

# Unicode for Slavic Medievalists

David J. Birnbaum (Pittsburgh, USA)

djbpitt+@pitt.edu

Ralph Cleminson (Portsmouth, UK)

ralph.cleminson@port.ac.uk

Pomorie, Bulgaria, September 2002

Part 1

# Outline, Part 1

- Bits, bytes, and characters
- Script and Alphabet
- Characters and Glyphs
- Unicode Basics
- Why Unicode?
- (Break)

# Outline, Part 2

- Lumpers and Splitters
- Early Cyrillic Writing and Unicode
- The Text Encoding Initiative (TEI) Writing System Declaration (WSD)
- What's New: Variation Selectors and Early Cyrillic Writing

# References

- Unless otherwise noted, all textual citations and reproductions of charts and diagrams are taken from *The Unicode Standard, Version 3.0*, Reading: Addison-Wesley, 2000.
- Available on-line at:  
<http://www.unicode.org/>

# Bits, Bytes, and Characters

- A computer doesn't know what “a” or “я” is
- Information in a computer is a sequence of *bits*
- Bits have only two possible values: 0 or 1
- To distinguish more than two things, use sequences of bits
- Two bits can make four ( $2^2$ ) distinctions  
00, 01, 10, 11
- Three bits can make eight ( $2^3$ ) distinctions  
000, 001, 010, 011, 100, 101, 110, 111
- Etc.

# Bytes

- Seven bits can make 128 ( $2^7$ ) distinctions
  - American Standard Code for Information Interchange (ASCII)
  - English
- Eight bits (one *byte*) can make 256 ( $2^8$ ) distinctions
  - ISO 8859-1 (Western European Latin)
  - KOI-8, CP 1251, ISO 8859-5 (all English + Cyrillic)
- By convention:  $256 - 33 = 223$  characters (or fewer)

# ISO 8859-1

	000	001	002	003	004	005	006	007
0			0	@	P	`	p	
1		!	1	A	Q	a	q	
2		"	2	B	R	b	r	
3		#	3	C	S	c	s	
4		\$	4	D	T	d	t	
5		%	5	E	U	e	u	
6		&	6	F	V	f	v	
7		'	7	G	W	g	w	
8		(	8	H	X	h	x	
9		)	9	I	Y	i	y	
A		*	:	J	Z	j	z	
B		+	;	K	[	k	{	
C		,	<	L	\	l		
D		.	=	M	]	m	}	
E		.	>	N	^	n	~	
F		/	?	O	_	o		

	008	009	00A	00B	00C	00D	00E	00F
0			◊	À	Ð	à	ð	
1		ı	±	Á	Ñ	á	ñ	
2		¢	²	Â	Ò	â	ò	
3		£	³	Ã	Ó	ã	ó	
4		¤	´	Ä	Ô	ä	ô	
5		¥	µ	Å	Õ	å	õ	
6		¦	¶	Æ	Ö	æ	ö	
7		§	·	Ç	×	ç	÷	
8		¨	,	È	Ø	è	ø	
9		©	ı	É	Ù	é	ù	
A		ª	º	Ê	Ú	ê	ú	
B		«	»	Ë	Û	ë	û	
C		¬	¼	Ì	Ü	ì	ü	
D		­	½	Í	Ý	í	ý	
E		®	¾	Î	Þ	î	þ	
F		¯	¿	Ï	ß	ï	ÿ	

# What if 256 Aren't Enough?

- Use multiple sets in the same document
- Not supported in 8-bit web pages
- Require *shift* or *escape* sequences
  - Stateful
  - How do you search?

# How about 65536?

- Unicode characters are 16 bits wide
- Sixteen bits can make 65536 ( $2^{16}$ ) distinctions
- “Surrogate pairs” increase the inventory still further
- Technical documentation may refer to characters by their numeric value ...

# Why Does “F” Think It’s a Number?

- Binary digits can be 0 or 1
- Decimal digits can be: 0 1 2 3 4 5 6 7 8 9
- Hexadecimal digits can be:  
0 1 2 3 4 5 6 7 8 9 A B C D E F
- Thorn (b) =
  - Decimal 254 (2 x 100 + 5 x 10 + 4 x 1)
  - Hexadecimal 00fe  
(0 x 4096 + 0 x 256 + 15 x 16 + 14 x 1)
  - Binary 00000000 11111110

# Why Is This a Good Thing?

- In an 8-bit world, decimal 254 may represent at various times Latin, Cyrillic, or other characters
- In a 16-bit world, no numerical value can have more than one meaning
- Supported simultaneously in a single web page
- Does not require shift or escape sequences

# Script and Alphabet

- Script
  - Cyrillic, Latin, Greek
  - Cultural Concept
- Alphabet
  - Russian, Bulgarian, Swedish, Greek
  - Early Cyrillic / Church Slavonic?
    - Different manuscripts use different inventories

# Greek, Coptic, and Cyrillic

- “The Greek script is used for writing the Greek language and (in an extended variant) the Coptic language.”
- “The Cyrillic script is a member of the family of scripts strongly influenced by the Greek script.”

# Same script but different ...

- Alphabet (modern Bulgarian, modern Serbian, modern Russian, pre-1918 Russian, etc.)
- Language (Bulgarian, Serbian, Russian, etc.)
- Orthography
  - And different languages (Russian, Bulgarian, Serbian, etc.)
  - But same language (modern orthographic reforms)
  - But same language? (Church Slavonic recensions)

# Script, not Language

- Unicode characters have script properties, but not language properties
- Language is specified in a higher-order protocol
  - language tagging (e.g., `xml:lang` attribute)
  - out of band
- Language information may be needed for case-mapping, collation, hyphenation, spell-checking, text-to-speech, fine typography, but not for legible (even if culturally-unexpected) plain text graphic rendering

# Terminology

- Text Element
- Text Process
- Character
- Grapheme
- Glyph

# Text Element and Text Process

- “The text elements in a given language depend upon the specific text process; a text element for spell-checking may have different boundaries from a text element for sorting purposes.”
- “... for each text process, written languages differ in what is considered a fundamental unit of [written] text, or *text element*.”

# Character

- “a member of a set of elements used for the organization, control, and representation of data”
- “fundamental unit of encoding”

# Graphemes and Characters

- “A *grapheme* is a minimal unit of a writing system, just as a phoneme is a minimal unit of a sound system. In some instances a grapheme is represented by more than one Unicode character ... .”
- “An important class of text elements is called a *grapheme*, which typically corresponds to what a user thinks of as a ‘character’.”
- “An abstract character does not necessarily correspond to what a user thinks of as a ‘character’ and should not be confused with a *grapheme*.”

# One Grapheme or Two?

- Swedish, German, and English are all based on Latin script
- Swedish *ö* is an independent character with its own sort position (at end of alphabet)
  - Different grapheme from *o*
- German *ö* is equivalent to *oe* and is sorted with it
- English *ö* is a variant of *o*, and is sorted with it

# One Character or Two?

## Canonical Equivalent Character Sequences

- U+00F6 LATIN SMALL LETTER O WITH DIAERESIS (ö)  
: 006F 0308
- U+006F LATIN SMALL LETTER O (o)
- U+0308 COMBINING DIAERESIS (¨)

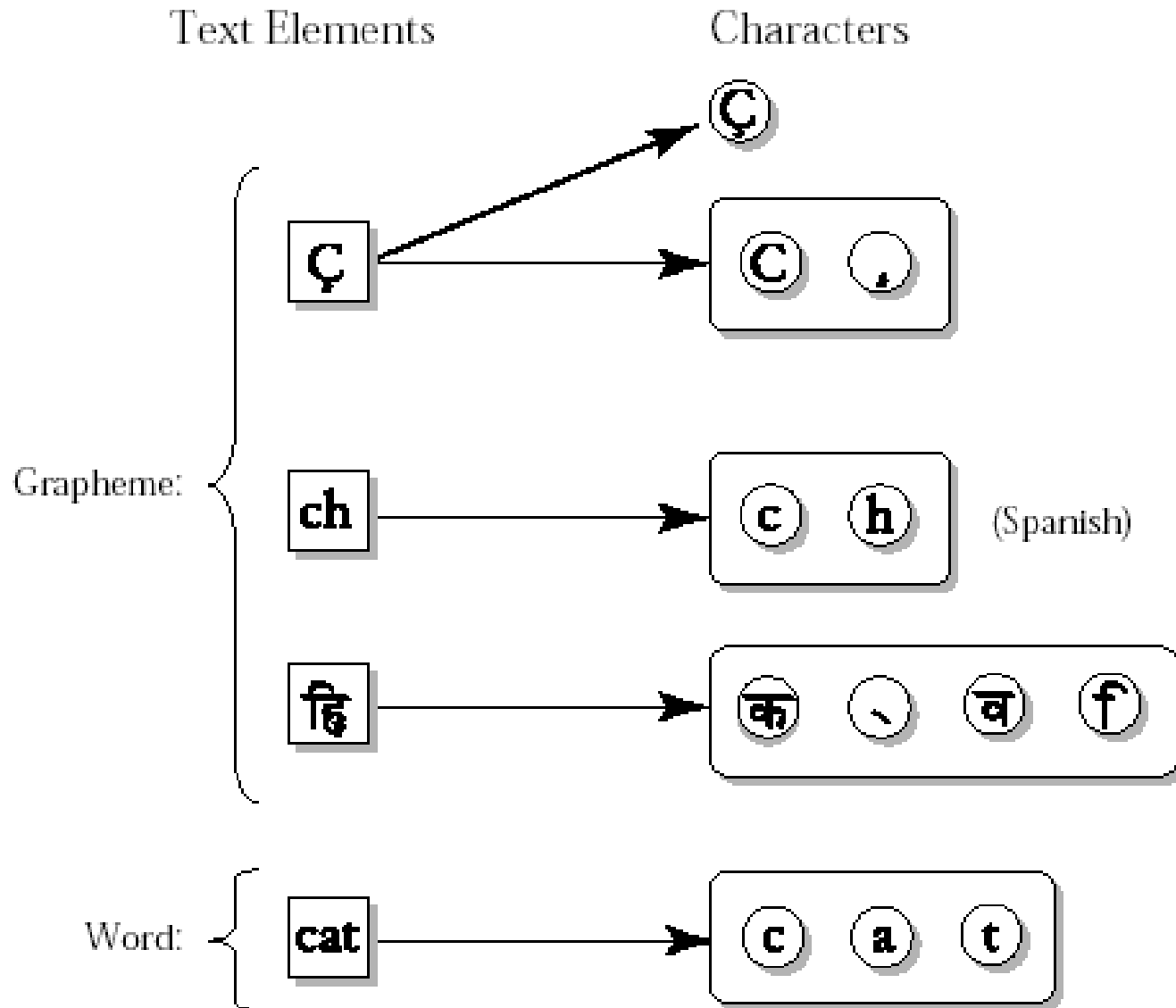
# Text Elements and Characters

“The design of the character encoding must provide precisely the set of code values that allows programmers to design applications capable of implementing a variety of text processes in the desired languages. These code values may not map directly to any particular set of text elements that is used by one of these processes.”

# Text Elements and Compromise

“No encoding can support all basic text processes equally well. As a result, some trade-offs are necessary. For example, ASCII defines separate codes for upper-case and lower-case letters. This choice causes some text processes, such as rendering, to be carried out more easily, but other processes, such as comparison, to become more difficult.”

# Text Elements vs Characters



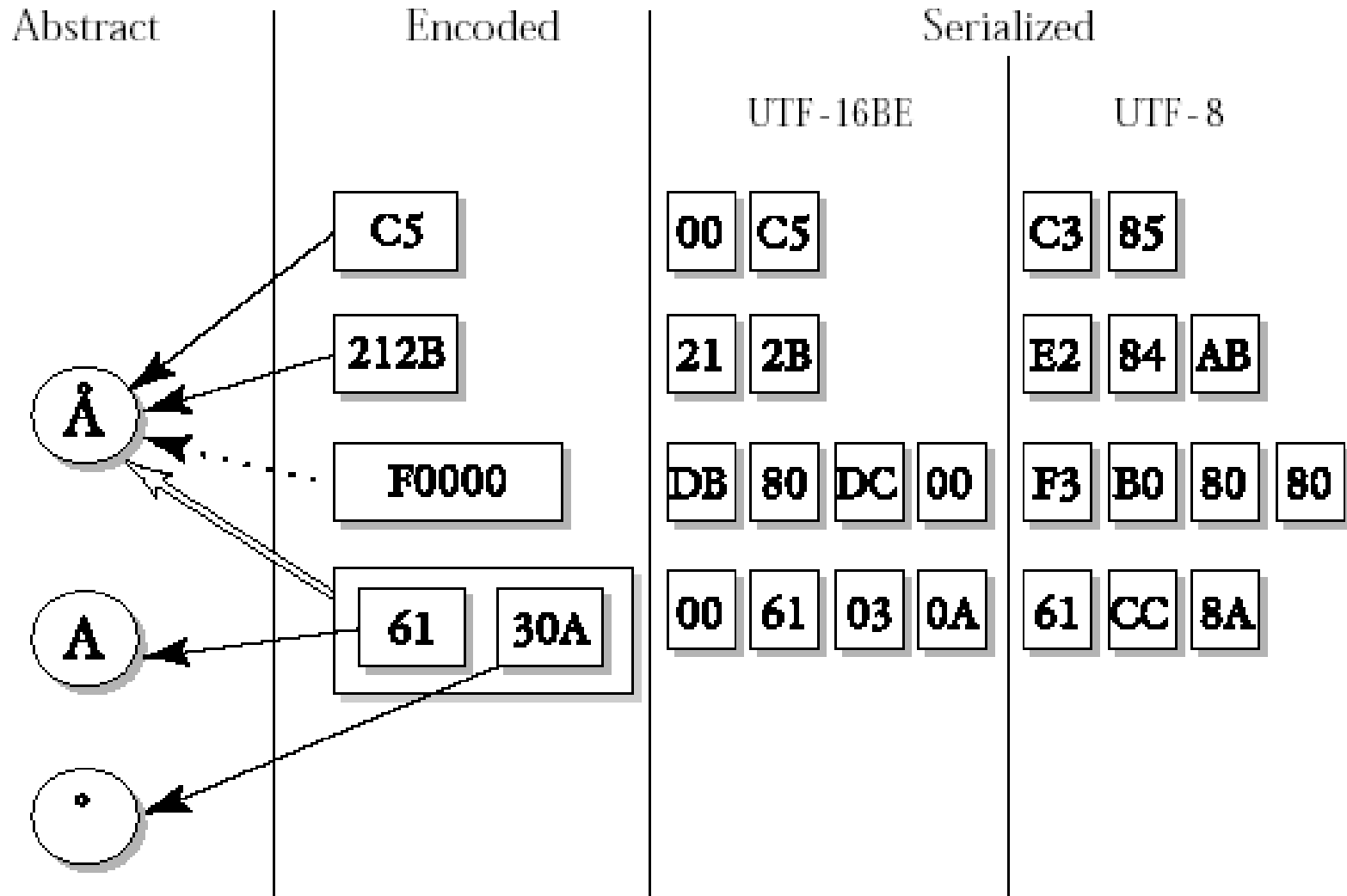
# Characters and Text Processes

- Bulgarian “A” and “a”
  - Rendered differently from each other
  - Conflated for (most) searching
  - Secondary distinction for sorting
- German “ck”
  - Hyphenated as “k-k”
  - Sorted as “c” + “k”
- Spanish “ll”
  - Rendered as “l” + “l”
  - Sorted between “l” and “m”
- Cf. Swedish, German, and English “ö”

# Abstract and Encoded Characters

- “Each Unicode abstract character is encoded one or more times. Each encoding, which consists of the relationship between an abstract character and its scalar value (see D28), is called an *encoded character*. Each scalar value is represented in either of two ways: as a single code value or as a sequence of two surrogate code values.”
- Multiple encodings of single abstract character
  - U+0341 COMBINING ACUTE TONE MARK
  - U+0301 COMBINING ACUTE ACCENT

# Abstract and Encoded Characters



# Decomposition: Variant Encodings

- Alternative representations
  - á
  - a + ´
  - ´ + a (not legal in Unicode)
- Character vs glyph

# Characters and Glyphs

<i>Set</i>	<i>Item</i>	<i>Representation</i>
Coded Character Set	(Encoded) Character	Information
Font	Glyph	Presentation

*á* vs *a + ´*

One or two characters? One or two glyphs?

# Variation

<i>Language</i>	<i>Example</i>	<i>Orientation</i>	<i>Function</i>
Bulgarian	й	Vertical	Independent collation
Russian	ë		Optional variant of e
French	ç		Variant of c
Russian	ы	Horizontal	Independent collation
Spanish	ch		Independent collation
English	ch		Sequence of c + h

# What is Unicode?

- “... the universal character encoding scheme for written characters and text.”
- “... provides the capacity to encode all characters used for the written languages of the world”
  - Private use area

# Unicode and ISO/IEC 10646

- “Version 3.0 of the Unicode Standard is code-for-code identical to ISO/IEC 10646-1:2000, Information Technology--Universal Multiple-Octet Coded Character Set (UCS)--Part 1: Architecture and Basic Multilingual Plane, which is also known as the Universal Character Set (UCS).”
- Merger 1992
- Ongoing liaison and collaboration

# Unicode Design Principles

- Sixteen-Bit Characters
- Stateless:  
Unambiguous and  
Self-Synchronizing
- Characters, not Glyphs
- Character Semantics
- Plain Text
- Logical Order
- Unification
- Dynamic Composition
- Equivalent Sequence
- Convertability
- Open repertoire

# Sixteen-Bit Characters

- 65536 ( $2^{16}$ ) values
  - 63486 single-character codes
  - 2048 “surrogates” permit an additional 1048544 characters (1024 high surrogates followed by 1024 low surrogates)
- Unicode Transformation Format (UTF)
  - UTF-16: all characters encoded in 16 bits
  - UTF-8: some characters (including ASCII) encoded in 8 bits, others in 16 (or more)

# Stateless: Unambiguous and Self-Synchronizing

- Unambiguous new character
  - Non-surrogates
  - 1024 high surrogates
- Unambiguous one-character backup
  - 1024 low surrogates
- No escapes or shifting required (stateless)
- Corruption does not propagate
- (Bit-wise synchronization is not automatic)

# Characters, Not Glyphs

- Characters
  - “Smallest components of written language that have semantic value”
- Glyphs
  - “represent the shapes that characters can have when they are rendered or displayed”

# Characters vs Glyphs

- “A character conveys distinctions in meaning or sounds. A character has no intrinsic appearance.”
- “A glyph conveys distinctions in form. A glyph has no intrinsic meaning.”
- “The relationship between character codes and glyph identifiers may be one-to-one, one-to-many, many-to-one, or many-to-many.”

(N 915)

# “*Distinctions* in meaning or sounds”

- “A character conveys distinctions in meanings or sounds ...”
- U+006A LATIN SMALL LETTER J (j)
  - Voiced palatal affricate (English ‘joy’)
  - Voiced palatal fricative (French ‘joie’)
  - Palatal glide (German ‘ja’)
- U+0308 COMBINING DIAERESIS (¨)  
= umlaut

# Characters and Glyphs

Unicode Identifier	Unicode Name	Representative Glyphs
U+0041	LATIN CAPITAL LETTER A	A A A <b>A A</b> A <i>A A</i> <i>A A A <b>A A</b> A A</i> <i>A A</i> A <b>A A A A</b> A <i>A A</i>
U+0061	LATIN SMALL LETTER A	a a a <b>a a</b> a <i>a a</i> <i>a a a <b>a a</b> a a</i> <i>a a a</i> a <b>a a a a</b> a <i>a a</i>

# Character Semantics

- Character names
  - LATIN CAPITAL LETTER A
  - LATIN CAPITAL LETTER A WITH ACUTE
  - COMMA
  - ACUTE ACCENT
  - COMBINING ACUTE ACCENT
- Properties
  - Spacing, combination, directionality, etc.

# Plain Text

- Vs. “rich” or “fancy” text
- Content stream to which formatting may be applied
- Excludes font (and font properties, such as size, weight, etc.), language, color
- Plain text “has no inherent appearance. It requires a rendering process to make it visible.”
- “Plain text must contain enough information to permit the text to be rendered legibly, and nothing more.”

# Plain Text and XML

- XML documents are in plain text
- XML uses specific plain-text characters to demarcate markup
- Characters within these delimiters are markup, distinct from data content
- Markup represents document properties
- Markup may be used to control rendering of document either directly (presentational markup) or—preferably—indirectly

# Logical Order

G i d i \_ s a i d , \_ " ם ם \_ ם ' ן \_ ם ' ל \_ ם ' ל \_ ם ' ל " .



- Characters have inherent directionality
- Unicode includes special characters that assert directionality
- All combining marks are encoded after their base characters in logical order
  - $a + ' + u = \acute{a}u$  (not  $a\acute{u}$ )

# Unification

- “Unicode unifies different items within scripts across languages.”
  - U+0430: CYRILLIC SMALL LETTER A
  - U+0463: CYRILLIC SMALL LETTER YAT
- No unification across scripts
  - U+0430: CYRILLIC SMALL LETTER A
  - U+0061: LATIN SMALL LETTER A

# Dynamic Composition

- Base characters and diacritics may be combined dynamically
- No formal restriction on combinations
- “Inside-out rule”
  - Diacritics are stacked outward from base  
( $c + \bar{\ } + ' \neq c + ' + \bar{\ }$ )
  - Order of non-interacting (geographically non-overlapping) diacritics is unrestricted  
( $c + , + ' = c + ' + ,$ )

# Round Trip (Source Set Rule)

- “If a form is included as a character in any of the character sets from which ISO/IEC 10646 is derived, then that form shall be included as a character in ISO/IEC 10646 ...” (N 915)
- Round trip rule: Round-trip lossless conversion is guaranteed between Unicode and standards in wide use prior to 1993


# Convertability

- Legacy characters
  - Included to satisfy round trip rule
  - May fail to qualify as independent characters
- Types
  - Precomposed characters (e.g., ö vs o + ¨ )
  - Compatibility characters (e.g., Arabic ligatures, Arabic contextual form glyphs)

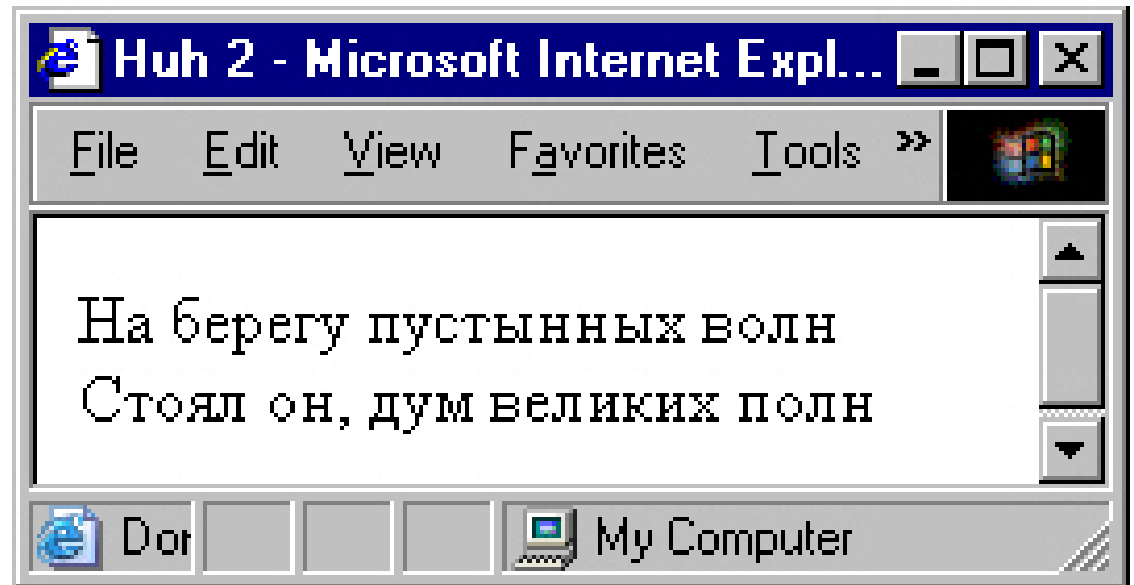
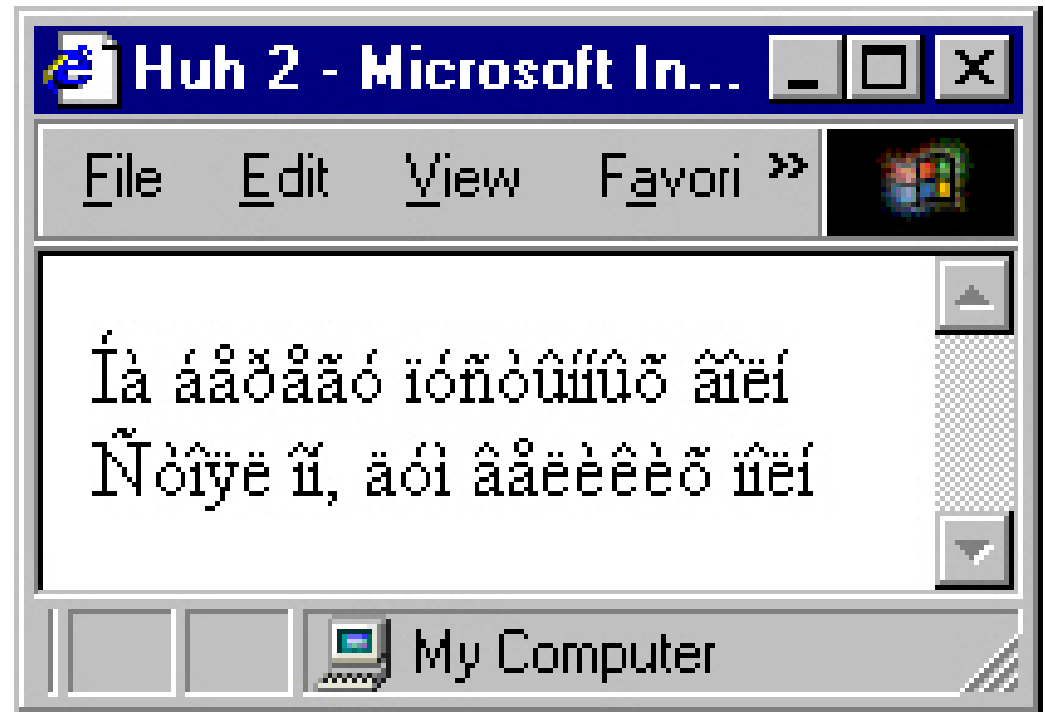
# Open Repertoire

- Characters will be added continuously
- Characters may be deprecated, but will not be moved or removed, and code points will not be reassigned
- New versions are supersets of old version
- Old data will remain valid and legible under new versions

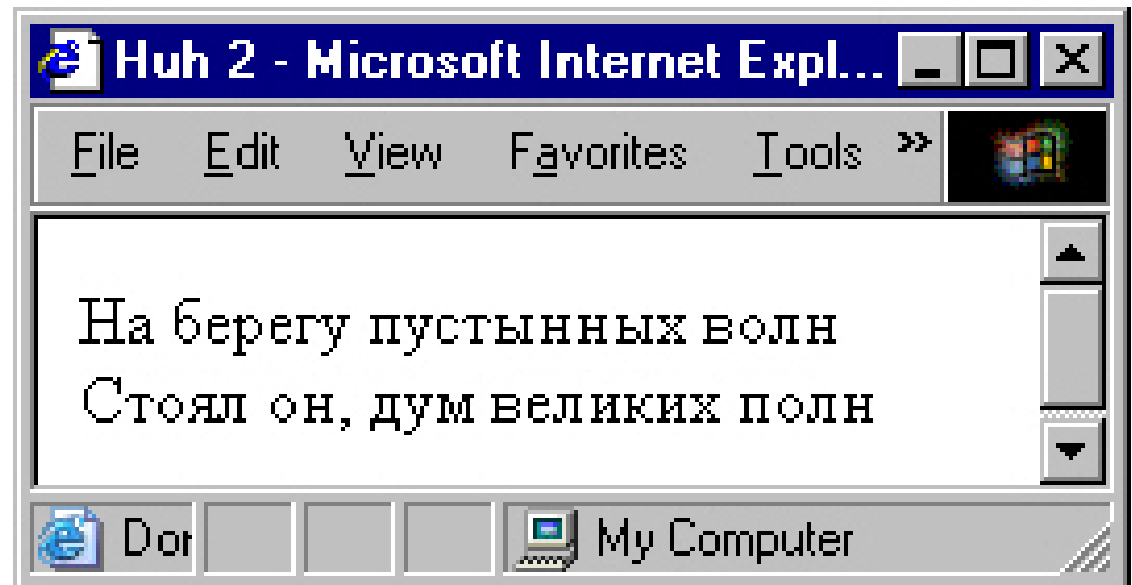
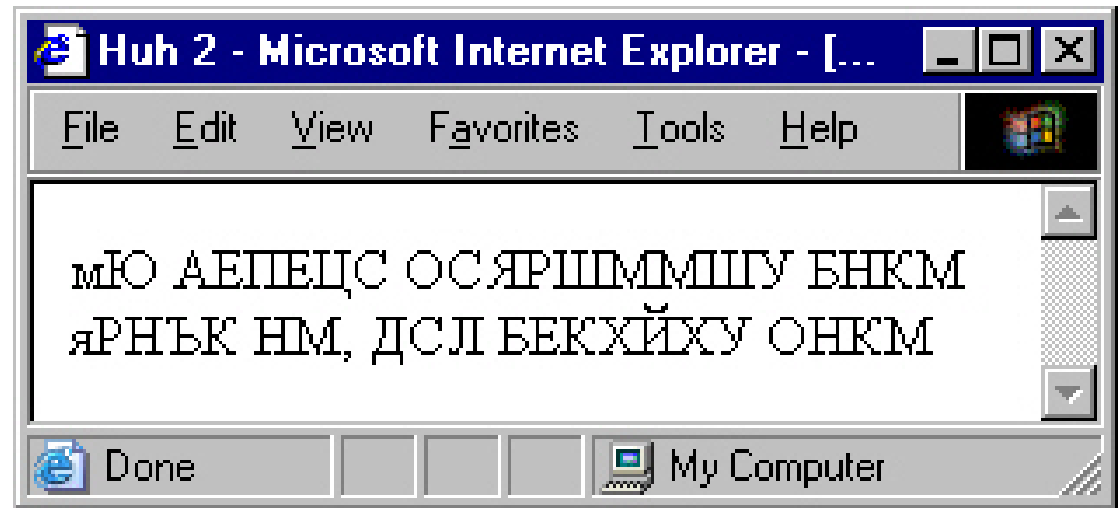
# Why Unicode?

- Superset of all existing character sets
- Addresses two problems
  - Overloading font selection for character-selection purposes
    - U+0061: a (Times New Roman)  $\neq$   (Wingdings)
  - Conflicting character set standards for same inventory (e.g., Cyrillic KOI-8 vs CP 1251 vs ISO 8859-5)

# Huh? (Part 1)



# Huh? (Part 2)



# Why Not Unicode?

- **Legacy software compatibility:** Software must be Unicode-aware
- **Conversion overhead:**
  - Character remapping required
  - Overloaded font selection poses additional complications
- **Storage space overhead:** File size is approximately doubled
- **Processing overhead:** Access, transfer, and other processing time increases with file size

# Why Unicode, Nevertheless?

- Separates character set information from font information
- Storage space and processing overhead are affordable
- Modern software is Unicode-aware
- XML requires Unicode
- Unicode makes multilingual documents as platform-independent, system-independent, and application-independent as ASCII English documents