Internationalization & Localization

بین‌المللی‌سازی و بومی‌سازی

$$\underbrace{\texttt{internationalizatio}}_{18}\texttt{n} \longrightarrow \texttt{i18n}$$

$$\texttt{l}\underbrace{\texttt{ocalizatio}}_{10}\texttt{n} \longrightarrow \texttt{l10n}$$

Local	محلی، بومی
Locale	محل، بوم
Localize	بومی کردن
Localization	بومی‌سازی

Definition

- Localization: Adapting software for location-dependent requirements
- Internationalization: Making the software localizable for any region / country / locale, without changing the code

Scopes

- Text / Language
- Time
- Other
  - Econimical
  - Cultural
  - Political

Scopes ⟶ **Text / Language**

- Language
  - Translation
  - Number Localization (Persian / Arabic / Urdu, Indian, Thai)
  - Other (Capitalization, Plural forms, Text sorting)
- Encoding
- Text Rendering (BiDi, CTL, ...)
- Keyboard Layouts and Shortcuts
- Optical Character Recognition (OCR)
- Text to Speech (T T S)
- Voice Recognition

POSIX (Unix-style) Locale Names

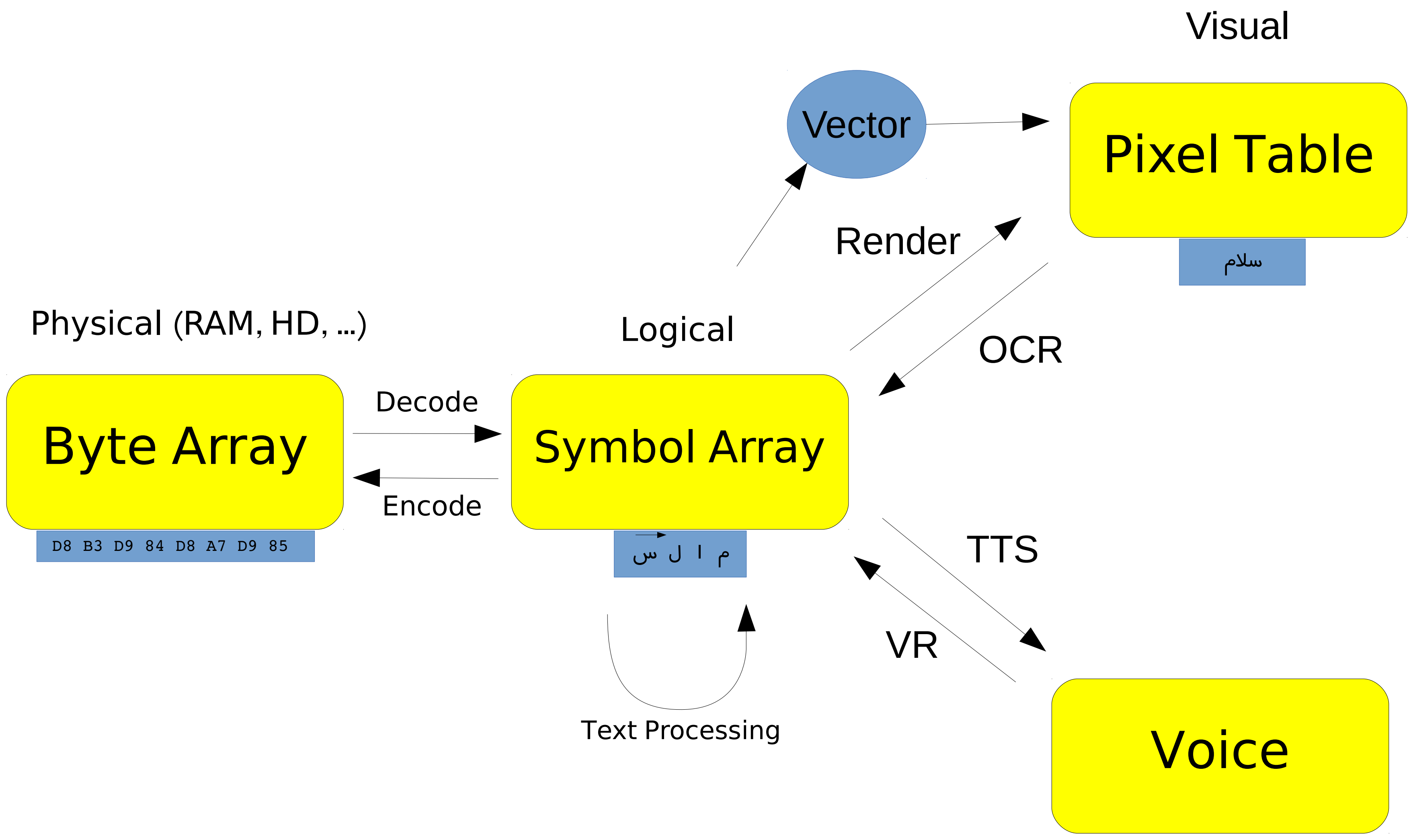
fa_IR.utf8

en_US.utf8

en_GB.utf8

```
$ less /etc/locale.alias
$ less /etc/default/locale
$ man locale
```

Scopes ⟶ Text / Language



Scopes ⟶ Text / Language ⟶ Encoding

Other Names

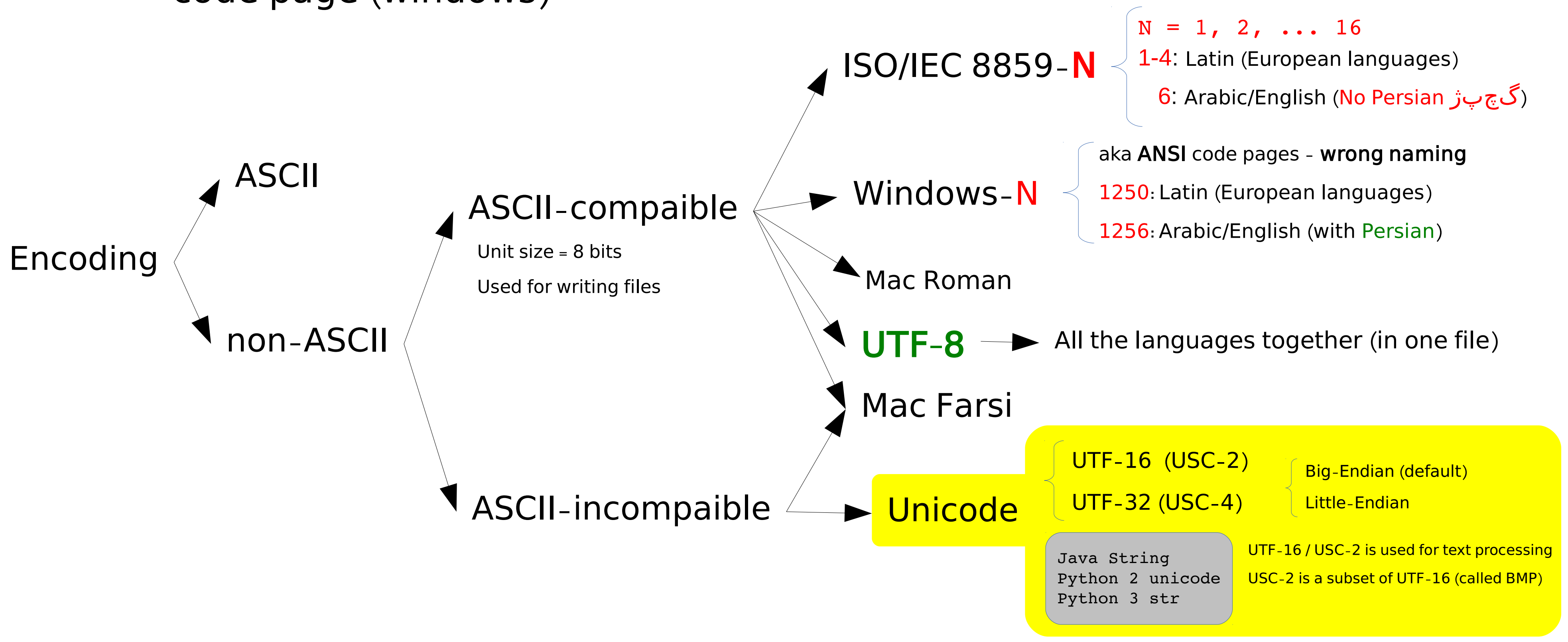
- character encoding
- character set = **charset**
- character map
- codeset
- code page (windows)

Notes:

UTC = Unicode Transformation Format

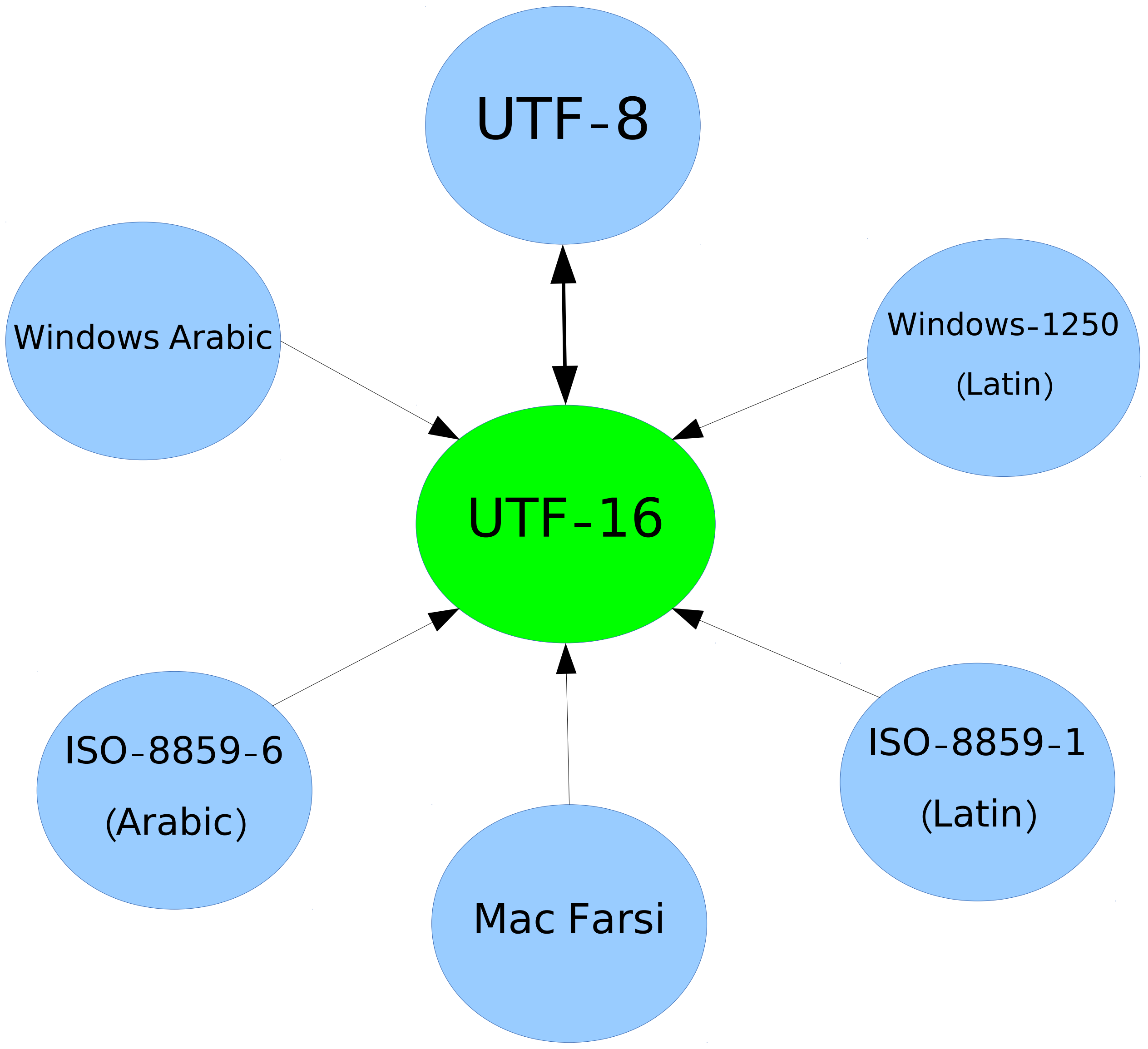
USC = Universal Character Set

BMP = Basic Multilingual Plane



Encoding
ASCII
non-ASCII
ASCII-compaible
Unit size = 8 bits
Used for writing files
ASCII-incompaible
ISO/IEC 8859-N
N = 1, 2, ... 16
1-4: Latin (European languages)
6: Arabic/English (No Persian گچپژ)
Windows-N
aka **ANSI** code pages - **wrong naming**
1250: Latin (European languages)
1256: Arabic/English (with Persian)
Mac Roman
UTF-8
All the languages together (in one file)
Mac Farsi
Unicode
UTF-16 (USC-2)
UTF-32 (USC-4)
Big-Endian (default)
Little-Endian
Java String
Python 2 unicode
Python 3 str
UTF-16 / USC-2 is used for text processing
USC-2 is a subset of UTF-16 (called BMP)

Scopes ⟶ Text/Language ⟶ Encoding



Problems arising from the use of code pages in Windows

Microsoft strongly recommends using Unicode in modern applications, but many applications or data files still depend on the legacy code pages. This can cause many problems, especially since the **Windows default is still not Unicode**:

- Programs need to know what code page to use in order to display the contents of files correctly. If a program uses the wrong code page it may show text as mojibake. Like: ï»¿Ø§Ù„Ø¥Ø¹Ù„Ø§Ù† Ø§Ù„Ø¹Ø§Ù„Ù…Ù‰ Ù„ØÙ‚Ù^Ù‚ Ø§Ù„Ø¥Ù†Ø³Ø§Ù†
- The code page in use may differ between machines, so files created on one machine may be unreadable on another.
- Data is often improperly tagged with the code page, or not tagged at all, making determination of the correct code page to read the data difficult.
- These Microsoft code pages differ to various degrees from some of the standards and other vendors' implementations. This isn't a Microsoft issue per se, as it happens to all vendors, but the lack of consistency makes interoperability with other systems unreliable in some cases.
- The use of code pages limits the set of characters that may be used.
- Characters expressed in an unsupported code page may be **converted to question marks (?)** or other replacement characters, or to a simpler version (such as removing accents from a letter). In either case, **the original character may be lost**.

From en.wikipedia.org/wiki/Windows_code_page

Note: Microsoft is using a seperate code page called **OEM** for DOS and Windows console

Windows-1256 Character Table

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	-	0	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80	€	پ	‚	ƒ	„	…	†	‡	ˆ	‰	ٹ	‹	Œ	چ	ژ	ڈ
90	گ	‘	’	“	”	•	–	—	ک	™	ڑ	›	œ			ں
A0		،	¢	£	¤	¥	¦	§	¨	©	ھ	«	¬	-	®	¯
B0	°	±	^{2}	^{3}	´	µ	¶	·	¸	^{1}	؛	»	¼	½	¾	؟
C0	ہ	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
D0	ذ	ر	ز	س	ش	ص	ض	×	ط	ظ	ع	غ	ـ	ف	ق	ك
E0	à	ل	â	م	ن	ه	و	ç	è	é	ê	ë	ى	ي	î	ï
F0	ً	ٌ	ٍ	َ	ô	ُ	ِ	÷	ّ	ù	ْ	û	ü			ے

About Mac Farsi

- Similar to **ISO 8859-6** in Arabic codes, but also includes Persian گچپژ
- Contains all the ASCII characters
- Not ASCII-compatible, why?

```
>>> import string
>>> for c in string.printable:
...   if unicode(c).encode('mac farsi') != c:
...     print c,
...
! " # $ & ' ( ) * + - . / : < = > [ \ ] ^ _ { | }
>>> ord(u'.'.encode('mac farsi')) - ord('.')
128
```

Mac Farsi Character Table

Font: XB Zar (IRMUG)

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
20		!	"	#	$	%	&	'	(	)	*	+	,	−	0	/
30	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
40	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
50	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
60	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
70	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
80	Ä		Ç	É	Ñ	Ö	Ü	á	à	â	ä	ں	«	ç	é	è
90	ê	ë	í	…	î	ï	ñ	ó	»	ô	ö	÷	ú	ù	û	ü
A0		!	"	#	$	٪	&	'	(	)	*	+	،	−	0	/
B0	۰	۱	۲	۳	۴	۵	۶	۷	۸	۹	:	؛	<	=	>	؟
C0	❊	ء	آ	أ	ؤ	إ	ئ	ا	ب	ة	ت	ث	ج	ح	خ	د
D0	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	[	\	]	^	_
E0	ـ	ف	ق	ك	ل	م	ن	ه	و	ى	ي	ً	ٌ	ٍ	َ	ُ
F0	ِ	ّ	ْ	پ	ٹ	چ	ە	ڤ	گ	ڈ	ڑ	{	|	}	ژ	ے

Scopes

- Text / Language
- Time
- Other

Scopes ⟶ **Time**

Calendaring Systems

- Gregorian
- Persian (Jalali)
- Arabic (Hijri)
- Indian
- Hebrew (Jewish)
- Ethiopian

Time Zone

- Asia/Tehran
- UTC + 3:30

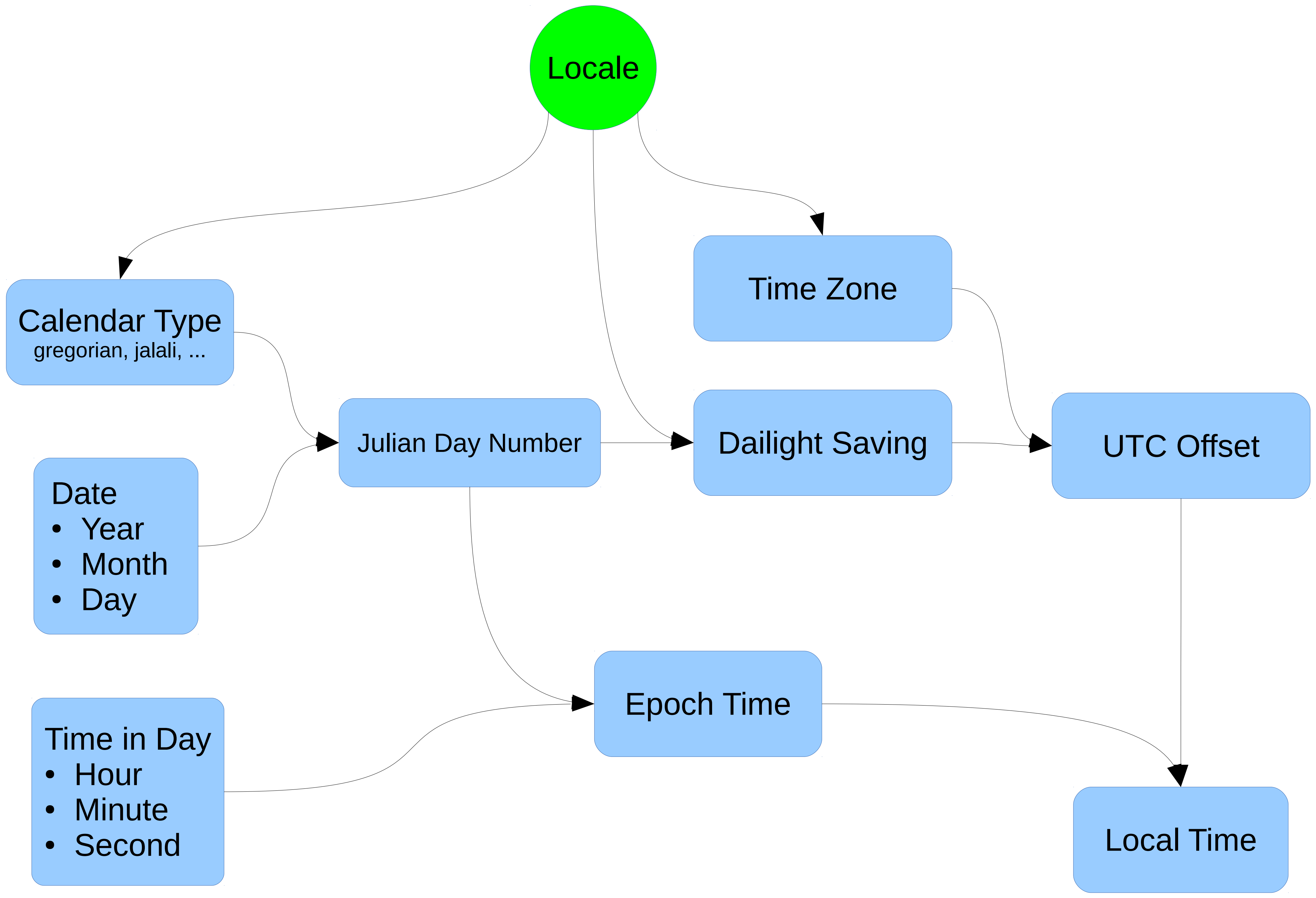
Daylight Saving

Date Format

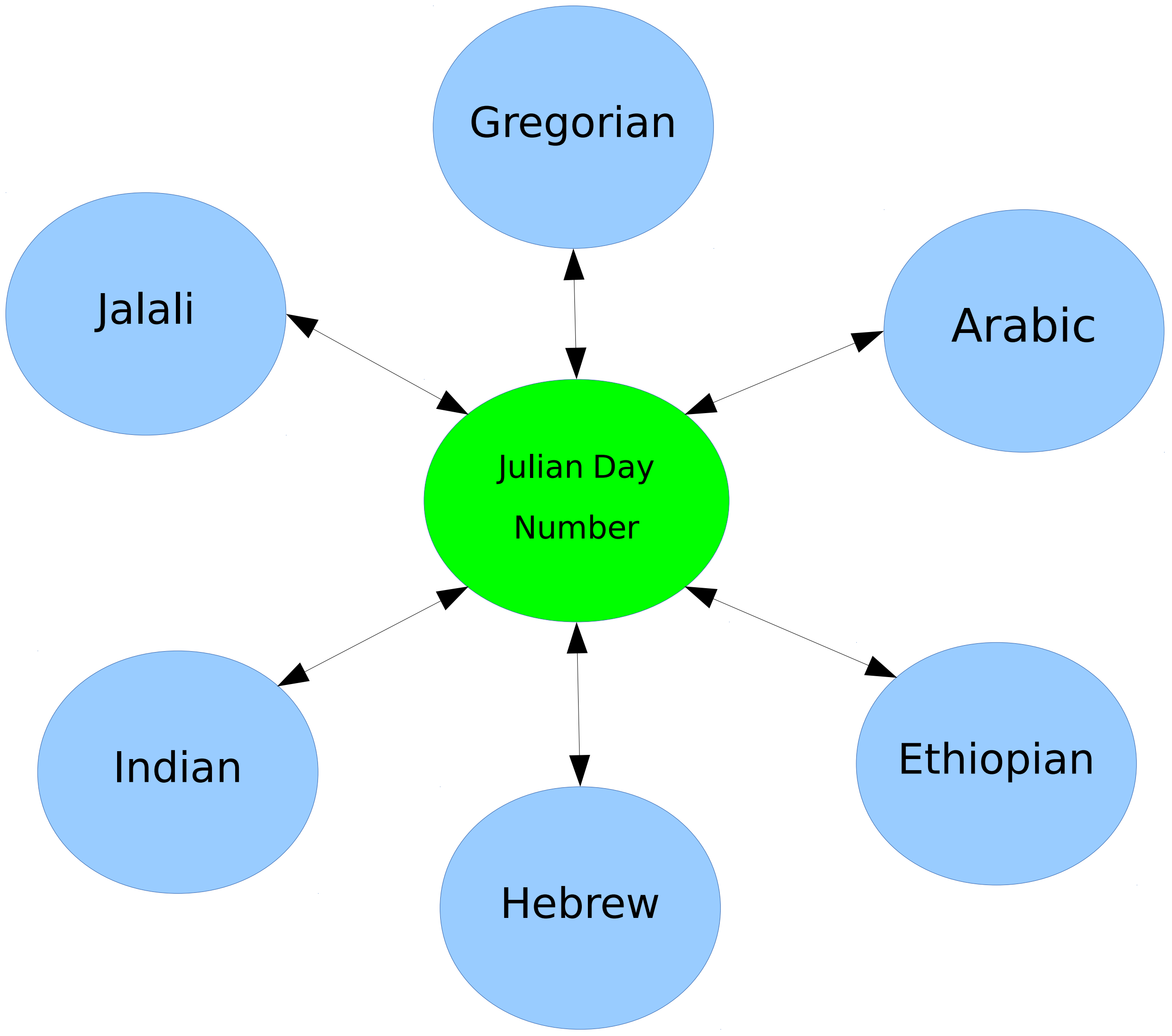
Week

- First Day of Week
- Work Days / Holiday(s)
- Week Numbering

Scopes ⟶ Time


Locale
Calendar Type
gregorian, jalali, ...
Time Zone
Julian Day Number
Dailight Saving
UTC Offset
Date
• Year
• Month
• Day
Epoch Time
Time in Day
• Hour
• Minute
• Second
Local Time

Scopes ⟶ Time ⟶ Calendaring Systems



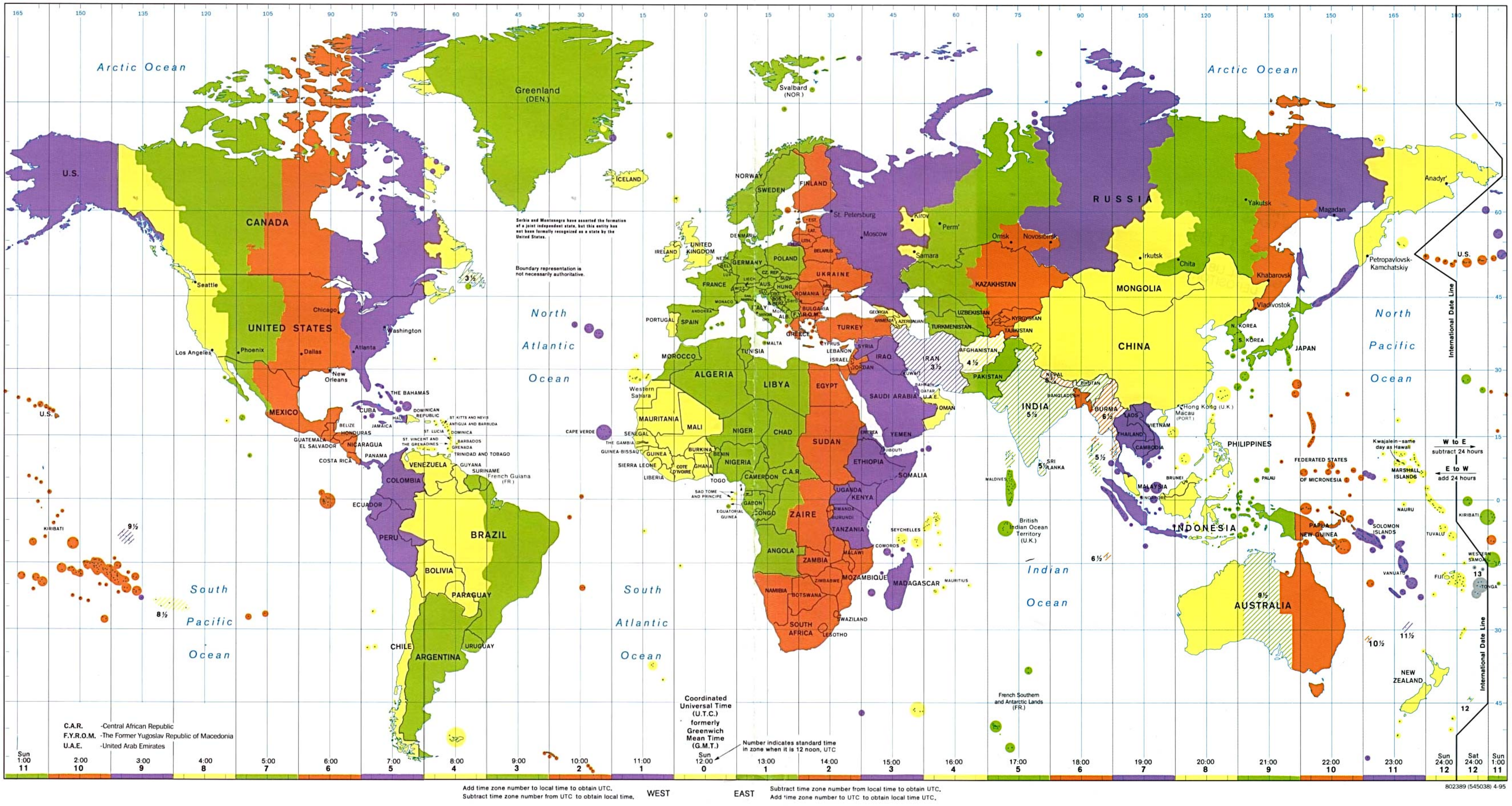
Scopes ⟶ Time ⟶ **Time Zone**

Asia/Tehran

UTC + 3:30

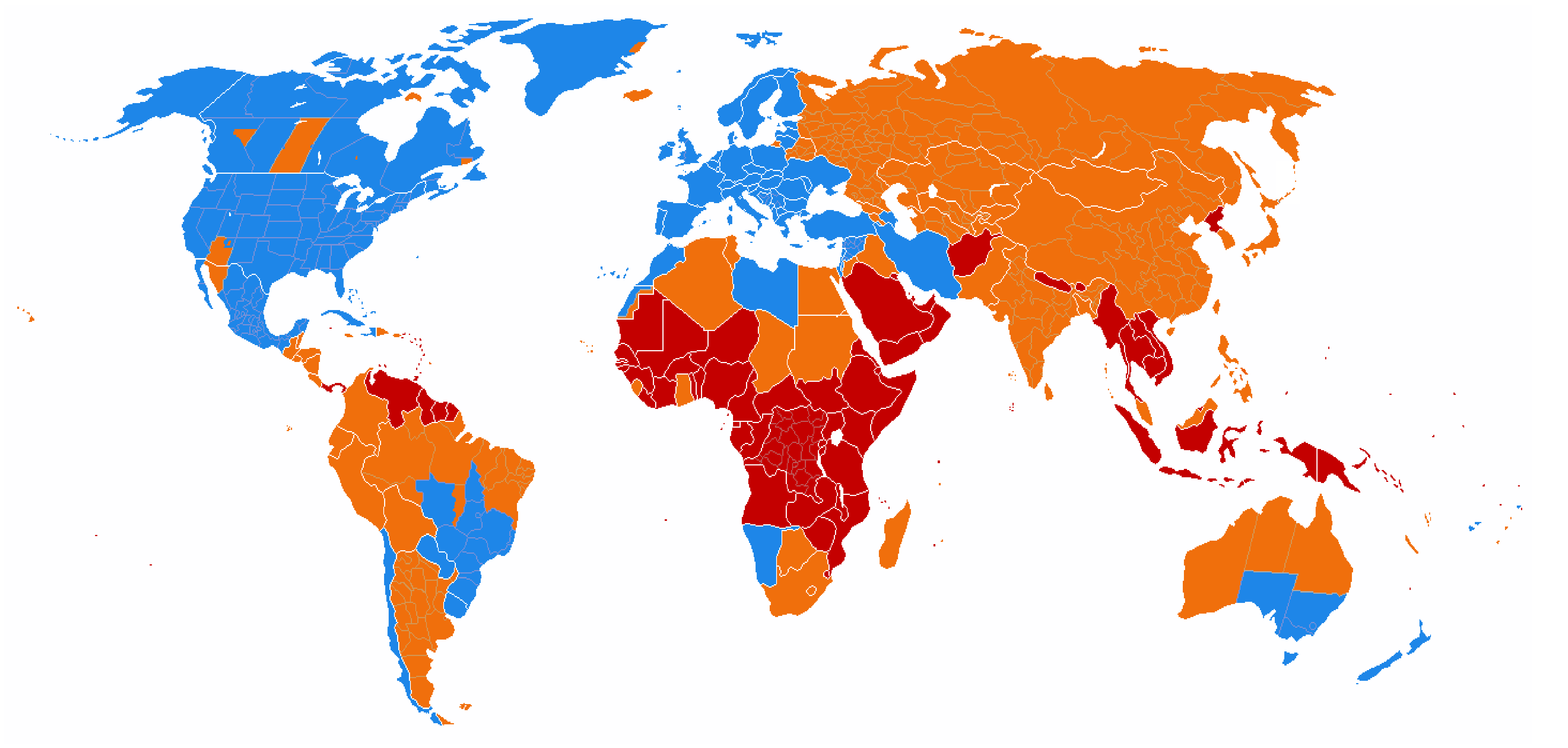
GMT + 3:30

Standard Time Zones of the World


Arctic Ocean
Greenland (DEN.)
Svalbard (NOR.)
U.S.
CANADA
UNITED STATES
MEXICO
ICELAND
NORWAY
SWEDEN
FINLAND
RUSSIA
KAZAKHSTAN
MONGOLIA
CHINA
JAPAN
IRAN 3½
AFGHANISTAN 4½
PAKISTAN
INDIA 5½
BURMA 6½
North Atlantic Ocean
North Pacific Ocean
ALGERIA
LIBYA
EGYPT
SUDAN
ZAIRE
SOUTH AFRICA
MADAGASCAR
BRAZIL
PERU
BOLIVIA
ARGENTINA
CHILE
South Pacific Ocean
South Atlantic Ocean
Indian Ocean
AUSTRALIA
INDONESIA
PHILIPPINES
NEW ZEALAND
International Date Line
Coordinated Universal Time (U.T.C.) formerly Greenwich Mean Time (G.M.T.)
Number indicates standard time in zone when it is 12 noon, UTC
W to E subtract 24 hours
E to W add 24 hours
C.A.R. -Central African Republic
F.Y.R.O.M. -The Former Yugoslav Republic of Macedonia
U.A.E. -United Arab Emirates
Serbia and Montenegro have asserted the formation of a joint independent state, but this entity has not been formally recognized as a state by the United States.
Boundary representation is not necessarily authoritative.
Add time zone number to local time to obtain UTC. Subtract time zone number from UTC to obtain local time.
WEST
EAST
Subtract time zone number from local time to obtain UTC. Add time zone number to UTC to obtain local time.
802389 (545038) 4-95

Scopes ⟶ Time ⟶

Daylight Saving



🟦 DST is used

🟧 DST is no longer used

🟥 DST has never been used

Scopes ⟶ Time ⟶

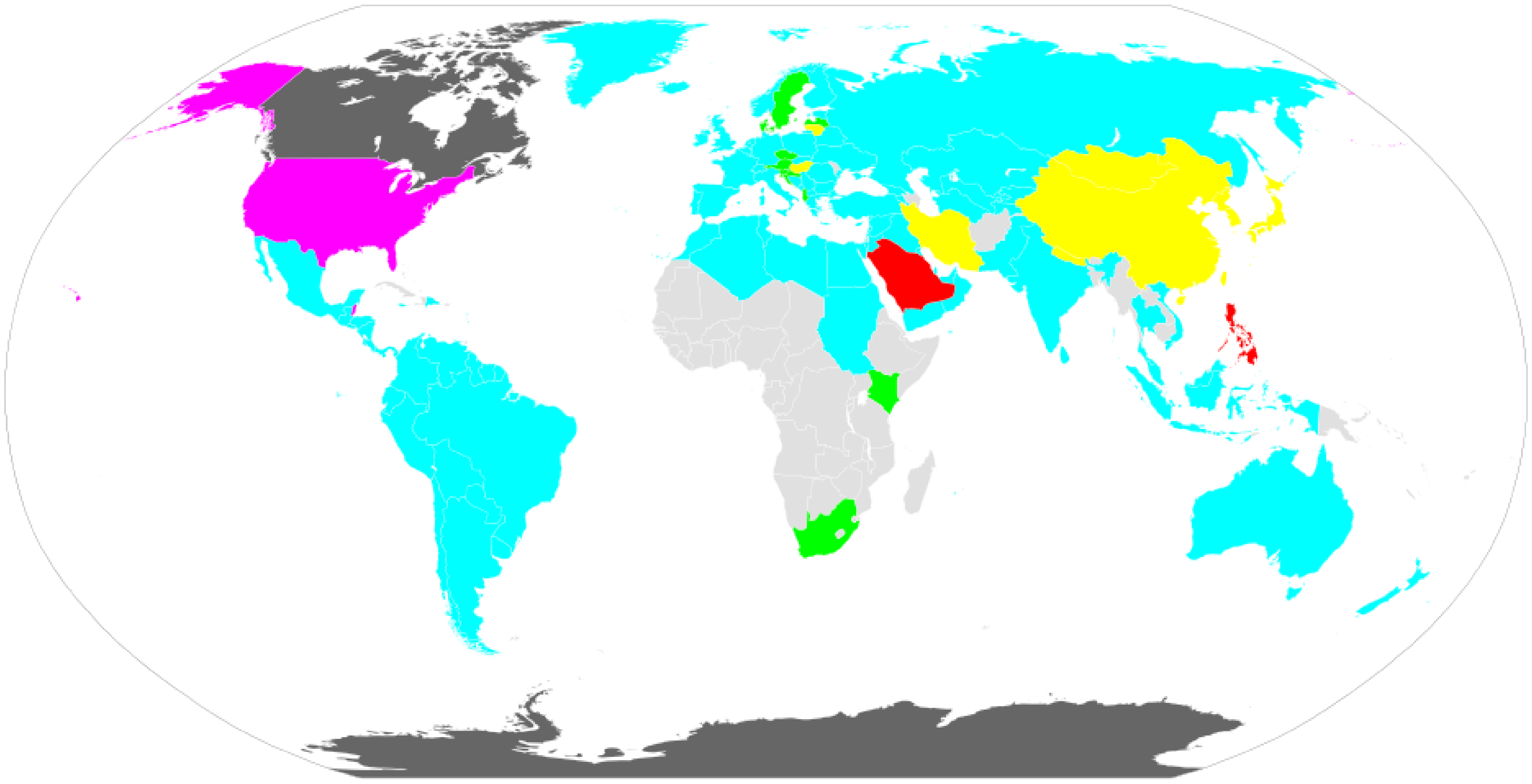
Date Format

YMD	2013/12/30	Iran, East Asia (CJK)
MDY	12/30/2013	USA, Belize
DMY	30/12/2013	Most of Asia & Europe , North Africa, South America, ...

🟩 YMD, DMY

🟥 DMY, MDY

⬛ YMD, DMY, MDY



Scopes ⟶ Time ⟶

Week

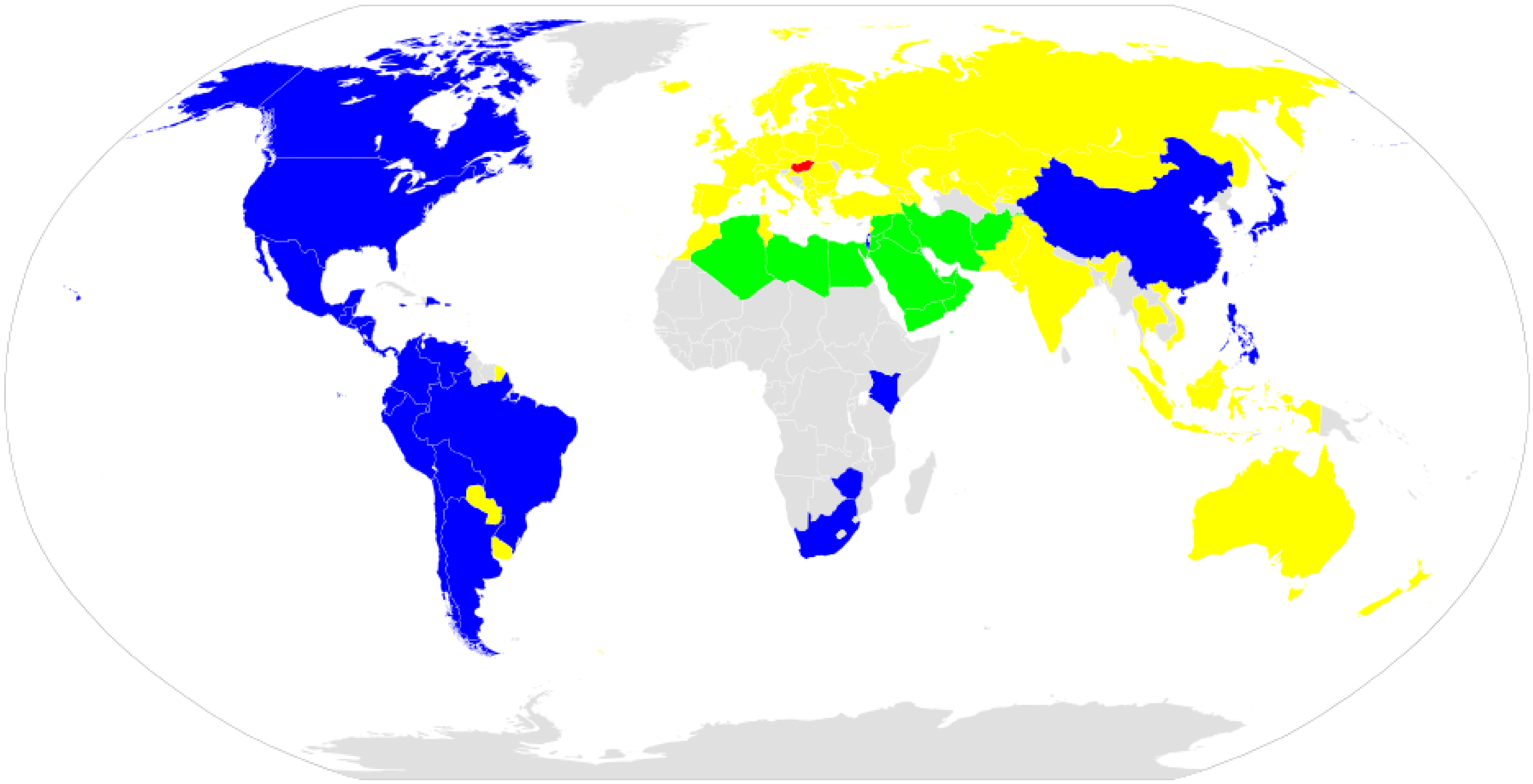
First day of week	First week of year contains			Used by/in
Saturday	1 January	1st Friday	1–7 days of year	Iran, Much of the Middle East
Sunday	1 January	1st Saturday	1–7 days of year	Canada, USA, Mexico
Monday	4 January	1st Thursday	4–7 days of year	Most of Europe, ISO 8601

At least six methods for week numberings are in use

http://www.pjh2.de/datetime/weeknumber/wnd.php?l=en

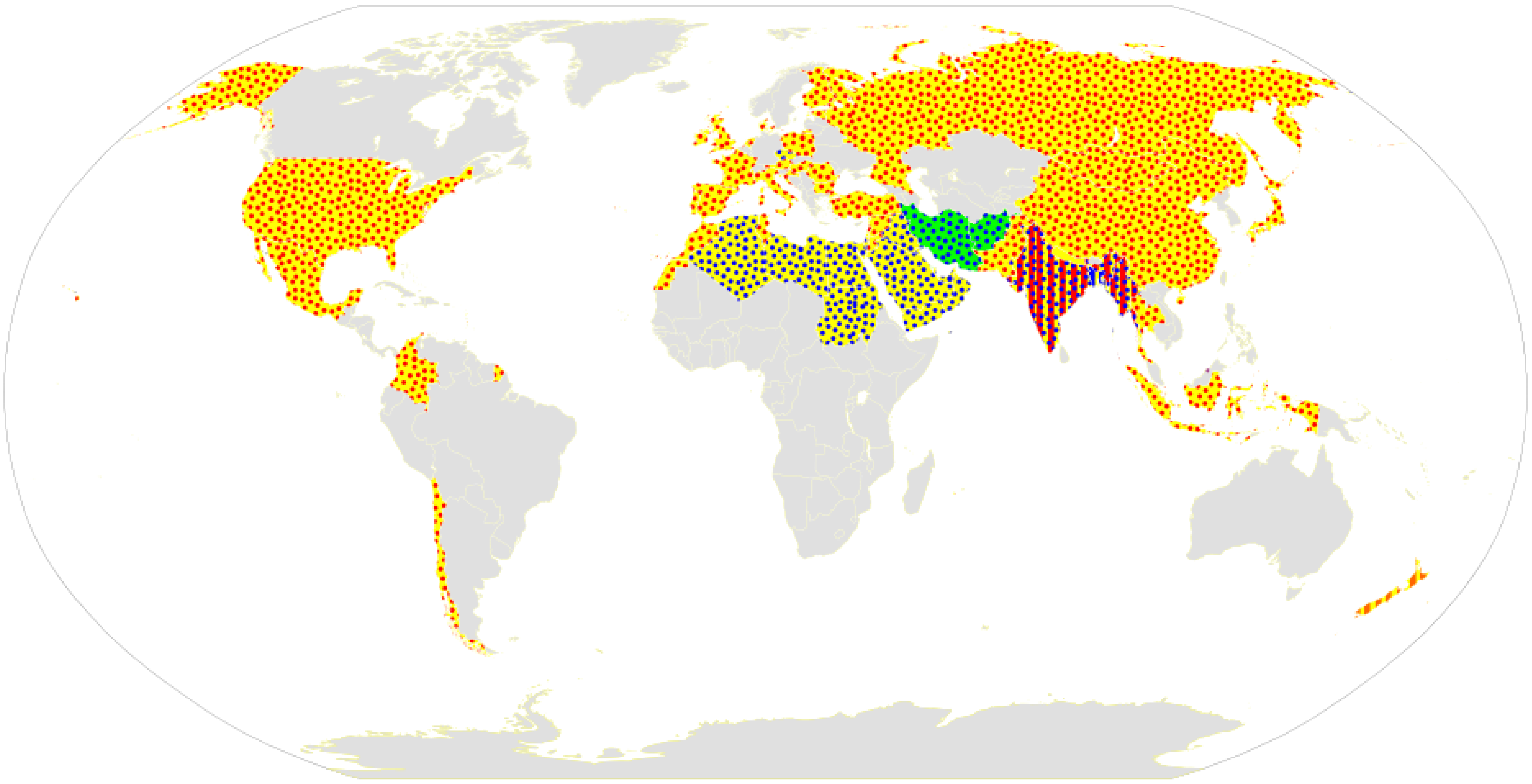
Scopes ⟶ Time ⟶ Week ⟶ **First Day of Week**

شنبه	🟩	Saturday
یکشنبه	🟦	Sunday
دوشنبه	🟨	Monday
چهارشنبه	🟥	Wednesday



Scopes ⟶ Time ⟶ Week ⟶ **Holidays**

پنجشنبه	🟩	Thursday
جمعه	🟦	Friday
شنبه	🟨	Saturday
یکشنبه	🟥	Sunday



Scopes

- Text / Language
- Time
- Other

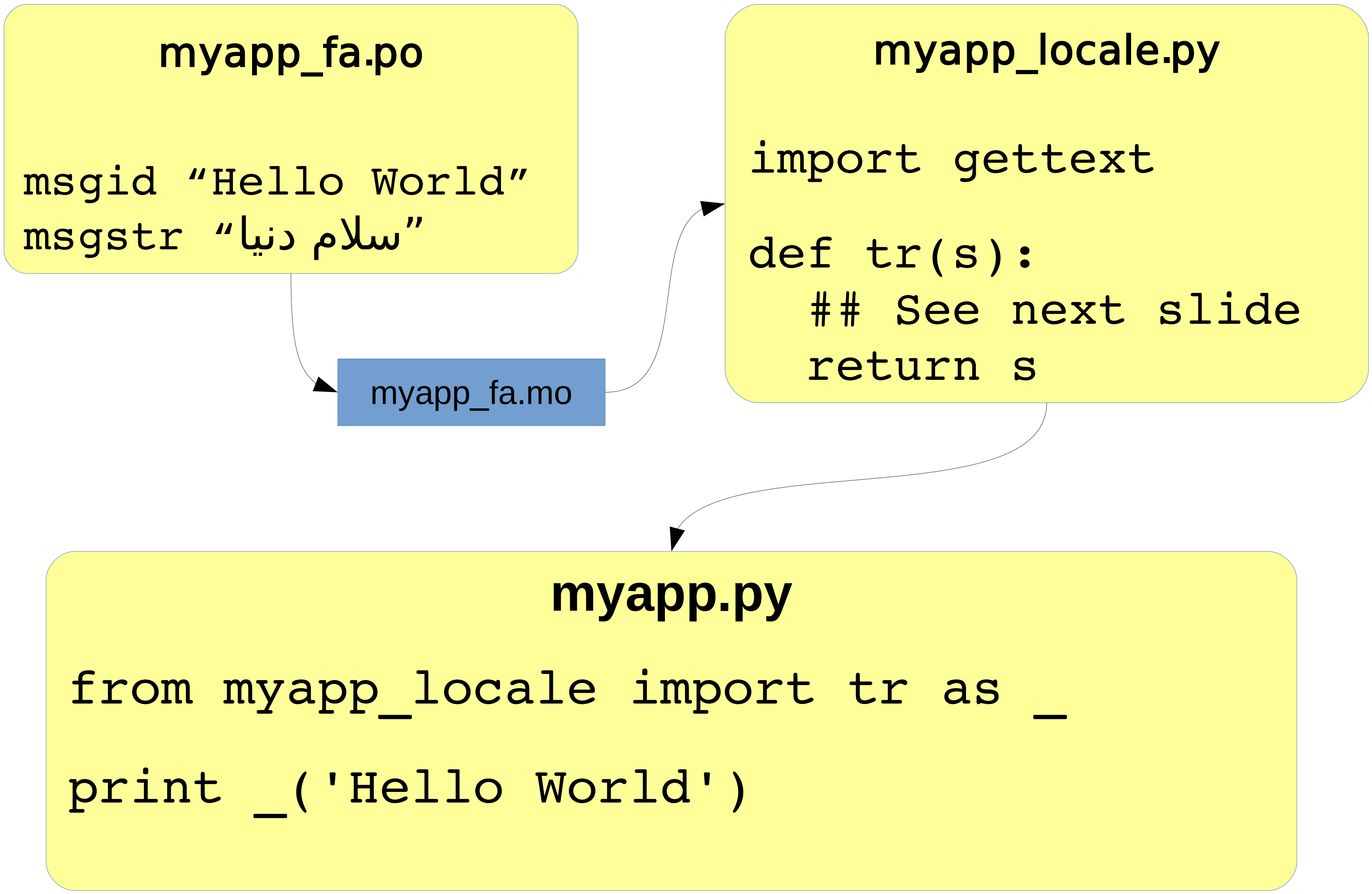
Other

- Currency, Tax and Economical differences
- Native Laws (Copyright, DRM, …)
- Phone Numbers, Area Codes, Zip Codes, …
- Native Themes and Styles

Now Let's Do the Code

Python Examples

Translation



myapp_fa.po

```
msgid “Hello World”
msgstr “سلام دنیا”
```

myapp_locale.py

```
import gettext

def tr(s):
   ## See next slide
   return s
```

myapp.py

```
from myapp_locale import tr as _

print _('Hello World')
```

Terminal

```
$ msgfmt "myapp_fa.po" -o “myapp_fa.mo”
```

myapp_locale.py

```
import gettext

lang = 'fa'

try:
    fd = open('myapp_%s.mo'%lang, 'rb')
except IOError:
    tr = str ## Fallback translator
else:
    transObj = gettext.GNUTranslations(fd)
    def tr(s):
        return transObj.gettext(toStr(s)).decode('utf-8')
```

Encoding

```
$ python2.7
>>> st = 'سلام'
>>> st
'\xd8\xb3\xd9\x84\xd8\xa7\xd9\x85'
>>> st[0]
'\xd8'
>>> print st[0]

>>> uni = st.decode('utf-8')
>>> uni
u'\u0633\u0644\u0627\u0645'
>>> print uni[0]
س
>>> len(st), len(uni)
(8, 4)
>>> for c in uni: print c
...
س
ل
ا
م
```

Python Examples ⟶ Encoding

```
>>> uni = u'سلام'
>>> uni
u'\u0633\u0644\u0627\u0645'
>>> uni.encode('utf-8')
'\xd8\xb3\xd9\x84\xd8\xa7\xd9\x85'
>>> uni.encode('windows-1256')## or 'cp1256'
'\xd3\xe1\xc7\xe3
>>> uni.encode('iso 8859-6')## or 'arabic'
'\xd3\xe4\xc7\xe5'
>>> uni.encode('mac farsi')## different from iso 8859-6
'\xd3\xe4\xc7\xe5'
>>> uni.encode('mac arabic')## the same as mac farsi
'\xd3\xe4\xc7\xe5'
>>> u'گچپژ'.encode('iso 8859-6', 'ignore')## or 'arabic'
''
>>> u'گچپژ'.encode('mac farsi', 'ignore')
'\xf8\xf5\xf3\xfe'
>>> u'گچپژ'.encode('windows-1256', 'ignore')
'\x90\x8d\x81\x8e'
```

```
>>> uni.encode('windows-1250')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib/python2.7/encodings/cp1250.py", line 12, in encode
    return codecs.charmap_encode(input,errors,encoding_table)
UnicodeEncodeError: 'charmap' codec can't encode characters in position 0-3:
character maps to <undefined>
>>> unicode('سلام')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeDecodeError: 'ascii' codec can't decode byte 0xd8 in position 0:
ordinal not in range(128)
>>> unicode('hello')
u'hello'
>>> str(u'سلام')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
UnicodeEncodeError: 'ascii' codec can't encode characters in position 0-3:
ordinal not in range(128)
>>> str(u'hello')
'hello'
```

ord, chr, unichr

```
>>> ord('a')
97
>>> ord(u'a')
97
>>> hex(ord('a'))
'0x61'
>>> ord(u'س')
1587
>>> chr(97)
'a'
>>> unichr(97)
u'a'
>>> unichr(1587)
u'\u0633'
>>> print unichr(1587)
س
>>> ord('a')-ord('A')
32
>>> chr(ord('b')-32)
'B'
```

```python
#!/usr/bin/python
# -*- coding: utf-8 -*-
# recode a file from arabic windows(windows-1256) to utf8

import sys, os
from os.path import splitext

def winArabicToUtf8(st):
    uni = st.decode('windows-1256')
    for ar, fa in [
        (u'ي', u'ی'),
        (u'ك', u'ک'),
        (u'هٔ', u'ۀ'),
    ]:
        uni = uni.replace(ar, fa)
    return uni.encode('utf8')

if __name__=='__main__':
    fname, ext = splitext(sys.argv[1])
    newName = fname + '.utf8' + ext
    st = open(sys.argv[1], 'rb').read()
    st = winArabicToUtf8(st)
    open(newName, 'w').write(st)
```

encoding2csv.py

Create a CSV file containing the character table of a given encoding

```
import sys
import csv

def getHex(n, fill=True):
    s = hex(n)[2:].upper()
    if fill and len(s) % 2 == 1:
        s = '0' + s
    return s

ext = '.csv'

encoding = sys.argv[1]
try:
    opath = sys.argv[2]
    if not opath.endswith(ext):
        opath += ext
except IndexError:
    opath = encoding + ext

writer = csv.writer(open(opath, 'wb'))
```

```python
writer.writerow(
    [''] + [getHex(i, False) for i in range(16)]
)

for i in range(2, 16):
    row = [
        getHex(16*i)
    ]
    for j in range(16):
        ordNum = 16*i + j
        ordHex = getHex(ordNum)
        try:
            cstr = chr(ordNum).decode(encoding).encode('utf8')
        except UnicodeDecodeError:
            print 'Unknown character %s'%ordHex
            cstr = ''
        row.append(cstr)
    writer.writerow(row)

del writer
```

Any Questions?

(C) 2013 by Saeed Rasooli <*saeed.gnu@gmail.com*>

http://saeedgnu.blog.ir