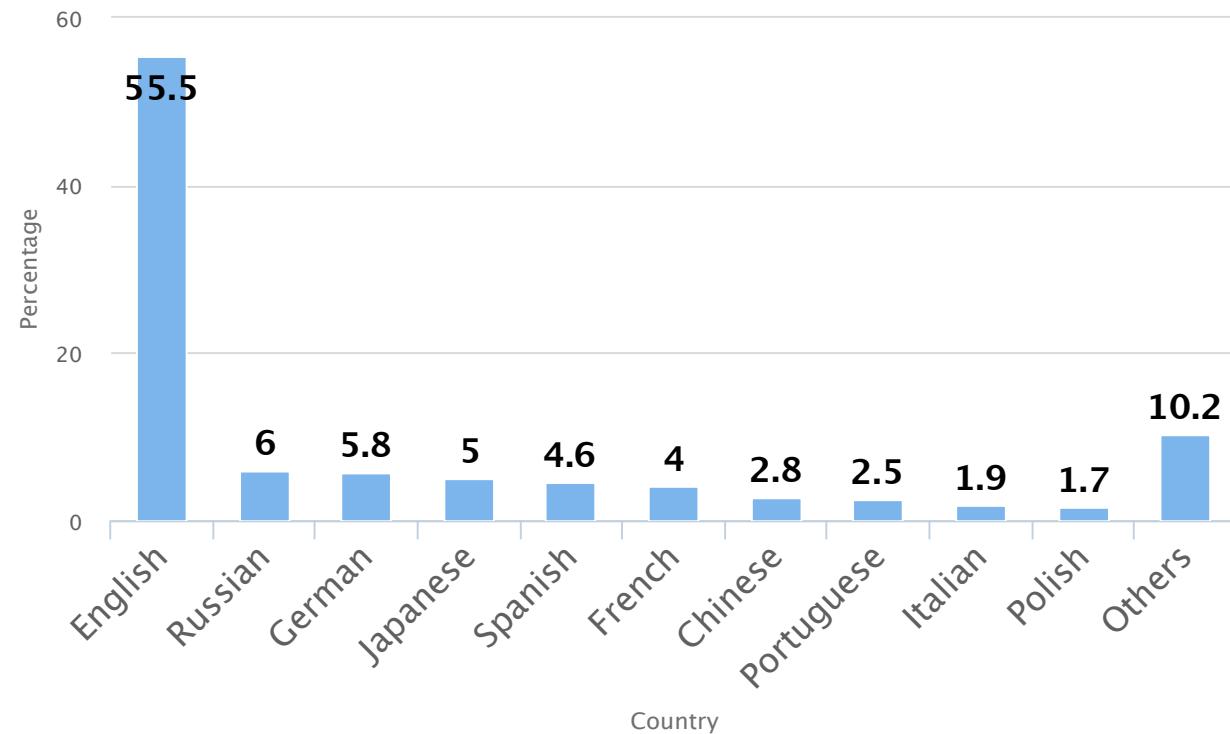


information in a global world

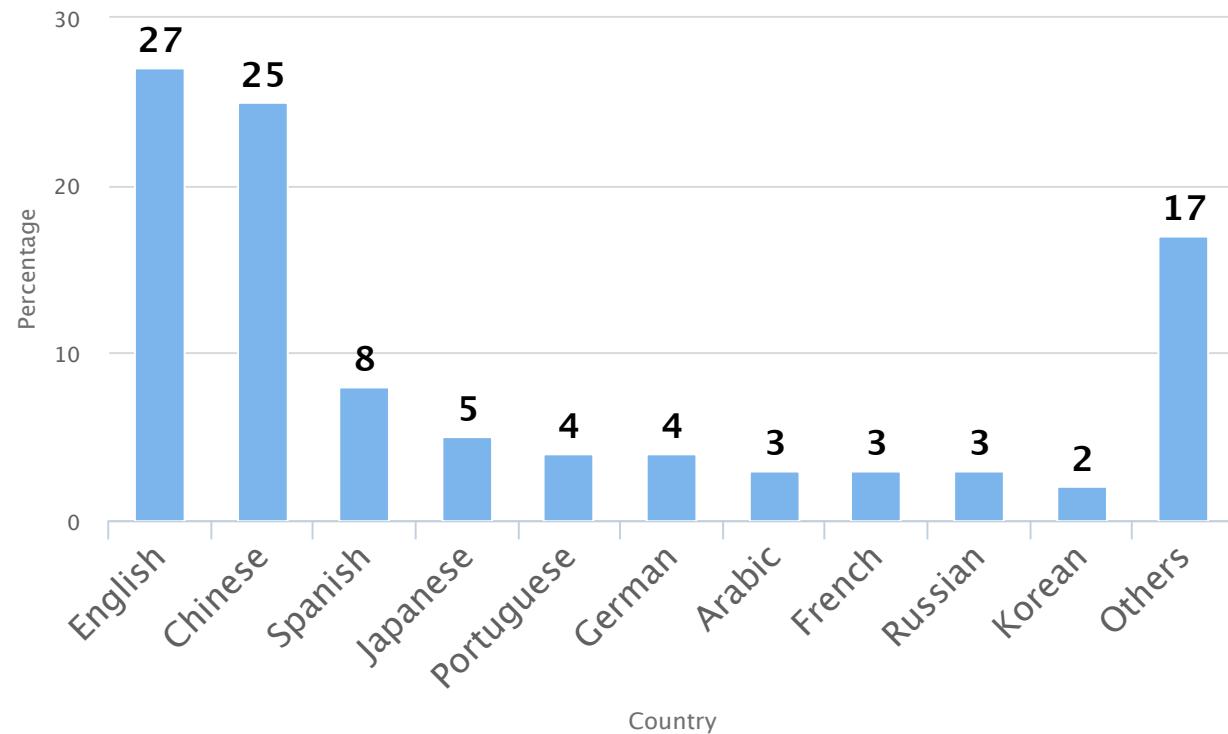
info-20002: foundations of informatics

Diversity of languages of website content



W3Techs.com. Usage of content languages for websites. Retrieved Mar 2015.

Diversity of languages of website users



Internet World Stats. [Number of Internet Users by Language as of 31 May 2011.](#)



Scumandà da traversar ils binaris !
Überschreiten der Gleise verboten !
Vietato attraversare i binari !
Do not cross the railway lines !
線路を横切ってはいけません !

<http://www.un.org>

مرحباً • 欢迎光临 • Welcome • Bienvenue • Добро пожаловать • Bienvenidos

2016 rio olympics

<http://www.rio2016.com/en>



i18n & L10n

localization

modifying or adapting an application to fit the requirements of a particular locale

a *locale* is a collection of parameters that defines a specific language, country, script, rules, and coded character set

internationalization

designing an application so that it can operate in multiple locales without any engineering change to the application.

[IBM Software - Globalize your business](#)

[Microsoft - Globalization step by step](#)

issues in i18n & L10n

characters set & encoding	Welcome or Καλώς Ήρθατε
language translations	Hello or Bonjour
currency format	\$ £ € ¥
numerical format	1,457 or 1.457
date/time format	08/09/2008 (<i>8 September</i> or <i>9 August</i> ?)

character sets

A character is a indivisible unit of text. A character set defines the range of a character collection. Examples:

charset	num. of chars	encoding
ASCII (Basic Latin)	128	7-bit ASCII
ISO-8859-1 (Latin-1 Western European)	191	8-bit ISO-8859-1
Windows-1252	251	8-bit Windows-1252
GB2312 (Simplified Chinese)	6,763	EUC-CN, HZ
Unicode	1,114,112	UTF-8, UTF-16, UTF-32

NUL 000000	SOH 000101	STX 000202	ETX 000303	EOT 000404	ENQ 000505	ACK 000606	BEL 000707	BS 000808	HT 000909	LF 000A0A	VT 000B0B	FF 000C0C	CR 000D0D	SO 000E0E	SI 000F0F
DLE 001010	DC1 001111	DC2 001212	DC3 001313	DC4 001414	NAK 001515	SYN 001616	ETB 001717	CAN 001818	EM 001919	SUB 001A1A	ESC 001B1B	FS 001C1C	GS 001D1D	RS 001E1E	US 001F1F
SP 002020	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0 003030	1 003131	2 003232	3 003333	4 003434	5 003535	6 003636	7 003737	8 003838	9 003939	:	;	<	=	>	?
@ 004040	A 004141	B 004242	C 004343	D 004444	E 004545	F 004646	G 004747	H 004848	I 004949	J 004A4A	K 004B4B	L 004C4C	M 004D4D	N 004E4E	O 004F4F
P 005050	Q 005151	R 005252	S 005353	T 005454	U 005555	V 005656	W 005757	X 005858	Y 005959	Z 005A5A	[\]	^	—
~ 006060	a 006161	b 006262	c 006363	d 006464	e 006565	f 006666	g 006767	h 006868	i 006969	j 006A6A	k 006B6B	l 006C6C	m 006D6D	n 006E6E	o 006F6F
p 007070	q 007171	r 007272	s 007373	t 007474	u 007575	v 007676	w 007777	x 007878	y 007979	z 007A7A	{		}	~	DEL 007F7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
Ao	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
Bo	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
Co	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
Do	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
Eo	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF
Fo	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

ASCII

```
! "#$%& ' ()*+, -./0123456789: ;<=>?@ABCDEFGHIJKLMNPQRSTUVWXYZ[ \ ]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

- American Standard Code for Information Interchange
- Supports Basic Latin characters
- Limited support for European languages
- é in *café* is not supported

How can we support other languages:

- Extended Latin charsets (Windows-1252 or ISO-8859-1 (Latin-1))
- Various charsets:
 - Shift-JIS: Japanese
 - Windows-1251: Cyrillic (Russian, Bulgarian, etc.)
 - Big5, GB2312: Chinese
- Problems: overlaps across encodings/representation of these proprietary character sets.

Comparison of ASCII, Windows 1252, and ISO-8859-1 (click to cycle)

NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0000 00	0001 01	0002 02	0003 03	0004 04	0005 05	0006 06	0007 07	0008 08	0009 09	000A 0A	000B 0B	000C 0C	000D 0D	000E 0E	000F 0F
DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010 10	0011 11	0012 12	0013 13	0014 14	0015 15	0016 16	0017 17	0018 18	0019 19	001A 1A	001B 1B	001C 1C	001D 1D	001E 1E	001F 1F
SP	!	"	#	\$	%	&	'	()	*	+	,	-	.	/
0020 20	0021 21	0022 22	0023 23	0024 24	0025 25	0026 26	0027 27	0028 28	0029 29	002A 2A	002B 2B	002C 2C	002D 2D	002E 2E	002F 2F
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0030 30	0031 31	0032 32	0033 33	0034 34	0035 35	0036 36	0037 37	0038 38	0039 39	003A 3A	003B 3B	003C 3C	003D 3D	003E 3E	003F 3F
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0040 40	0041 41	0042 42	0043 43	0044 44	0045 45	0046 46	0047 47	0048 48	0049 49	004A 4A	004B 4B	004C 4C	004D 4D	004E 4E	004F 4F
P	Q	R	S	T	U	V	W	X	Y	Z	[\]	^	-
0050 50	0051 51	0052 52	0053 53	0054 54	0055 55	0056 56	0057 57	0058 58	0059 59	005A 5A	005B 5B	005C 5C	005D 5D	005E 5E	005F 5F
‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0060 60	0061 61	0062 62	0063 63	0064 64	0065 65	0066 66	0067 67	0068 68	0069 69	006A 6A	006B 6B	006C 6C	006D 6D	006E 6E	006F 6F
p	q	r	s	t	u	v	w	x	y	z	{		}	~	DEL
0070 70	0071 71	0072 72	0073 73	0074 74	0075 75	0076 76	0077 77	0078 78	0079 79	007A 7A	007B 7B	007C 7C	007D 7D	007E 7E	007F 7F
80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F
90	91	92	93	94	95	96	97	98	99	9A	9B	9C	9D	9E	9F
A0	A1	A2	A3	A4	A5	A6	A7	A8	A9	AA	AB	AC	AD	AE	AF
B0	B1	B2	B3	B4	B5	B6	B7	B8	B9	BA	BB	BC	BD	BE	BF
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
D0	D1	D2	D3	D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF
E0	E1	E2	E3	E4	E5	E6	E7	E8	E9	EA	EB	EC	ED	EE	EF

Unicode

- Provide a universal character set that encapsulates almost all characters in the world's languages
- A collection of abstract characters, each identified by an integer/number called **code point** and **unique name** (coded character set)
- In Unicode, code point is NOT the way computer represents (encodes) the character, but it is used to identify the character

é U+00E9 'LATIN SMALL LETTER E WITH ACUTE'

- 00E9 is the code point of the character
- LATIN SMALL LETTER E WITH ACUTE is the name of the character

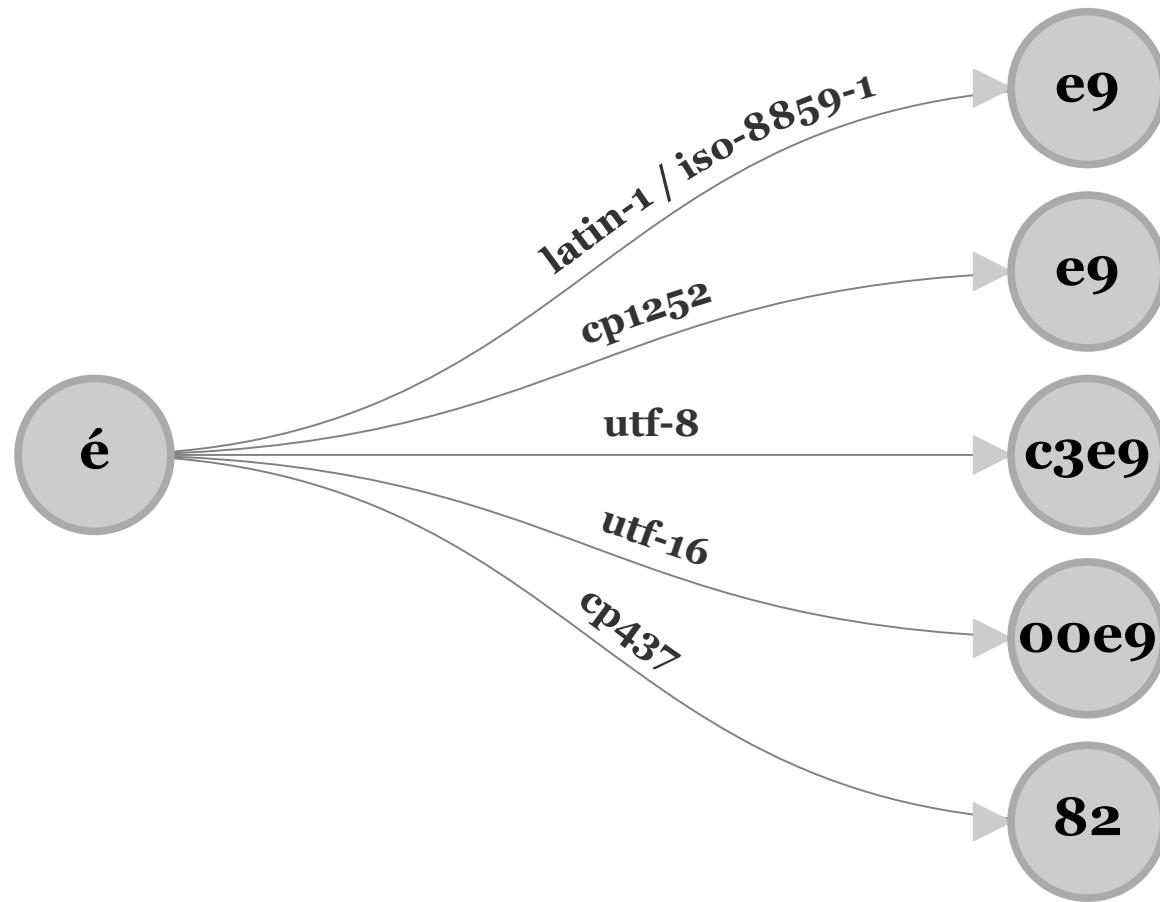
character encoding

- A character is NOT the physical (binary) representation in a file, in a web page or in a device. A character is an abstract entity, its actual representation is determined by **character encoding**; encoding maps a logical character to a numerical value
- The terms character set and encoding are often used interchangeably (while it is not a problem for ASCII & Windows 1252, it is problematic for Unicode)

character encoding

- Encoding is a mapping from a logical representation of a character to its physical representation
- Encoding maps a character to (one or more) bytes

various encodings of é



encoding of input and output

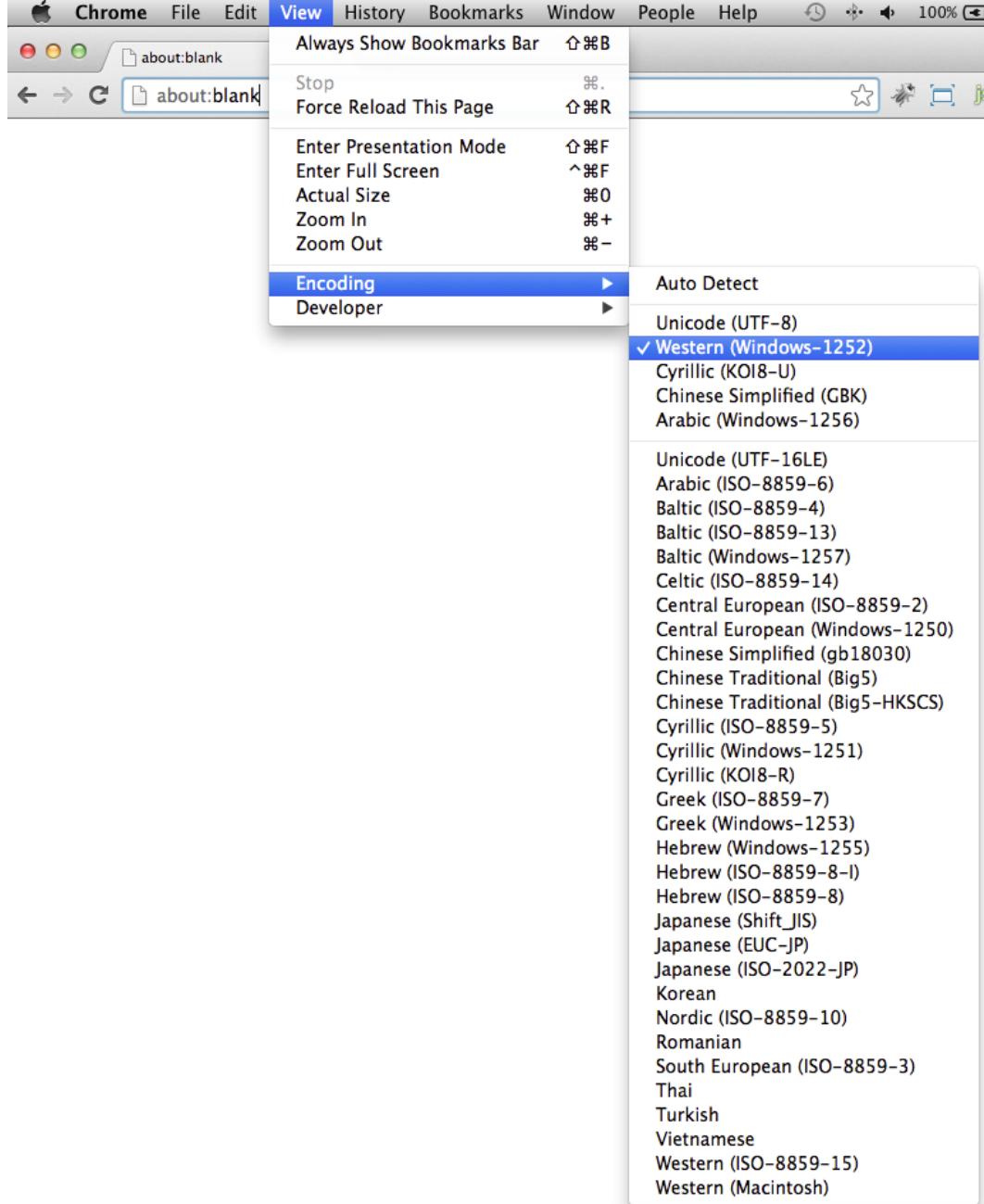
- Encoding is needed for interpreting a piece of data: text file, XML file, web page (which are blobs of bytes)
- Programmers need to know the encoding of data that goes into or comes from a device (keyboard, screen, or web browser)

Example: it's best to specify the encoding used in your HTML page.

```
<DOCTYPE html>
<html>
<head>
  <meta http-equiv="content-type"
        content="text/html; charset=UTF-8" />
</head>
<body>
  ...

```

chrome browser encoding



me File Edit

View History Bookmarks Window People Help

100% 100% Sat 8:00 pm Ivo Widjaja

about:blank

about:blank

Always Show Bookmarks Bar ⌘B

Stop ⌘.

Reload This Page ⌘R

Enter Presentation Mode ⌘F

Enter Full Screen ⌘F

Actual Size ⌘0

Zoom In ⌘+

Zoom Out ⌘-

Encoding ▶

Developer ▶

Auto Detect

Unicode (UTF-8) ✓

Western (Windows-1252)

Cyrillic (KOI8-U)

Chinese Simplified (GBK)

Arabic (Windows-1256)

Unicode (UTF-16LE)

Arabic (ISO-8859-6)

Baltic (ISO-8859-4)

Baltic (ISO-8859-13)

Baltic (Windows-1257)

Celtic (ISO-8859-14)

Central European (ISO-8859-2)

Central European (Windows-1250)

Chinese Simplified (gb18030)

Chinese Traditional (Big5)

Chinese Traditional (Big5-HKSCS)

Cyrillic (ISO-8859-5)

Cyrillic (Windows-1251)

Cyrillic (KOI8-R)

Greek (ISO-8859-7)

Greek (Windows-1253)

Hebrew (Windows-1255)

Hebrew (ISO-8859-8-I)

Hebrew (ISO-8859-8)

Japanese (Shift_JIS)

Japanese (EUC-JP)

Japanese (ISO-2022-JP)

Korean

Nordic (ISO-8859-10)

Romanian

South European (ISO-8859-3)

Thai

Turkish

Vietnamese

新闻 hao123 地图 视频 贴吧 登录 设置

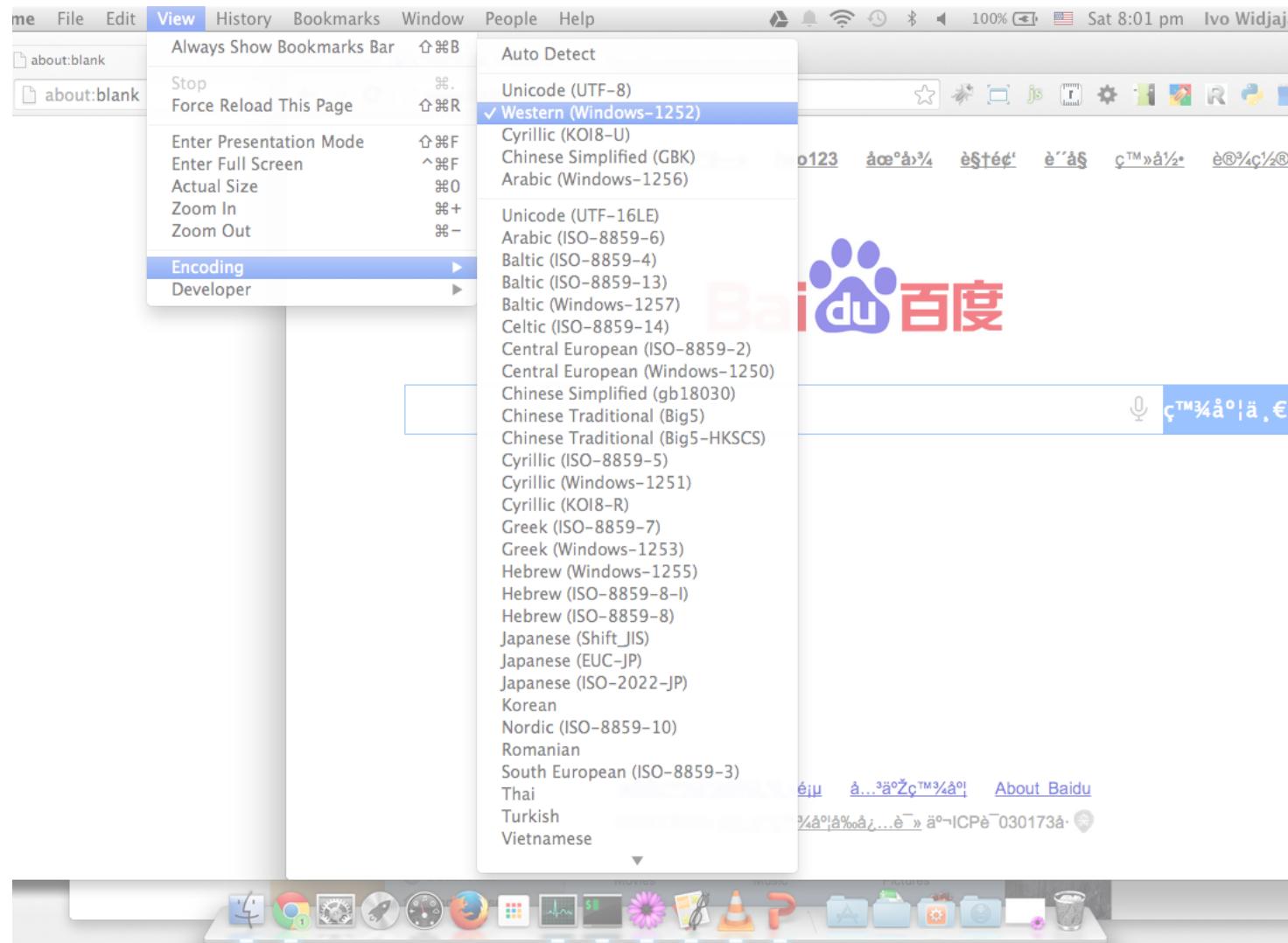


百度一下

关于百度 About Baidu

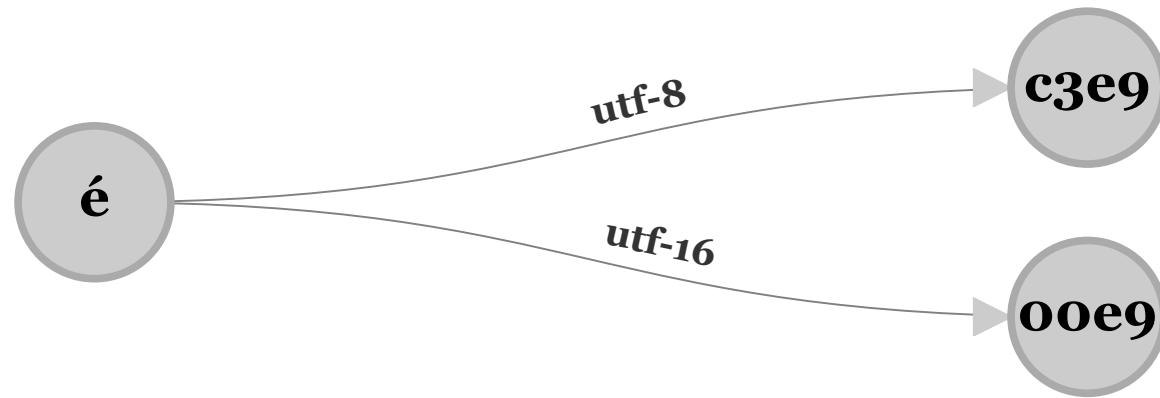
百度前必读 京ICP证030173号





Unicode encoding

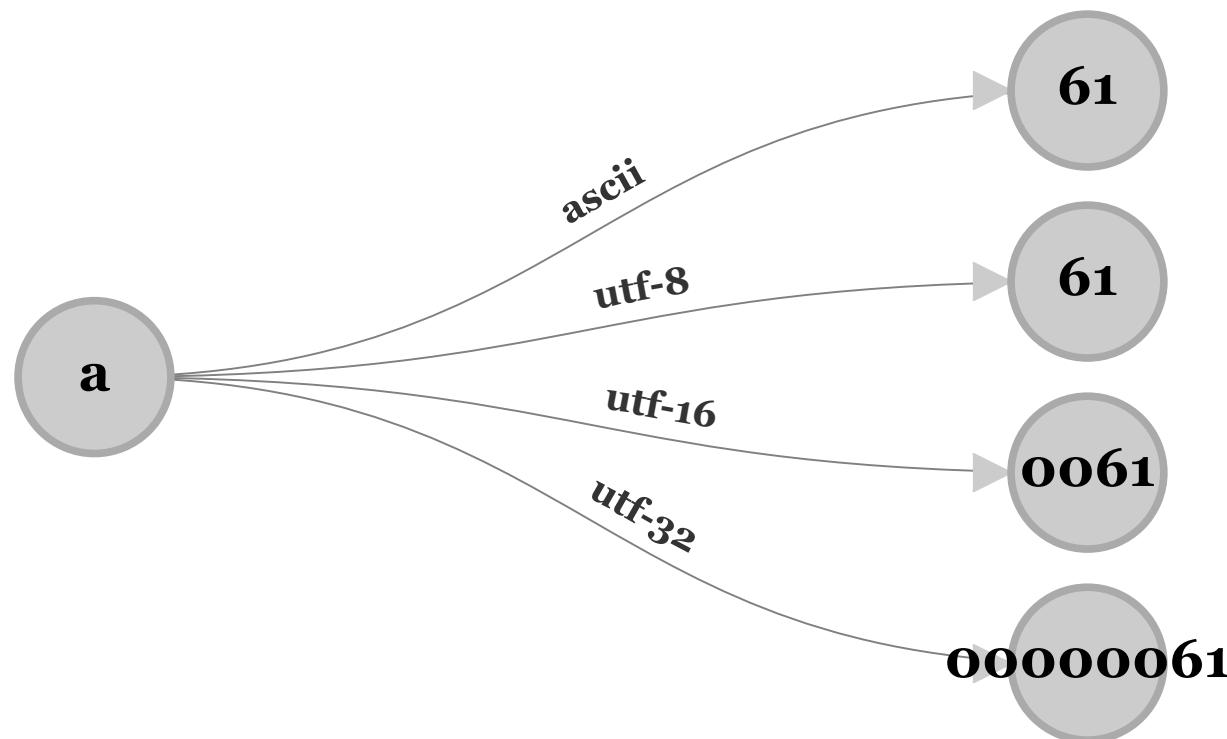
- ASCII characters are encoded as 1 byte (a single way of encoding)
- Unicode uses variable length encoding: 1-4 bytes (multiple ways of encoding: UTF-8, UTF-16, UTF-32)



- **é** (code point U+00E9) belongs to [Latin-1 Supplement group in Unicode](#).
- [Unicode encoder](#)
- [Mapping codepoints to Unicode encoding](#)

UTF-8 encoding

- The most commonly used encoding is UTF-8
- Backward compatible with ASCII; all ASCII data are also UTF-8 encoded



PROTECTED PLACE

NO
ADMITTANCE
TO UNAUTHORISED
PERSONS

LARANG MASOK
JIKA TIADA
KEBENARAN



EMPAT LARANGAN

பாதுகாப்பு உள்ள இட

保护区

闲人
免进

உத்தரவின்டு
உள்ளே
பிரவேசிக்க
கூடாது

Python and Unicode

- Python supports (1) **byte string** and (2) **Unicode string**
- Unicode string starts with u prefix

```
>>> e_acute = u'\u00e9' # Unicode string
>>> e_acute.encode('utf-8')
'\xc3\xa9' # byte string
```

- Unicode string is an abstract or logical representation of characters.
- Python byte string is a physical representation.
- Sending to a file, web page, device, you need to encode.

encoding in Python

To convert Unicode string to byte string use `encode(encoding)` function:

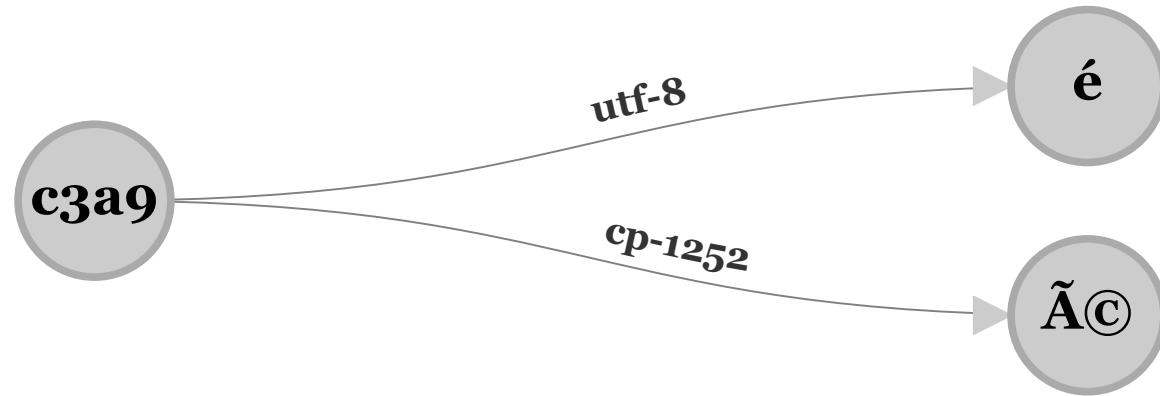
```
>>> e_acute = u'\u00e9'
>>> e_acute.encode('utf-8')
'\xc3\xaa'
>>> e.encode('latin-1')
'\xe9'
>>> e.encode('cp1252')
'\xe9'
```

decoding in Python

To convert byte string to Unicode string use `decode(encoding)` function:

```
>>> something = '\xc3\xa9'  
>>> print something.decode('utf-8')  
é  
>>> print something.decode('cp1252')  
Ã©
```

decode: interpreting bytes



consuming unicode

demo.1.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<demo>
  <sentence lang="chinese">我爱你</sentence>
  <sentence lang="kazakh">Фыууда</sentence>
  <sentence lang="iceland">Björk</sentence>
</demo>
```

consuming utf-8 xml

lxml's etree provides Unicode string where necessary

```
from lxml import etre
etre = etre.parse( "demo-1.xml" )
root = tree.getroot()
for sentence in root:
    print type(sentence.text) # the type of is <unicode>
```

producing web page in utf-8

```
str = '''<html>
<head>
    <meta http-equiv="content-type" content="text/html; charset=UTF-8">
    <title>Unicode demo</title>
</head>
<body>
    ...
    # specify the web page is in UTF-8,
    # so that the browser can interpret correctly
    ...
    <h1>Hello, 世界!</h1>
    <h2>This page is in UTF-8!</h2>
</body>
</html>'''
```

producing data in utf-8 (output)

```
**UnicodeEncodeError: 'ascii' codec can't encode characters in
position 0-2: ordinal not in range(128)**
```

issue

- Everytime you send a Unicode string to a device, a network, or a disk (e.g. using `print` or `file.write()`), the *Unicode string* needs to be encoded to as *byte string*. If programmer do not do that explicitly, Python does this for you automatically.
- If no encoding is specified, it uses the default encoding (sometimes `ascii`).

```
>>> import sys
>>> sys.getdefaultencoding()
'ascii'
```

- Not all Unicode characters can be encoded in target encoding (e.g. ASCII (0-127))

Solution: Explicitly encode using supported encoding (e.g UTF-8).

```
...
print a_unicode_string.encode('utf-8')
...
```

supporting internationalization in Python

- Logically store all characters in Unicode
- Perform all string processing in Unicode string
- **Decode early:** Data from file or from user input should be stored/processed as Unicode string (decode your python byte string to Unicode string).
- **Encode last:** If you have to serialize a Unicode string to a device (to a web page using print or to a file using `write()`), encode your Unicode strings using appropriate encoding supported by the device (e.g. "utf-8").
- In receiving/displaying characters from user input/output, you need to know the supported encoding of the I/O device (keyboard, screen, or web browser).

locale

- support software ability to deal with certain cultural issues in various locales (countries) without requiring the programmer to know all the detail about those locales
- the issues: formatting of numerical value, monetary value, and date
- use Python's `locale` module

```
>>> import locale
>>> import time
>>> locale.setlocale(locale.LC_ALL, 'English_Australia')
'English_Australia.1252'
>>> time.strftime('%X %x %A') # time, date, day of week
'5:40:19 PM 16/09/2008 Tuesday'
>>> locale.setlocale(locale.LC_ALL, 'English_US')
'English_United States.1252'
>>> time.strftime('%X %x %A')
'5:40:35 PM 9/16/2008 Tuesday'
```

managing translation

English

Please enter your name

Italian

Il vostro nome per favore

Sinhalese

ආදාළඟාවන්

lost in translations

```
...
fs = cgi.FieldStorage()
lang = fs.get_first("lang")
if lang == "English":
    print "Hello"
elif lang == "French":
    print "Bonjour"
elif lang == "Italian":
    print "Alo"
...
```

separation of concerns

