an excuse to intervene because of OIL and US hegemoney in the Middle East?"

If it was the whole "excuse" thing why did that not apply to George Bush Senior? Saddam had to sell his oil on the same market as everyone else. Childish conspiracy theories don't do much for me.

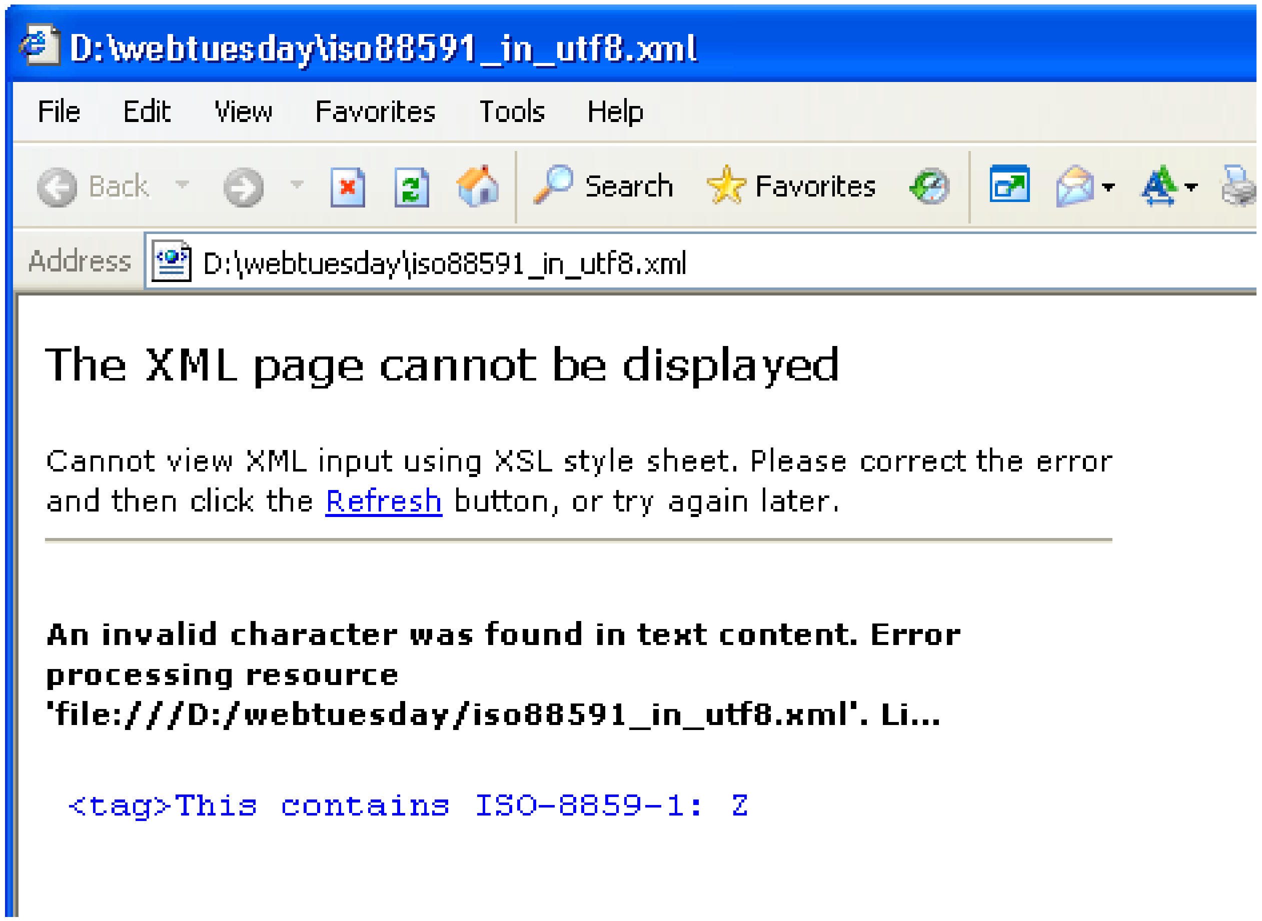
DWearing:"For example, you ask how the various quotes etc that I give �prove Mr Palast's claims?�. Odd question since I�m not trying to prove Mr Palast�s claims. The clue�s in the bit where I say �Palast's assessment of the significance of oil misses the point�."

Page declared as UTF-8 but comment form page ISO-8859-1

Fear and Loathing

- What not getting it means...
  - Ugly output – what are those funny characters?
  - Denial of service? XML feeds “broken”?

Fear and Loathing


D:\webtuesday\iso88591_in_utf8.xml
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
D:\webtuesday\iso88591_in_utf8.xml

The XML page cannot be displayed

Cannot view XML input using XSL style sheet. Please correct the error and then click the Refresh button, or try again later.

An invalid character was found in text content. Error processing resource 'file:///D:/webtuesday/iso88591_in_utf8.xml'. Li...

```
<tag>This contains ISO-8859-1: Z
```

...IE / msxml

Fear and Loathing

PHP's SimpleXML:

```
Warning: simplexml_load_file(): iso88591_in_utf8.xml:4:
parser error :
Input is not proper UTF-8, indicate encoding ! Bytes: 0xFC
0x72 0x69 0x63 in
D:\webtuesday\phpsimplexml_iso88591_in_utf8.php on line 2

Warning: simplexml_load_file():  <tag>This contains ISO-
8859-1: Zch</tag>
 in D:\webtuesday\phpsimplexml_iso88591_in_utf8.php on
line 2

Warning: simplexml_load_file():
      ^
in D:\webtuesday\phpsimplexml_iso88591_in_utf8.php on line
2
bool(false)
```

Fear and Loathing

- What not getting it means...
  - Ugly output – what are those funny characters?
  - Denial of service? XML feeds “broken”
  - Accessibility: can Chinese, Japanese, Russians add comments to your blog?

Fear and Loathing

- What not getting it means...
  - Ugly output – what are those funny characters?
  - Denial of service? XML feeds “broken”
  - Accessibility: can Chinese, Japanese, Russians add comments to your blog?
  - Humiliation...

    “*When I discovered that the popular web development tool PHP has almost complete ignorance of character encoding issues, blithely using 8 bits for characters, making it darn near impossible to develop good international web applications, I thought, enough is enough.*”

    http://www.joelonsoftware.com/articles/Unicode.html

Grokking Unicode

- Keyword: i18n
- Difficult to find relevant information
  - Assumed knowledge
  - Scattered
  - Few bloggers getting it (be careful who you trust!)
- MUST READ: "A tutorial on character code issues"
  http://www.cs.tut.fi/~jkorpela/chars.html
- “Do you know your character encodings?”
  http://www.sitepoint.com/blogs/2006/03/15/do-you-know-your-character-encodings/

Terminology

- **Character**: smallest semantic component of written language
- **Character set**: (abstract) group of characters
  - Characters assigned (abstract) numbers (by committees)
  - May correspond to alphabets (e.g. Latin or Cyrillic)
  - Unicode – one character set to rule them all

WARNING: terminology may vary, depending on context, who you listen to and sometimes through carelessness

e.g. (HTTP header) `Content-Type: text/html; charset=UTF-8`

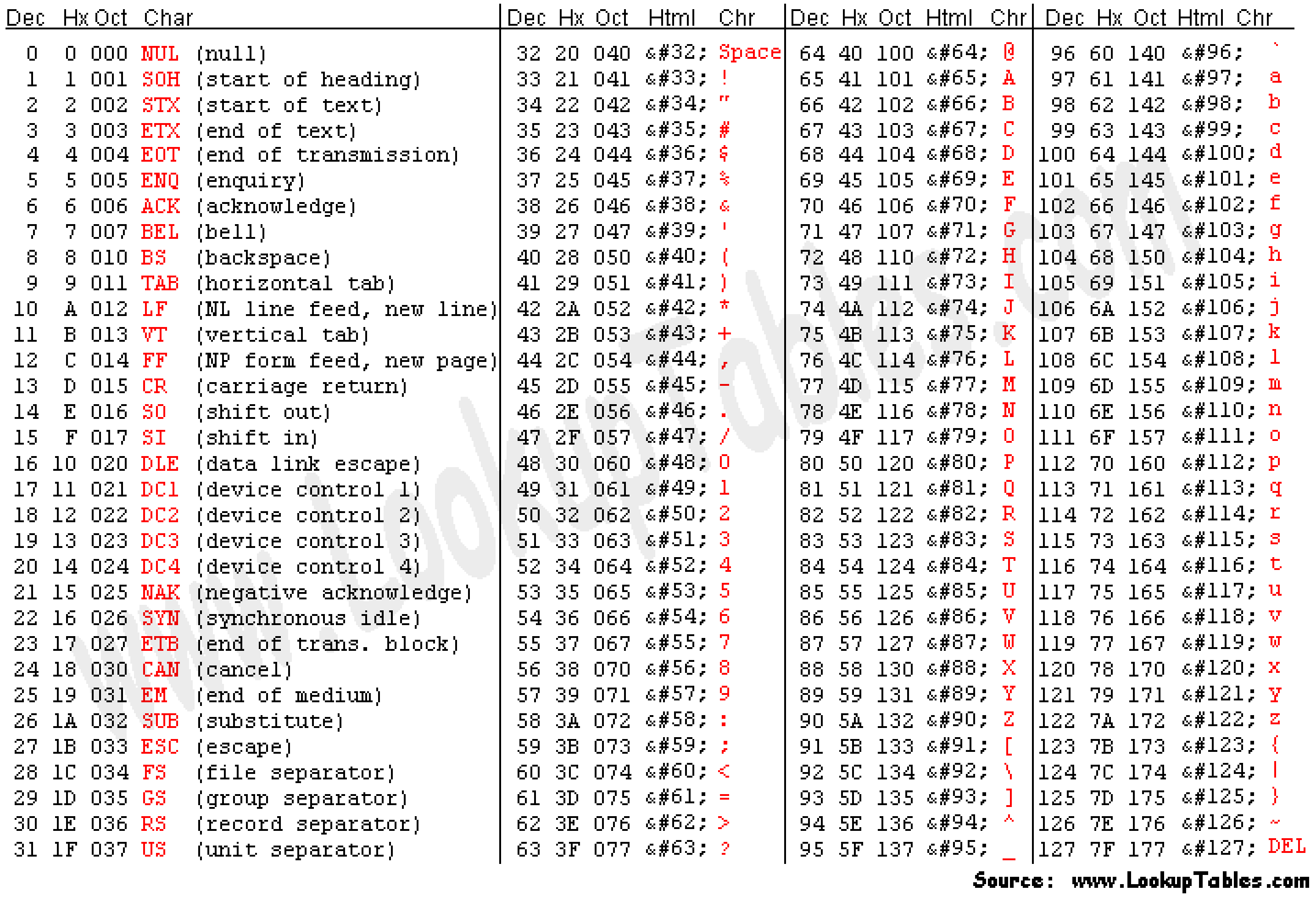
Terminology

- **Character encoding**: how to represent a character *set* using 1's and 0's
  - Use algorithm / lookup table to translate between character *set* and character *encoding*
  - Example: Unicode is a character *set*. It has multiple *encodings*: UTF-7, UTF-8, UTF-16, UTF-32 (etc.)
- **Font**: collection of images (glyphs) used by an application to display characters
  - Your text editor may support Unicode (via, say, UTF-8) but it's current font may not support all Unicode characters
  - WYSI**N**WYG

Encodings you need to know about

- ISO-8859-1 (Latin 1)
  - Western Europe (minus the Euro sign)
  - Something like the “default” encoding for the web
- CP-1252 (the evil twin of ISO-8859-1)
  - IE + Windows 98 = CP-1252
  - http://intertwingly.net/stories/2004/04/14/i18n.html#CleaningWindows
- ASCII as in US-ASCII (7 bits)
  - **ISO-8859-1 != ASCII !!!**
- UTF-8 : encoding of Unicode

ASCII Reminder



Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	&#32;	Space	64	40	100	&#64;	@	96	60	140	&#96;	`
1	1	001	SOH (start of heading)	33	21	041	&#33;	!	65	41	101	&#65;	A	97	61	141	&#97;	a
2	2	002	STX (start of text)	34	22	042	&#34;	"	66	42	102	&#66;	B	98	62	142	&#98;	b
3	3	003	ETX (end of text)	35	23	043	&#35;	#	67	43	103	&#67;	C	99	63	143	&#99;	c
4	4	004	EOT (end of transmission)	36	24	044	&#36;	$	68	44	104	&#68;	D	100	64	144	&#100;	d
5	5	005	ENQ (enquiry)	37	25	045	&#37;	%	69	45	105	&#69;	E	101	65	145	&#101;	e
6	6	006	ACK (acknowledge)	38	26	046	&#38;	&	70	46	106	&#70;	F	102	66	146	&#102;	f
7	7	007	BEL (bell)	39	27	047	&#39;	'	71	47	107	&#71;	G	103	67	147	&#103;	g
8	8	010	BS (backspace)	40	28	050	&#40;	(	72	48	110	&#72;	H	104	68	150	&#104;	h
9	9	011	TAB (horizontal tab)	41	29	051	&#41;	)	73	49	111	&#73;	I	105	69	151	&#105;	i
10	A	012	LF (NL line feed, new line)	42	2A	052	&#42;	*	74	4A	112	&#74;	J	106	6A	152	&#106;	j
11	B	013	VT (vertical tab)	43	2B	053	&#43;	+	75	4B	113	&#75;	K	107	6B	153	&#107;	k
12	C	014	FF (NP form feed, new page)	44	2C	054	&#44;	,	76	4C	114	&#76;	L	108	6C	154	&#108;	l
13	D	015	CR (carriage return)	45	2D	055	&#45;	-	77	4D	115	&#77;	M	109	6D	155	&#109;	m
14	E	016	SO (shift out)	46	2E	056	&#46;	.	78	4E	116	&#78;	N	110	6E	156	&#110;	n
15	F	017	SI (shift in)	47	2F	057	&#47;	/	79	4F	117	&#79;	O	111	6F	157	&#111;	o
16	10	020	DLE (data link escape)	48	30	060	&#48;	0	80	50	120	&#80;	P	112	70	160	&#112;	p
17	11	021	DC1 (device control 1)	49	31	061	&#49;	1	81	51	121	&#81;	Q	113	71	161	&#113;	q
18	12	022	DC2 (device control 2)	50	32	062	&#50;	2	82	52	122	&#82;	R	114	72	162	&#114;	r
19	13	023	DC3 (device control 3)	51	33	063	&#51;	3	83	53	123	&#83;	S	115	73	163	&#115;	s
20	14	024	DC4 (device control 4)	52	34	064	&#52;	4	84	54	124	&#84;	T	116	74	164	&#116;	t
21	15	025	NAK (negative acknowledge)	53	35	065	&#53;	5	85	55	125	&#85;	U	117	75	165	&#117;	u
22	16	026	SYN (synchronous idle)	54	36	066	&#54;	6	86	56	126	&#86;	V	118	76	166	&#118;	v
23	17	027	ETB (end of trans. block)	55	37	067	&#55;	7	87	57	127	&#87;	W	119	77	167	&#119;	w
24	18	030	CAN (cancel)	56	38	070	&#56;	8	88	58	130	&#88;	X	120	78	170	&#120;	x
25	19	031	EM (end of medium)	57	39	071	&#57;	9	89	59	131	&#89;	Y	121	79	171	&#121;	y
26	1A	032	SUB (substitute)	58	3A	072	&#58;	:	90	5A	132	&#90;	Z	122	7A	172	&#122;	z
27	1B	033	ESC (escape)	59	3B	073	&#59;	;	91	5B	133	&#91;	[	123	7B	173	&#123;	{
28	1C	034	FS (file separator)	60	3C	074	&#60;	<	92	5C	134	&#92;	\	124	7C	174	&#124;	|
29	1D	035	GS (group separator)	61	3D	075	&#61;	=	93	5D	135	&#93;	]	125	7D	175	&#125;	}
30	1E	036	RS (record separator)	62	3E	076	&#62;	>	94	5E	136	&#94;	^	126	7E	176	&#126;	~
31	1F	037	US (unit separator)	63	3F	077	&#63;	?	95	5F	137	&#95;	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

Watch out ASCII for control codes and XML!
http://hsivonen.iki.fi/producing-xml/#controlchar
Perl: `s/[^\x09\x0A\x0D\x20-\x7E]//g`

Which encoding for storing text?

- Store different encodings for different character sets? **BAD IDEA!**
  - What if a web page needs multiple character sets?
    - e.g. Blog entry posted by a Russian with comments by someone Japanese
  - Leads to developer insanity – radical complexity
- Use an encoding of Unicode? :)
  - Foregone conclusion (on the web): use UTF-8

Why UTF-8?

- Encoding of Unicode
  - Represent any character – one i18n issue solved
- Designed by Ken Thompson and Rob Pike
  - The nerd generation that really knew stuff
- Backward compatible with ASCII (7 bit)
- Wide support
  - Best survival chances with PHP
  - Modern browsers do a good job
- It's what “everyone” is doing

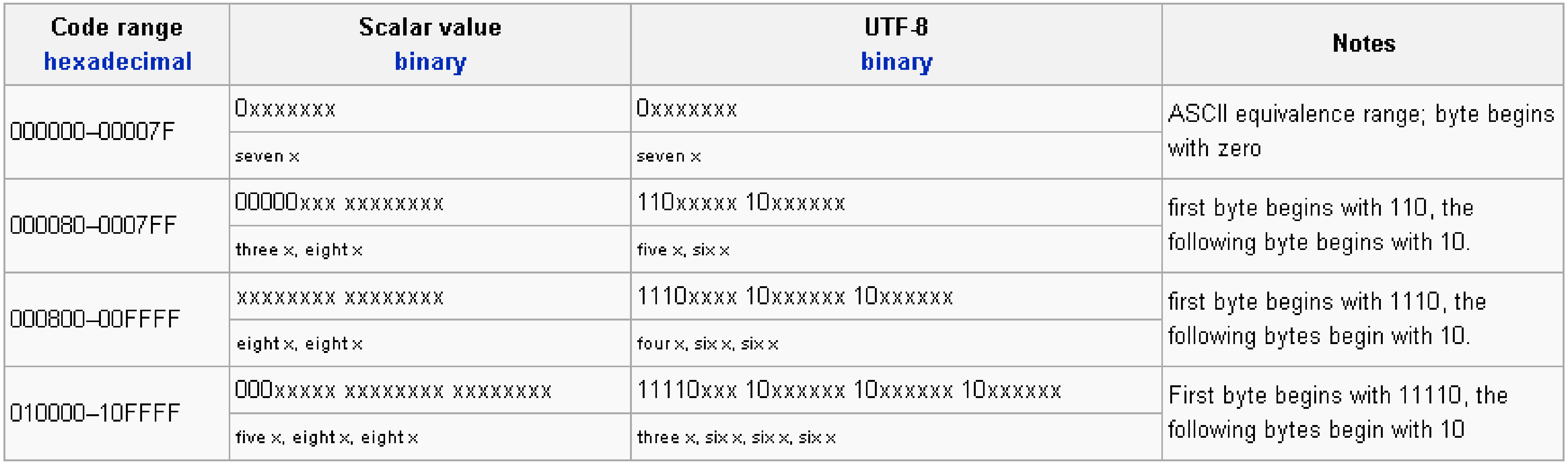
Why not UTF-8

- Racist?
  - http://www.tbray.org/ongoing/When/200x/2003/04/26/UTF
- Performance?
  - Handling more expensive than, say, UTF-32
  - Memory vs. performance

UTF-8 Design

- ASCII in UTF-8 encoded same as ASCII
  - If all you've got is ASCII, “just” relabel as UTF-8
- Everything else (every non-ASCII character) encoded using sequences of 2-6 bytes
  - First byte indicates length of sequence

UTF-8 Design



Code range hexadecimal	Scalar value binary	UTF-8 binary	Notes
000000–00007F	0xxxxxxx	0xxxxxxx	ASCII equivalence range; byte begins with zero
	seven x	seven x	
000080–0007FF	00000xxx xxxxxxxx	110xxxxx 10xxxxxx	first byte begins with 110, the following byte begins with 10.
	three x, eight x	five x, six x	
000800–00FFFF	xxxxxxxx xxxxxxxx	1110xxxx 10xxxxxx 10xxxxxx	first byte begins with 1110, the following bytes begin with 10.
	eight x, eight x	four x, six x, six x	
010000–10FFFF	000xxxxx xxxxxxxx xxxxxxxx	11110xxx 10xxxxxx 10xxxxxx 10xxxxxx	First byte begins with 11110, the following bytes begin with 10
	five x, eight x, eight x	three x, six x, six x, six x	

Source: http://en.wikipedia.org/wiki/UTF-8

UTF-8 Design

- UTF-8 has a *structure*
  - “Well formed” in the same sense as XML
  - So need to check for *badly* formed UTF-8
- No sequence can be mistaken for part of a longer sequence (if it's well formed)
  - Simplified algorithms
  - Backwards compatibility with ASCII routines
  - ...which means this works:
    `explode('à','Iñtërnâtiônàlizætiøn');`

UTF-8 Gotchas

- Unicode needs only 2-4 byte UTF-8 sequences.
  - 5-6 byte sequences have no meaning in Unicode but are supported by UTF-8
- UTF-7 can be mistaken for UTF-8
  - XSS potential – declare you character sets! http://shiflett.org/archive/177
- Failing to check for well formedness
- Performance: missed optimizations through API overuse
  - Failing to understand the design

UTF-8 Survival

- Check your text editor!
  - http://www.phpwact.org/php/i18n/charsets#editors_with_utf-8_support
  - PHP doesn't like BOM in scripts
    - http://en.wikipedia.org/wiki/Byte_Order_Mark
- Server Locale?
  - `$ locale`
- Where are you storing data?
  - Database? Text files (e.g. Error messages, l10n)? PDF? Images? etc.
- What are your interfaces?
  - RSS? Email? etc.

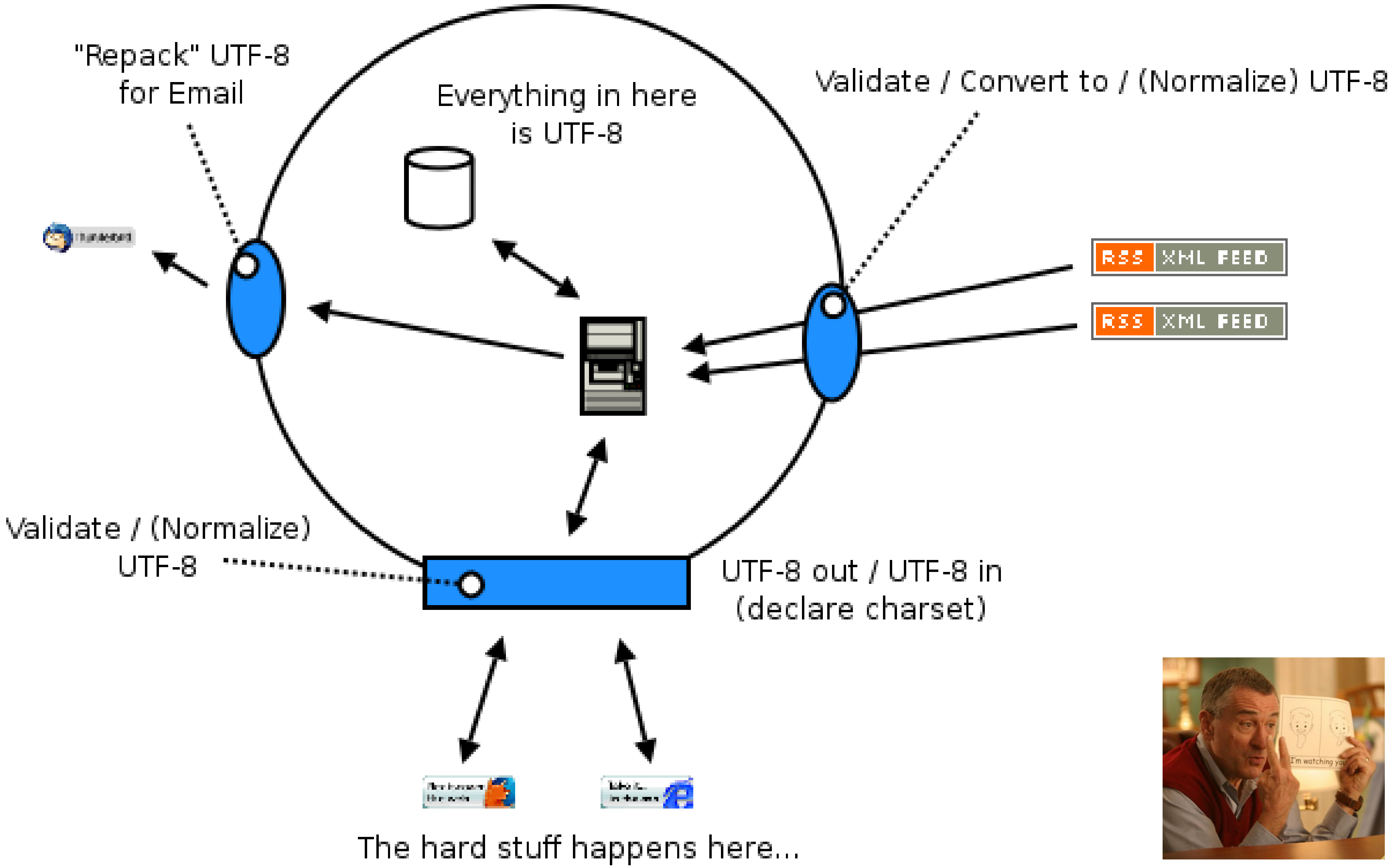
UTF-8 Survival: Strategy

- Make it someone else's problem
  - Browsers have good support
- Store everything as UTF-8
  - Consequences?
    - e.g. More space required for VARCHARs
- Publish everything as UTF-8
- Migrate (big bang); don't iterate

UTF-8 Survival: Issues

- Security?
  - The joy of phishing
    - recommend: restrict primary keys, identifiers to ASCII
  - UTF-7 XSS
    - Declare UTF-8
- Unicode Normalization?
  - e.g. identifiers, searching
- Sorting (Collation)
  - How smart is your UTF-8 library?
  - How much do you care?

UTF-8 Circle of Trust



UTF-8 Survival in PHP

- PHP (< v6) assumes 1 character = 1 byte
  - `strlen('Iñtërnâtiônàlizætiøn') == 27`
- Also tries to be “smart” using server locale
  - e.g. `strtolower()` / `strtoupper()`
- Lists of UTF-8 “dangerous” PHP functions
  - http://www.phpwact.org/php/i18n/utf-8
  - http://wiki.silverorange.com/UTF-8_Notes
- General reading
  - http://www.phpwact.org/php/i18n/charsets
- Good news: PHP 6 + ICU

Useful PHP Functionality

- mbstring http://www.php.net/mbstring
  - Alternatives for (some) native PHP string functions
  - Includes function “overloading”
  - Best chance of survival
- iconv http://www.php.net/iconv
  - Character set *conversion*
  - Some (slow) replacement functions
  - Useful for specific tasks e.g. migrating to UTF-8, converting input (e.g. RSS)

Useful PHP Functionality

- `utf8_encode() utf8_decode()`
  http://www.php.net/xml
  - *Only* for converting between ISO-8859-1 and UTF-8
  - Useful hack for `strlen()` replacement;
    - `strlen(utf8_decode($str));`
      - given valid UTF-8, converts all multi byte sequences to one byte
- PCRE /u
  - Regards pattern and subject as UTF-8
  - Doesn't understand Unicode!

Useful PHP Functionality

- GNU Recode: http://www.php.net/recode
  - Similar role to iconv
  - In reality, causes conflicts with other PHP extensions, unpopular
  - Forget it!

Userland Helpers

- PHP UTF-8
  http://sourceforge.net/projects/phputf8
  - “*PHP UTF-8 is a UTF-8 aware library of functions mirroring PHP's own string functions. Does not require PHP mbstring extension though will use it, if found, for a (small) performance gain.*”
  - mbstring fallback powered by PCRE /u
  - Adds functions mbstring misses
- Unicode Normalization (Mediawiki)
  http://svn.wikimedia.org/viewvc/mediawiki/trunk/phase3/includes/normal/
  - Implemented in PHP and (or) PHP extension

Guard the Circle

- Check UTF-8 form early (e.g. as part of input validation) *once*
  - Validating well formedness is expensive
  - Clean or complain?
  - PHP UTF-8 has tools for this
- Normalize?
- Conversion (iconv)
  - Should only need for non-browser input

Declare

- `Header('Content-Type: text/html; charset=utf-8');`
  - For browser authoritative over meta tags
    `<meta http-equiv="Content-Type" content="text/html; charset=utf-8">`
- Forms...
  `<form accept-charset="utf-8" ... >`
  - Helps prevent users override server
  - Validate anyway...
  - Reading...
    http://ppewww.ph.gla.ac.uk/~flavell/charset/form-i18n.html
- XML...
  `<?xml version="1.0" encoding="utf-8"?>`

Configuration

- Apache: AddDefaultCharset
  - defaults to ISO-8859-1 in Apache 1.3.x
  - Disable or explicitly declare;
    - `AddDefaultCharset OFF`
    - `AddDefaultCharset utf-8`
- PHP: default_charset
  - Disable or explicity declare
    - `default_charset = "utf-8"`

HTML Entities

- With UTF-8, we don't need no stinkin' entities
  - Watch out for `htmlentities()`
  - Use `htmlspecialchars()` instead

Font Pains

- Images with text (via GD)
  - Needs a TrueType font that supports Unicode
  - http://www.slovo.info/unifonts.htm
- PDF
  - Watch this space: http://framework.zend.com/manual/en/zend.pdf.html
  - TCPDF
    http://www.tecnick.com/public/code/cp_dpage.php?aiocp_dp=tcpdf
    http://sourceforge.net/projects/tcpdf

Preparing: Code Analysis

- PHPXref: http://phpxref.sourceforge.net/
  - Identify functions on the dangerous list
    http://www.phpwact.org/php/i18n/utf-8
    http://wiki.silverorange.com/UTF-8_Notes
  - Replace with mbstring or PHP UTF-8 implementations
    - But watch for optimizations!
- Database schemas?
  - VARCHARs big enough?

Watch for Optimizations

Compile time: native string functions can be OK (assuming well formed UTF-8)...

```
<?php
header ('Content-type: text/html; charset=utf-8');
$haystack = 'Iñtërnâtiônàlizætiøn';
$needle = 'ô';

$pos = strpos($haystack, $needle);

print "Position in bytes is $pos<br>";

$substr = substr($haystack, 0, $pos);

print "Substr: $substr<br>";
```

Watch for Optimizations

Runtime: if 99%+ is in the ASCII range....

```
<?php
require_once '/path/to/utf8/utf8.php';
require_once UTF8 . '/utils/ascii.php';

if ( utf8_is_ascii($string) ) {
    # use native PHP string functions
} else {
    # use utf8_* string functions
}
```

Big Bang

- Migrate all code and data in a single shot
  - Downtime needed
  - Trial runs needed
  - Watch for cp1252 vs. ISO-8859-1
    - Use encoding detector
- Live happily ever after ...

PHP 6

- Native Unicode support
  - International Components for Unicode (ICU)
  - Reading
  - http://cvs.php.net/viewvc.cgi/php-src/README.UNICODE?view=markup
    http://www.derickrethans.nl/files/php6-unicode.pdf
    http://www.gravitonic.com/downloads/talks/oscon-2006/php-6-and-unicode_oscon
    http://www.gravitonic.com/blog/archives/000149.html

UTF-8 Survival in MySQL

- MySQL < 4.1.x
  - Defaults to ISO-8859-1
  - Poor support for UTF-8
- MySQL > 4.1.x (4.1.2 ?)
  - Getting better (but read the bug reports / use latest version)
- Find out what you've got
  - `SHOW VARIABLES LIKE 'character_set_database';`
  - `SHOW VARIABLES LIKE 'character_set_client';`
- ...and what's available;
  - `SHOW CHARACTER SET;`
  - `SHOW COLLATION LIKE 'utf8%';`

Strategy for MySQL < 4.1.x

- In general shouldn't “break” UTF-8
  - ...at least if you're using ISO-8859-1 or ASCII
- Avoid MySQL string related functions
- Expect collation (e.g. SORT BY) to be problematic for non-ASCII
- Upgrade!

MySQL 4.1.x +

- Server
  - In `/etc/my.cnf`

```
[mysqld]
 ...
default-character-set=utf8
default-collation=utf8_general_ci
```

- Database
  - `(CREATE | ALTER) DATABASE ... DEFAULT CHARACTER SET utf8`
- Table
  - (CREATE | ALTER) TABLE ... DEFAULT CHARACTER SET utf8
- Connection
  - SET NAMES 'utf8';
- Also check out the `CONVERT()` function

PHP MySQL Client

- ...defaults to ISO-8859-1
- Override (per connection) like;
  - `mysql_query("SET NAMES 'utf8'");`

MySQL Further Reading

- http://mysql.com/doc/refman/5.0/en/charset.html
- http://www.phpwact.org/php/i18n/utf-8/mysql
- “Practical i18n with PHP and MySQL”

  http://www.mysqluc.com/presentations/mysql06/winstead_practical.pdf

UTF-8 Survival in Perl

- Native support for Unicode since Perl 5.6.1
  - Best use 5.8.x +
- Uses UTF-8 internally to represent Unicode
- Scalar types flagged: utf8 or 1byte=1char
- Reading
  - http://www.ahinea.com/en/tech/perl-unicode-struggle.html (best place to start)
  - `$ man perlunintro`

Perl as Unicode hammer

- Stuff you don't want to reinvent
  - Unicode::Collate
  - Unicode::Normalize
  - Encode::Guess

Python

- Native support since 1.6
- Uses UTF-16 to represent Unicode
  - ...or optionally UTF-32
- Two different string types
  - byte (default) str = 'Hello World'
  - Unicode: str = u'Iñtërnâtiônàlizætiøn'
- Reading
  - http://trac.edgewall.org/wiki/TracDev/UnicodeGuidelines (nice intro)
  - http://downloads.egenix.com/python/Unicode-EPC2002-Talk.pdf

Python for RSS & Screen Shaping

- Universal Encoding Detector
  - http://chardet.feedparser.org/
  - Based on Mozilla implementation
- Universal Feed Parser
  - http://www.feedparser.org
- Beautiful Soup
  - http://www.crummy.com/software/BeautifulSoup/
  - Uses Universal Encoding Detector

Other Stuff / Tools

- Simredo (is a free Java Unicode editor)
  - http://www4.vc-net.ne.jp/~klivo/sim/simeng.htm
- http://intertwingly.net/stories/2006/07/04/clean_utf8_for_xml.c
  - Needs turning into a PHP extension
- UTF-8 Samples / Test Pages
  - http://www.columbia.edu/kermit/utf8.html
  - http://www.cl.cam.ac.uk/~mgk25/ucs/examples/UTF-8-test.txt
  - http://www.w3.org/2001/06/utf-8-wrong/