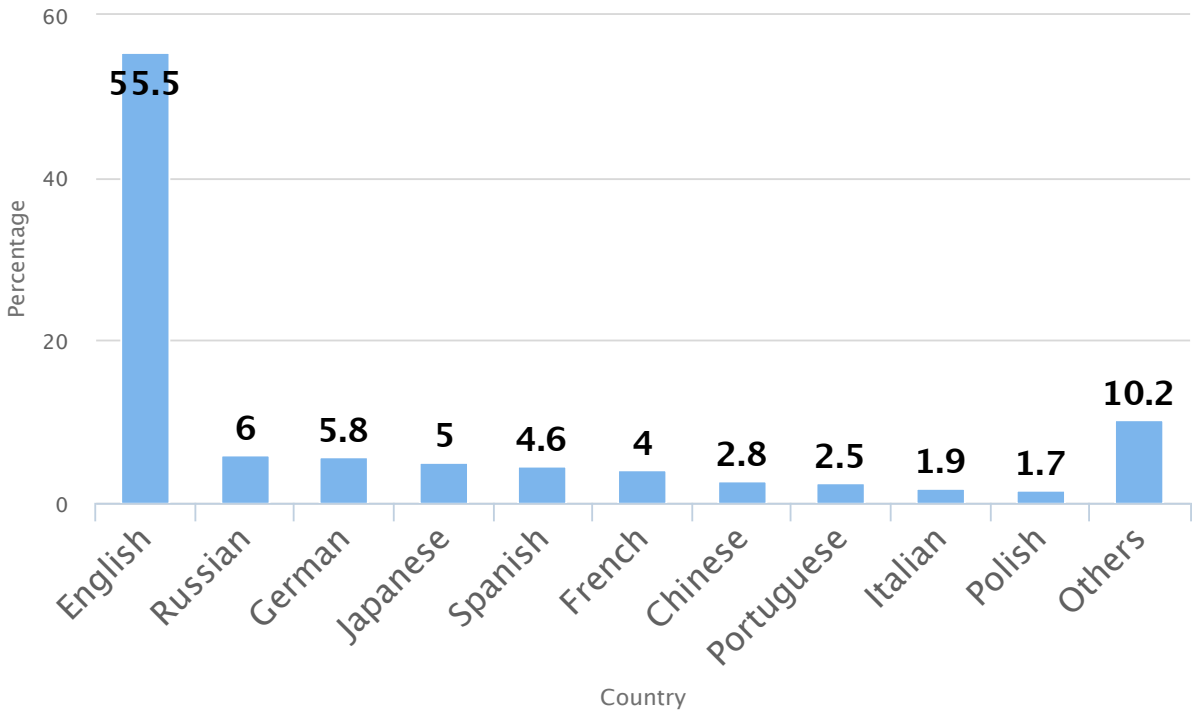
information in a global world

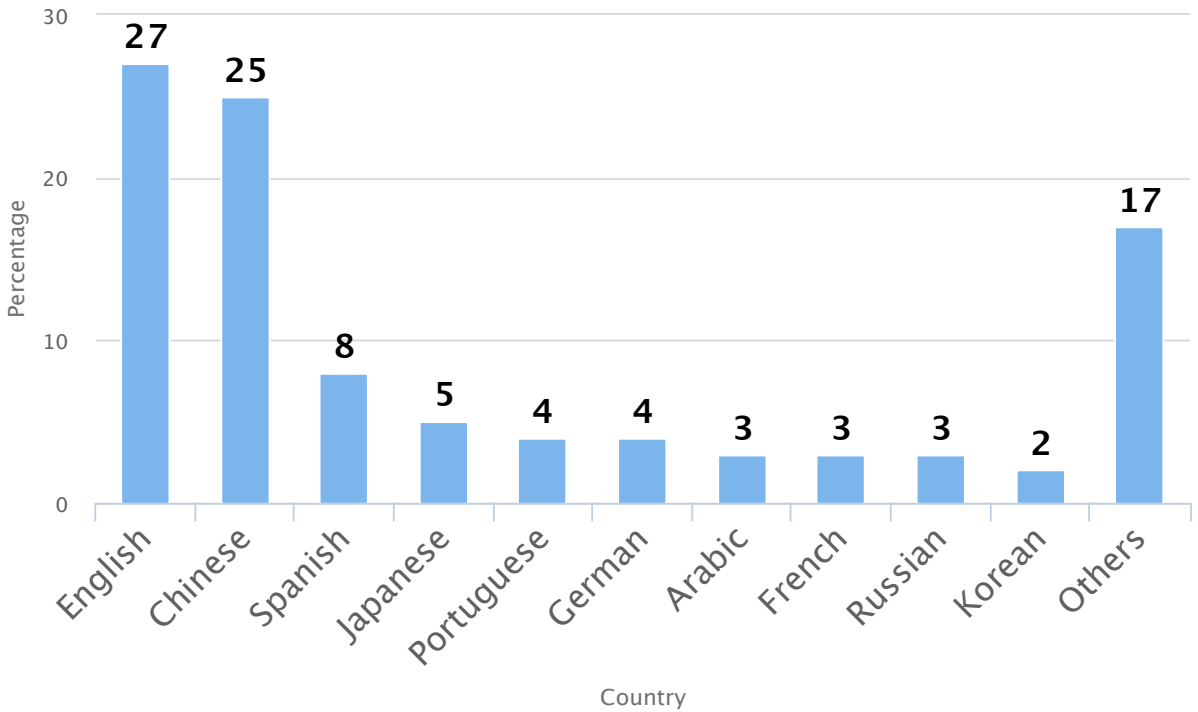
info-20002: foundations of informatics

Diversity of languages of website content

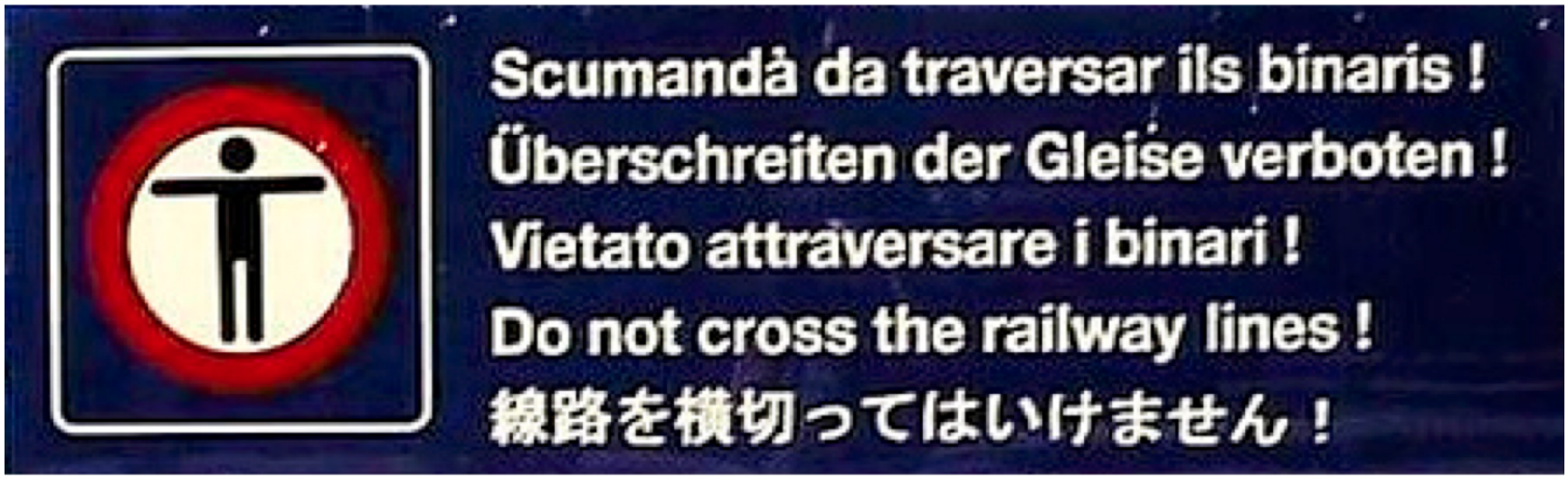


W3Techs.com. **Usage of content languages for websites.** Retrieved Mar 2015.

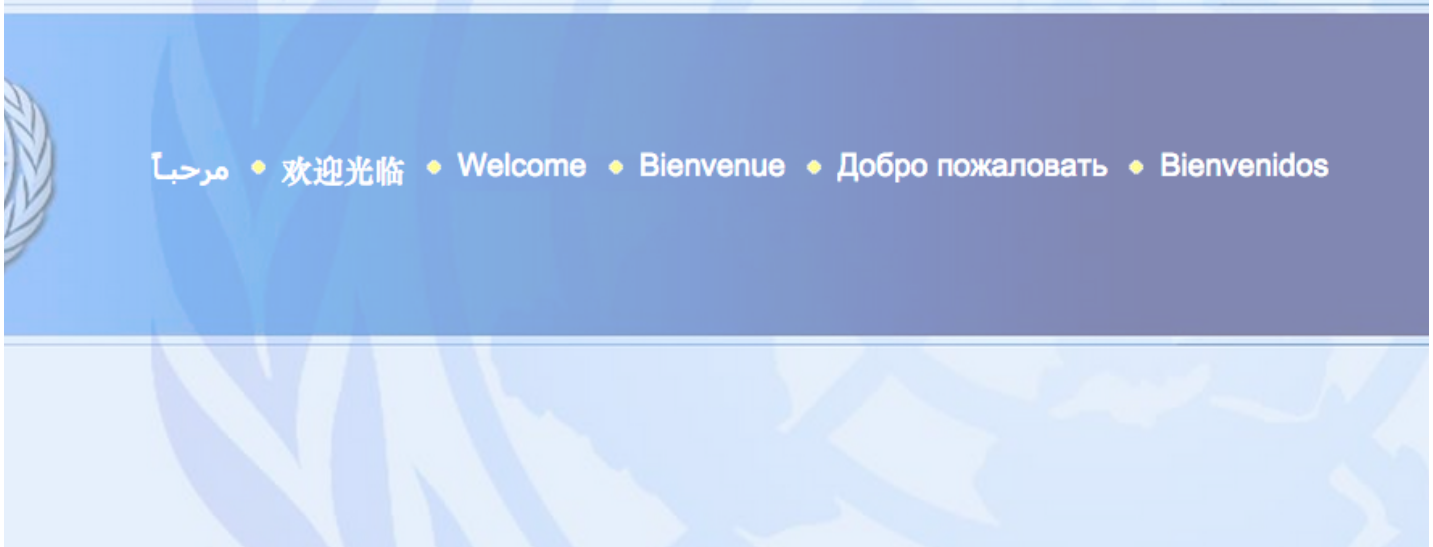
Diversity of languages of website users



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



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 the 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

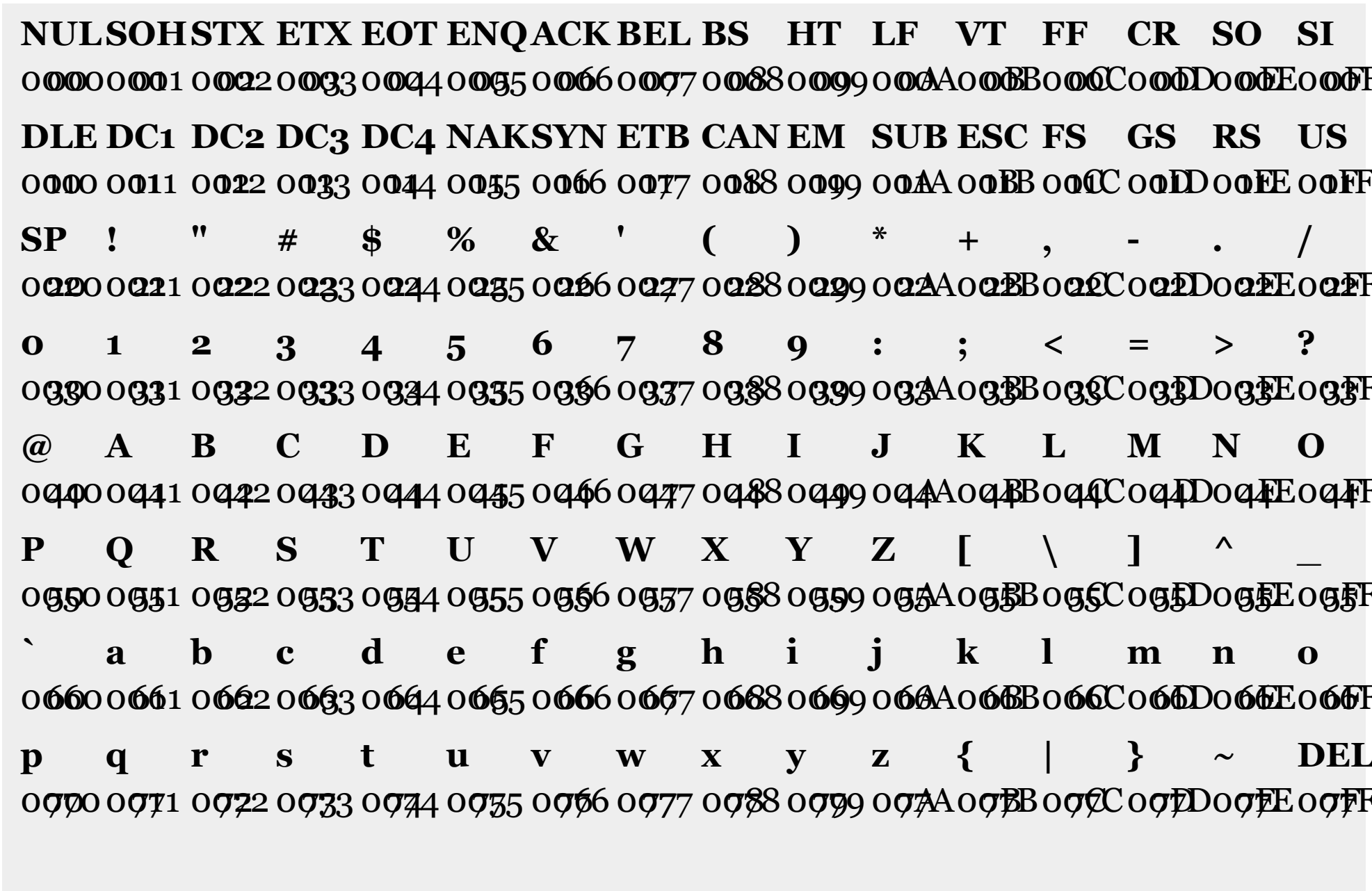
issues in i18n & L10n

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 (*8 September* or *9 August*?)

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 SOH STX ETX EOT ENQ ACK BEL BS HT LF VT FF CR SO SI
DLE DC1 DC2 DC3 DC4 NAK SYN ETB CAN EM SUB ESC FS GS RS US
SP ! " # $ % & ' ( ) * + , - . /
0 1 2 3 4 5 6 7 8 9 : ; < = > ?
@ A B C D E F G H I J K L M N O
P Q R S T U V W X Y Z [\ ] ^ _
` a b c d e f g h i j k l m n o
p q r s t u v w x y z { | } ~ DEL
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
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

ASCII

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

DLE 0010 10	**DC1** 0011 11	**DC2** 0012 12	**DC3** 0013 13	**DC4** 0014 14	**NAK** 0015 15	**SYN** 0016 16	**ETB** 0017 17	**CAN** 0018 18	**EM** 0019 19	**SUB** 001A 1A	**ESC** 001B 1B	**FS** 001C 1C	**GS** 001D 1D	**RS** 001E 1E	**US** 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 0030 30	**1** 0031 31	**2** 0032 32	**3** 0033 33	**4** 0034 34	**5** 0035 35	**6** 0036 36	**7** 0037 37	**8** 0038 38	**9** 0039 39	**:** 003A 3A	**;** 003B 3B	**<** 003C 3C	**=** 003D 3D	**>** 003E 3E	**?** 003F 3F
@ 0040 40	**A** 0041 41	**B** 0042 42	**C** 0043 43	**D** 0044 44	**E** 0045 45	**F** 0046 46	**G** 0047 47	**H** 0048 48	**I** 0049 49	**J** 004A 4A	**K** 004B 4B	**L** 004C 4C	**M** 004D 4D	**N** 004E 4E	**O** 004F 4F
P 0050 50	**Q** 0051 51	**R** 0052 52	**S** 0053 53	**T** 0054 54	**U** 0055 55	**V** 0056 56	**W** 0057 57	**X** 0058 58	**Y** 0059 59	**Z** 005A 5A	**[** 005B 5B	**** 005C 5C	**]** 005D 5D	**^** 005E 5E	**_** 005F 5F
` 0060 60	**a** 0061 61	**b** 0062 62	**c** 0063 63	**d** 0064 64	**e** 0065 65	**f** 0066 66	**g** 0067 67	**h** 0068 68	**i** 0069 69	**j** 006A 6A	**k** 006B 6B	**l** 006C 6C	**m** 006D 6D	**n** 006E 6E	**o** 006F 6F
p 0070 70	**q** 0071 71	**r** 0072 72	**s** 0073 73	**t** 0074 74	**u** 0075 75	**v** 0076 76	**w** 0077 77	**x** 0078 78	**y** 0079 79	**z** 007A 7A	**{** 007B 7B	**|** 007C 7C	**}** 007D 7D	**~** 007E 7E	**DEL** 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
F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	FA	FB	FC	FD	FE	FF

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

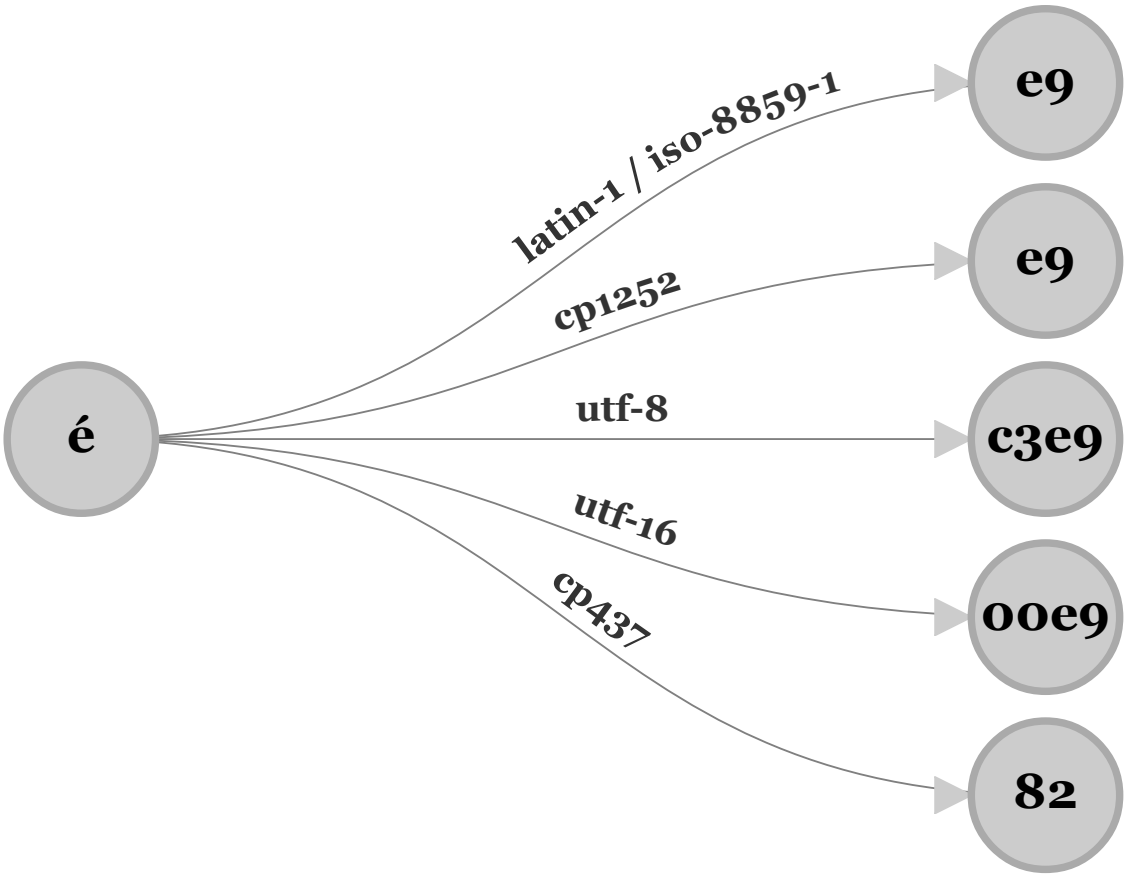
character encoding

- 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**



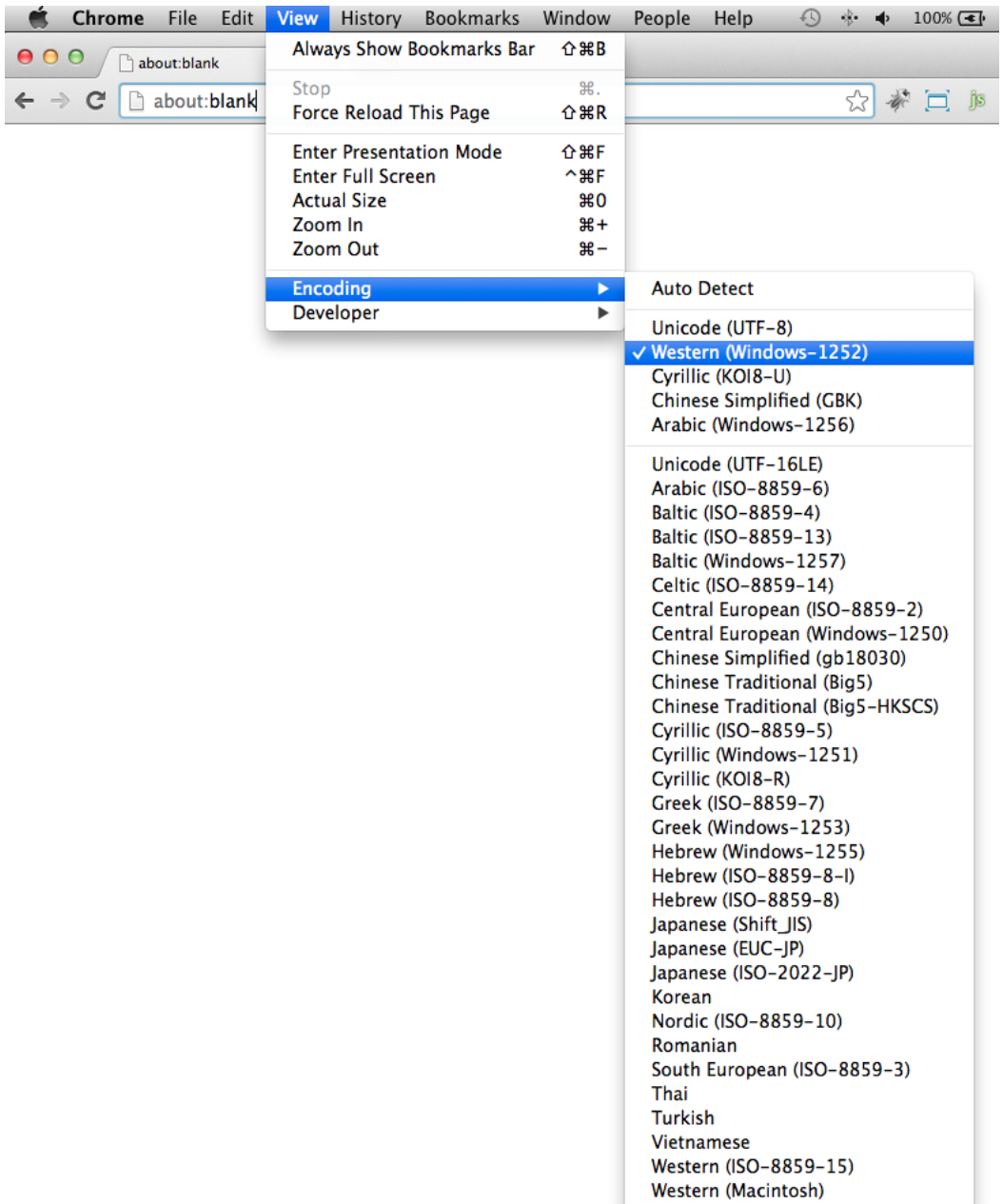
encoding of input and output

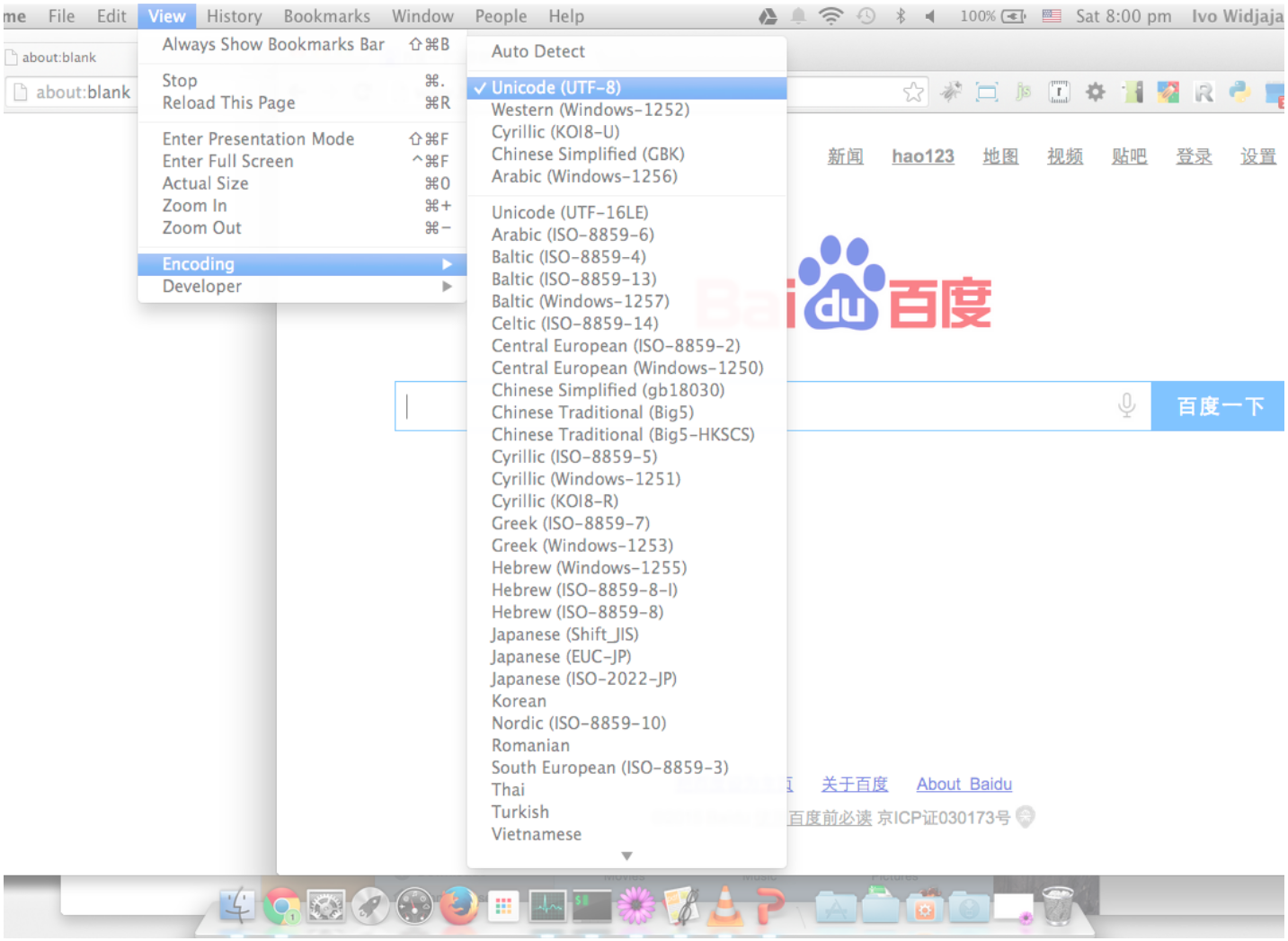
- Encoding is needed for interpreting a piece of data: text file, XML file, web page (which are blobs of bytes)
- Programmers need to know the encoding of data that goes into or comes from a device (keyboard, screen, or web browser)

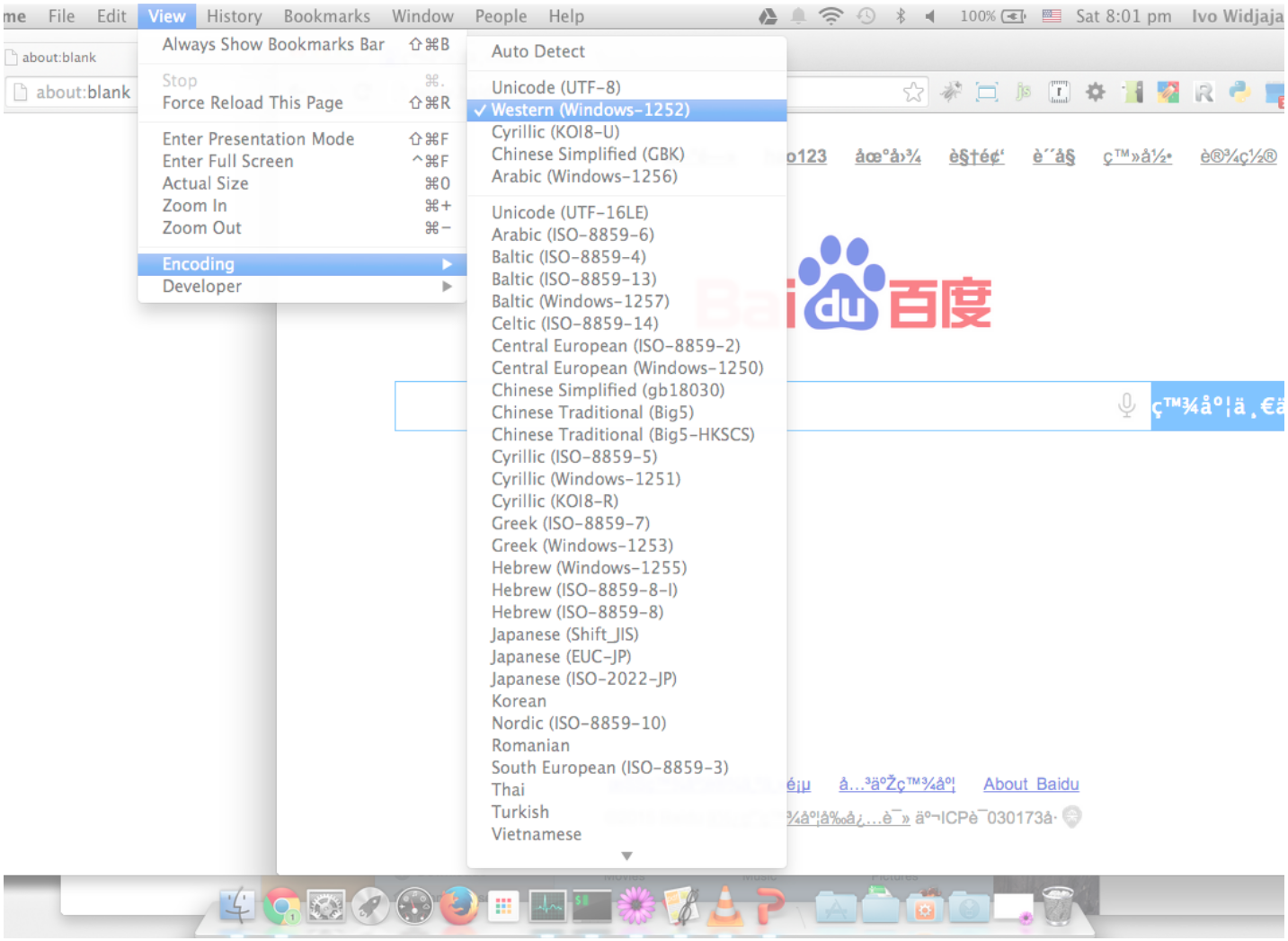
Example: it's best to specify the encoding used in your HTML page.

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

chrome browser encoding

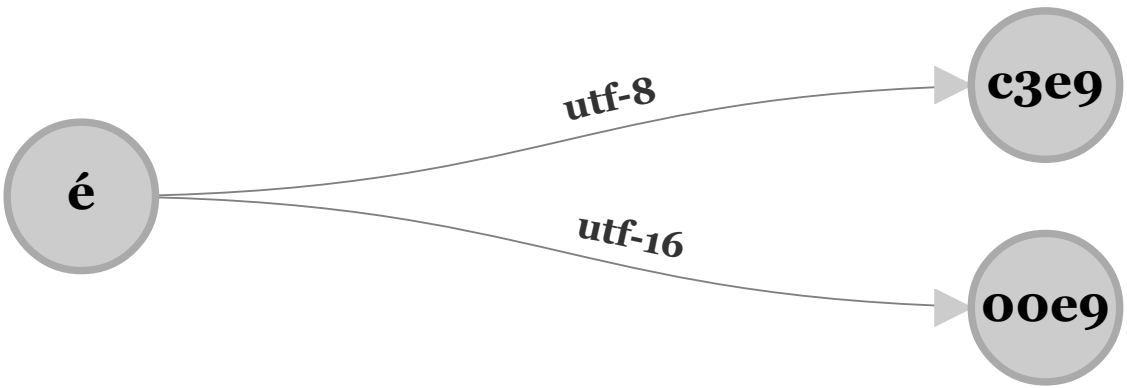



File Edit View History Bookmarks Window People Help
Sat 8:00 pm Ivo Widjaja
about:blank
Always Show Bookmarks Bar ⇧⌘B
Stop ⌘.
Reload This Page ⌘R
Enter Presentation Mode ⇧⌘F
Enter Full Screen ^⌘F
Actual Size ⌘0
Zoom In ⌘+
Zoom Out ⌘−
Encoding
Developer
Auto Detect
Unicode (UTF-8)
Western (Windows-1252)
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


File Edit View History Bookmarks Window People Help
Sat 8:01 pm Ivo Widjaja
about:blank
Always Show Bookmarks Bar ⇧⌘B
Stop ⌘.
Force 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
About Baidu

Unicode encoding

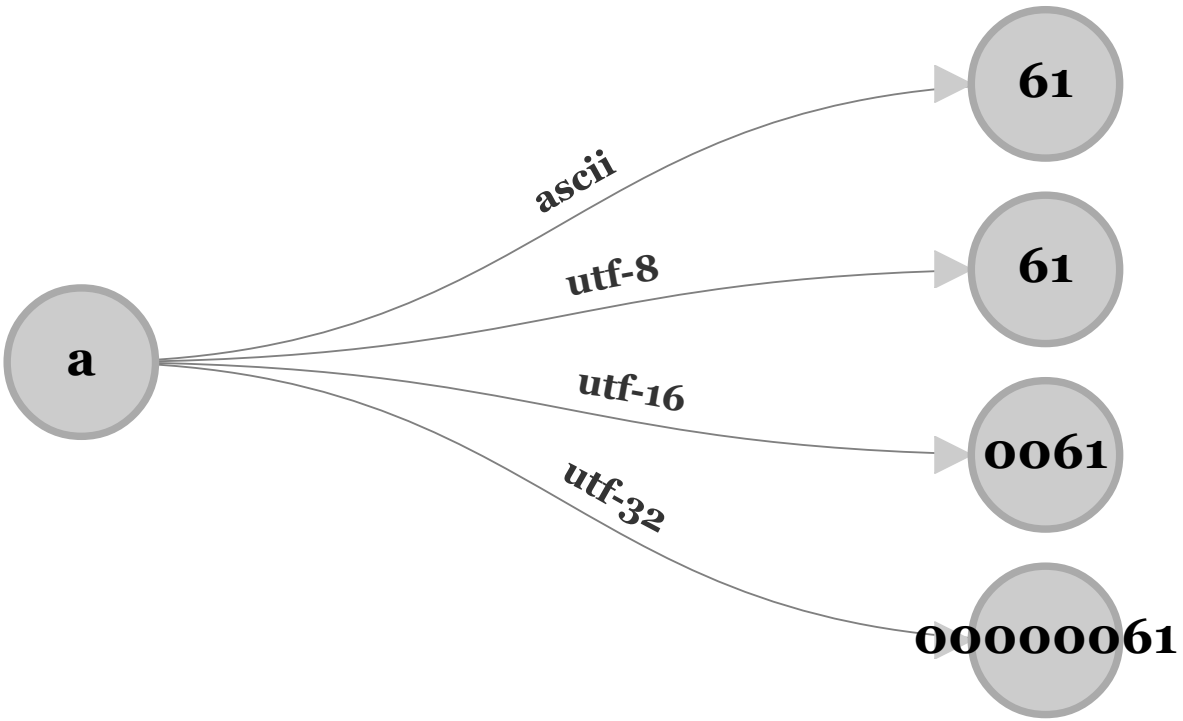
- ASCII characters are encoded as 1 byte (a single way of encoding)
- Unicode uses variable length encoding: 1-4 bytes (multiple ways of encoding: UTF-8, UT-16, UTF-32)



- **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




ROTECTED PLACE
保护区
NO
ADMITTANCE
O UNAUTHORISED
PERSONS
闲人
免进
LARANG MASOK
JIKA TIADA
KEBENARAN
EMPAT LARANGAN
பாதுகாப்பு உள்ள இட

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:

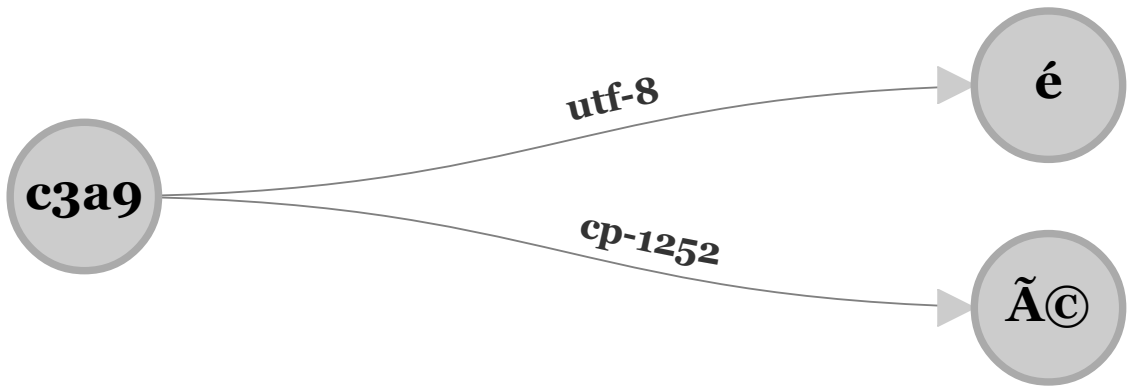
```
>>> e_acute = u'\u00e9'
>>> e_acute.encode('utf-8')
'\xc3\xa9'
>>> e.encode('latin-1')
'\xe9'
>>> e.encode('cp1252')
'\xe9'
```

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">ғыыуда</sentence>
  <sentence lang="iceland">Björk</sentence>
</demo>
```

consuming utf-8 xml

`lxml`'s `etree` provides Unicode string where necessary

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

producing data in utf-8

```
from lxml import etre
etree = etree.parse("demo.1.xml")
root = tree.getroot()
print "Content-Type: text/html\n"

print '<html>'
print '  <head>'
# specify the web page as UTF-8
# so that the browser can interpret correctly
print '    <meta http-equiv="content-type" '
print '          content="text/html; charset=UTF-8">'
print '    <title>Unicode demo</title>'
print '  </head>'
print '  <body>'
...
```

producing data in utf-8 (cont'd)

```
...
print '<ul>'
for sentence in root:
  # attempt to print Unicode string directly
  print '<li>', sentence.text, '</li>'
print '</ul>'
...
```

producing data in utf-8 (output)

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

in line:

```
print '<li>', sentence.text, '</li>'
```

diagnosis

1. `sentence.text` is a Unicode string
2. 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.
3. Since no encoding is specified, it uses the default `ascii`.

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

4. But not all Unicode characters can be encoded as ASCII (0-127)

Solution: Explicitly encode using UTF-8 encoding.

```
print '<li>', sentence.text.encode('utf-8'), '</li>'
```

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 Select languages

Please enter your name Go!

Italian Select languages

Il vostro nome per favore Go!

Sinhalese Select languages

ආයුබෝවන් Go!

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