CS193E
Lecture 16

Internationalization and Localization

Announcements

- Final Project Due:
- Wed, March 19th at 11:59 PM

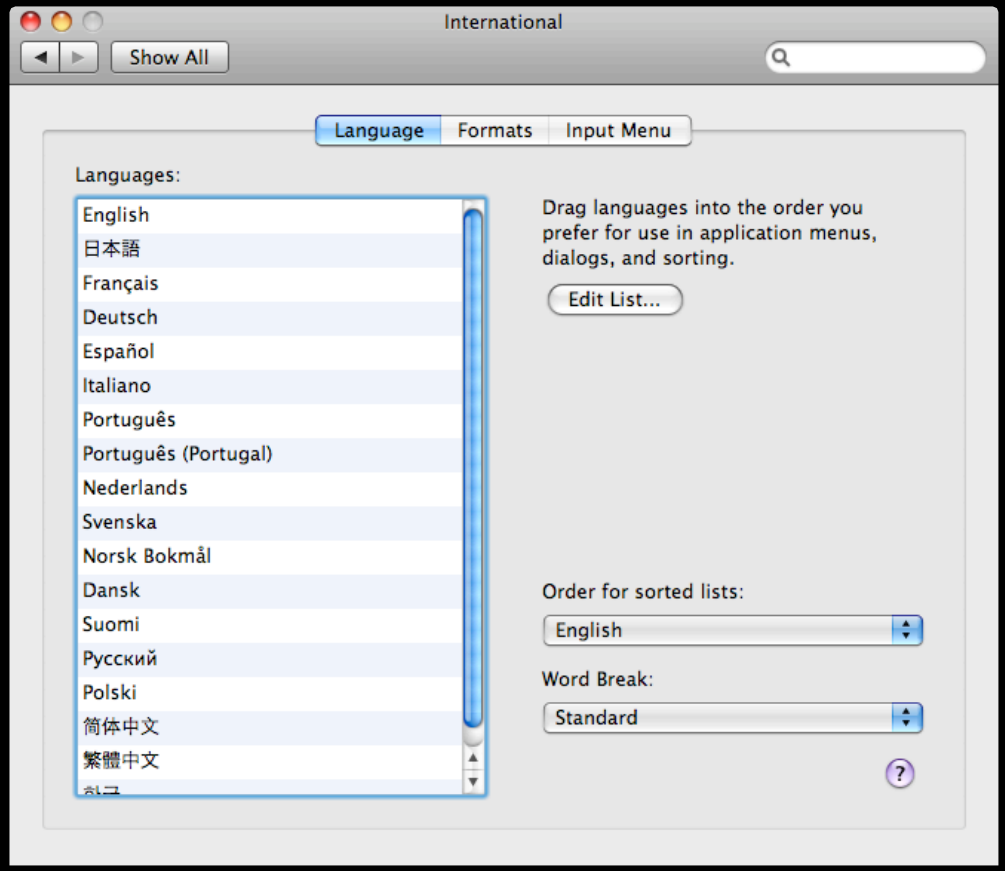
Announcements

- Final Project Demos
  - Thurs, March 20th, 3:30 - 6:30
  - Same room
- Plan for about a 5 minute demo
  - Show off what you've built

Internationalization and Localization

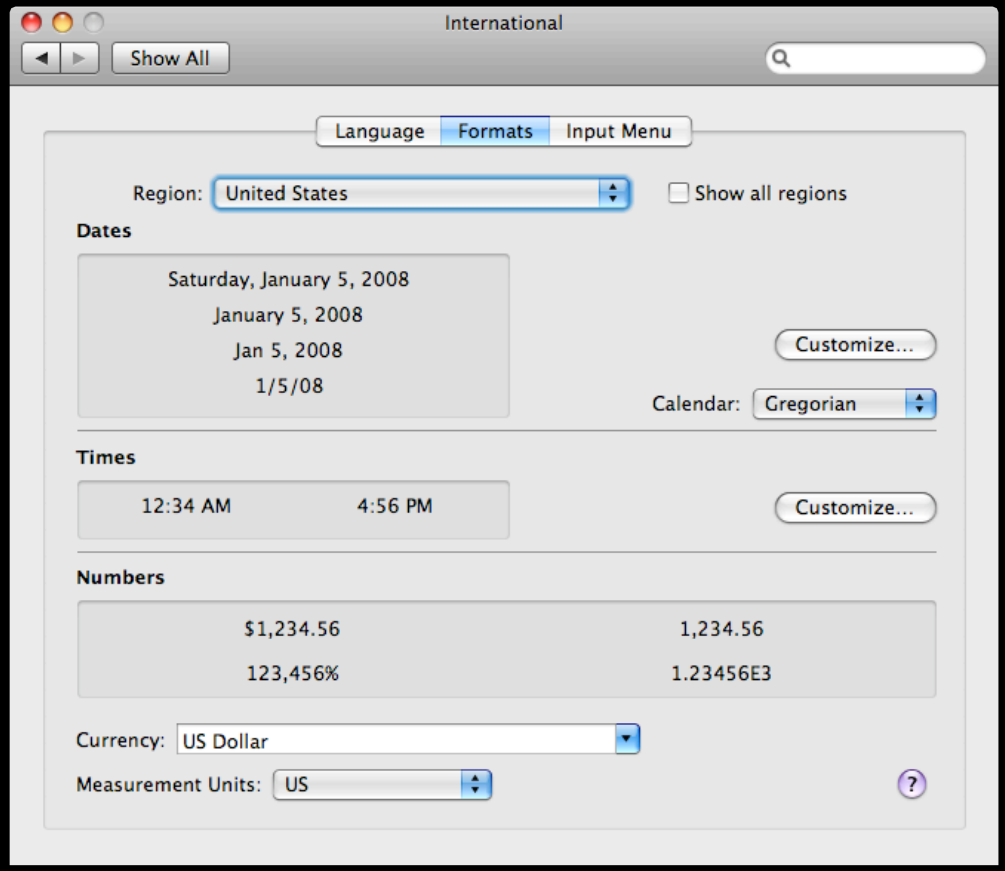
Mac OS X is International

18 localizations in one release for Leopard



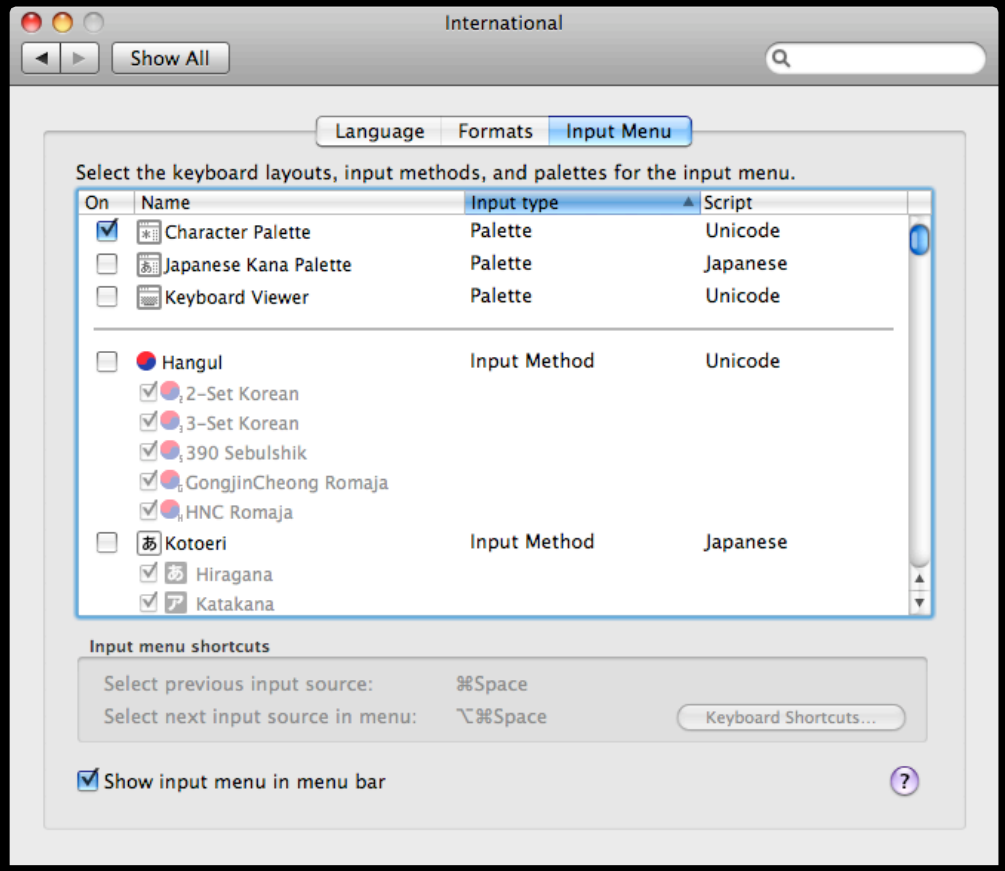
Mac OS X is International

Regional settings



Mac OS X is International

Input Methods



Internationalization and Localization

- Internationalization is the process of designing/modifying software to facilitate localization
- Localization is the process of adapting software for use in different locales

What is a locale?

- Locales encapsulate information about linguistic, cultural, and technological conventions and standards
- Represented historically by language names (e.g. “English”, “Japanese”, “Spanish”)
- Standard strings defined in ISO 639, 3166

en	English
en_US	US English
en_GB	UK English
fr_FR	French (France)
fr_CA	Canadian French

Locale Sensitive Stuff

- Interface layouts (nib files)
- User visible strings
- Images
- Other files such as documentation and help files
- Text handling
- Spoken-text sound files
- Dates, Numbers, Currencies

Localizing Resources

Resource Organization

- All resources live inside the bundle, or application wrapper (e.g. the PersonalTimeline.app folder)
- Locale specific resources live inside a locale specific subdirectory with the .lproj extension

```
PersonalTimeline.app/
        Contents/
            Info.plist
            MacOS/
            PkgInfo
            Resources/
                en.lproj/
                fr.lproj/
```

NSBundle

- Object representation of a bundle (e.g. app wrapper).
- +[NSBundle mainBundle] returns the NSBundle for the app
- Resources can be resolved from a bundle using -[NSBundle pathForResource:ofType:]
- Finds the most appropriate localized version of the file by searching .lproj directories.

NSBundle example

- Looking up a localized image:

```
NSString *path =
    [[NSBundle mainBundle]
              pathForResource:@"ThumbsUp"
                                ofType:@"tiff"];

NSImage *image = [[NSImage alloc]
                     initWithContentsOfFile:path];
```

Localizing Nib Files

- A different nib file for each localization, if necessary
- Don’t localize until your application’s interface is done
- Note that the same word or concept can take up more room in different languages
- nibtool pulls strings out of a nib file, and allows the localized strings to be pushed back into the nib

Localizing Strings

Strings

- Any user-readable string should not be hard-coded into the app, but looked up as a localized resource
- Strings need to be externalized in .strings files
  - Default name 'Localizeable.strings'
  - Can create strings files with other names
  - A strings file is referred to in the API as a 'table'
- Format .strings files as UTF-16
- Can be created in Xcode or TextEdit
- Can be automatically generated using genstrings tool

Format of a .strings file

- In an English .strings file

```
/* This is a comment */
"Yes" = "Yes";
"Hello" = "Hello";
```

- In the corresponding French .strings file

```
/* This is a comment */
"Yes" = "Oui";
"Hello" = "Bonjour";
```

Accessing localized strings

- Code looks up strings rather than hardcode them
- NSBundle has API to look up localized strings:
  - -localizedStringForKey:value:table:
- More convenient to use defined macros:
  - NSLocalizedString(@"Welcome!", @"Enthusiastic Greeting");
- NSLocalizedString looks up @"Welcome!" in Localizable.strings, returns @"Welcome!" if not found.
- If you use the macros, you just write your code, then run genstrings to automatically create the .strings file for you.

Order Independant String Formatting

- Consider:

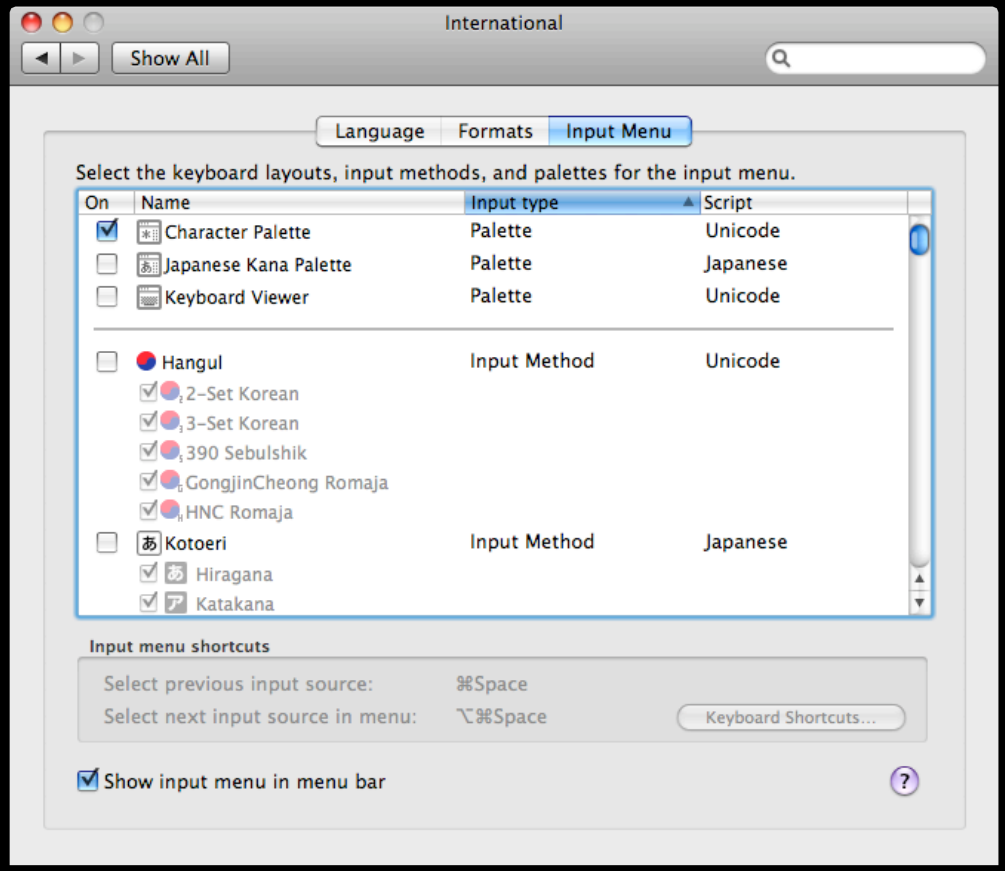
```
[NSString stringWithFormat:@"Sorry %@, the file
%@ could not be opened", firstName, file];
```

- In some languages, the arguments might be reversed.
- Refer to argument order explicitly

```
[NSString
stringWithFormat:NSLocalizedString(@"Sorry
%1$@, the file %2$@ could not be opened",
@"Error"), firstName, file];
```

Text Input

Input Methods



Text Input

- Composing English is easy, just type the characters you want
- Composing Japanese/Chinese is more difficult because there are so many characters
- Always use NSTextView, NSTextField to get advanced input management for free

Character Encodings

Character Encodings

- There are many different ways to store 16 bit unicode in a file, some ways are lossy
- These encodings are represented by NSStringEncoding:

NSASCIIStringEncoding
NSISOLatin1StringEncoding
NSJapaneseEUCStringEncoding
NSMacOSRomanStringEncoding
NSNonLossyASCIIStringEncoding
NSShiftJISStringEncoding
NSUTF8StringEncoding
NSUnicodeStringEncoding

Character Encodings

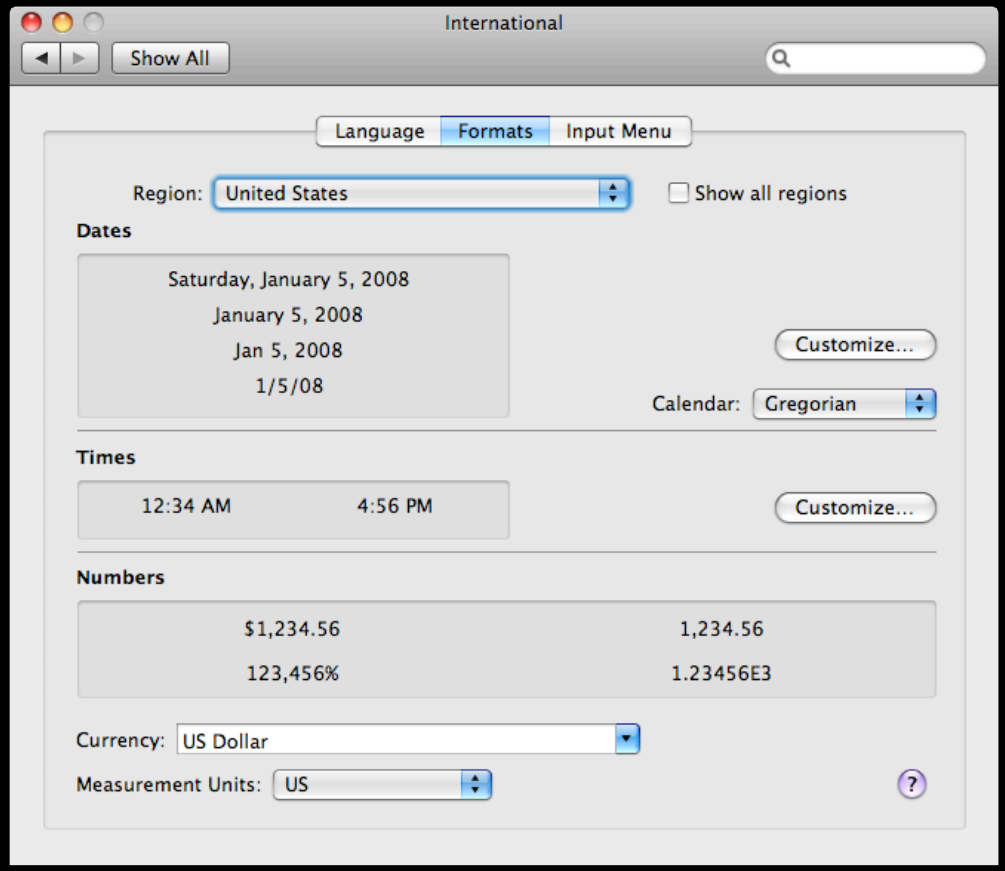
- The system has a default encoding that expects files to be stored as, returned be +[NSString defaultCStringEncoding]
- Can encode a string explicitly using -[NSString dataUsingEncoding:(NSStringEncoding)encoding]
- Can initialize from any encoding using -[NSString initWithData:(NSData *)data encoding:(NSStringEncoding)]

Text Handling

- Unicode provides a unique way to represent every character, no matter what the platform, no matter what the program, no matter what the language.
- NSString's native representation is Unicode
- Use NSString instead of char *, strcmp, strlen, etc.

Formatters

Regional Settings



NSFormatter

- Converts value objects such as NSDates, NSNumbers, etc to strings
  - `- [NSFormatter stringForObjectValue:(id)]`
- Also parses strings into objects

```
-(BOOL)[NSFormatter getObjectValue:(id *)
forString:(NSString *) errorDescription:
(NSString **)]
```

Tiger Date Formatter

- Can specify a style which will use formats set by user in International preferences

  `NSDateFormatterShortStyle`

  `NSDateFormatterMediumStyle`

  `NSDateFormatterLongStyle`

  `NSDateFormatterFullStyle`

- Can also specifiy new style format string as described in Unicode Technical Standard #35

Demo

Questions?