# Internationalization Ⓣ̈ı̇p̈š and Ť̈r̈i̇ç̈k̲̈ṥ

Session 244

**Dave DeLong**
iOS Frameworks Engineer

365 Million

155 Countries

# One App

Everyone, Everywhere

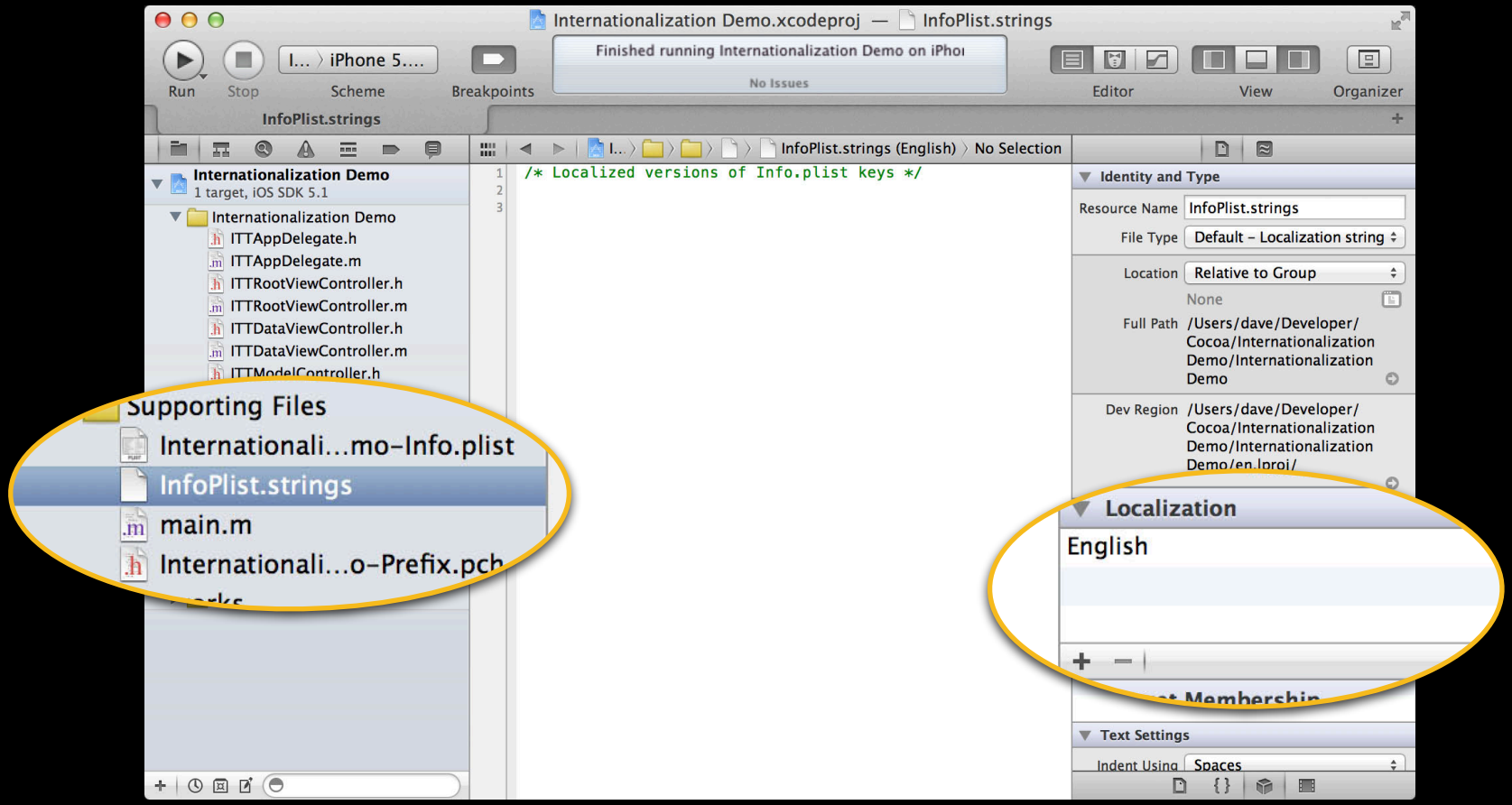
# Internationalization

## Preparing Your App for Translated Content

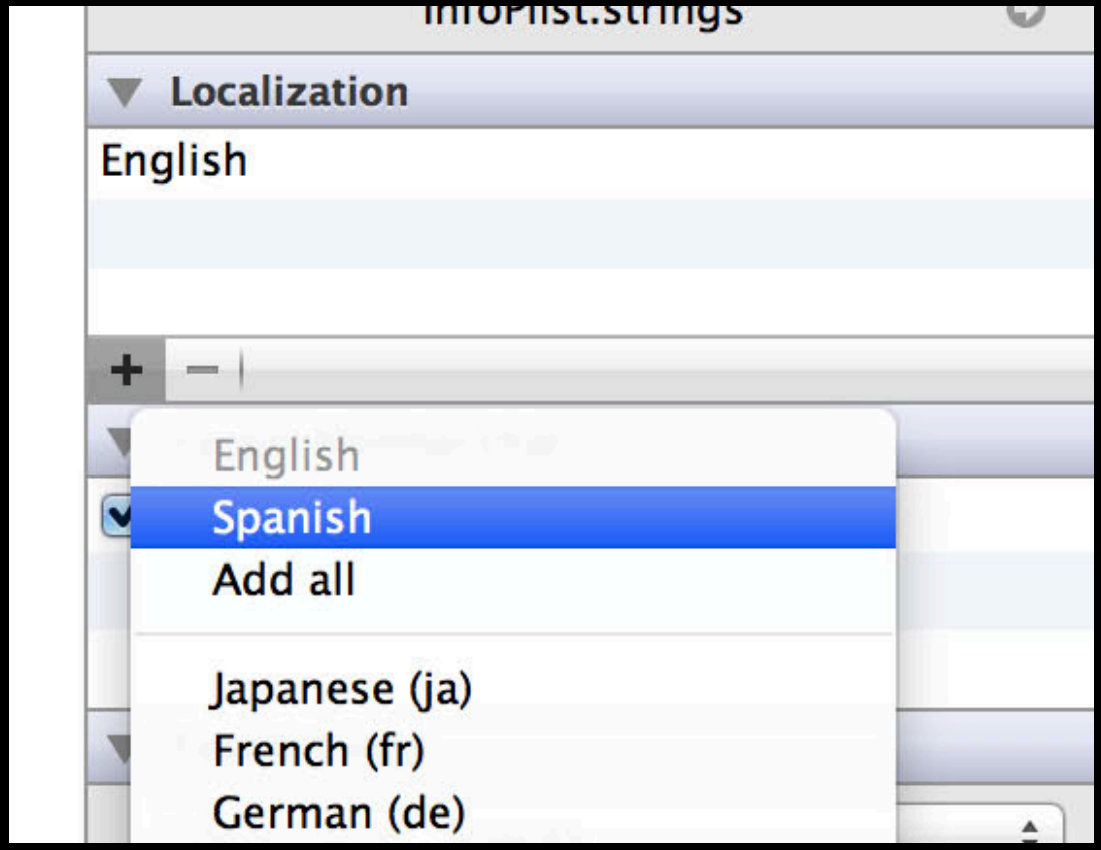
# Localization

## Integrating Translated Content

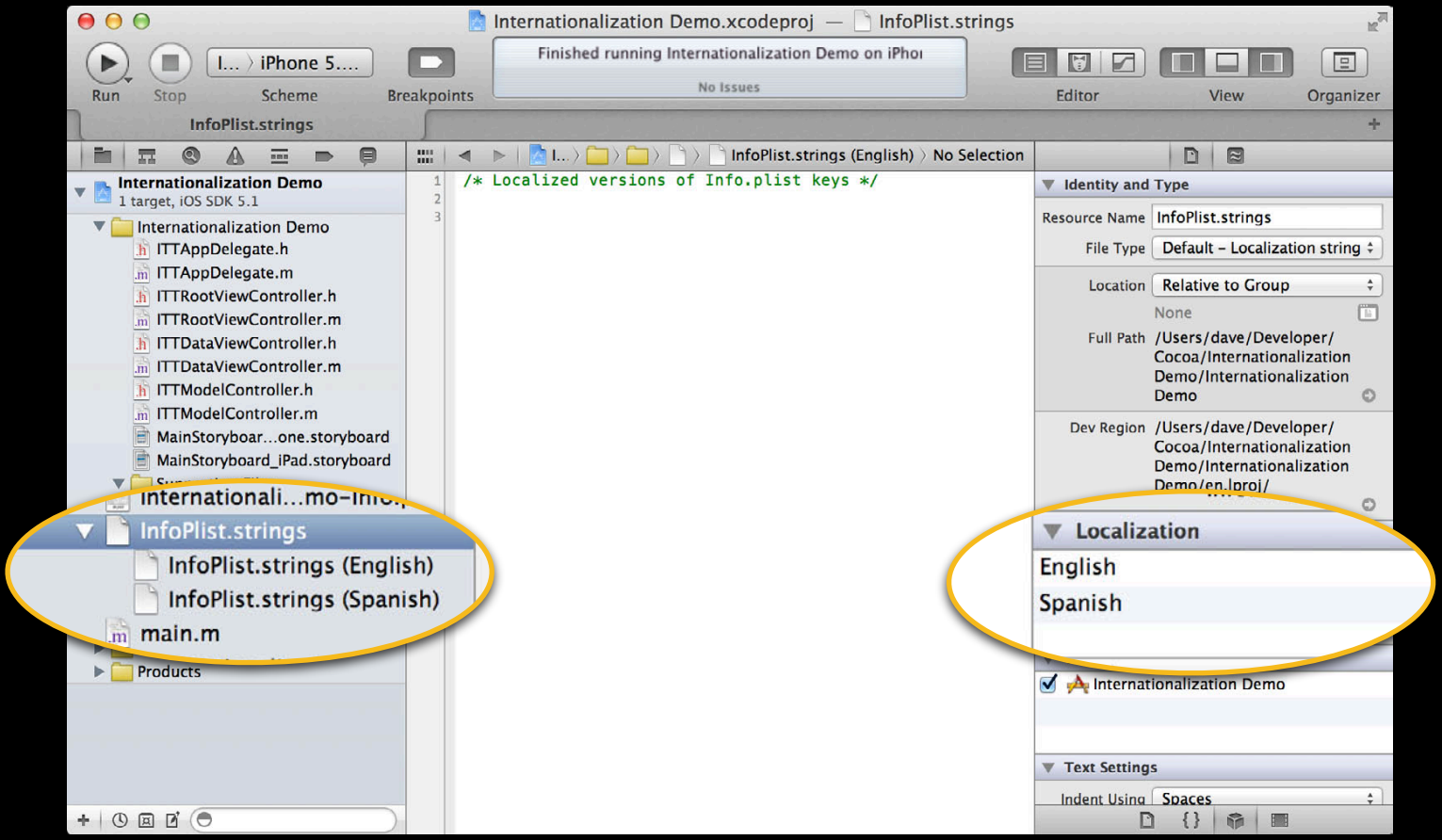
# Localizing Resources



# Localizing Resources

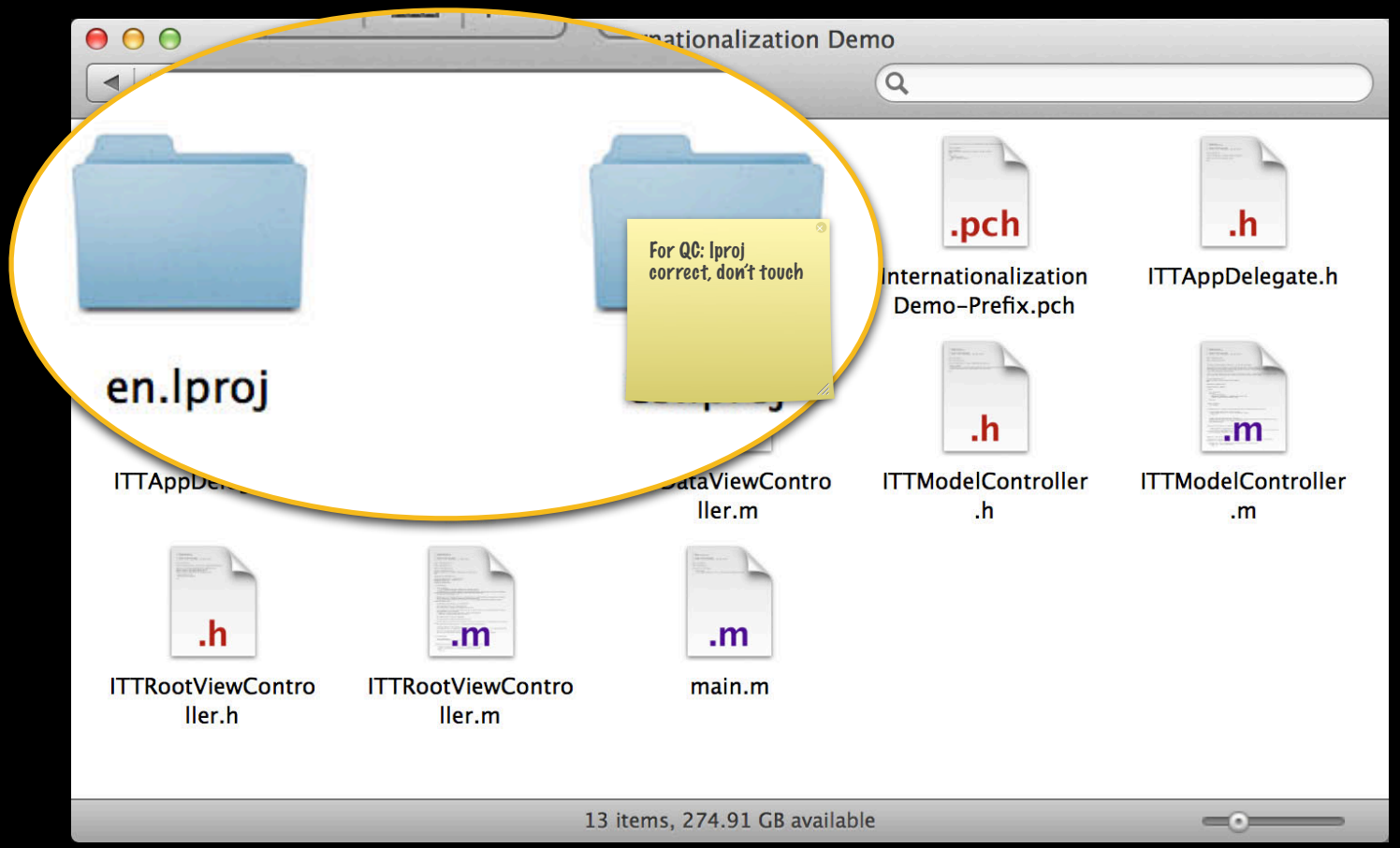


# Localizing Resources

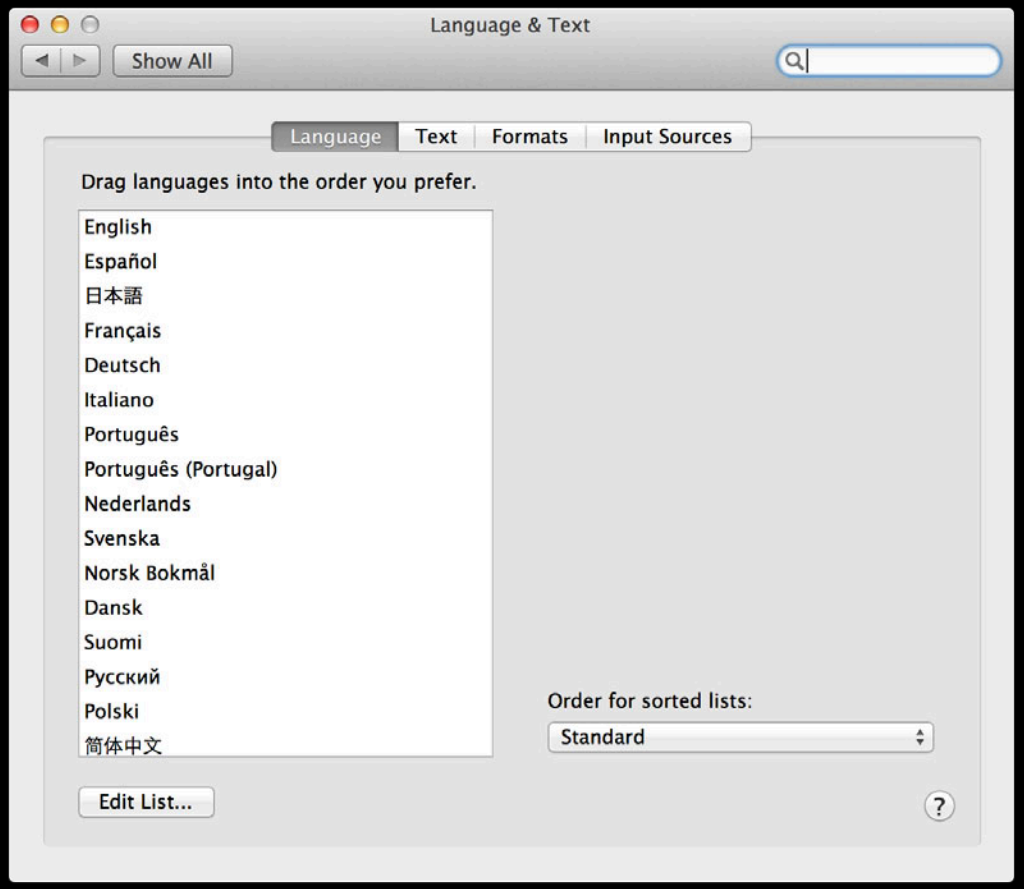


# Language-specific Project Directory

## lproj



# Language and Locale




# Language and Locale

# NSLocale

- Conventions and standards
- `+currentLocale`
  - `+autoupdatingCurrentLocale`
- `NSCurrentLocaleDidChangeNotification`

# Overview

- Text
- Dates
- Numbers
- Images

# Related Sessions

| | |
|---|---|
| Introduction to Auto Layout for iOS and OS X | Mission<br>Tuesday 10:15AM |
| Best Practices for Mastering Auto Layout | Mission<br>Thursday 9:00AM |

# Text

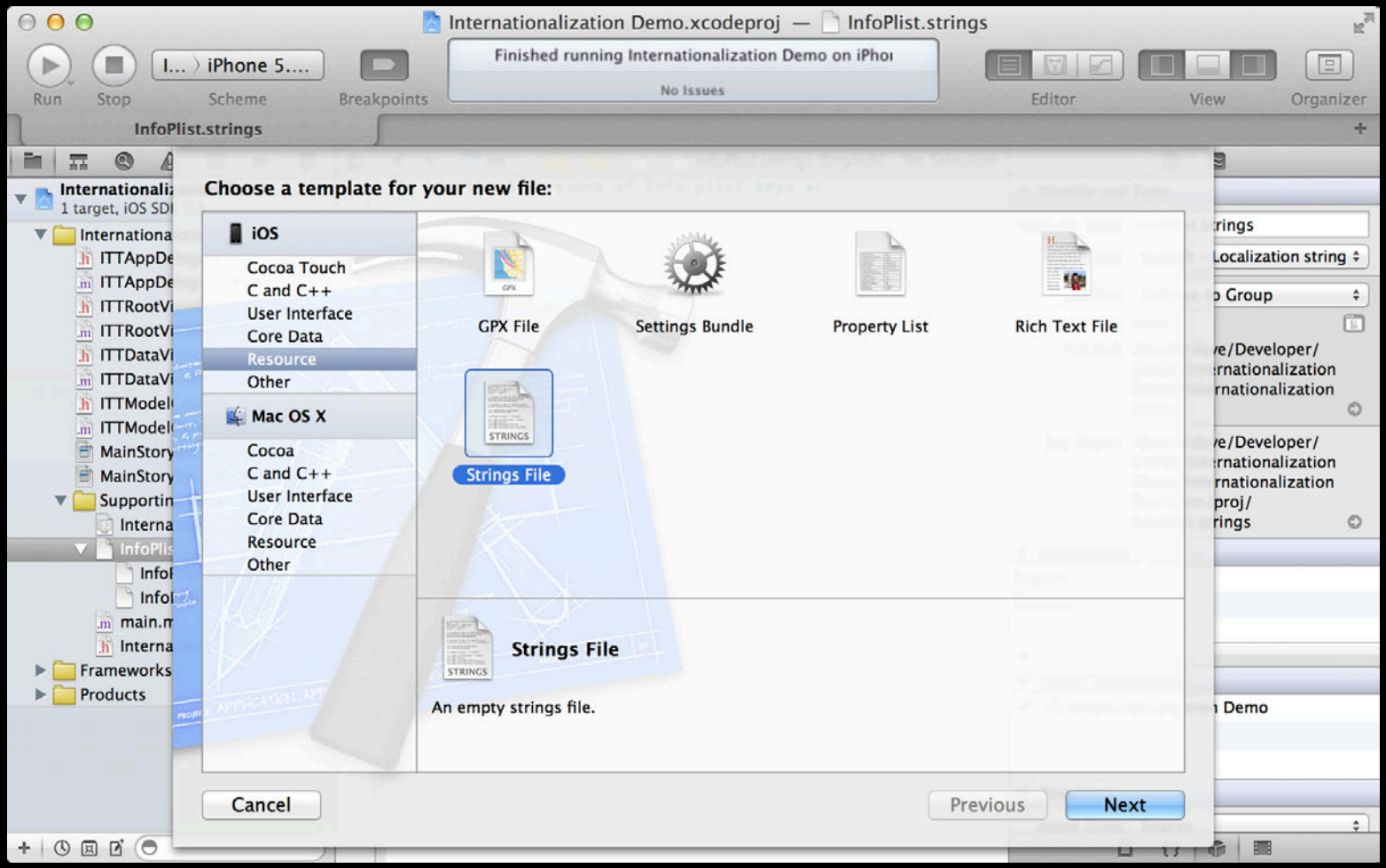
# Text

- Bulk of internationalized content

```
NSLocalizedString
    NSLocalizedStringFromTable
    NSLocalizedStringFromTableInBundle
    NSLocalizedStringWithDefaultValue
-[NSBundle localizedStringForKey:value:table:]
```

# String Tables

- Localized strings stored in “tables”
- Localizable.strings
- Stored as key-value pairs
  - `“Key” = “Value”;`


Internationalization Demo.xcodeproj — InfoPlist.strings
Run
Stop
Scheme
Breakpoints
Finished running Internationalization Demo on iPhoı
No Issues
Editor
View
Organizer
InfoPlist.strings
Choose a template for your new file:
iOS
Cocoa Touch
C and C++
User Interface
Core Data
Resource
Other
Mac OS X
Cocoa
C and C++
User Interface
Core Data
Resource
Other
GPX File
Settings Bundle
Property List
Rich Text File
Strings File
Strings File
An empty strings file.
Cancel
Previous
Next

# genstrings

- Generates strings tables
- Easily integrated into scripts
- Customizable
- `man genstrings`

# genstrings

```
find . -name \*.m | xargs genstrings -o en.lproj/
```

# genstrings

```
NSLocalizedString(
    @”RequiredEnergyAmount”,
    @”the energy needed for the machine to operate“);
```

# genstrings

```
NSLocalizedString(
    @”RequiredEnergyAmount”,
    @”the energy needed for the machine to operate“);
```

```
/*                                                    */
                          =                          ;
```

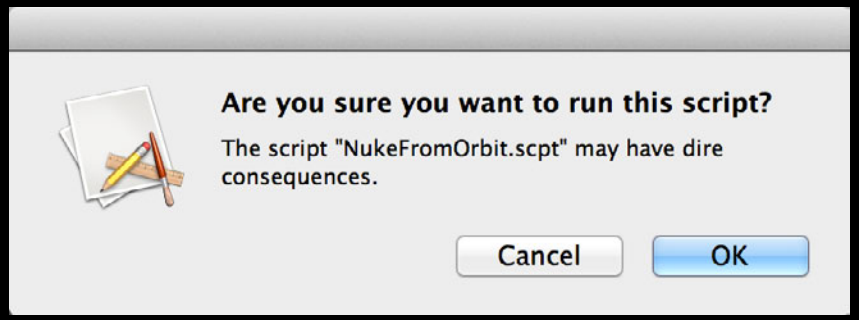
# genstrings

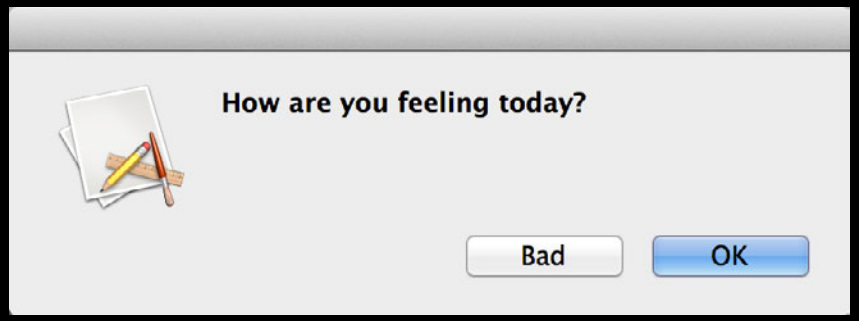
```
NSLocalizedString(
    @”RequiredEnergyAmount”,
    @”the energy needed for the machine to operate“);
```

```
/* the energy needed for the machine to operate */
“RequiredEnergyAmount” = “1.21 Gigawatts”;
```

# Text Gotchas

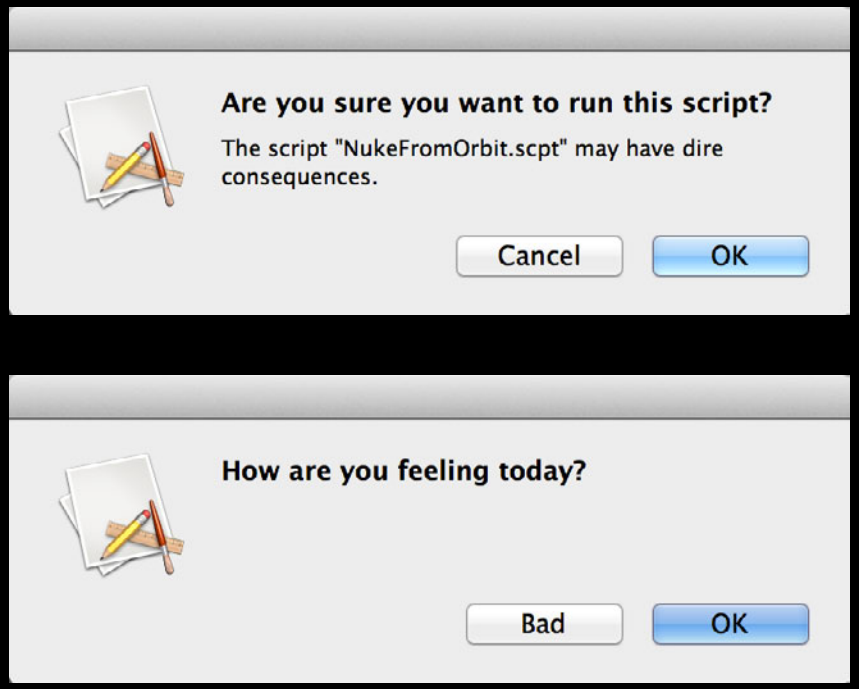
# Overloading Keys



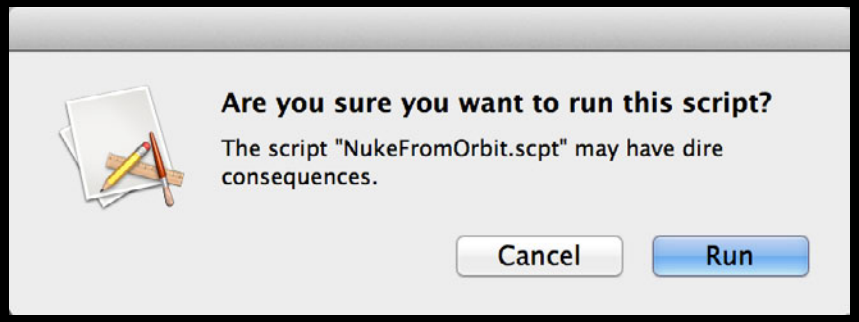

```
NSLocalizedString(@"OK",
 @"Alert button title")
```

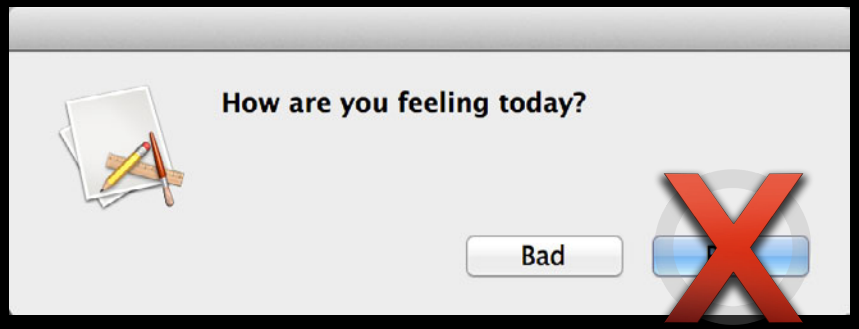
# Overloading Keys




```
NSLocalizedString(@”Run”,
  @”Alert button title”)
```

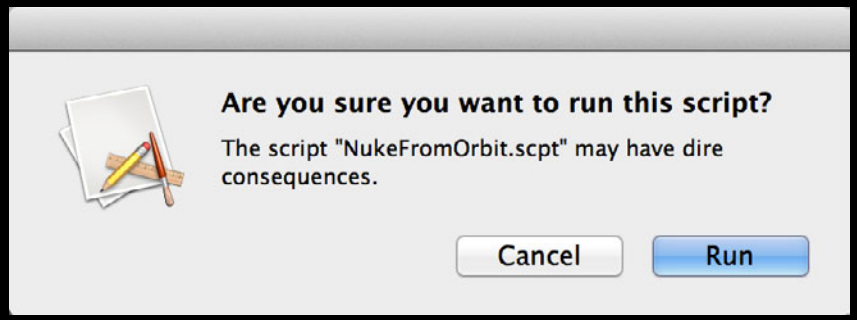
# Overloading Keys



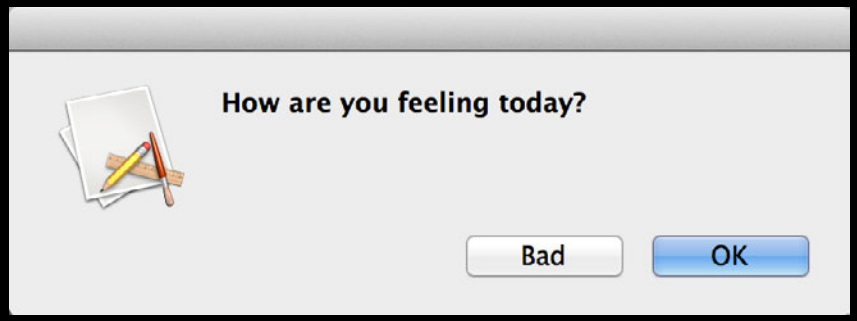




```
NSLocalizedString(@"Run",
  @"Alert button title")
```

# Overloading Keys




```
NSLocalizedString(@”Run”,
 @”Indicates a script may
 be run”)
```




```
NSLocalizedString(@”OK”,
 @”Affirmative response
 to how a user is
 feeling”)
```

# Vague Context

- Comments are there for a reason

```
/* No comment provided by engineer */
“Show” = “Show”;
```

# Vague Context

- Comments are there for a reason

```
/* One of a list of activities to go do.
   Ex: “go see a show”  */
“ActivityListOptionShow” = “Show”;
```

# Composing

- The unit of translation is the sentence
- Number and gender must agree

# Composing

```
/* Go to next [chapter, page] */
“GoToNext” = “Go to next %@”;
“Page”     = “page”;
“Chapter”  = “chapter”;
```

Go to next page

Fem. Fem.

# Composing

```
/* Go to next [chapter, page] */
“GoToNext” = “Go to next %@”;
“Page”     = “page”;
“Chapter”  = “chapter”;
```

Go to the next page

Fem. Fem. Fem.

# Composing


Go to the next chapter

Mas. Mas. Mas.

# Composing

```
/* button title: Go to next page */
“GoToNextPage”    = “Go to next page”;

* button title: Go to next chapter */
“GoToNextChapter” = “Go to next chapter”;
```

# Sorting

- `-compare:` is wrong
- Use `-localizedStandardCompare:`
- Avoid case or diacritic insensitivity

# Sorting (b, Ä, c, ä, A, C, a, B) in Swedish

| | |
|---|---|
| `compare:` | A, Ä, B, C, a, ä, b, c |
| `localizedCaseInsensitiveCompare:` | A, a, b, B, c, C, ä, Ä |
| `localizedStandardCompare:` | a, A, b, B, c, C, ä, Ä |

# Searching

- May use case and diacritic insensitivity
- Specify the locale

```
-[NSString rangeOfString:]
```

```
-[NSString rangeOfString:options:range:locale:]
```

# Directionality

- Arabic and Hebrew are predominantly right-to-left
- RTL text can contain LTR text

Visual


Logical

# Parsing

- Not all characters fit in a unichar

```
[@"É" length]  = 1
[@"É" length]  = 2
[@"🍔" length] = 2
[@"🇺🇸" length] = 4
```

# Parsing

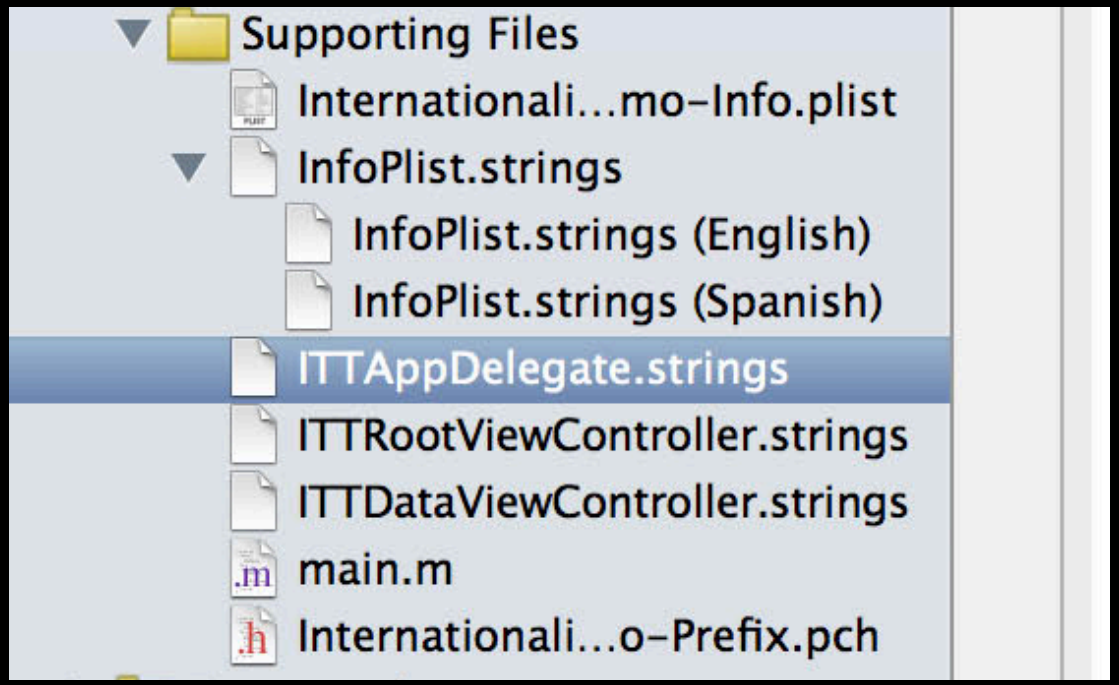
- Avoid
  - `-characterAtIndex:`
  - `-length`
  - `-precomposedStringWithCanonicalMapping`

- `-enumerateSubstringsInRange:options:usingBlock:`

# Text Tips

# Multiple Tables

## Encapsulation



# Pseudolocalize

## Easily find untranslated text

# Pseudolocalize

## Easily find untranslated text

# Custom Macros

```
NSLocalizedString(@”BackButtonTitle”,
 @”button title”);
```

# Custom Macros

```
ITTRootViewControllerString(@”BackButtonTitle”,
 @”button title”);
```

# Custom Macros

```
NSString* ITTRootViewControllerString(
 NSString *key,
 NSString *comment)
{
 NSBundle *b = [NSBundle mainBundle];
 NSString *str = [b localizedStringForKey:key
  value:nil table:@"ITTRootViewController"];
 #if DEBUG
  str = ITTPseudolocalize(str);
 #endif
 return str;
}
```

# Custom Macros

```
NSString* ITTRootViewControllerString(
 NSString *key,
 NSString *comment)
{
 NSBundle *b = [NSBundle mainBundle];
 NSString *str = [b localizedStringForKey:key
  value:nil table:@"ITTRootViewController"];
 #if DEBUG
  str = ITTPseudolocalize(str);
 #endif
 return str;
}
```

# Dates

# NSDate

- Wraps an `NSTimeInterval`
- Independent of any calendar

# NSDateFormatter

Converts a date to and from human-readable form

Initialized with current system settings

Locale-sensitive

`+localizedStringFromDate:dateStyle:timeStyle:`

# Date Gotchas

# Format Strings

- Avoid explicit format strings for user-visible dates

```
NSDateFormatter *df = [NSDateFormatter new];
[df setDateFormat:@"M/dd/yyyy"];
```

| | |
|---|---|
| English (US) | 6/15/2012 |
| Chinese (China) | 6/15/2012 |

# Format Strings

- Avoid explicit format strings for user-visible dates

```
NSDateFormatter *df = [NSDateFormatter new];
[df setDateStyle:NSDateFormatterShortStyle];
```

| | |
|---|---|
| English (US) | 6/15/2012 |
| Chinese (China) | 12年6月15日 |

# Localizing Format Strings

- When the default styles don't meet your needs

```
NSDateFormatter *df = [NSDateFormatter new];
[df setDateFormat:@"MMM d"];
```

| | |
|---|---|
| French (France) | juin 15 |
| Chinese (China) | 6月 15 |

# Localizing Format Strings

- When the default styles don't meet your needs
- +dateFormatFromTemplate:options:locale:

```
NSString *format = [NSDateFormatter
  dateFormatFromTemplate:@"MMM d"
  options:0 locale:[NSLocale currentLocale]];
[dateFormatter setDateFormat:format];
```

| | | |
|---|---|---|
| French (France) | "d MMM" | 15 juin |
| Chinese (China) | "M月d日" | 6月15日 |

# Parsing Date Strings

- Date formatters should be explicitly configured

```
NSDateFormatter *df = [NSDateFormatter new];
[df setDateFormat:@”MM-dd-yyyy”];
NSDate *date = [df stringFromDate:@”06-15-2012”];
```

# Parsing Date Strings

- Date formatters should be explicitly configured

```
NSDateFormatter *df = [NSDateFormatter new];
NSCalendar *gregorian = [[NSCalender alloc]
  initWithCalendarIdentifier:NSGregorianCalendar];
[df setCalendar:gregorian];
[df setDateFormat:@”MM-dd-yyyy”];
NSDate *date = [df stringFromDate:@”06-15-2012”];
```

# Parsing Date Strings

- Use `strptime_l` with unlocalized dates

# NSCalendar

- Represents a calendaring system
- Calendrical calculations

# NSCalendar

- Represents a calendaring system
- Calendrical calculations

# Timezones

- Storing/parsing a date without fixing the timezone

```
[dateFormatter setDateFormat:@"yyyy-MM-dd HH:mm:ss"];
date = [dateFormatter dateFromString:@"2012-06-15 09:42:00"];
```

| | |
|---|---|
| **In San Francisco (GMT-07:00)** | June 15, 2012 9:42:00 AM |
| **In Beijing (GMT+08:00)** | 2012年06月15日 09:42:00 |

# Timezones

- Storing/parsing a date without fixing the timezone

```
[df setTimeZone:[NSTimeZone timeZoneWithSecondsFromGMT:0]];
[df setDateFormat:@"yyyy-MM-dd HH:mm:ss"];
date = [df dateFromString:@"2012-06-15 09:42:00"];
```

| | |
|---|---|
| **In San Francisco (GMT-07:00)** | June 15, 2012 2:42:00 AM |
| **In Beijing (GMT+08:00)** | 2012年06月15日 17:42:00 |

# NSTimeZone

- Represents an offset from UTC

# Date Tips

## Proper care and feeding of calendars

# 86,400

## (24 * 60 * 60)

- 1 day ≠ 86,400 seconds

```
NSDate *start = ...; //   3/10/2012 11:50 PM
NSDate *end = [start dateByAddingTimeInterval:86400];
// end is “3/12/2012 00:50 AM”
```

# 86,400

## (24 * 60 * 60)

- 1 day ≠ 86,400 seconds

```
NSDate *start = ...; //   3/10/2012 11:50 PM
NSDateComponents *oneDay = [NSDateComponents new];
[oneDay setDay:1];

NSCalendar *cal = [NSCalendar currentCalendar];
NSDate *end = [cal dateByAddingComponents:oneDay
  toDate:start options:0];
// end is “3/11/2012 11:50 PM”
```

# NSDateComponents

- Represents portions of a calendrical date
- Absolute or relative

```
NSDateComponents *c = [NSDateComponents new];
[c setMonth:13];
[c setDay:3];
[c setYear:2000];
```

# Unit Lengths

```
-[NSCalendar maximumRangeOfUnit:]
-[NSCalendar rangeOfUnit:inUnit:forDate:]
```

# AM/PM

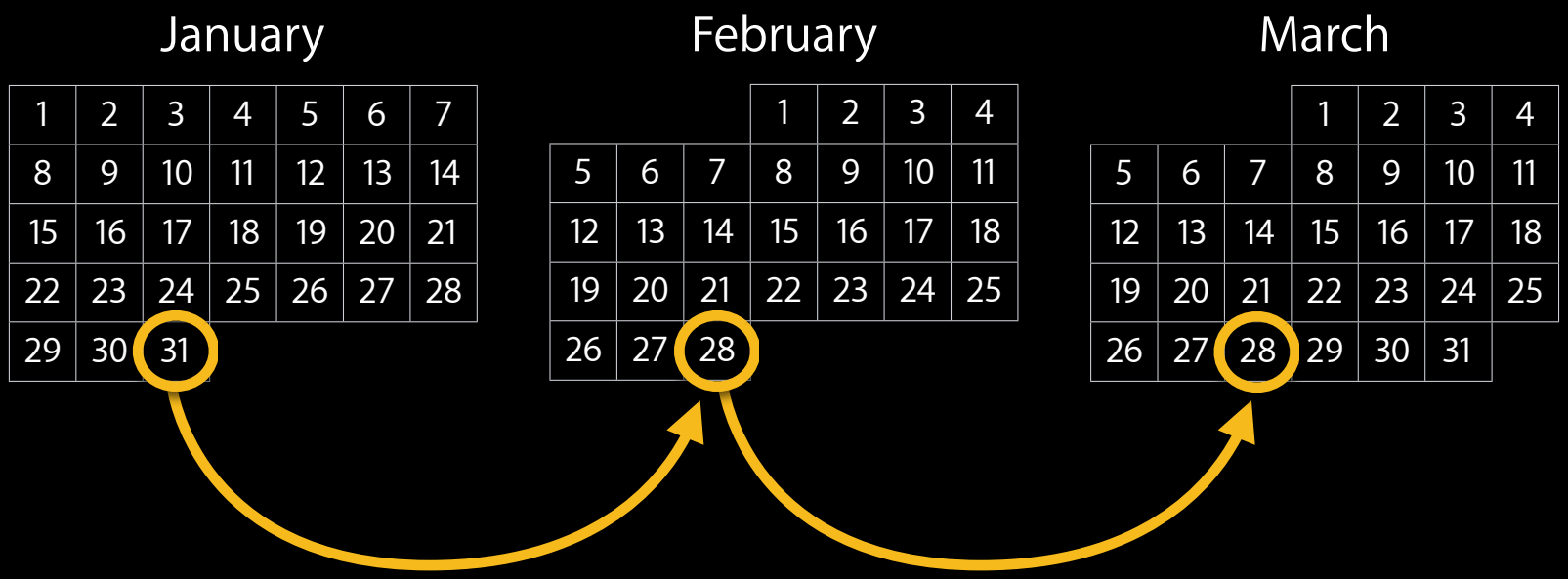
```
@”j”
```

```
NSString *timeFormat = [NSDateFormatter
  dateFormatFromTemplate:@”j”
  options:0
  locale:[NSLocale currentLocale]];
```

| | |
|---|---|
| United States | h a |
| Japan | H |
| France | HH |
| China | ah |
| Korea | a h |

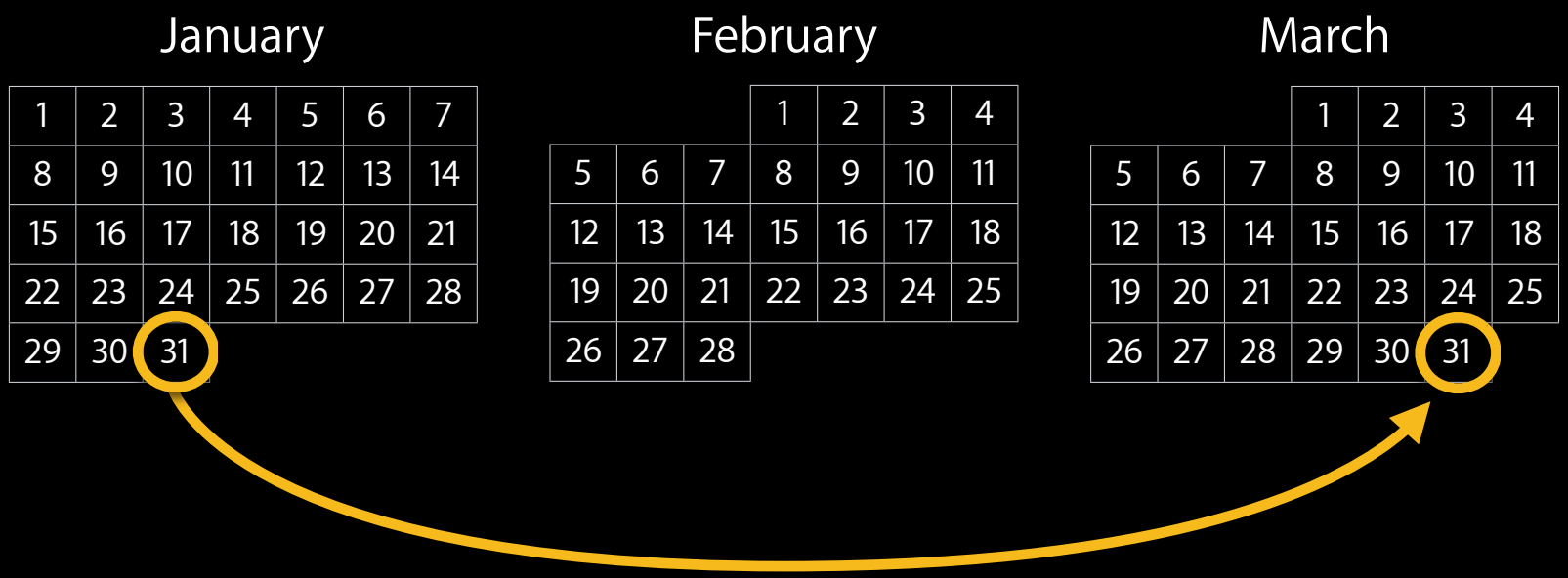
# Date Math

## 1+1≠2



# Date Math

## 1+1≠2



# Numbers

# Localizing Numbers

`NSNumberFormatter`

Locale-sensitive

`+localizedStringFromNumber:numberStyle:`

# Number Styles

`@1234.56`

| | |
|---|---|
| **General** | 1,234.56 |
| **Currency** | $1,234.56 |
| **Percentage** | 123,456% |
| **Scientific** | 1.23456E+03 |
| **Spell-out** | One thousand two hundred thirty-four point five six |

# Locale Sensitivity

`@1234.56`

| | | |
|---|---|---|
| **Digits** | 1,234.56 | ١,٢٣٤,٥٦ |
| **Decimal point and grouping separator** | 1,234.56 | 1 234,56 |
| **Currency** | $123.45 | ¥123.45 |
| **Percentage** | 45% | ٤٥٪ |

# Number Gotchas

# Parsing

- Avoid –`intValue`, `scanf`, etc

```
[@”asdf” intValue] → 0
[@”0” intValue]    → 0
```

# Parsing

- Avoid -intValue, scanf, etc

```
NSNumberFormatter *nf = [NSNumberFormatter new];
[nf setNumberStyle:NSNumberFormatterDecimalStyle];
NSNumber *asdf = [nf numberFromString:@”asdf”]; // nil
NSNumber *zero = [nf numberFromString:@”0”];    // @0
```

# Printing

- Avoid +stringWithFormat:, printf, etc

```
[NSString stringWithFormat:@"%3.2f", myNumber];
```

| | |
|---|---|
| English (US) | 241.23 |
| Arabic (Egypt) | 241.23 |

# Printing

- Avoid +stringWithFormat:, printf, etc

```
[NSNumberFormatter localizedStringFromNumber:myNumber
     numberStyle:NSNumberFormatterDecimalStyle];
```

| | |
|---|---|
| English (US) | 241.23 |
| Arabic (Egypt) | ٢٤١٫٢٣ |

# Printing


```
[NSString localizedStringWithFormat:@"%3.2f", myNumber];
```

| | |
|---|---|
| English (US) | 241.23 |
| Arabic (Egypt) | ٢٤١٫٢٣ |

# Losing the Locale

- Setting a format overrides the locale

```
[numberFormatter setFormat:@"$#,##0.00"];
[numberFormatter stringFromNumber:@241.23];
```

| | |
|---|---|
| English (US) | $241.23 |
| Chinese (China) | $241.23 |

# Losing the Locale

- Setting a format overrides the locale

```
[numberFormatter setNumberStyle:NSNumberFormatterCurrencyStyle];
[numberFormatter stringFromNumber:@241.23];
```

| | |
|---|---|
| English (US) | $241.23 |
| Chinese (China) | ¥241.23 |

# Number Tips

# Tips

- Don’t forget unit conversion
  - $42 ≠ €42 ≠ ¥42
  - 1 km ≠ 1 mile
  - Ask `NSLocale`
- Use `strto*_l` functions for unlocalized numbers
- Use predefined formats as much as possible

# Images

Beware


ver

# Abeja Inteligente

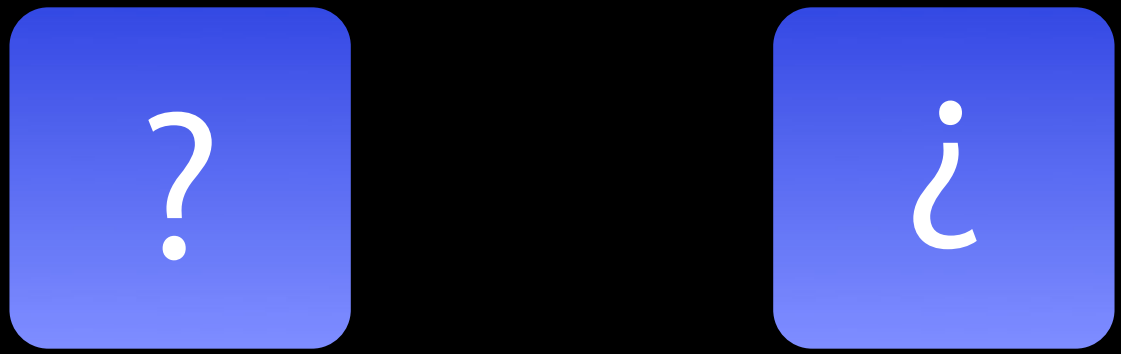
"Smart honeybee"

# Culture-Specific Content



```
[UIImage imageNamed:@”StopSign”]
[NSImage imageNamed:@”StopSign”]
```

# Text in Images



# Summary

- Standards vary
- Cultures vary
- Test, test, test
- Built-in APIs

# More Information

**Paul Marcos**
Application Services Evangelist
pmarcos@apple.com

**Documentation**
Introduction to Internationalization Programming Topics
https://developer.apple.com/

**Apple Developer Forums**
http://devforums.apple.com

# Related Sessions

| | |
|---|---|
| Introduction to Auto Layout for iOS and OS X | Mission<br>Tuesday 10:15AM |
| Accessibility for iOS | Russian Hill<br>Wednesday 9:00AM |
| Best Practices for Mastering Auto Layout | Mission<br>Thursday 9:00AM |

# Labs

| | |
|---|---|
| Internationalization Lab | App Services Lab A<br>Friday 11:30AM |


WWDC2012