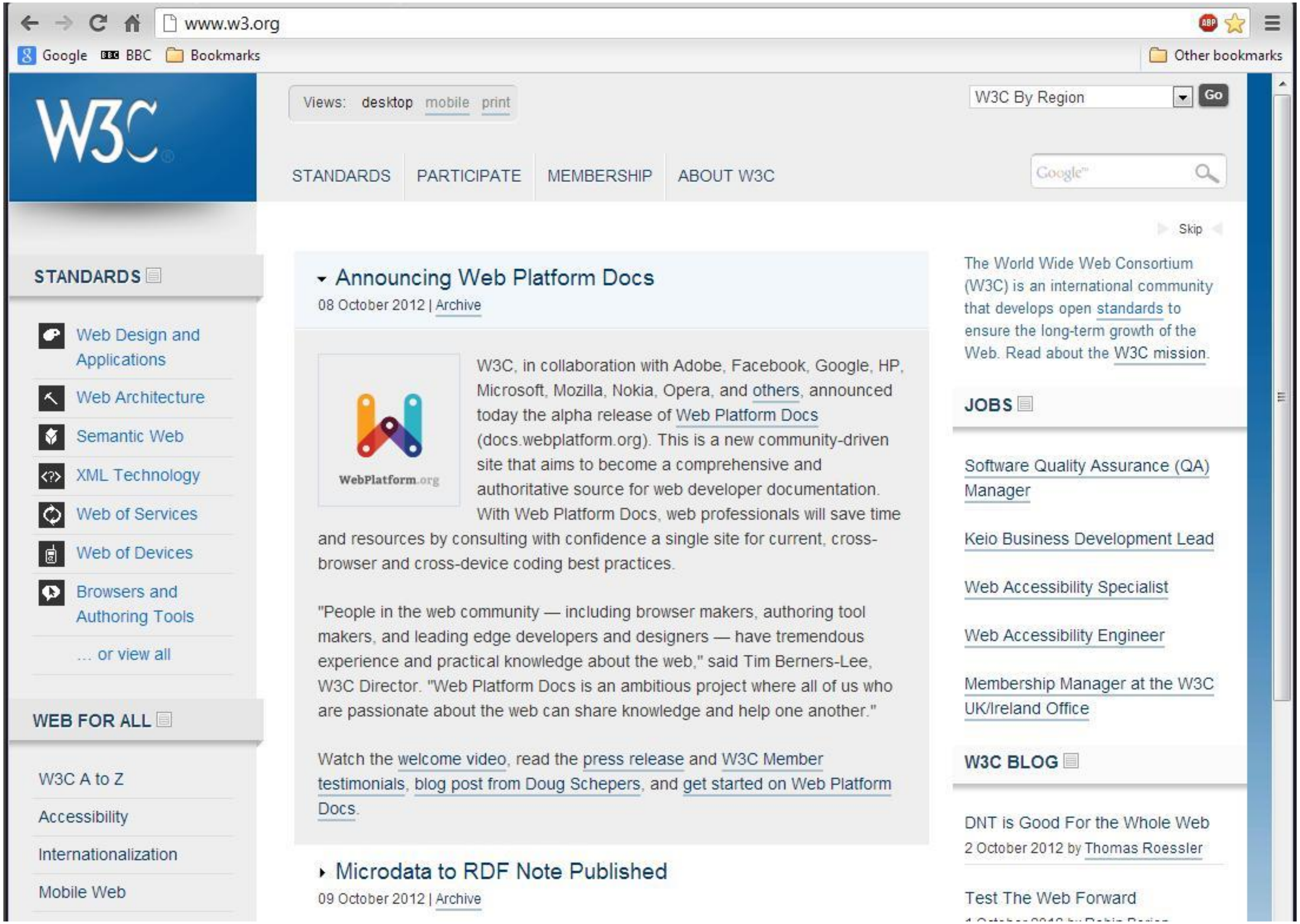
Dynamic Web Development

Lecture 3

Data Representation

www.w3.org



www.w3.org
Google
BBC
Bookmarks
Other bookmarks
W3C
Views: desktop mobile print
W3C By Region
Go
STANDARDS
PARTICIPATE
MEMBERSHIP
ABOUT W3C
Google
Skip
STANDARDS
Web Design and Applications
Web Architecture
Semantic Web
XML Technology
Web of Services
Web of Devices
Browsers and Authoring Tools
... or view all
WEB FOR ALL
W3C A to Z
Accessibility
Internationalization
Mobile Web
Announcing Web Platform Docs
08 October 2012 | Archive
WebPlatform.org
W3C, in collaboration with Adobe, Facebook, Google, HP, Microsoft, Mozilla, Nokia, Opera, and others, announced today the alpha release of Web Platform Docs (docs.webplatform.org). This is a new community-driven site that aims to become a comprehensive and authoritative source for web developer documentation. With Web Platform Docs, web professionals will save time and resources by consulting with confidence a single site for current, cross-browser and cross-device coding best practices.
"People in the web community — including browser makers, authoring tool makers, and leading edge developers and designers — have tremendous experience and practical knowledge about the web," said Tim Berners-Lee, W3C Director. "Web Platform Docs is an ambitious project where all of us who are passionate about the web can share knowledge and help one another."
Watch the welcome video, read the press release and W3C Member testimonials, blog post from Doug Schepers, and get started on Web Platform Docs.
Microdata to RDF Note Published
09 October 2012 | Archive
The World Wide Web Consortium (W3C) is an international community that develops open standards to ensure the long-term growth of the Web. Read about the W3C mission.
JOBS
Software Quality Assurance (QA) Manager
Keio Business Development Lead
Web Accessibility Specialist
Web Accessibility Engineer
Membership Manager at the W3C UK/Ireland Office
W3C BLOG
DNT is Good For the Whole Web
2 October 2012 by Thomas Roessler
Test The Web Forward

The Semantic Web

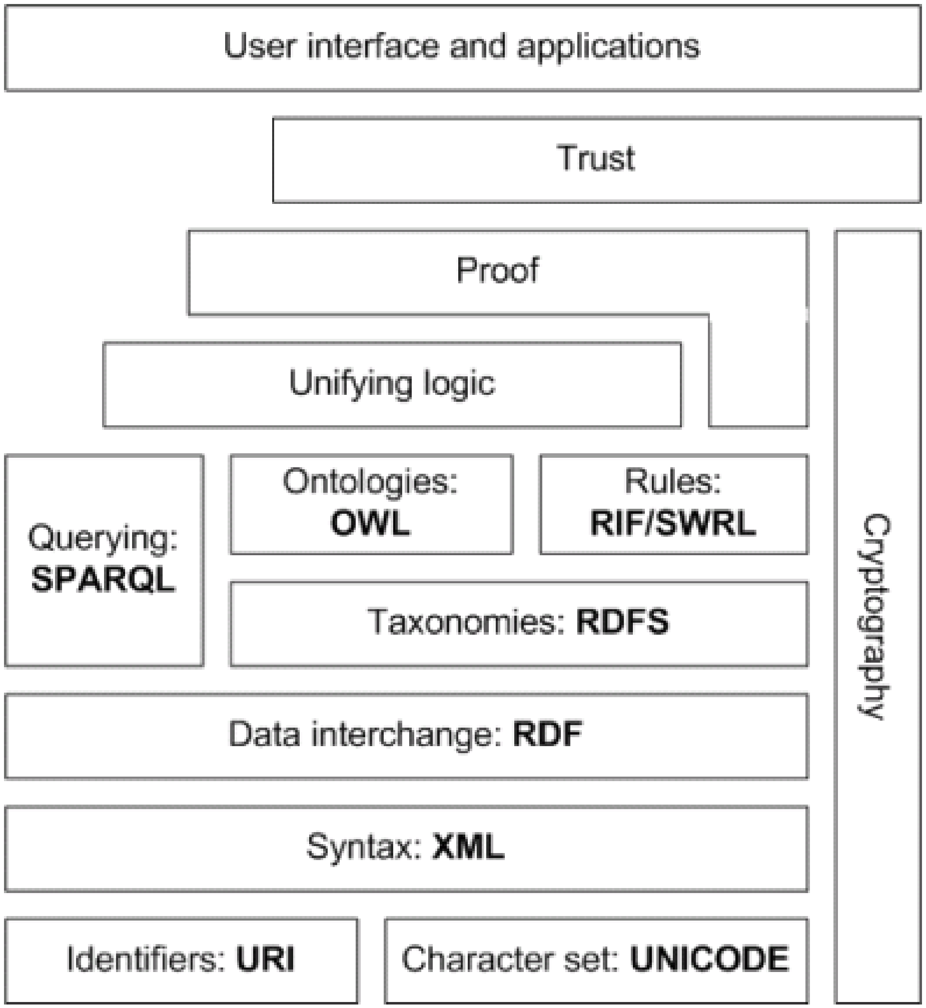
"In addition to the classic “Web of documents” W3C is helping to build a technology stack to support a “Web of data,” the sort of data you find in databases. "

"The ultimate goal of the Web of data is to enable computers to do more useful work and to develop systems that can support trusted interactions over the network. The term “Semantic Web” refers to W3C’s vision of the Web of linked data. "

"Semantic Web technologies enable people to create data stores on the Web, build vocabularies, and write rules for handling data. Linked data are empowered by technologies such as RDF, SPARQL, OWL, and SKOS."

www.w3.org/standards/semanticweb

Semantic Web Diagram



RDF
Resource Description Framework

RDFS
RDF Schema

OWL
Web Ontology Language

RIF
Rule Interchange Format

SPARQL
An RDF query language

http://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

Representing Text

A huge amount of the data transmitted across the web is in the form of text. Why?

A huge amount of the data that people use in the real world is in the form of text.

This means that an accurate and compact way of storing and transmitting text, in digital form, is of great importance.

ASCII

American Standard Code for Information Interchange

Originally a 7 bit code which included codes for 33 control characters and 95 printable characters

The printable characters were those of the English alphabet

The control characters related to the electro-mechanical teletype termimals, and fell out of use when VDUs became widespread.

7 bit ASCII

0	0000000	NULL	32	0100000		64	1000000	@	96	1100000	`
1	0000001	SOH	33	0100001	!	65	1000001	A	97	1100001	a
2	0000010	STX	34	0100010	"	66	1000010	B	98	1100010	b
3	0000011	ETX	35	0100011	#	67	1000011	C	99	1100011	c
4	0000100	EOT	36	0100100	$	68	1000100	D	100	1100100	d
5	0000101	ENQ	37	0100101	%	69	1000101	E	101	1100101	e
6	0000110	ACK	38	0100110	&	70	1000110	F	102	1100110	f
7	0000111	BEL	39	0100111	'	71	1000111	G	103	1100111	g
8	0001000	BS	40	0101000	(	72	1001000	H	104	1101000	h
9	0001001	HT	41	0101001	)	73	1001001	I	105	1101001	i
10	0001010	LF	42	0101010	*	74	1001010	J	106	1101010	j
11	0001011	VT	43	0101011	+	75	1001011	K	107	1101011	k
12	0001100	FF	44	0101100	,	76	1001100	L	108	1101100	l
13	0001101	CR	45	0101101	-	77	1001101	M	109	1101101	m
14	0001110	SOH	46	0101110	.	78	1001110	N	110	1101110	n
15	0001111	SI	47	0101111	/	79	1001111	O	111	1101111	o
16	0010000	DLE	48	0110000	0	80	1010000	P	112	1110000	p
17	0010001	DC1	49	0110001	1	81	1010001	Q	113	1110001	q
18	0010010	DC2	50	0110010	2	82	1010010	R	114	1110010	r
19	0010011	DC3	51	0110011	3	83	1010011	S	115	1110011	s
20	0010100	DC4	52	0110100	4	84	1010100	T	116	1110100	t
21	0010101	NAK	53	0110101	5	85	1010101	U	117	1110101	u
22	0010110	SYN	54	0110110	6	86	1010110	V	118	1110110	v
23	0010111	ETB	55	0110111	7	87	1010111	W	119	1110111	w
24	0011000	CAN	56	0111000	8	88	1011000	X	120	1111000	x
25	0011001	EM	57	0111001	9	89	1011001	Y	121	1111001	y
26	0011010	SUB	58	0111010	:	90	1011010	Z	122	1111010	z
27	0011011	ESC	59	0111011	;	91	1011011	[	123	1111011	{
28	0011100	FS	60	0111100	<	92	1011100	\	124	1111100	|
29	0011101	GS	61	0111101	=	93	1011101	]	125	1111101	}
30	0011110	RS	62	0111110	>	94	1011110	^	126	1111110	~
31	0011111	US	63	0111111	?	95	1011111	_	127	1111111	DEL

Extended ASCII

If we use 8 bit character codes, what does this mean about the number of characters we can represent?

00000000 – 01111111	Standard ASCII
10000000 – 11111111	Extended ASCII

The extra 128 codes can be used to represent extra characters such as:

ã ä ò ø ý ă Ě Ĝ	accented characters
¢ ¥ £	currency symbols (other than $)
© ® ¼ ½ ¾	other symbols

Unfortunately, there were a lot of proprietary variations on the extended character set.

ISO 8859

Eventually the ISO stepped in and produced a standard for 8 bit character codes.

So, for example, ISO 8859-1 is called Latin-1

Part 1	Latin-1 Western European
Part 2	Latin-2 Central European
Part 3	Latin-3 South European
Part 4	Latin-4 North European
Part 5	Latin plus Cyrillic
Part 6	Latin plus Arabic
Part 7	Latin plus Greek
Part 8	Latin plus Hebrew
Part 9	Latin-5 Turkish
Part 10	Latin-6 Nordic
Part 11	Latin plus Thai
Part 12	Latin plus Devanagari
Part 13	Latin-7 Baltic Rim
Part 14	Latin-8 Celtic
Part 15	Latin-9 variation of Latin-1 which includes the Euro symbol
Part 16	Latin-10 South-Eastern European

Using Character Encoding

This is why you will sometimes see this sort of thing at the top of an XML file:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>
```

or at the top of an HMTL file:

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

Other countries national standards organisations have produced character sets for their languages:

KS C 5601-1992	Korean
JIS X 0213	Japanese
PASCII	Perso-Arabic Indian languages (Kashmiri, Sindhi, Urdu)
GB18030	Chinese
Big5	Taiwan, Hong Kong, Macau

Unicode

An attempt to create a standard set of universal character codes for all languages.

Started in 1987 by Joe Becker (Xerox) and Mark Davies (Apple).

Current version: 8.0 (June 2015)

120,737 characters from 129 scripts

Each character is given a unique code point (number).

How that character is visually rendered (size, shape, font, style, glyph) is left to other software (browser, word processor).

The first 256 code points were made identical to ISO-8859-1 for backward compatibility.

Unicode Planes

00000000 – 0000FFFF	Basic Multilingual Plane Almost all modern languages	Plane 0
00010000 - 0001FFFF	Supplementary Multilingual Plane Historic languages also Mathematical Symbols	Plane 1
00020000 – 0002FFFF	Supplementary Ideographic Plane More Chinese/Japanese/Korean ideographs	Plane 2
00030000 – 0003FFFF	Plane 3 to	
:	: not used	
00100000 - 0010FFFF	Plane 16	

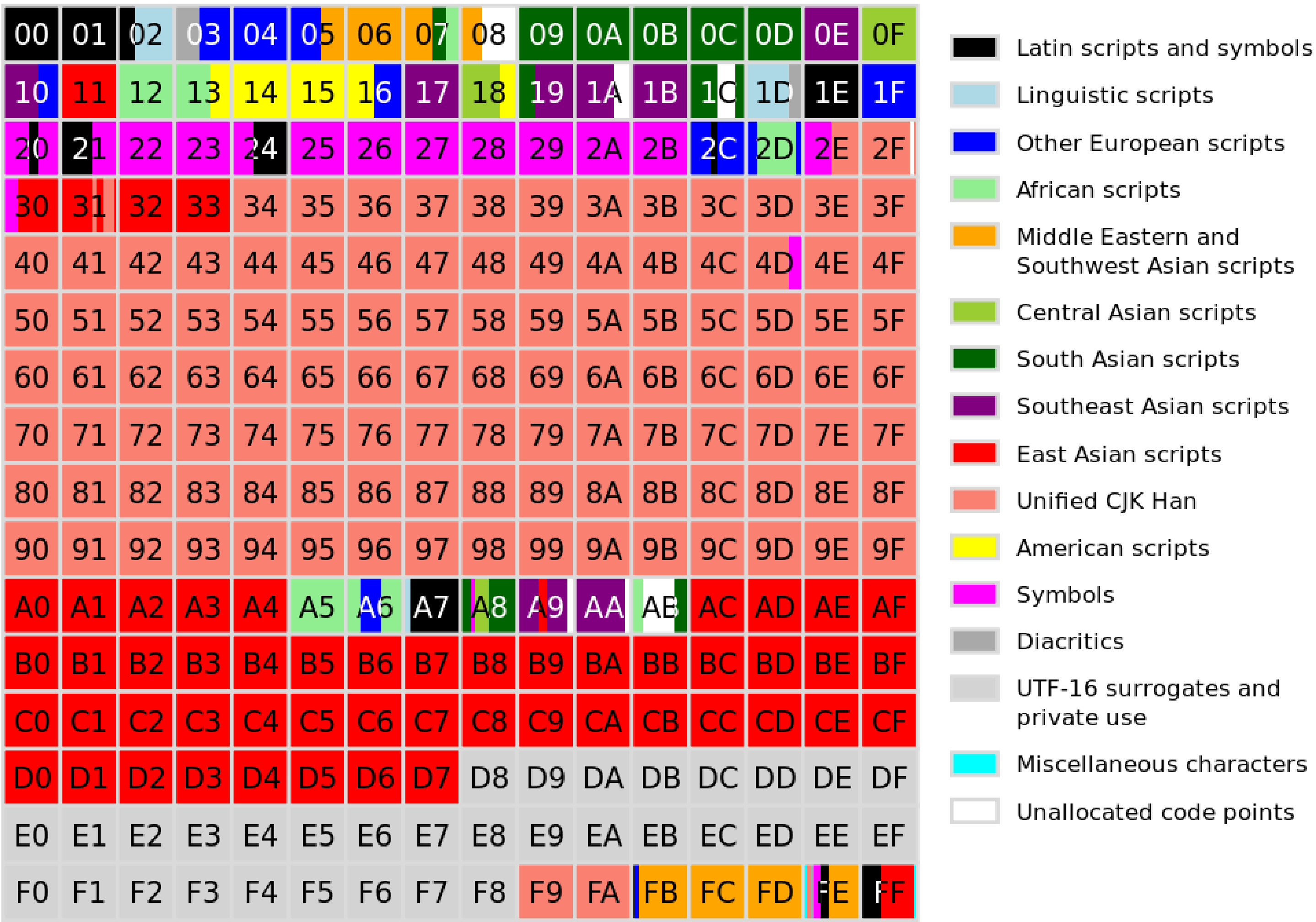
See en.wikipedia.org/wiki/Unicode_plane for details.

"After a few beers, some developers are willing to admit that they're preparing for a day when we're part of a Galactic Federation of thousands of intelligent species"

XML in a Nutshell, Page 75.

Plane 0

A map of the Basic Multilingual Plane.
Each numbered box represents 256 code points



The first part of Plane 0

C0 Controls and Basic Latin (0000–007F)
Latin-1 Supplement (0080–00FF)
Latin Extended-A (0100–017F)
Latin Extended-B (0180–024F)
IPA Extensions (0250–02AF)
Spacing Modifier Letters (02B0–02FF)
Combining Diacritical Marks (0300–036F)
Greek and Coptic (0370–03FF)
Cyrillic (0400–04FF)
Cyrillic Supplement (0500–052F)
Armenian (0530–058F)
Hebrew (0590–05FF)
Arabic (0600–06FF)
Syriac (0700–074F)
Arabic Supplement (0750–077F)
Thaana (0780–07BF)
N'Ko (07C0–07FF)
Samaritan (0800–083F)
Mandaic (0840–085F)
Arabic Extended-A (08A0–08FF)
Devanagari (0900–097F)
Bengali (0980–09FF)
Gurmukhi (0A00–0A7F)
Gujarati (0A80–0AFF)
Oriya (0B00–0B7F)
Tamil (0B80–0BFF)
Telugu (0C00–0C7F)
Kannada (0C80–0CFF)
Malayalam (0D00–0D7F)
Sinhala (0D80–0DFF)
Thai (0E00–0E7F)
Lao (0E80–0EFF)
Tibetan (0F00–0FFF)
Myanmar (1000–109F)
Georgian (10A0–10FF)
Hangul Jamo (1100–11FF)
Ethiopic (1200–137F)
Ethiopic Supplement (1380–139F)
Cherokee (13A0–13FF)
Canadian Aboriginal Syllabics (1400–167F)
Ogham (1680–169F)
Runic (16A0–16FF)
Philippine scripts:
Tagalog (1700–171F)
Hanunoo (1720–173F)
Buhid (1740–175F)
Tagbanwa (1760–177F)
Khmer (1780–17FF)
Mongolian (1800–18AF)
Canadian Aboriginal Extended (18B0–18FF)
Limbu (1900–194F)
Tai Le (1950–197F)
Tai Lue (1980–19DF)
Khmer Symbols (19E0–19FF)
Buginese (1A00–1A1F)
Tai Tham (1A20–1AAF)

….. and so on

Plane 1

Supplementary Multilingual Plane

Linear B Syllabary (10000–1007F)
Linear B Ideograms (10080–100FF)
Aegean Numbers (10100–1013F)
Ancient Greek Numbers (10140–1018F)
Ancient Symbols (10190–101CF)
Phaistos Disc (101D0–101FF)
Lycian (10280–1029F)
Carian (102A0–102DF)
Old Italic (10300–1032F)
Gothic (10330–1034F)
Ugaritic (10380–1039F)
Old Persian (103A0–103DF)
Deseret (10400–1044F)
Shavian (10450–1047F)
Osmanya (10480–104AF)
Cypriot Syllabary (10800–1083F)

Imperial Aramaic (10840–1085F)
Phoenician (10900–1091F)
Lydian (10920–1093F)
Meroitic Hieroglyphs (10980–1099F)
Meroitic Cursive (109A0–109FF)
Kharoshthi (10A00–10A5F)
Old South Arabian (10A60–10A7F)
Avestan (10B00–10B3F)
Inscriptional Parthian (10B40–10B5F)
Inscriptional Pahlavi (10B60–10B7F)
Old Turkic (10C00–10C4F)
Rumi Numeral Symbols (10E60–10E7F)
Brahmi (11000–1107F)
Kaithi (11080–110CF)
Sora Sompeng (110D0–110FF)
Chakma (11100–1114F)
Sharada (11180–111DF)
Takri (11680–116CF)
Cuneiform (12000–123FF)
Cuneiform Numbers and Punctuation (12400–1247F)
Egyptian Hieroglyphs (13000–1342F)
Bamum Supplement (16800–16A3F)
Miao (16F00–16F9F)

Kana Supplement (1B000–1B0FF)
Byzantine Musical Symbols (1D000–1D0FF)
Musical Symbols (1D100–1D1FF)
Ancient Greek Musical Notation (1D200–1D24F)
Tai Xuan Jing Symbols (1D300–1D35F)
Counting Rod Numerals (1D360–1D37F)
Mathematical Alphanumeric Symbols (1D400–1D7FF)
Arabic Mathematical Symbols (1EE00–1EEFF)
Mahjong Tiles (1F000–1F02F)
Domino Tiles (1F030–1F09F)
Playing Cards (1F0A0–1F0FF)
Enclosed Alphanumeric Supplement (1F100–1F1FF)
Enclosed Ideographic Supplement (1F200–1F2FF)
Miscellaneous Symbols And Pictographs (1F300–1F5FF)
Emoticons (1F600–1F64F)
Transport And Map Symbols (1F680–1F6FF)
Alchemical Symbols (1F700–1F77F)

Unicode Encoding

Given that each character has a unique code point number, there are a variety of ways of encoding (packaging) this number.

UTF-32

Fixed length method. Store the code point number in 4 bytes. This allows a direct representation of the code point. It doesn't need to be 'packaged'

Disadvantages?

UTF-8

UTF-16

Variable length methods. Store the code point number in a variable number of bytes, depending on the size of the code point value.

UTF-8 Encoding

Bits	Last code point	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6
7	U+007F	0xxxxxxx					
11	U+07FF	110xxxxx	10xxxxxx				
16	U+FFFF	1110xxxx	10xxxxxx	10xxxxxx			
21	U+1FFFFF	11110xxx	10xxxxxx	10xxxxxx	10xxxxxx		
26	U+3FFFFFF	111110xx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	
31	U+7FFFFFFF	1111110x	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx	10xxxxxx

1. One-byte codes are used only for the ASCII values 0 through 127. In this case the UTF-8 code has the same value as the ASCII code. The high-order bit of these codes is always 0.

2. Codepoints larger than 127 are represented by multi-byte sequences, composed of a *leading byte* and one or more *continuation bytes*. The leading byte has two or more high-order 1s followed by a 0, while continuation bytes all have '10' in the high-order position.

3. The remaining bits of the encoding are used for the bits of the codepoint being encoded, padded with high-order 0s if necessary. The number of bytes in the encoding is the minimum required to hold all the significant bits of the codepoint.

4. The number of high-order 1s in the leading byte of a multi-byte sequence indicates the number of bytes in the sequence, so that the length of the sequence can be determined without examining the continuation bytes.

5. Three bytes are needed for characters in the rest of the Basic Multilingual Plane (which contains virtually all characters in common use