# What Developers Should Know about Character Sets, Unicode etc.

**Stefan Heymann**

www.destructor.de
stefan@destructor.de

# Topics

- Why?
- Fonts, Glyphs
- ASCII & Co.
- Unicode
- Application
- Firebird and Character Sets

# Why?

# Did you ever have to specify ...

- `Content-Type` in HTML?
- `encoding` in XML?
- `CHARACTER SET` in Firebird?
- `NLS_LANG` in Oracle?
- ...

# Growing Requirements

- ASCII is enough ...
- Internationalization of User Interfaces
- Internationalization of Websites
- Handling of Texts from various languages
- Development tools built (mainly) by native English speakers

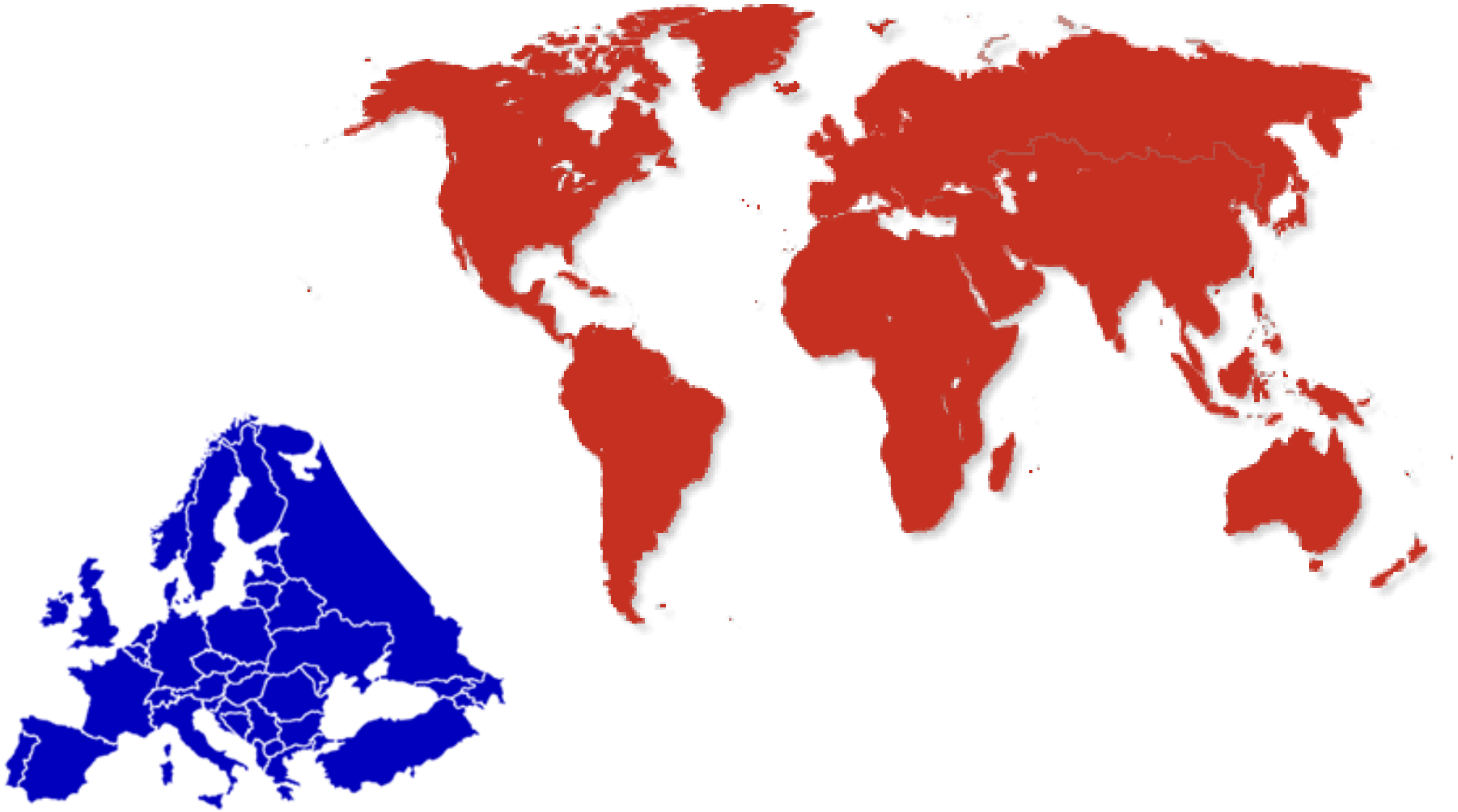
# Joel on Software

So I have an announcement to make: if you are a programmer working in 2003 and you don't know the basics of characters, character sets, encodings, and Unicode, and I catch you, I'm going to punish you by making you peel onions for 6 months in a submarine. I swear I will.

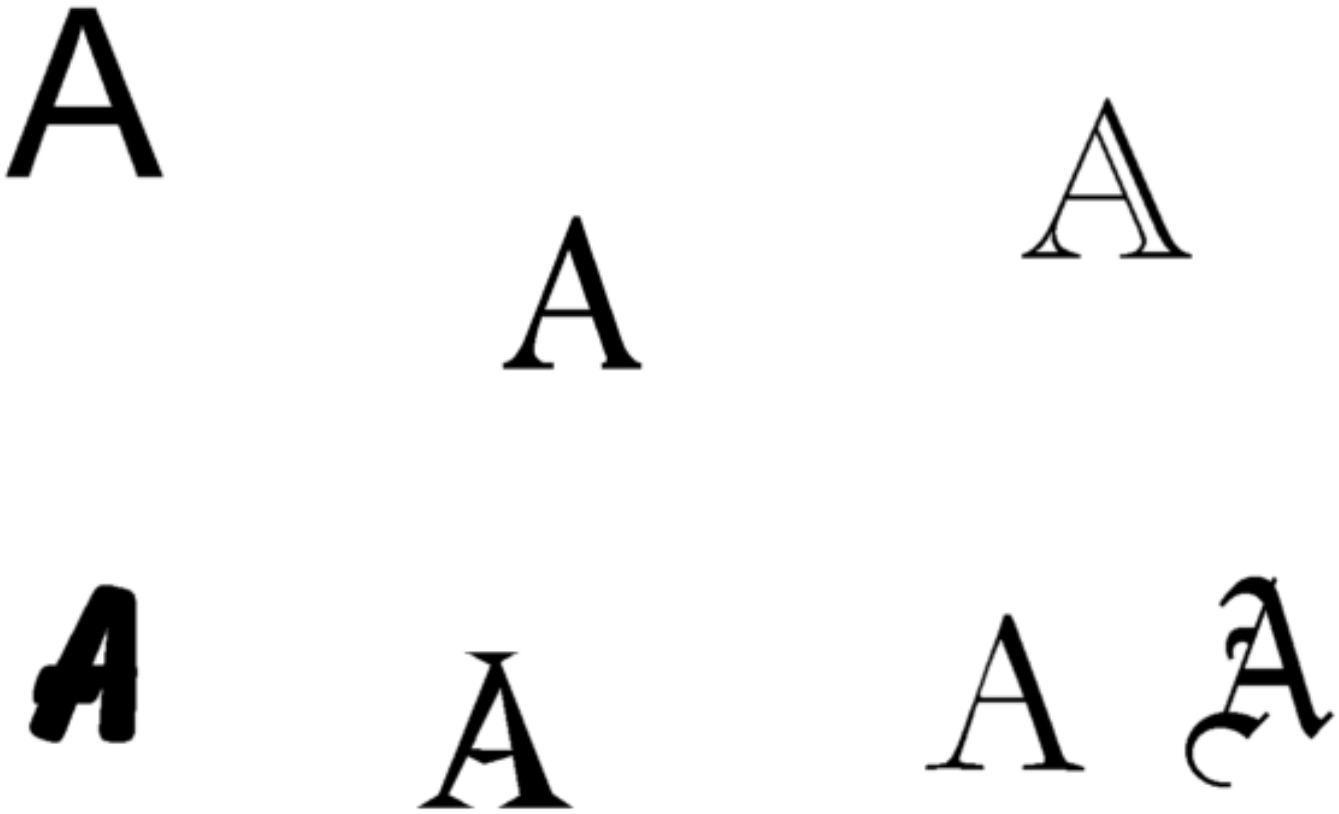
-- Joel Spolsky, www.joelonsoftware.com

# Fonts, Glyphs

# Characters of the World



# Glyphs vs. Characters



# Glyph, Character, Character Set

- Rendering of characters as glyphs is the job of the rendering machine (Postscript, GDI, TrueType, etc.)
- As developers we mostly care for processing the characters
- A character set encodes characters as numerical values
  A = 65

# Glyphs

- Not all languages display glyphs as a string of left-to-right, contiguous rectangles
- Right-to-left (Arabic, Hebrew), top-to-bottom (Japanese, Chinese)
- Several characters can „melt" into one glyph

| Keystrokes | ل | ا | لا | | غ | غ |
|---|---|---|---|---|---|---|
| Input characters | ل | ا | ل | ا | غ | غ |
| Encoded characters (byte values in hex) | 0644 | 0627 | 0644 | 0627 | 0639 | 0639 |
| Display | لاغغ | | | | | |

# ASCII & Co.

# The Mother of Character Sets

- American Standard Code for Information Interchange: ASCII, ISO 646
- 7 bits wide, characters ranging from 0 to 127
- 32 invisible control characters (NUL, CR, LF, FF, BEL, ESC, ...)
- Alphabet, Digits, Punctuation (;.-?)
- Optimized for English
- No accents and umlauts

# ASCII for Europe

- Seldom used characters get new assignment
- [=Ä \=Ö ]=Ü
- Problem: Printer and screen must have same setting
- No mixing in one text possible: „Amélie knackt gerne die Kruste von Crème Brulé mit dem Löffel“

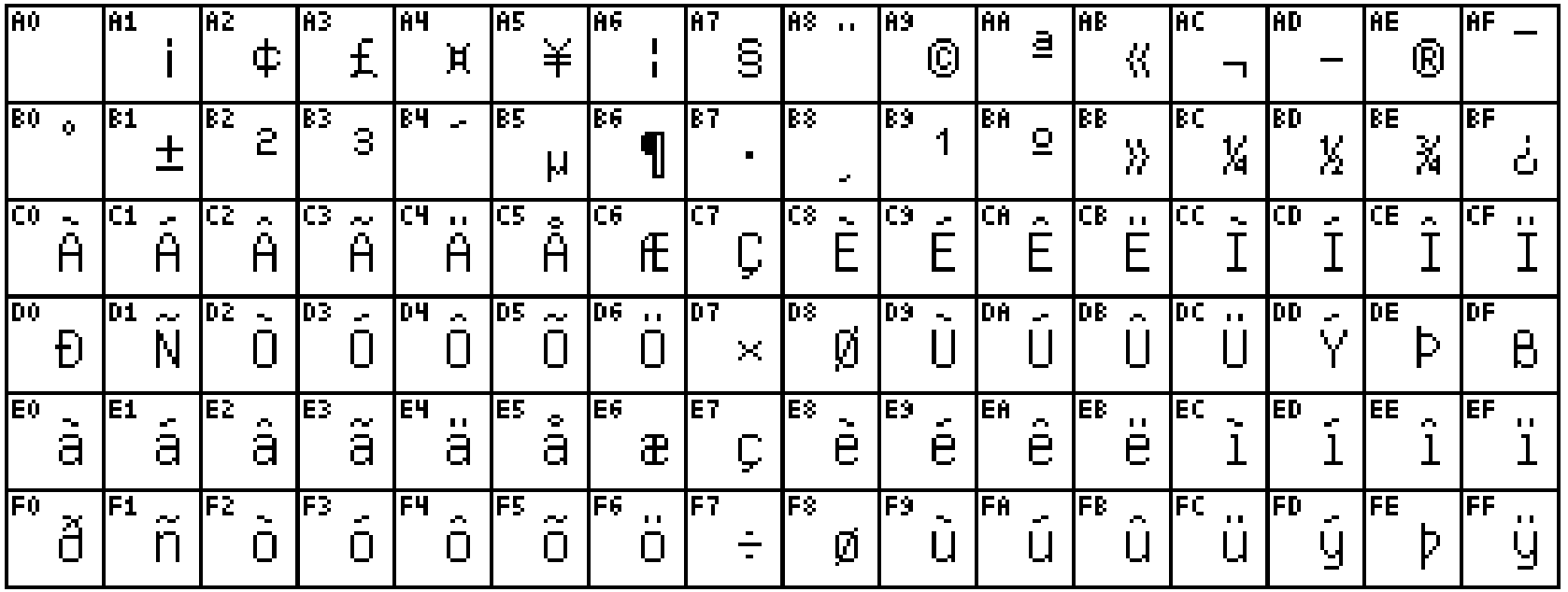
# Use the 8th Bit

- 128 new characters
- A lot of character sets have evolved
- ISO 8859-x = ASCII + 160..255
- ISO 8859-1 = Latin-1
  (Western European languages)
- Windows 1252 = ISO 8859-1 + 128..159

# ISO 8859

- Characters 0..127 identical to ASCII
- 128..159 undefined control characters
- 160..255 individually assigned

# ISO 8859-1 / Latin-1




Afrikaans, Albanese, Basque, Danish, German, English, Faroese, Finnish, French, Icelandic, Italian, Catalan, Dutch, Norwegian, Portuguese, Rhaeto-Romance, Scottish Gaelic, Schwedish, Spanish, Suaheli

*Large parts of the world – wide-spread use*

# ISO 8859-2 / Latin-2

| A0 | A1 Ą | A2 ˘ | A3 Ł | A4 ¤ | A5 Ľ | A6 Ś | A7 § | A8 ¨ | A9 Š | AA Ş | AB Ť | AC Ź | AD - | AE Ž | AF Ż |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0 ° | B1 ą | B2 ˛ | B3 ł | B4 ´ | B5 ľ | B6 ś | B7 ˇ | B8 ¸ | B9 š | BA ş | BB ť | BC ź | BD ˝ | BE ž | BF ż |
| C0 Ŕ | C1 Á | C2 Â | C3 Ă | C4 Ä | C5 Ĺ | C6 Ć | C7 Ç | C8 Č | C9 É | CA Ę | CB Ë | CC Ě | CD Í | CE Î | CF Ď |
| D0 Đ | D1 Ń | D2 Ň | D3 Ó | D4 Ô | D5 Ő | D6 Ö | D7 × | D8 Ř | D9 Ů | DA Ú | DB Ű | DC Ü | DD Ý | DE Ţ | DF ß |
| E0 ŕ | E1 á | E2 â | E3 ă | E4 ä | E5 ĺ | E6 ć | E7 ç | E8 č | E9 é | EA ę | EB ë | EC ě | ED í | EE î | EF ď |
| F0 đ | F1 ń | F2 ň | F3 ó | F4 ô | F5 ő | F6 ö | F7 ÷ | F8 ř | F9 ů | FA ú | FB ű | FC ü | FD ý | FE ţ | FF ˙ |

Central and Eastern Europe (Czech, Polish, etc.)

# ISO 8859-4 / Latin-4

| A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | AA | AB | AC | AD | AE | AF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0 ° | B1 ą | B2 ˛ | B3 ŗ | B4 ´ | B5 ĩ | B6 ļ | B7 ˇ | B8 ¸ | B9 š | BA ē | BB ģ | BC ŧ | BD Ŋ | BE ž | BF ŋ |
| C0 Ā | C1 Á | C2 Â | C3 Ã | C4 Ä | C5 Å | C6 Æ | C7 Į | C8 Č | C9 É | CA Ę | CB Ë | CC Ė | CD Í | CE Î | CF Ī |
| D0 Đ | D1 Ņ | D2 Ō | D3 Ķ | D4 Ô | D5 Õ | D6 Ö | D7 × | D8 Ø | D9 Ų | DA Ú | DB Û | DC Ü | DD Ũ | DE Ū | DF ß |
| E0 ā | E1 á | E2 â | E3 ã | E4 ä | E5 å | E6 æ | E7 į | E8 č | E9 é | EA ę | EB ë | EC ė | ED í | EE î | EF ī |
| F0 đ | F1 ņ | F2 ō | F3 ķ | F4 ô | F5 õ | F6 ö | F7 ÷ | F8 ø | F9 ų | FA ú | FB û | FC ü | FD ũ | FE ū | FF ˙ |

Northern Europe, Baltic, Greenlanic, Sami

# ISO 8859-5 / Cyrillic

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A0 | A1 Ё | A2 Ђ | A3 Ѓ | A4 Є | A5 Ѕ | A6 І | A7 Ї | A8 Ј | A9 Љ | AA Њ | AB Ћ | AC Ќ | AD  | AE Ў | AF Џ |
| B0 А | B1 Б | B2 В | B3 Г | B4 Д | B5 Е | B6 Ж | B7 З | B8 И | B9 Й | BA К | BB Л | BC М | BD Н | BE О | BF П |
| C0 Р | C1 С | C2 Т | C3 У | C4 Ф | C5 Х | C6 Ц | C7 Ч | C8 Ш | C9 Щ | CA Ъ | CB Ы | CC Ь | CD Э | CE Ю | CF Я |
| D0 а | D1 б | D2 в | D3 г | D4 д | D5 е | D6 ж | D7 з | D8 и | D9 й | DA к | DB л | DC м | DD н | DE о | DF п |
| E0 р | E1 с | E2 т | E3 у | E4 ф | E5 х | E6 ц | E7 ч | E8 ш | E9 щ | EA ъ | EB ы | EC ь | ED э | EE ю | EF я |
| F0 № | F1 ё | F2 ђ | F3 ѓ | F4 є | F5 ѕ | F6 і | F7 ї | F8 ј | F9 љ | FA њ | FB ћ | FC ќ | FD § | FE ў | FF џ |

Cyrillic (Russia, Ukraine, etc.)
More important: KOI8-R (Russian), KOI8-U (Ukrainian)

# ISO 8859-6 / Arabic

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A0 | | | | A4 ¤ | | | | | | | | AC ، | AD ـ | | |
| | | | | | | | | | | | BB ػ | | | | BF ؟ |
| | C1 ء | C2 آ | C3 أ | C4 ؤ | C5 إ | C6 ئ | C7 ا | C8 ب | C9 ة | CA ت | CB ث | CC ج | CD ح | CE خ | CF د |
| D0 ذ | D1 ر | D2 ز | D3 س | D4 ش | D5 ص | D6 ض | D7 ط | D8 ظ | D9 ع | DA غ | | | | | |
| E0 ـ | E1 ف | E2 ق | E3 ك | E4 ل | E5 م | E6 ن | E7 ه | E8 و | E9 ى | EA ي | EB ً | EC ٌ | ED ٍ | EE َ | EF ُ |
| F0 ِ | F1 ّ | F2 ْ | | | | | | | | | | | | | |

Arabic
Does not contain all characters, so it's not used often)

# ISO 8859-9 / Latin-5

| A0 | A1 ¡ | A2 ¢ | A3 £ | A4 ¤ | A5 ¥ | A6 ¦ | A7 § | A8 ¨ | A9 © | AA ª | AB « | AC ¬ | AD - | AE ® | AF ¯ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| B0 ° | B1 ± | B2 ² | B3 ³ | B4 ´ | B5 µ | B6 ¶ | B7 · | B8 ¸ | B9 ¹ | BA º | BB » | BC ¼ | BD ½ | BE ¾ | BF ¿ |
| C0 À | C1 Á | C2 Â | C3 Ã | C4 Ä | C5 Å | C6 Æ | C7 Ç | C8 È | C9 É | CA Ê | CB Ë | CC Ì | CD Í | CE Î | CF Ï |
| D0 Ğ | D1 Ñ | D2 Ò | D3 Ó | D4 Ô | D5 Õ | D6 Ö | D7 × | D8 Ø | D9 Ù | DA Ú | DB Û | DC Ü | DD İ | DE Ş | DF ß |
| E0 à | E1 á | E2 â | E3 ã | E4 ä | E5 å | E6 æ | E7 ç | E8 è | E9 é | EA ê | EB ë | EC ì | ED í | EE î | EF ï |
| F0 ğ | F1 ñ | F2 ò | F3 ó | F4 ô | F5 õ | F6 ö | F7 ÷ | F8 ø | F9 ù | FA ú | FB û | FC ü | FD ı | FE ş | FF ÿ |

Turkish

# Windows Character Sets

- Mostly congruent to ISO-8859
- Additional assignment of characters 128..159 with visible (non-control) characters
- Hyphen, typographic „quotation marks“, etc.
- Windows character sets officially registered at IANA

# windows-1252

- Congruent to ISO 8859-1
- Additional characters in the 128..159 range:

€ ‚ ƒ „ … † ‡ ˆ ‰ Š ‹ Œ Ž
‘ ’ “ ” • – — ˜ ™ š › œ ž Ÿ

€ sign since 2000

# Multi-Byte Character Sets MBCS

- Multiple Bytes per Character
- Eastern Asian Languages (CJK)
- String Length <> Length of character chain
- Making sub-strings more difficult

# Unicode

# Why Unicode?

- One single Character Set for all languages
- No code overlaps
- Hardware and OS independant
- Standardisation ISO **10**646 (vs. ASCII = ISO 646)

# Unicode

- Started with 16 Bits/Character, now 32 Bits/Char
- Possibility to code 1,114,112 characters
- Only a fraction is used
- Current version: 4.1.0
- Defines Characters, *not* Glyphs
- Practically equal to ISO/IEC 10646

# Character Definition

- Unicode defines a numerical *code point* (scalar value) and an *Identifier* for every character

0041 LATIN CAPITAL LETTER A
00E4 LATIN SMALL LETTER A WITH DIAERESIS
0391 GREEK CAPITAL LETTER ALPHA
05D0 HEBREW LETTER ALEF
0950 DEVANAGARI OM
1D56C MATHEMATICAL BOLD FRAKTUR CAPITAL A

# Unicode Code Points

- Codespace: 0..10FFFF
- Usual Notation: hexadecimal, with preceding „U+“, at least 4 digits
- U+0020
- U+0041
- U+1D56C

# Unicode Character Names

- Consisting of the uppercase letters A..Z, digits 0..9, hyphen (-) and whitespace.
- BYZANTINE MUSICAL SYMBOL LEIMMA ENOS CHRONOU
- DESERET CAPITAL LETTER OW
- BRAILLE PATTERN DOTS-1245

# Unicode Coding

- Storage of Code Points in memory
- 32 Bits/Character too fat
- There are various codings around:
  - 8-Bit (UTF-8)
  - 16-Bit (UCS-2, UTF-16)
  - 32-Bit (UCS-4, UTF-32)
  - UTF-7, UTF-1, PunyCode, ...

# UCS-2

- 16 Bits per Character
- Code Area: 0000..FFFF
  = Basic Multilingual Plane (BMP)
- Since Unicode 3.1, not all characters can be coded
- Replaced by UTF-16
- „Unicode“ often used as a synonym for UCS-2
  (which is wrong and can lead to misunderstanding)

# UTF-16

- 16 Bits per Character
- All characters > FFFF must be coded as a „Surrogate Pair“ and occupy 2 contiguous 16-Bit words
- Complete Code Space can be coded
- Difficult to calculate string length or substrings

# Endianness

- Problem with UCS-2, UTF-16: Low/High-Byte ordering
- Differentiate in UTF-16BE and UTF-16LE in metadata
- Byte Order Mark (BOM) U+FEFF
- U+FEFF set to the very beginning of each text
- U+FFFE is (and will be) an invalid code point

# UCS-2 vs. UTF-16

- UTF-16 ist backwards compatible
- UCS-2 often synonym for „Unicode“
- WideString, wchar_t
- NT3, NT4: UCS-2
- Since Windows 2000: UTF-16

# UTF-8

- Coding as 8-Bit strings
- US-ASCII characters untouched, all others need 2 to 4 contiguous bytes
- Complete codespace can be coded
- Advantage: „Latin“ texts quite compact
- Problem: string length, substrings, etc.

# UTF-8 coding

- US-ASCII characters untouched,
  Bit 7 reset `0xxxxxxx`
- All others are sequences of bytes with Bit 7 set
  `1xxxxxxx`
- First Byte: As many leading bits set as length of sequence: `110xxxxx`
- Following Bytes: Bit 7 set, Bit 6 reset: `10xxxxxx`

# UTF-8 coding

- You can determine the type of each byte:
  - Complete character: 0xxxxxxx
  - Part of a sequence: 1xxxxxxx
  - Sequence Start: 11xxxxxx
  - Sequence body: 10xxxxxx

# UTF-8 coding

- Bits after the type bits remain for coding the Coding Point

ä – LATIN SMALL LETTER A WITH DIAERESIS

$00E4_{16} = 11100100_2$

| |
|---|
| 00000 – 00007F: 0xxxxxxx |
| 00080 – 0007FF: 110xxxxx 10xxxxxx |
| 00800 – 00FFFF: 1110xxxx 10xxxxxx 10xxxxxx |
| 10000 – 1FFFFF: 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx |

```
110xxxxx 10xxxxxx
   ooo11   100100
-------- --------
11000011 10100100 bin
C3       A4       hex
195      164      dez
Ã        ¤        Latin-1
```

# UTF-8 with ISO rendering

JÃ¼rgen Klinsmann
(German Soccer Coach)

# UCS-4, UTF-32

- 32 Bit per Character
- Every code point as one word in memory
- Fat but handy (1 word = 1 character)
- Endianness issues (UTF-32BE, UTF-32LE)
- Complete codespace can be coded
- No practical differences between UCS-4 and UTF-32

# What is Plain Text?

- For every string, for every text (file, e-mail, attachment, download, etc) the encoding must be known.
- Plain text can begin with a BOM

> There Ain't No Such Thing As Plain Text.
> -- Joel Spolsky

# Encodings Overview

- ISO 8859-x
- Windows-12xx
- KOI8-R, KOI8-U
- Shift-JIS
- etc.
- Unicode: UTF-8, (UCS-2), UTF-16, UTF-32
- There is *no* „Unicode“ encoding!

# Unicode Specials

Skip

# Character Ranges

- General Scripts (Latin, Greek, Cyrillic, etc.)
- Symbols (Symbol, Dingbat, Punctuation, Math, etc.)
- CJK Phonetics and Symbols
- CJK Ideographs
- Yi Syllables, Hangul Syllables
- Surrogate Area, Private Use Area
- Compatibility and Specials

# Control Characters

- 0000..001F: ASCII control characters
- 007F: ASCII Delete/Rubout
- 0080..009F: Control characters

# Combined Characters

- There are 2 ways to code an ä:
- 00E4: LATIN SMALL LETTER A WITH DIAERESIS
- 0061: LATIN SMALL LETTER A +
  0308: COMBINING DIAERESIS
- You can „compose“ characters/glyphs using Combining Characters

# Normalization

- Problem: You cannot compare text byte-for-byte
- Normalisation/Canonicalisation required
- Translation tables
- Composition, Decomposition
- Technical Report #15: Unicode Normalization Forms

# Bi-directionality

- Unicode defines how characters are arranged in memory
- Some languages go from right-to-left
- E.g. Arabic, Hebrew
- Rules and control characters for left-to-right and right-to-left text

# Transcoding

- Transform from special character sets to Unicode and back
- Windows-1252 –> Unicode –> ISO 8859-1
- Translation tables: www.unicode.org
- Characters can get lost (ك becomes ?)
- Characters can get transformed (ç becomes c)

# Surrogate Pairs for UTF-16

- For coding characters > U+FFFF
- Code Points D800..DFFF reserved
- Two contiguous 16-Bit words used: High and Low Surrogate = Surrogate Pair
- High Surrogate: U+D800..U+DBFF
- Low Surrogate: U+DC00..U+DFFF

# Surrogate Pairs for UTF-16

- You can see if a 16-Bit word ...
  - is a non-surrogate character
  - belongs to a surrogate pair
  - is the high or low surrogate

# Sorting

- Sorting rules also apply for searching
- There are cultural differences
  - treat ä like a
  - treat ä like ae
  - treat ä as a seperate character after z
- Unicode defines algorithms and delivers tables for sorting

# Case Mappings

- Not available in every language
- Not always reversible
- Turkish: ı –> I, i –> İ
- No necessarily 1:1 ß –> SS
- Case Mappings are language depending
- Unicode Technical Report #21
  „Case Mappings“

# The Unicode Standard

- Unicode Consortium www.unicode.org
- ISO 10646
- Book: The Unicode Standard, Version 4.0
  ISBN 0-321-18578-1
- All chapters and files, complete character database available for free at www.unicode.org
- Technical Reports

# Application

Skip

# Windows Notepad

- Save as ... > Encoding
  - ANSI (= Windows 1252)
  - Unicode (= UCS-2 or UTF-16)
  - Unicode Big Endian
  - UTF-8

# XML

- Every XML entity is Unicode by definition. Default is UTF-16 or UTF-8.
- <?xml version="1.0" encoding="utf-8" ?>
- With *every* encoding, every Unicode character can be inserted as a *Character Reference*
  - € €
  - ॐ ॐ „DEVANAGARI OM"

# HTML

- Unicode aware since HTML 4.0
- Character References like XML
- <head>

```
<meta http-equiv="Content-Type"
      content="text/html; charset=utf-8">
```

# Windows API

- Win32 API is Unicode aware from the beginning
- All string function also available as „W“ function
- UCS-2 up to Windows NT 4.0
- UTF-16 since Windows 2000

# Borland Delphi

- VCL controls are not Unicode aware (local character sets usable)
- Troy Wolbrink: Tnt Delphi Unicode Controls http://tnt.ccci.org/
- Mike Lischke (Virtual Treeview, etc.) http://www.delphi-gems.com/
- JCL: JclUnicode.pas
- Delphi Roadmap: VCL controls Unicode aware some time in the future

# Oracle

- Character set defined when database is created
- Client can have other characters set, automatic transliteration
- Environment variable NLS_LANG
- WE8MSWIN1252, WE8ISO8859P1, UTF8, etc.

# Firebird and Character Sets

# Character Sets

- Default Character Set defined at database create time:

```
CREATE DATABASE 'employee.gdb'
DEFAULT CHARACTER SET ISO8859_1;
```

- Different Character Set for every table column

```
CREATE TABLE users (
czech_name VARCHAR(50) CHARACTER SET ISO8859_2
```

- Default Collation (sort order) for every table column

```
CREATE TABLE test (
  name VARCHAR(50) COLLATE DE_DE);
```

- Available collations depending on character set used

# Collations

- Define sort ordering
- There is no default collation for the database
- COLLATE DE_DE: German sorting, applying the rules for Germany
- COLLATE FR_CA: French sorting, applying the rules for Canada

```
ORDER BY LASTNAME COLLATE FR_CA, FIRSTNAME COLLATE FR_CA
WHERE LASTNAME COLLATE FR_CA = :lastnametosearch
```

# UPPER()

- The UPPER() function only works correctly if there is a collation defined for the parameter field
- Situation will improve with Firebird 2.0 (which will also bring a new LOWER() function)

# Local Character Set

- Client side character set can be different from server side character set. Automatically transliterated (sort of ;-)
- Conversions between character sets are always done CHARSET1 -> UNICODE -> CHARSET2
- With NONE or OCTETS bytes are just copied:
  NONE/OCTETS -> CHARSET2 and CHARSET1 -> NONE/OCTETS.
- IB Objects
  Ib_Connection1.CharSet := 'ISO8859_1';
- IBX:
  IbDatabase1.Params.Add ('lc_ctype=ISO8859_1');

# Special Character Sets

- **NONE:** No character set applied. With this character set setting, Firebird is unable to perform conversion operations like UPPER() correctly on anything other than the standard 26 latin letters.
- **OCTETS:** Same as NONE. Cannot be used as client connection character set. Space character is #x00
- **ASCII:** US-ASCII

# „Normal“ Character Sets

- **ISO8859_*x*** (e.g. ISO8895_1, ISO8859_2)
  Collations: DE_DE, FR_FR, CS_CZ, etc.
- **WIN125*x*** (e.g. WIN1252, WIN1250)
  Collations: WIN1252, WIN_PTBR, PXW_CSY, etc.
- **DOS*xxx*** (e.g. DOS850, DOS852)
  Collations: DB_DEU850, PDOX_CSY

# Other Character Sets

- BIG_5: Chinese
- CYRL, KOI8-R, KOI8-U: Russian, Ukrainian
- KSC_5601: Korean
- SJIS_0208: Japanese
- EUCJ_0208: Japanese
- GB_2312: Chinese

# Unicode „Character Sets“

- **UNICODE_FSS:** Unicode UTF-8. Old implementation that accepts malformed strings and does not enforce correct max. string length. *DEPRECATED.*
- **UTF8:** New in Firebird 2.0
  - Collation **UCS_BASIC:** sorts in Unicode code-point order
  - Collation **UNICODE:** Sorts using the Unicode Collation Algorithm (UCA)

# Which one to select?

- DOS, dBASE and Paradox only for legacy support
- WINxxx is extension of corresponding ISOxxx, but may lead to problems on non-Windows systems.
- ISOxxx is missing a few characters of WINxxx (e.g. typographic dash signs) –> prepare to handle this
- You don't care much about character set issues? And you work for one platform anyway? Select the appropriate ISOxxx character set. It will store everything correctly (you don't care about typographic dashes, do you ;-) -- and you have the best selection of collations.

# Links

- Unicode specification, tables, articles: www.unicode.org
- My Firebird Website (with an article about and a complete listing of Firebird character sets): www.destructor.de/firebird

# Thank You!

**Stefan Heymann**

www.destructor.de
stefan@destructor.de