Unicode

Unicode consortium

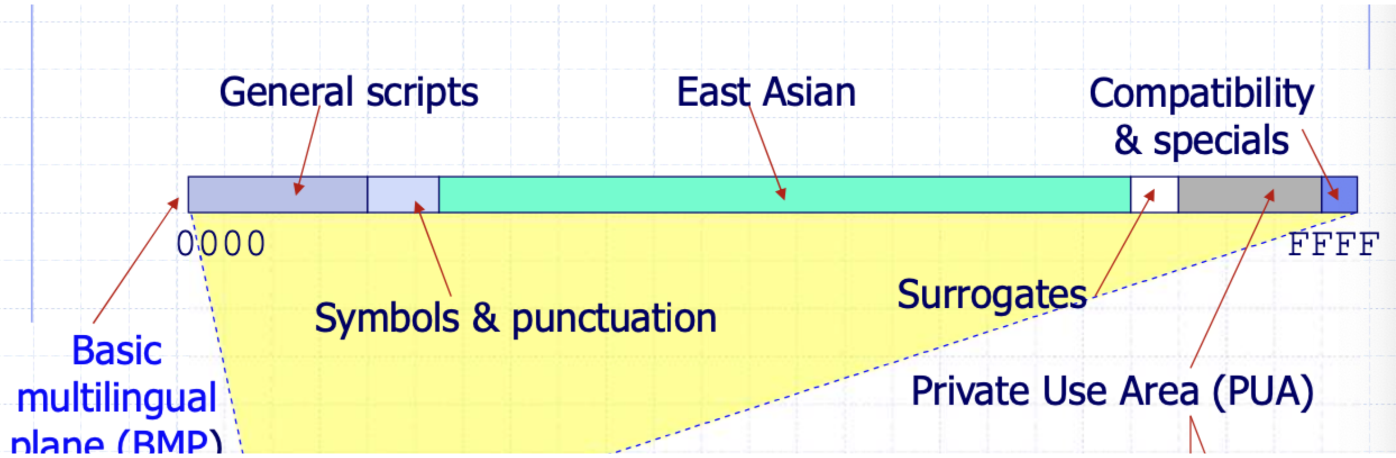
Unicode principles

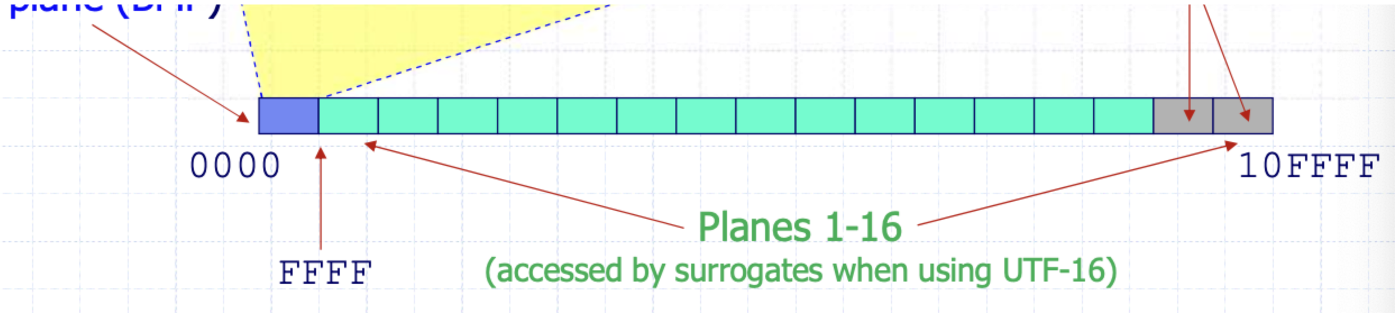
- 21 - bit character codes
    - 16 se les quedo chico
- Efficiency
- Characters, not glyphs
- Well-defined semantics
- Dynamic composition
- Plain text
- Logical ordering
- Unification
- Equivalence
- Convertibility

Character Codes and efficiency

- Character codes
    - Unicode 4.0 had 57,129 16-bit characters out of a total maximum of 63,470
    - A further 45,718 rare or archaic characters are encoded with two consecutive 16-bit code units from reserved ranges (called "surrogates")
    - Son números
    - cada caracter es un número
- Efficiency
    - No special escape or shift characters required
        - \t
    - All representations of unicode are self-synchronizing and can be randomly accessed
        - ir a cualquier sitio y saber que pone ahi
    - Formatting characters are kept to a minimum

Unicode code space





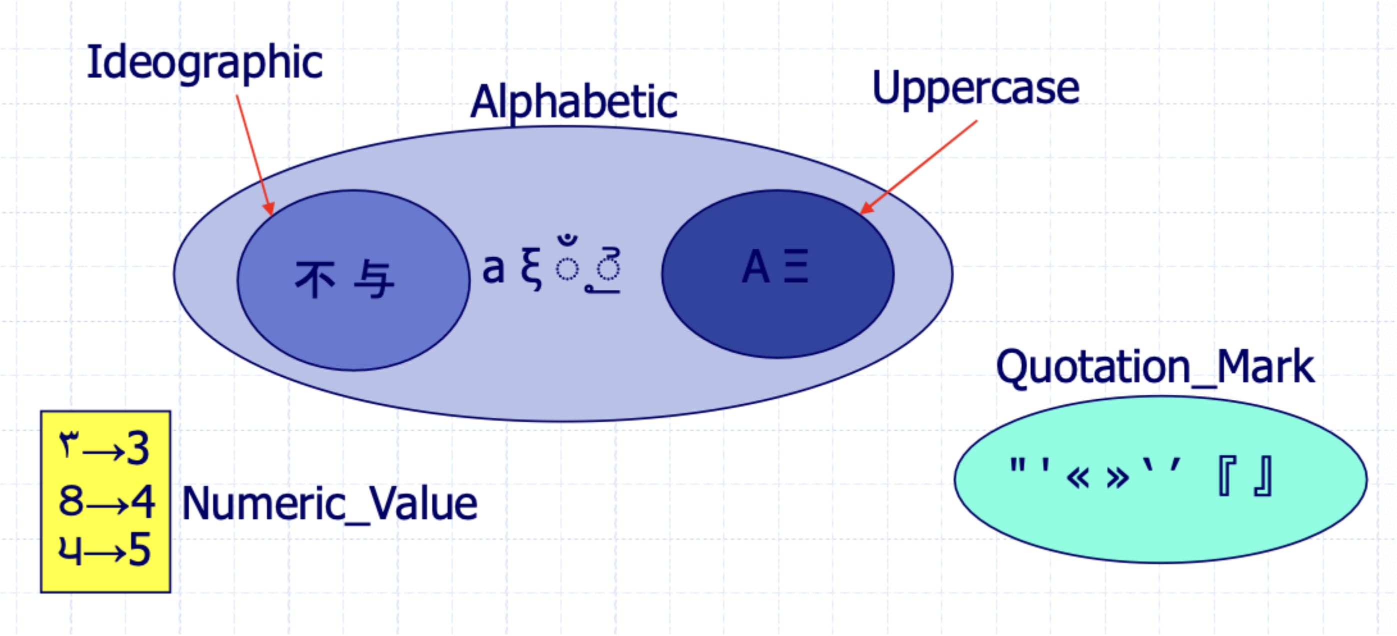
- Bloque azul
  - 0000 => FFFF 2^16
- General Scripts
  - fonemas
- East Asia
  - Caracteres representan ideas
- Private use Area
  - emojis cosas unicas de cada persona
- El resto verde estan para lenguajes inventados, muertos
- Verde tambien se usan para emoji
  - plano astrales

Characters vs Glyphs

- Character
  - the smallest component of written language that has semantic value
  - a es un caracter
    - a latina
    - a
      - glypho
    - U+00061
      - code point
      - para el mismo caracter hay carios glifos
    - U+00041
      - A
      - en grecia alpha
  - glifo que representa dos letras
    - fi
- Glyph
  - represents the shape of a character when rendered or displayed
- Fonts contain glyphs, not characters
  - font es un diccionario de characters a glyphs
- Latin A and Greek A (alpha) are distinct characters with the same glyph
- Arabic letters need up to four glyphs (initial, medial, final, isolated)
- "f" plus "i" is rendered with a single merged glyph in fine typesetting

Well-defined Semantics

- Tables generated by the unicode consortium give the properties of characters
    - Letter, number, punctuation mark, symbol, diacritic, whitespace ...
    - todos los caracteres estan clasificados

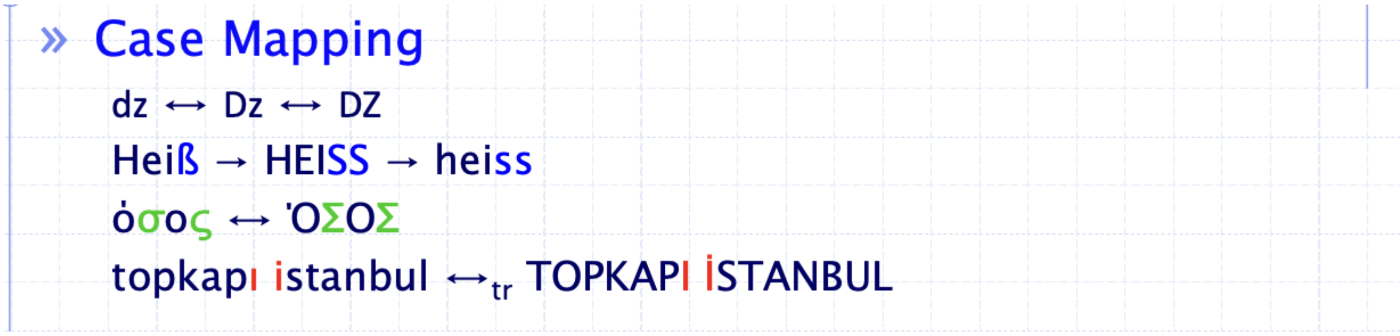


- Case mappining, arabic shaping, normalization
    - normalization
        - raul sin acento

Unicode General Categories

- Letters: upper, lower, title, modifier, other (syllables, ideographs, etc)
- Numbers: Digit, letter, other
- Punctuation: connector, ash, open, close, initial-quote, final-quote, other
- Marks: non-spacing, enclosing, other
- Symbols: math, currency, modifier, other
- Separators: space, line, paragrpah
- Other: control, format, surrogate, private-use

Case Mapping and Normalization



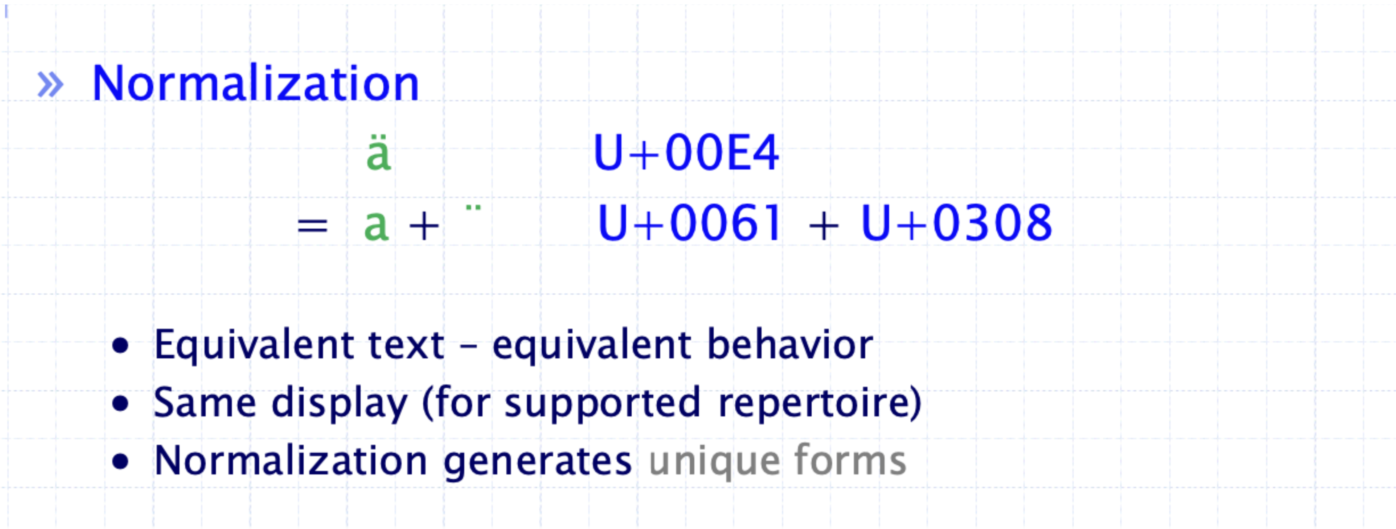
» Case Mapping

dz ↔ Dz ↔ DZ

Heiß → HEISS → heiss

όσος ↔ ΌΣΟΣ

topkapı istanbul $\leftrightarrow_{tr}$ TOPKAPI İSTANBUL



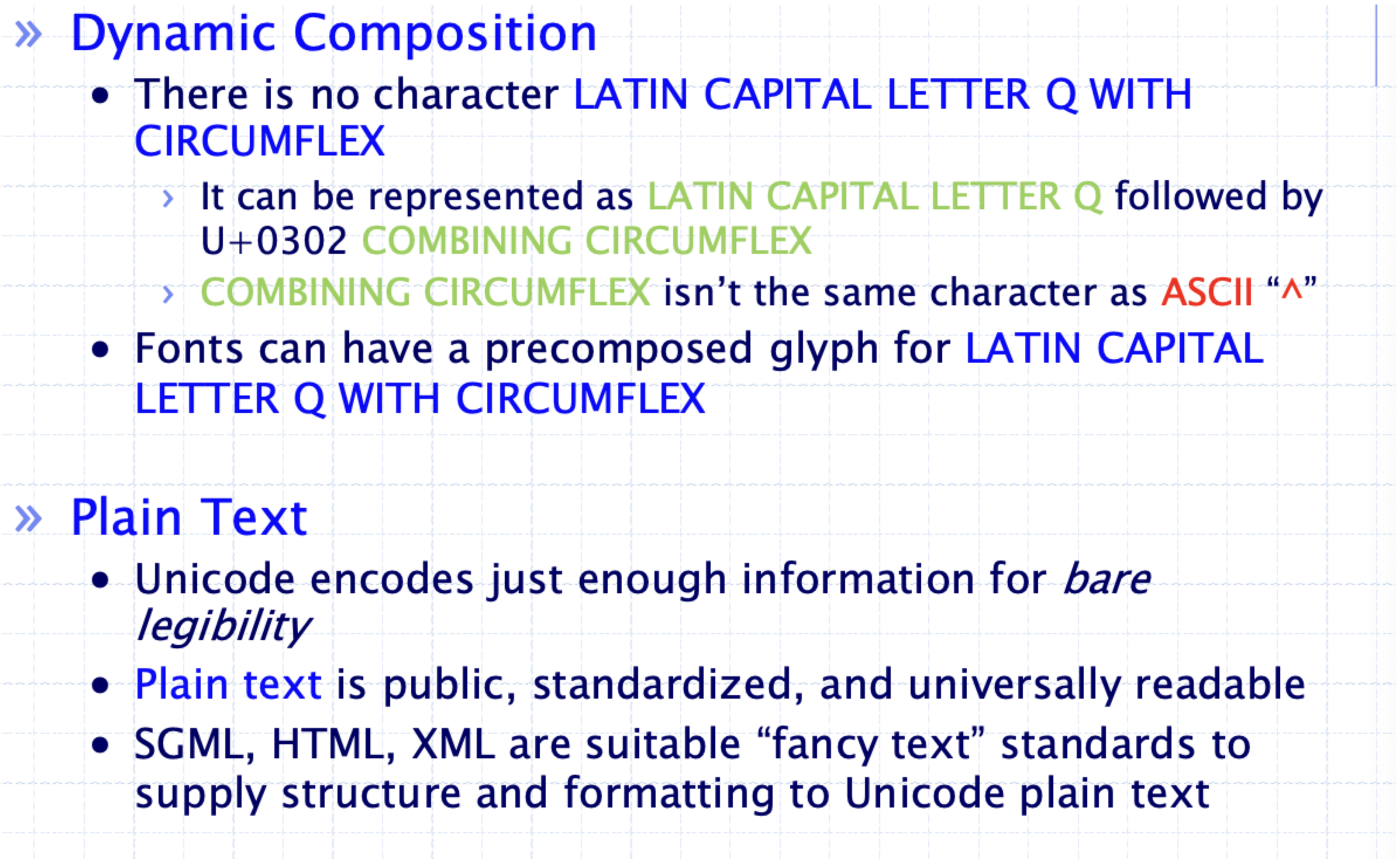
» Normalization

ä U+00E4

= a + ̈ U+0061 + U+0308

- Equivalent text – equivalent behavior
- Same display (for supported repertoire)
- Normalization generates unique forms

Dynamic Composition and Plain Text



» Dynamic Composition

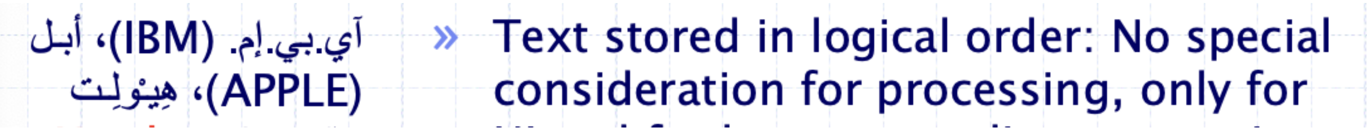
- There is no character LATIN CAPITAL LETTER Q WITH CIRCUMFLEX
  - › It can be represented as LATIN CAPITAL LETTER Q followed by U+0302 COMBINING CIRCUMFLEX
  - › COMBINING CIRCUMFLEX isn't the same character as ASCII "^"
- Fonts can have a precomposed glyph for LATIN CAPITAL LETTER Q WITH CIRCUMFLEX

» Plain Text

- Unicode encodes just enough information for *bare legibility*
- Plain text is public, standardized, and universally readable
- SGML, HTML, XML are suitable "fancy text" standards to supply structure and formatting to Unicode plain text

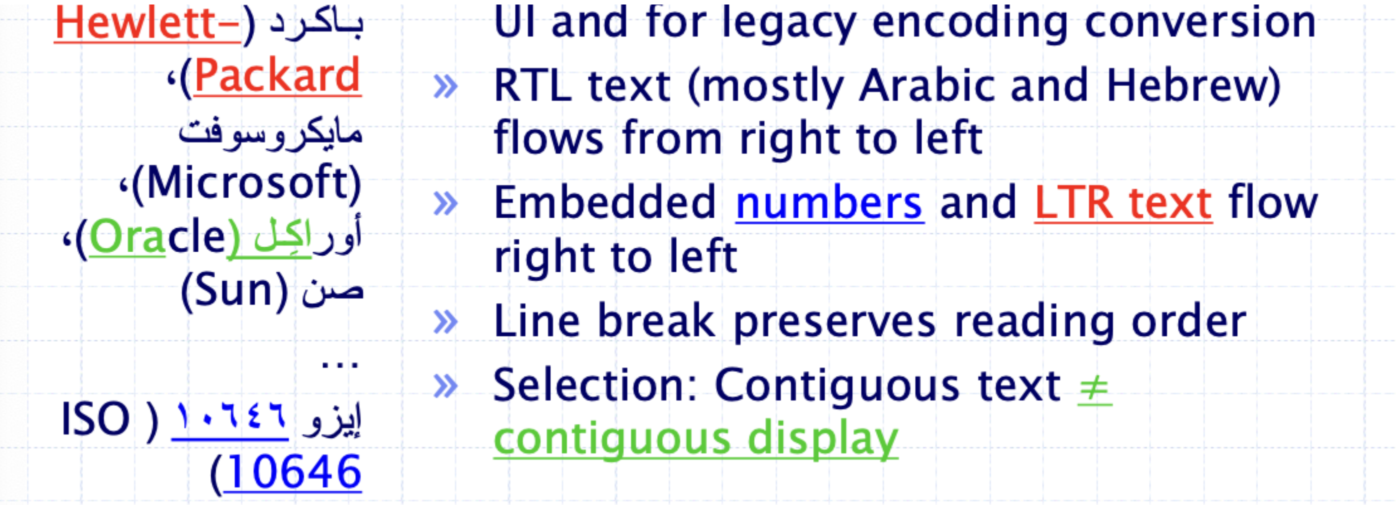
Logical Ordering

- With one minor exception, characters are represented in unicode in logical order (the order they are typed or spoken)
  - Unicode provides a table-driven algorithm for reordering text into proper reading order, including mixed directions



آي.بي.إم. (IBM)، أبل
هيوْلِت (APPLE)،

» Text stored in logical order: No special consideration for processing, only for



Hewlett–) باكرد
،(Packard
مايكروسوفت
،(Microsoft)
،(Oracle) أوراكِل
(Sun) صن
...
ISO) ١٠٦٤٦ إيزو
(10646

UI and for legacy encoding conversion

» RTL text (mostly Arabic and Hebrew) flows from right to left

» Embedded numbers and LTR text flow right to left

» Line break preserves reading order

» Selection: Contiguous text ≠ contiguous display

Unification

- "A difference that makes no difference is no difference"
- If characters look the same, and are from different source standards, they are a single Unicode character
  - common letters, punctuation marks, symbols, and diacritics are unified
  - Differences in language, font, size, and positioning are not represented
  - Identical-looking characters (a,alpha) from different scripts are not unified
  - Characters that were distinct in a major national or industry standard are kept distinct for rounding-tripping purposes

Han Unification

- Chinese, Japanese, Korean (CJK) all use the 3000 - year - old Chinese characters (hanzi, kanji, hanja)
  - Each national character set encodes the characters in its own way
- If it looks similar and is historically the same, Unicode unifies it!
  - Unicode orders Han characters using the traditional Kang Xi dictionary and other dictionaries
- Language differences, which control the choice of fonts, are expressed by a higher-level protocol
- Simplified and traditional characters are not unified in Unicode

Equivalence and Convertibility

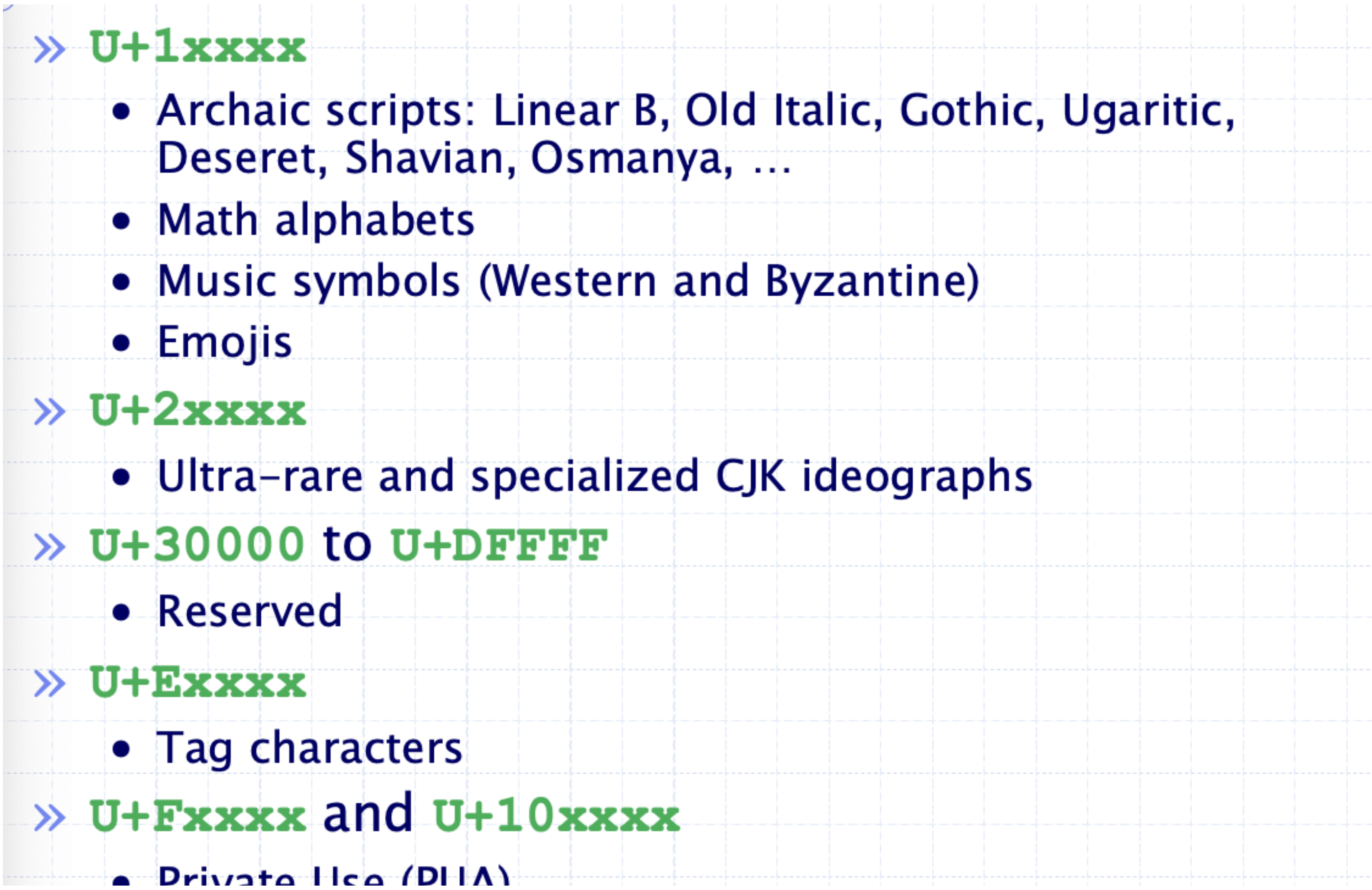
- Equivalence
  - Different way of representing the same characters are equally valid
  - Normalization forms allow documents to be compared and easily by suppressing irrelevant enconding differences
- Convertibility
  - Characters in other character sets can be converted to and from Unicode, usually 1:1
  - ASCII and Latin-1 map codepoint for codepoint
  - Conversions are done by mapping tables

Unicode Map : Basic Multilingual Plane

- U+0xxx
  - ASCII, Latin, Greek, Cyrillic Armenian, Hebrew, Arabic, Syriac, Thaana, Indic scripts, Thai Lao, Tibetan
- U+1xxx

- U+1xxx
    - Myanmar, Georgian, Hangul, Ethiopic, Cherokee, Canadian, Aboriginal, Ogham, Runic, Phillippine scripts, Khmer, Mongolian, Limbu, Tai Le, Extended Latin, Extended Greek
- U+2xxx
    - Symbols
        - Punctuation, super/subscripts, currency, letter-like, boxes, numerical, arrows, math, technical, OCR, dingbats, Braille
    - CJK radicals
- U+3xxx
    - CJK symbols, Hiragana, Katakana, Bopomofo
- U+3400 to U+9FFF
    - CJK Unified Idegraphs
- U+A000 to U+D7A3
    - Yi,Hangul Syllables
- U+D800 to U+DFFF
    - surrogates (no characters)
- U+E000 to U+F8FF
    - Private Use
- U+Fxxx
    - CJK Compatibility Ideographs
    - Presentation Forms
    - Halfwidth / Fullwidth

Unicode Map: "Astral Planes"



» U+1xxxx

- Archaic scripts: Linear B, Old Italic, Gothic, Ugaritic, Deseret, Shavian, Osmanya, …
- Math alphabets
- Music symbols (Western and Byzantine)
- Emojis

» U+2xxxx

- Ultra-rare and specialized CJK ideographs

» U+30000 to U+DFFFF

- Reserved

» U+Exxxx

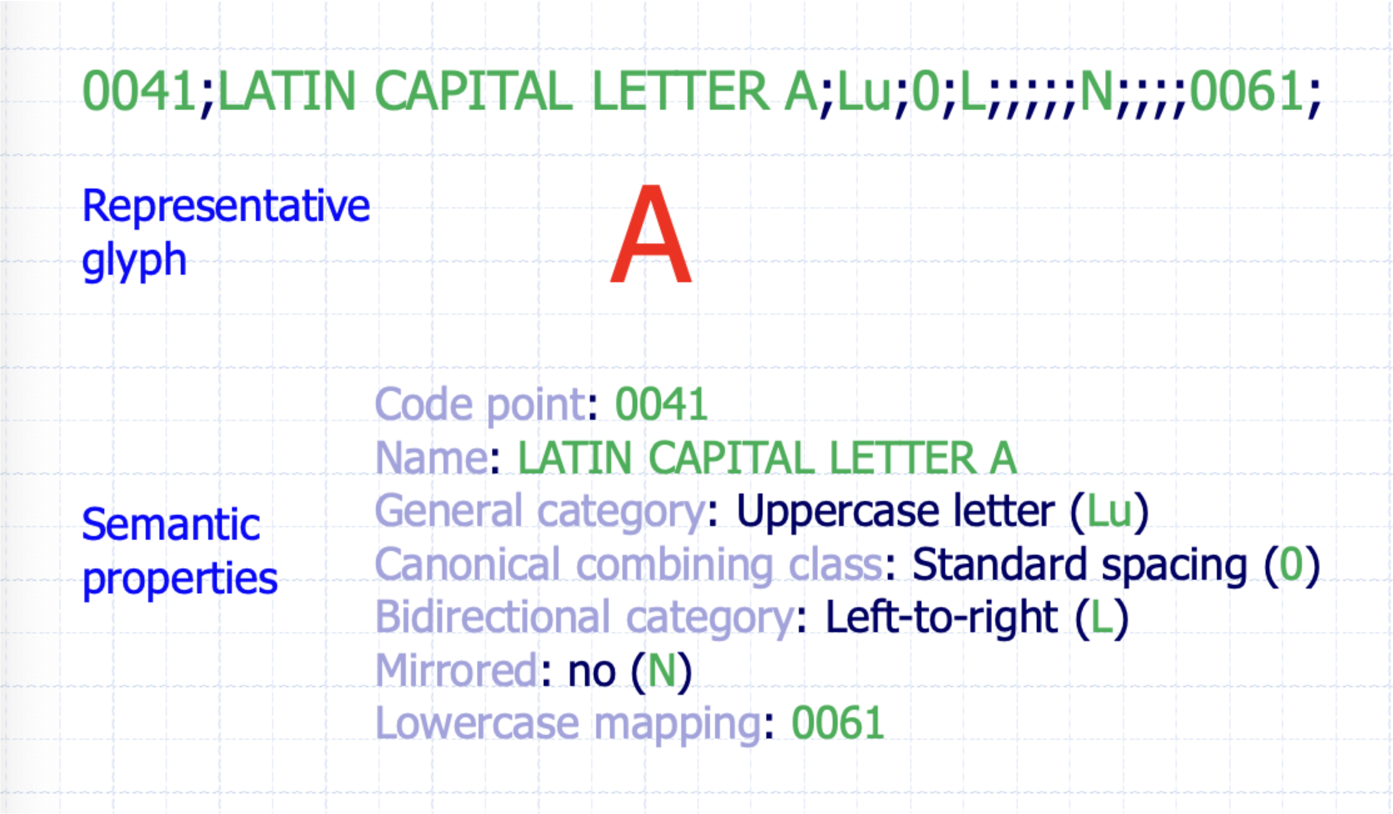
- Tag characters

» U+Fxxxx and U+10xxxx

- Private Use (PUA)

- Private Use (PUA)

Unicode Properties



0041;LATIN CAPITAL LETTER A;Lu;0;L;;;;;N;;;;0061;

Representative glyph

Semantic properties

Code point: 0041
Name: LATIN CAPITAL LETTER A
General category: Uppercase letter (Lu)
Canonical combining class: Standard spacing (0)
Bidirectional category: Left-to-right (L)
Mirrored: no (N)
Lowercase mapping: 0061

- Se representan con un code point
  - codepoint = DNI
    - unico
  - Name
    - nombre que lo describe

Puedo poner pile of poo como variable en java

- no porque no es del tipo que pide java para declarar variables

Encodings

- Pre-Unicode
  - ASCII is a 7-bit encoding for about 100 characters
  - ISO-8859-1 is an 8-bit encoding for about 200 characters
  - Shift-JIS is a mixed 8/16 bit encoding for about 8,000 characters
  - How to best encode Unicodes 2097152 (2^21) possible codepoints

2^21

- Necesito 2^21 bits pero son potencias de 2 entonces 32
- Que problemas hay con esa codificación
  - problemas de endianess
  - tamaño
    - no calzan con la memoria

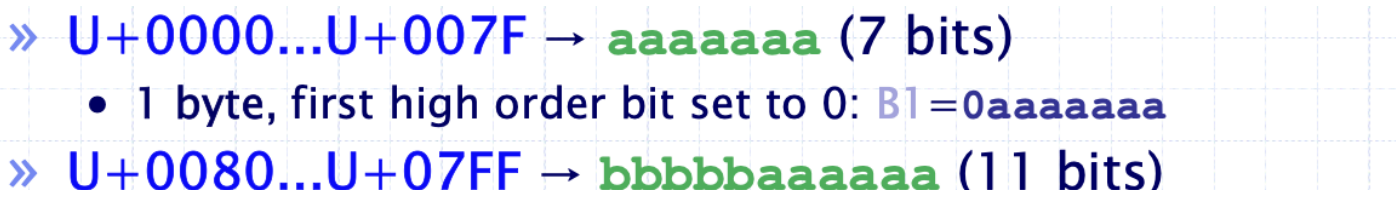
Three Unicode Encoding

- The Unicode Standard has Unicode Transformation Formtats (UTF) that are algorithmic mappings from every Unicode code point (except surrogate code points) to a unique byte sequence (8, 16 or 32-bits pero code point)
  - All encode the same common character repertoire and can be efficiently transformed into one another without loss of data: so they have equal representation power
  - All have advantages and disadvantages
- The three most famous are:
  - UTF- 8: 8 bit code units
    - enlace de datos de baja capacidad
    - no tiene problemas de endianess
      - no tener que explicar como estan codificado
  - UTF- 16: 16 bit code units
  - UTF-32: 32- bit code units
    - más rápido
    - fácil
    - desperdicia espacio
- All three need at most 4 bytes (or 32-bits) of data for each character
- Todos los utfs guardan cualquier caracter

UTF - 8

- Popular for HTML and similar protocols.
- Way of transforming all unicode characters into a variable length encoding of bytes (1,2,3 or 4 bytes to encode a character)
- Advantages:
  - The unicode characters corresponding to the familiar ASCII set have the same byte values as ASCII
  - Unicode characters transformed into UTF-8 can be uses with much existing software without modifications
  - No byte-ordering
    - Examples
      - A is 41 (same as ASCII)
        - unicode 000041
      - Alpha is CE 91
      - Katakana A is E3 82 A2
      - Gothic Ahsa is F0 90 8C B0

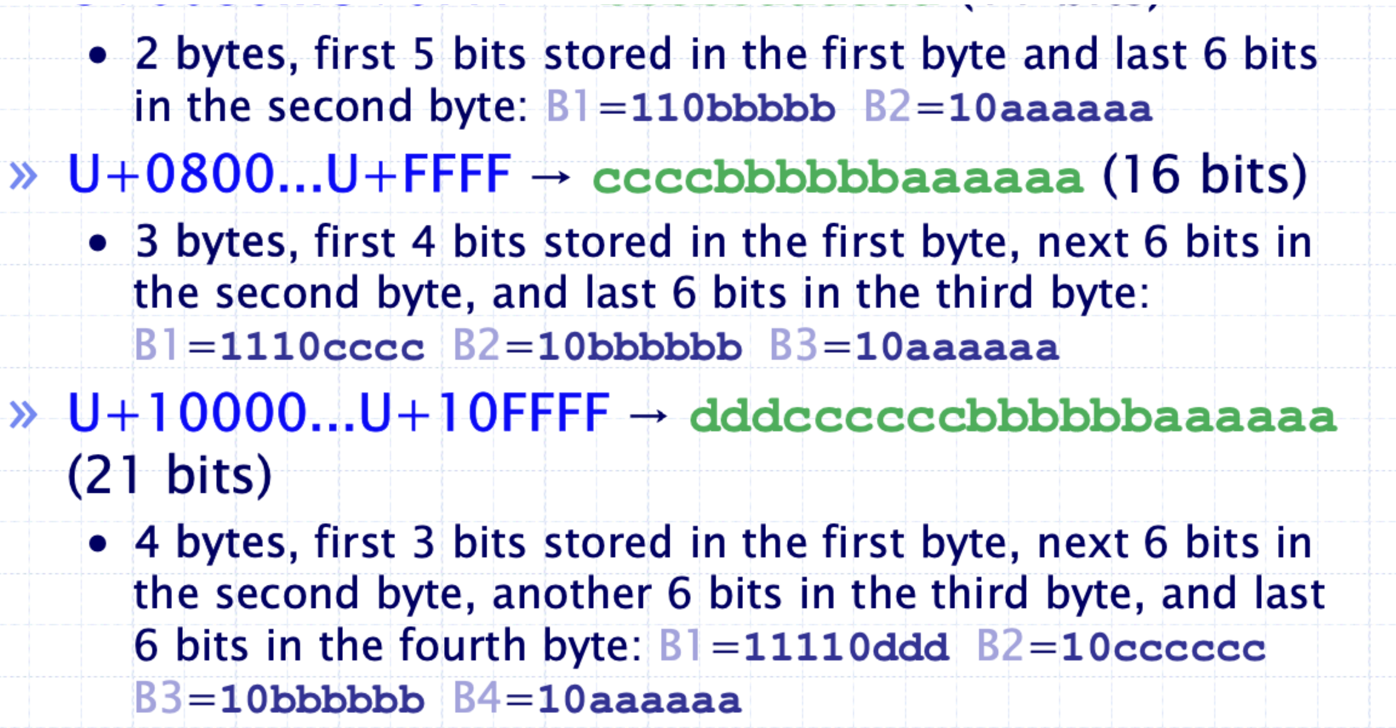
UTF-8 Encoding algorithm



» U+0000...U+007F → aaaaaaa (7 bits)

- 1 byte, first high order bit set to 0: B1=0aaaaaaa

» U+0080...U+07FF → bbbbbaaaaaa (11 bits)



- 2 bytes, first 5 bits stored in the first byte and last 6 bits in the second byte: B1=110bbbbb B2=10aaaaaa

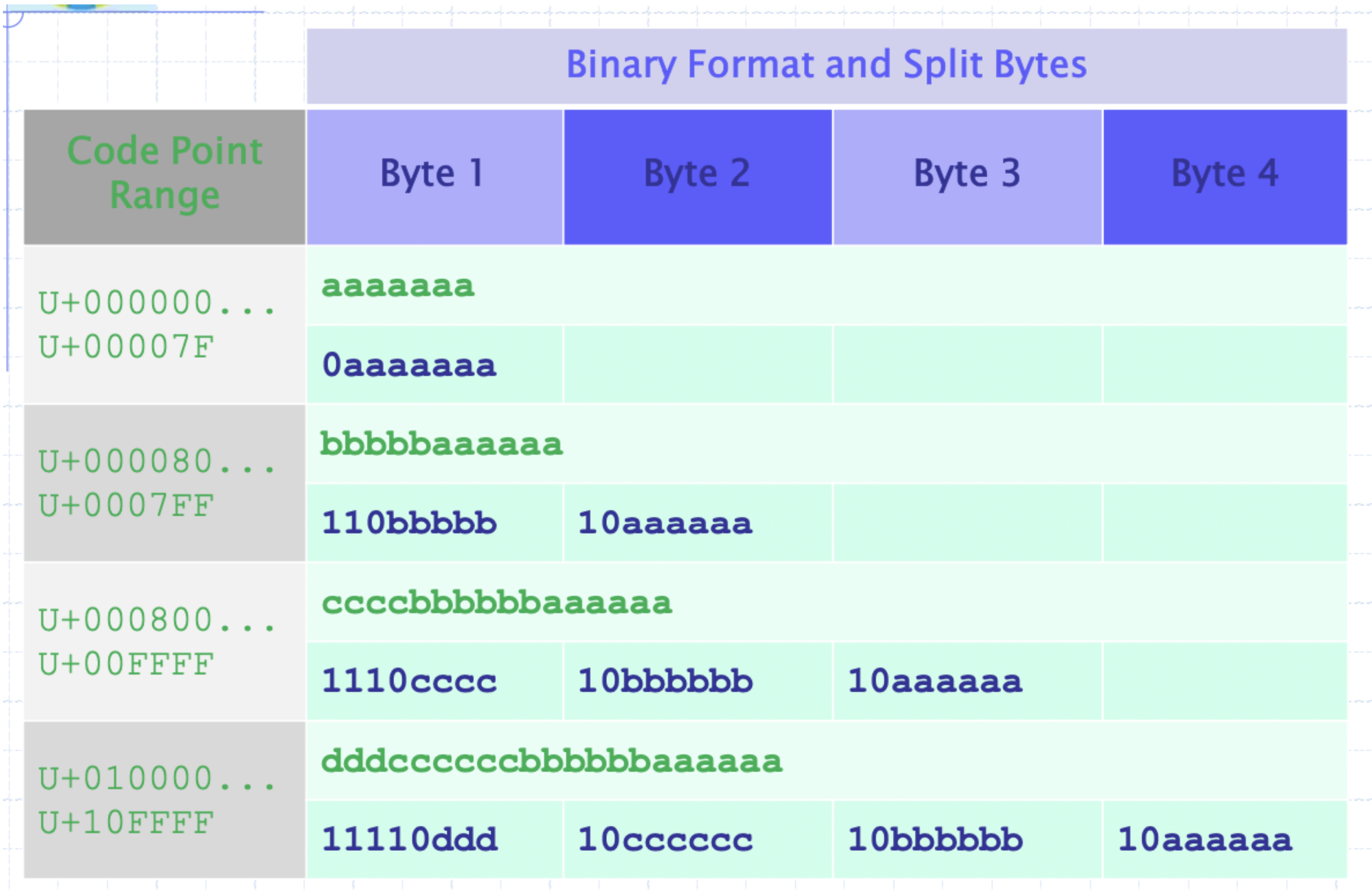
» U+0800...U+FFFF → ccccbbbbbbaaaaaa (16 bits)

- 3 bytes, first 4 bits stored in the first byte, next 6 bits in the second byte, and last 6 bits in the third byte: B1=1110cccc B2=10bbbbbb B3=10aaaaaa

» U+10000...U+10FFFF → dddccccccbbbbbbaaaaaa (21 bits)

- 4 bytes, first 3 bits stored in the first byte, next 6 bits in the second byte, another 6 bits in the third byte, and last 6 bits in the fourth byte: B1=11110ddd B2=10cccccc B3=10bbbbbb B4=10aaaaaa

- Todos los bytes extra siempre empiezan por 10
- Kent thompson
    - unix



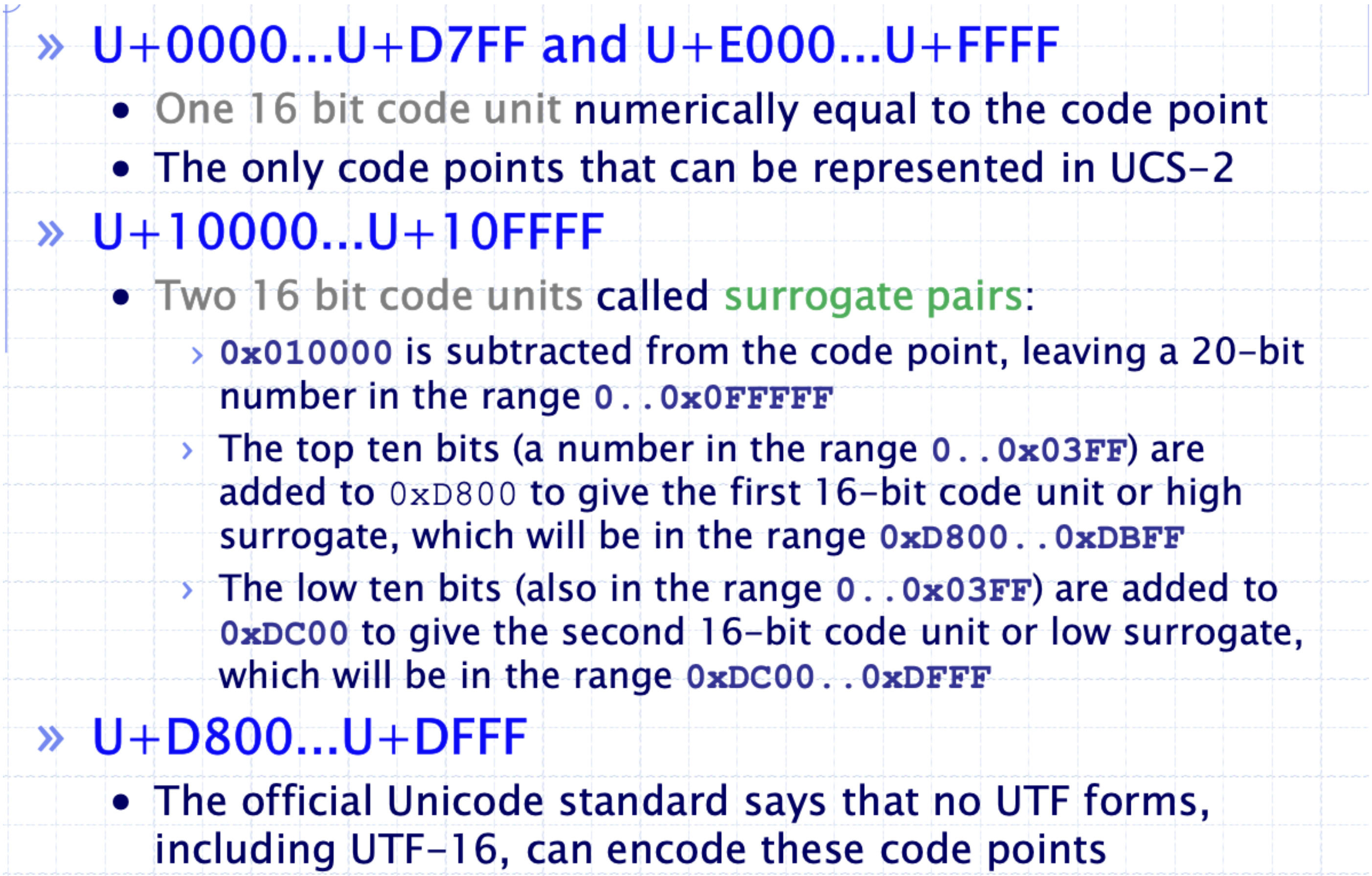
	Binary Format and Split Bytes			
Code Point Range	Byte 1	Byte 2	Byte 3	Byte 4
U+000000... U+00007F	aaaaaaa			
	0aaaaaaa			
U+000080... U+0007FF	bbbbbaaaaaa			
	110bbbbb	10aaaaaa		
U+000800... U+00FFFF	ccccbbbbbbaaaaaa			
	1110cccc	10bbbbbb	10aaaaaa	
U+010000... U+10FFFF	dddccccccbbbbbbaaaaaa			
	11110ddd	10cccccc	10bbbbbb	10aaaaaa

UTF-16

- Popular in many environments that need to balance efficient access to characters with economical use of storage
    - It is reasonably compact

    - It is reasonably compact
    - Each BMP character is represented by the obvious 16-bit code unit
    - Other characters are represented by two consecutive 16-bit code units using surrogates
        - surrogates
            - son ilegales
                - no son characters validos
                - ayudan a codificar codepoint de 32 bits
            - delegados
            - substitutos
    - Examples
        - A is 0041
        - Alpha is 0391
        - Gothic Ahsa (U+10330)

UTF-16 encoding algorithm



» U+0000...U+D7FF and U+E000...U+FFFF
- One 16 bit code unit numerically equal to the code point
- The only code points that can be represented in UCS-2

» U+10000...U+10FFFF
- Two 16 bit code units called surrogate pairs:
    - 0x010000 is subtracted from the code point, leaving a 20-bit number in the range 0..0x0FFFFF
    - The top ten bits (a number in the range 0..0x03FF) are added to 0xD800 to give the first 16-bit code unit or high surrogate, which will be in the range 0xD800..0xDBFF
    - The low ten bits (also in the range 0..0x03FF) are added to 0xDC00 to give the second 16-bit code unit or low surrogate, which will be in the range 0xDC00..0xDFFF

» U+D800...U+DFFF
- The official Unicode standard says that no UTF forms, including UTF-16, can encode these code points

UTF-16 Byte Ordering

- By default, Unicode uses big-endian
    - this can be overridden by local conventions(eg. on Windows)
- UTF-32 Byte Ordering
    - U+0000 u+FeFF, the byte order mark or BOM, can be placed at the beggining of a file to unambiguously indicate the byte order, as U+FFFE U+0000 does not exist
- UTF-16 Byte Ordering
    - Analogously, U+FEFF, the UTF-16 BOM, can be placed at the beginning of a file to

unambiguosly indicate the byte order, as U+FFFE does not exist

- UTF-8 BOM
  - UTF-8 does not need a bom to determine byte order
  - BOM byte sequence (EF BB BF) may still be useful in auto detecting UTF-8

UTF-32

- Useful where memory space is no concern, but fixed width, single code unit access to characters is desired
- Each Unicode character is encoded in a single 32 bit (4 bytes) code unit
- Same byte ordering issues as UTF-16
- Proper subset of UCS-4 (Universal Character Set 4) in ISO 10646
- The main advantage of UTF-32, versus variable- length encodings, is that the unicode code points are directñy indexable
  - Exammining the n'th code point is a constant time operation
- The main disadvantage of UTF-32 is that it is space inefficient, using four bytes per code point point

Comparison of encoding

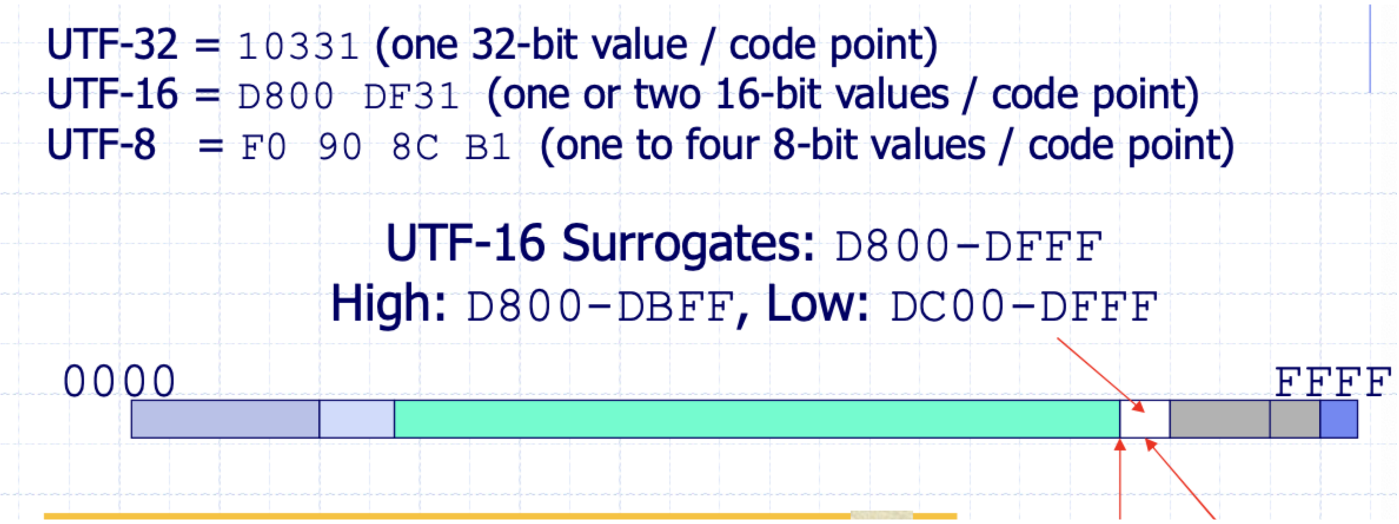
- Advantages of UTF-8
  - Fully ASCII- compatible, including control characters (but not Latin-1 compatible)
  - First byte of any character indicates the number of trailing bytes to follow
  - Sortable, searchable, compressible with 8 bit algorithms
  - Desventajas
    - no tienes random access
    - más complicado
    - gasta mas en CJK
- Advantages of UTF-16
  - Almost fixed-width encoding (non-BMP characters are expected to be rare in most documents)
  - As compact as national CJK encodings
    - UTF-8 costs 50% more
  - Good compromise between space and ease of use
- Advantages of UTF-32
  - Guaranteed fixed-width encoding (directly indexable)
  - Good for internal rather than external (file or network) use
  - desventajas
    - endianess
    - espacio

Todo fuera del BMP ocupa 32 bits

JVM para ejecutar una app

- Se crea un usario para cada applicacioón

Objective C para apps en ios



UTF-32 = 10331 (one 32-bit value / code point)
UTF-16 = D800 DF31 (one or two 16-bit values / code point)
UTF-8 = F0 90 8C B1 (one to four 8-bit values / code point)

UTF-16 Surrogates: D800–DFFF
High: D800–DBFF, Low: DC00–DFFF