Workshop: Unicode and UTF-8

Jonas Sternisko

April 12, 2018

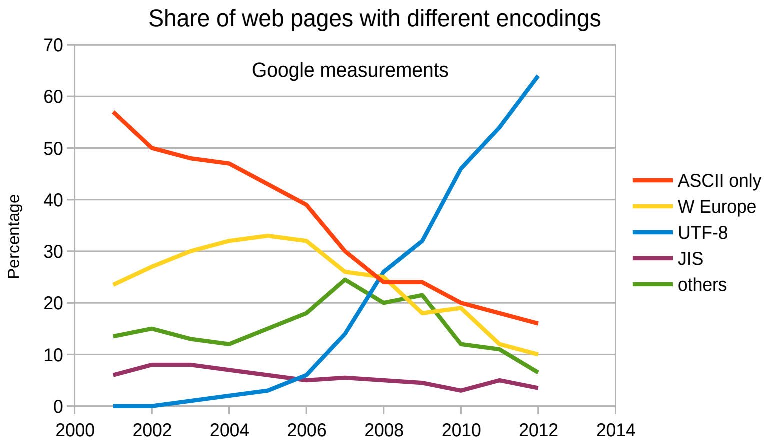
ADITION technologies AG

Representing Text with Unicode and UTF-8

Outline

1. Unicode
2. UTF-8

Why does this matter?



Unicode

- Character set like ASCII, ANSI Latin1, Windows-1252, ...
  - Ordered set of characters
- Contains characters of all written languages
- Contains 1,112,064 code points
- Examples
  - Letter A: has codepoint 65, written as U+0041 (hexadecimal)
  - Letter a: U+0061
  - Symbol $\sum$: U+01A9
  - Symbol €: U+20AC

UTF-8

- Encoding scheme for unicode (Universal character set Transformation Format)
  - Prefix-free code
  - Variable-Byte encoding
  - See workshop on lossless compression
  - Every unicode codepoint is represented by 1 to 4 Bytes (hence the “8”)
- Example letter “A” in binary: 0100 0001

UTF-8 Encoding Scheme

U+0000 - U+007F	0xxxxxxx			
U+0080 - U+07FF	110xxxxx	10xxxxxx		
U+0800 - U+FFFF	1110xxxx	10xxxxxx	10xxxxxx	
above	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx

- Leading byte encodes number of bytes respresenting the code point
- Continuation bytes always start with 10

UTF-8 Examples

A	0100 0001		
$\sum$	1100 0110	1010 1001	
€	1110 0010	1000 0010	1010 1100

Compatibility

- Unicode vs. ASCII
  - Codepoints are equal
- ANSI Latin1 et al.
  - ???
- UTF-16 and UTF-32
  - What are these anyway?

Take-home message

There Ain't No Such Thing As Plain Text.

- Supply encoding when sending text.

Further reading

Sources

- https://en.wikipedia.org/wiki/UTF-8
- http://www.fileformat.info/info/unicode/char/20aC/index.htm
- https://www.joelonsoftware.com/2003/10/08/
  the-absolute-minimum-every-software-developer-absolutely-positively-mu