CSCC09F
Programming on the Web

i18n and Unicode

extending the Web to all
written languages

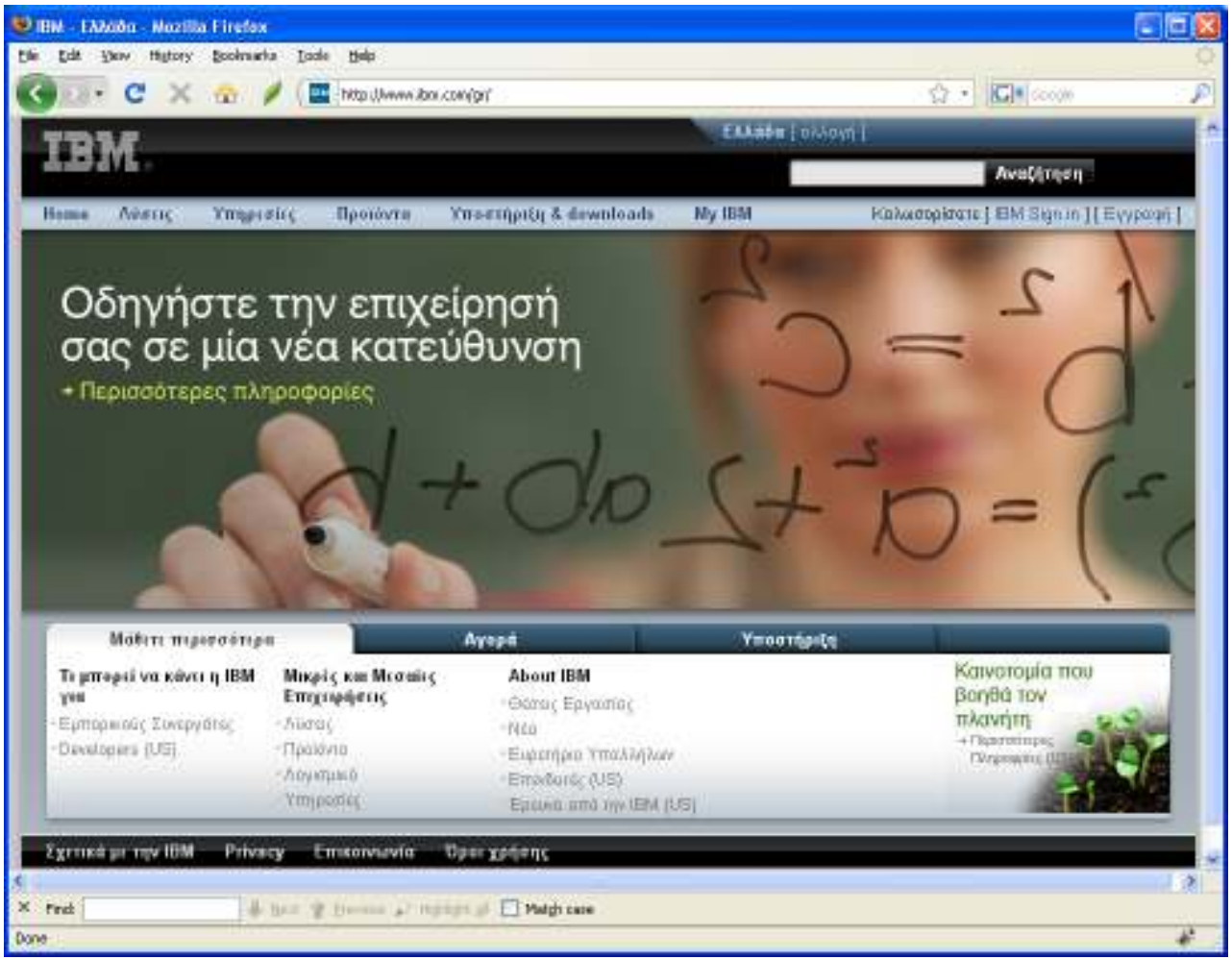
Internationalization

- 'i18n' for short (because we like acronyms!)
- deals with:
  - character sets (“code-points”)
  - text direction
  - local formats (dialects)
  - number formats (e.g. “.” vs “,”)
  - word-break rules (hyphenation conventions)
  - sorting order
  - calendars and dates
- the Web must be i18n, why?

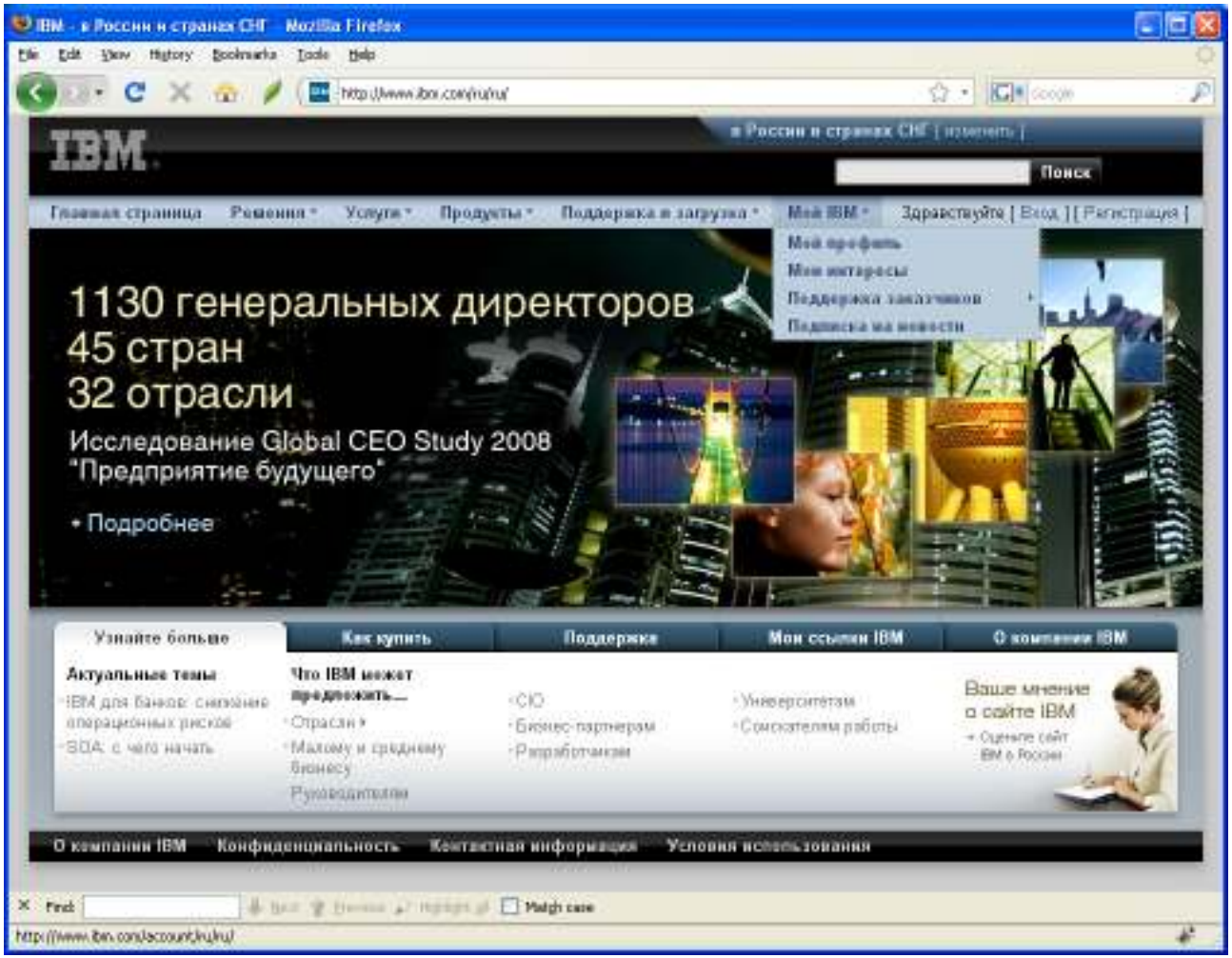
Internationalization (i18n)

- want worldwide reach for products and services
- increase user/customer satisfaction, which in turn can increase business
- improve quality of customer support communication, in turn reducing cost
- open new markets for goods and services, where English does not meet local needs
- decreased expenses: develop once, deploy in multiple markets

Internationalization (i18n)



Internationalization (i18n)



Internationalization (i18n)

Internationalization (i18n)

Language Transliteration

- அலன் (Tamil)
- অ্যালেন (Bengali)
- Алэн (Russian)
- 阿蘭 (Chinese)
- Алан (Mongolian)
- آلان (Arabic)
- אלן (Hebrew)
- एलन (Nepali)
- Αλαν (Greek)
- アラン (Japanese)

Character Encodings

- Character Set:
  - a collection of characters grouped together
    - code point ↔ abstract character
      - e.g. the lower-case letter 'n' as used in English
- Character Encoding:
  - a way of encoding code points into a byte stream
    - byte stream ↔ code points
- Font
  - a collection of character "pictures"
    - code point ↔ glyph

Single-Byte Character Encoding

- ASCII
  - American Standard Code for Information Interchange
    - 7-bit code (0-127), why 7 bits?
    - because it was designed in the 1960's to work on systems (h/w) with 7-bit bytes, and for 8-bit bytes with one bit as a parity (error checking) bit
  - most computers today use 8-bit bytes, opening up an additional 128 values for use as character encodings

Single-Byte Character Encoding

- ISO-8859 Series
  - A set of 8-bit code-points backward compatible with ASCII
    - 96 extra character slots (other 32 slots for control codes)
  - ISO-8859-0 through ISO-8859-15 (new default)
    - e.g., ISO-8859-1 / Latin 1
      - Afrikaans, Albanian, Basque, Catalan, Danish, Dutch, English, Faroese, Finish, French, German, Icelandic, Irish, Italian, Norwegian, Portuguese, Rhaeto-Romanic, Scottish, Spanish, Swahili, Swedish
      - Default code set for HTML
  - ISO-8859-6 (Arabic) and ISO-8859-7 (Greek) do not fully handle all their respective languages

ISO-8859 Series

- For all members of the series, code-points 0-127 identical to US-ASCII
- 128-159 for less-used control characters
  - ISO-6429
- Upper portion of code-points, 160-255, differ for each variant (Latin, Cyrillic, Arabic, etc.)
- Good information at

  http://czyborra.com/

ISO-8859-1 (Latin1)

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 Ð	D1 Ñ	D2 Ò	D3 Ó	D4 Ô	D5 Õ	D6 Ö	D7 ×	D8 Ø	D9 Ù	DA Ú	DB Û	DC Ü	DD Ý	DE Þ	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 ð	F1 ñ	F2 ò	F3 ó	F4 ô	F5 õ	F6 ö	F7 ÷	F8 ø	F9 ù	FA ú	FB û	FC ü	FD ý	FE þ	FF ÿ

Latin1 is also represented as the first 256 code points of ISO-10646 (Unicode)

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

ISO-8859-6 (Arabic)

A0				A4 ¤								AC ،	AD ـ		
											BB ػ				BF ؿ
	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 ْ													

Multi-byte Code Sets

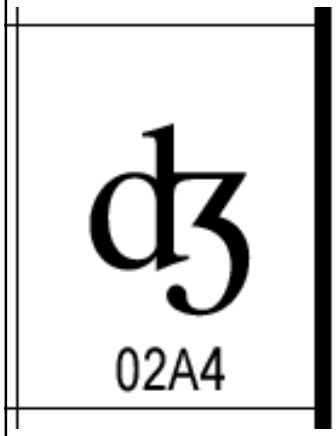
- Not every language can be shoehorned into 256 code-points (8-bit rep)
  - e.g. languages such as Chinese, Japanese, and Korean (CJK) require tens of thousands of code-points
  - also desireable to represent multi-language documents – thus need to have enough code points to segregate languages.
- 32-bits could handle up to 4 billion characters
  - more than sufficient, but … don't want to impose this space inefficiency except where needed – not a reasonable generic standard
- Is there a happy medium between these?

The Unicode Standard

- Defines a code-point set with a fixed-width, 16-bit character-encoding scheme
  - characters from all the world's major scripts are uniformly supported
    - can combine Arabic, French, Japanese, and Russian characters all in the same string/document
    - even musical symbols, mathematics, historical languages, aboriginal languages, ecclesiastical scripts, …
  - 65,536 unique slots (16-bits)
  - note, total estimated # characters introduced since the dawn of writing ≈ 500,000 (more on this later)
  - for ease of interoperability, first 128 Unicode code point values are assigned to match ASCII, first 256 Unicode code point values match ISO-8859-1 (Latin-1)

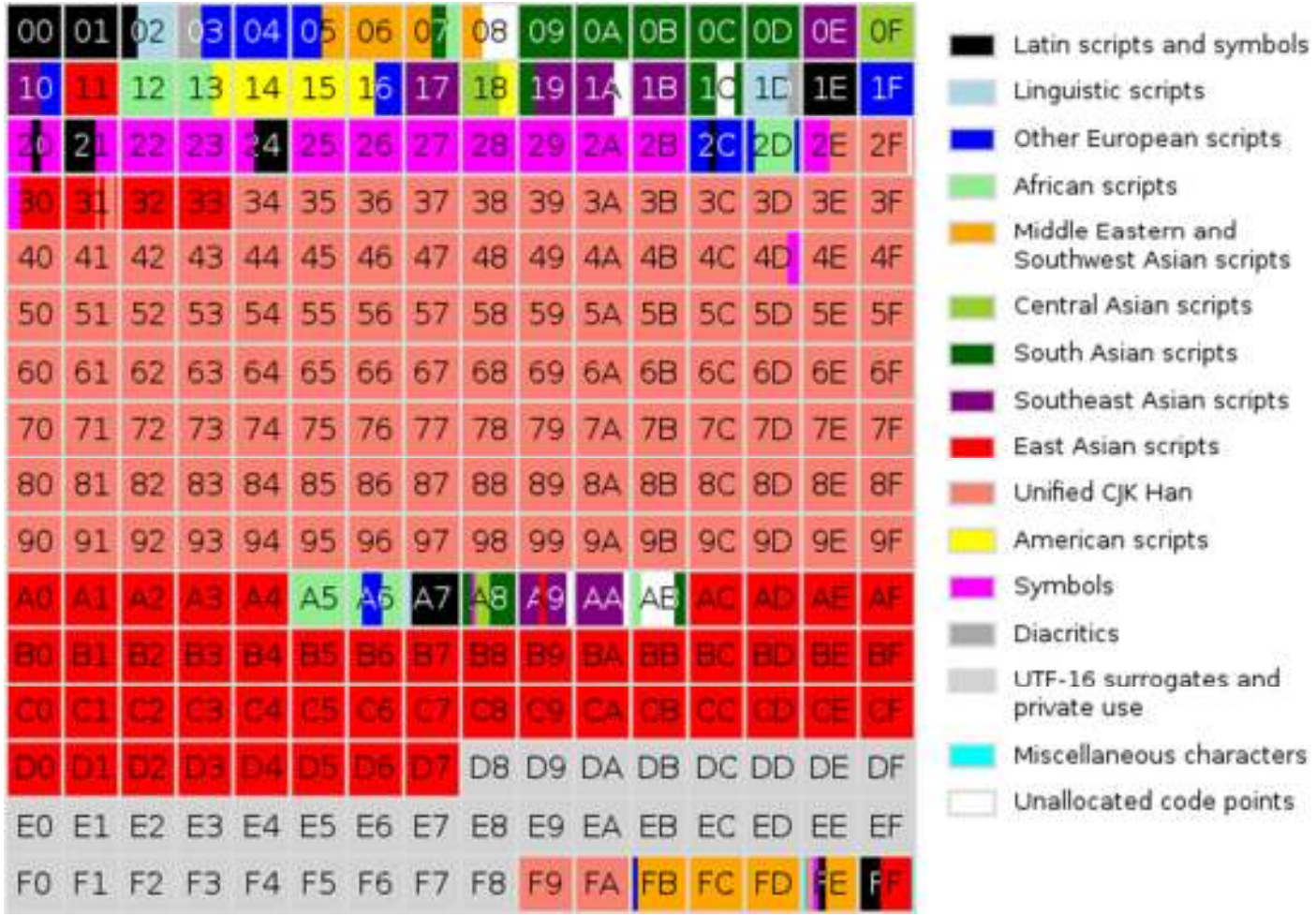
The Unicode Specification

- http://www.unicode.org/
- e.g. 025F
  - LATIN SMALL LETTER DOTLESS J WITH STROKE
    - voiced palatal stop
    - typographically a flipped f, but better thought of as a form of j
    - "gy" in Hungarian orthography
    - also archaic phonetic for palatoalveolar affricate 02A4



- See www.unicode.org/charts for code charts including all encoded languages

Unicode Code-point Distribution



Unicode Design Principles

- 16-bit characters (mostly)
- non-modal – no notion of being in certain state or language mode
- characters, not glyphs (a glyph is the visual presentation of one or more characters)
- Semantic properties associated with character
  - map lower to upper
  - maps mirrored character ( ), { }, etc
  - indicates numeric values
  - indicates directionality
  - sample glyph, …

Unicode Encoding Methods

- UTF-16 (UCS Transformation Format), UCS-2
  - default encoding method; straightforward
  - suited for storing UNICODE in memory (easy access)
- UTF-7
  - modal encoding method that uses only first 7-bits of a byte – introduced to cope with 7-bit apps like e-mail
- UTF-8
  - 8-bit, variable-length encoding
    - 1, 2, or 3 bytes could represent a UNICODE code element
    - ASCII compatible, mandated for use by Internet protocols
    - space efficient for Euro-text but costs 50% more for Asian
- UTF-32, UCS-4
  - Simplest, easy to access on 32-bit machines: 32-bit integer per Unicode character, inefficient for Latin/ASCII

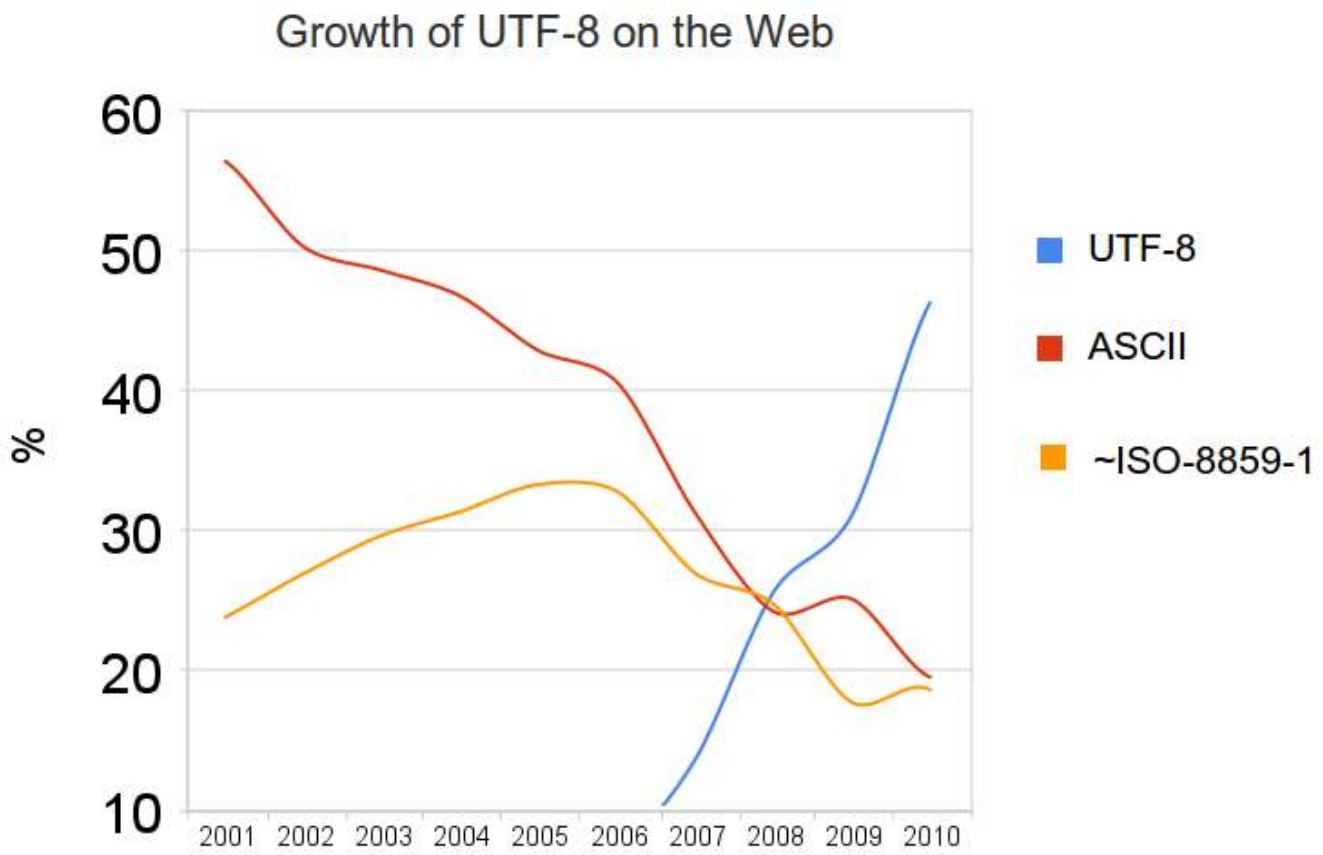
UCS-2 in UTF-8

UCS-2 Encoding Range	UTF-8 Bit Patterns
\u0000 - \u007F	0xxxxxxx
\u0080 - \u07FF	110xxxxx 10xxxxxx
\u0800 - \uFFFF	1110xxxx 110xxxxx 10xxxxxx

Choosing an Encoding

- Commonly used character encodings on the Web:
  - ASCII
  - ISO-8859-1 (Latin-1)
  - ISO-8859-n (e.g. n=5 supports Cyrillic)
  - UTF-8
  - UTF-16
  - SHIFT_JIS (a Japanese encoding)
  - EUC_JP (another Japanese encoding)
- Conforming UA's must correctly map supported encodings to UNICODE code-points

Character Encodings on the Web



18 - Unicode CSCC09 29

Specifying Document Character Encoding

- In the HTTP header

```
Content-Type: text/html; charset=EUC-JP
```

  - server must somehow determine encoding of document

- To address server or config limitations, HTML documents may include explicit info in meta

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

  - may only be used where the character-encoding is ASCII compatible (at least until the META element is parsed)

- For certainty, may include a 'charset' attribute on an element that designates an external resource (e.g. <a> and <link> elements)

Character References

- Numeric (decimal or hex)
  - &#D; refers to UNICODE char decimal D
  - &#xH; refers to UNICODE char hex H
    - &#229 = &#xE5 = &#Xe5 = å
- Character Entity References (defined in XML DTD)
  - symbolic names rather than code positions
  - small subset of char's predefined as "entities", e.g.,
    - **<** (useful to escape a <)
    - ** ** (a non-breaking space)
    - **©** (copyright symbol ©)
  - can define own, e.g. **<!ENTITY euro "&x20AC;">**

Example

- Example file iso8859.html inserts a symbol in 3 ways:
  - ¾
  - ¾
  - the raw byte value 190 inserted into the file
- The character encoding should only effect the way the last character is displayed.
- iso8859.html
  - change encoding to Cyrillic (ISO-8859-5) to get a big O
  - change encoding to Greek (ISO-8859-7) to get a big Y

Unicode Surrogate Pairs

- Unicode goal is to represent all language chars, but have only 16-bit code space, how does this work?
- 2048 unused slots in the Unicode hex range D800-DFFF are reserved to accommodate the overflow characters
- provide room for 1,048,576 (less commonly used) extra characters via "surrogate pairs"
  - Unicode 3.1 assigns more than 40,000 characters to the surrogate pair coding
- D800-DBFF followed by DC00-DFFF
  - no meaning in isolation (unpaired)