Today’s Topics

- Localization
- Accessibility
- Undo
- Copy Paste
Localization
Your International Users

• Choose a language

• Choose a region

Region Format Example
Tuesday, January 5, 2010
12:34 AM
(408) 555-1212

Exemple de format pour cette région
mardi 5 janvier 2010
00:34
(408) 555-1212
Your International Users

- Choose a language
- Choose a region
Your International Users

- Choose a language
- Choose a region
Your International Users

- Choose a language
- Choose a region
Creating International Apps

• Internationalization – i18n
  ▪ Prepare your app to be used in different languages and locales
  ▪ Developers write code that can later be localized

• Localization – L10n
  ▪ Add localized data for specific languages and locales
  ▪ Localizers provide translations of resources
  ▪ Some localization done by developers – dates, currency, ...
Localized Resources

• Language and locale dependent values
  ▪ Static Text
  ▪ Dynamic Text
  ▪ Formats
  ▪ Graphics
  ▪ Audio with spoken language
  ▪ Sorting rules
What Should Be Localized

- Strings

<table>
<thead>
<tr>
<th>English</th>
<th>French</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hello</td>
<td>Bonjour</td>
</tr>
</tbody>
</table>

- Graphics
What Needs Localizing

- Dates

<table>
<thead>
<tr>
<th>English / US</th>
<th>French / FR</th>
<th>French / Canada</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jan 5, 2010</td>
<td>5 janv. 2010</td>
<td>2010-01-05</td>
</tr>
</tbody>
</table>

- Numbers

<table>
<thead>
<tr>
<th>English / US</th>
<th>French / FR</th>
<th>French / Canada</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1,000.50</td>
<td>1 000,00 E</td>
<td>1 000,00 $</td>
</tr>
</tbody>
</table>

- Time

<table>
<thead>
<tr>
<th>English / US</th>
<th>French / FR</th>
<th>French / Canada</th>
</tr>
</thead>
<tbody>
<tr>
<td>12:34 AM</td>
<td>00:34</td>
<td>00:34</td>
</tr>
</tbody>
</table>
Where Do Localized Resources Go?

- In your application bundle
- MyApp.app/
  - MyApp
  - English.lproj/
    - Localizable.strings
    - MyView.nib
  - Japanese.lproj/
    - Localizable.strings
    - MyView.nib
Accessing Localized Resources

• General Resources
  ▪ Use NSBundle API to load localized versions

[[NSBundle mainBundle] pathForResource:@"CapitalCities" ofType:@"plist"]

• Strings
  ▪ Localized versions of strings stored in “Localizable.strings”
  ▪ Use NSLocalizedString API

NSLocalizedString(@"Hello", ...)

• Formats
  ▪ Use NSFormatter subclasses (NSDateFormatter, NSNumberFormatter)
International Code – Strings

- Wrong

```objective-c
NSString *title = @"Download Error";
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:title ...];
```

- Correct

```objective-c
NSString *title = NSLocalizedString(@"Download Error", ...);
UIAlertView *alert = [[UIAlertView alloc] initWithTitle:title ...];
```
Strings File Example

```objective-c
NSString *title = NSLocalizedString("Download Error", "download error message");
```

**English.lproj/Localizable.strings**

```plaintext
/* download error message */
"Download Error" = "Download Error"
```

**French.lproj/Localizable.strings**

```plaintext
/* download error message */
"Download Error" = "Erreur de téléchargement"
```
Strings File Example

```objective-c
NSString *title = NSLocalizedString(@"DOWNLOAD_ERROR", @"download error message");
```

**English.lproj/Localizable.strings**

```objective-c
/* download error message */
"DOWNLOAD_ERROR" = "Download Error"
```

**French.lproj/Localizable.strings**

```objective-c
/* download error message */
"DOWNLOAD_ERROR" = "Erreur de téléchargement"
```
Strings File Example

- Positional Parameters
  - In some languages, the order of parameters may need to change
  - Localizers can specify the order!

- Example

```objective-c
"HELLO_FIRST_AND_LAST_NAME_FORMAT" = "Hello %1$@ %2$@!";

NSString *fmt = NSLocalizedString(@"HELLO_FIRST_AND_LAST_NAME_FORMAT", @"");
NSString *helloString = [NSString stringWithFormat:fmt, @"Adam", @"Porter"]
```

Result: “Hello Adam Porter!”
Strings File Example

• Positional Parameters
  
  • In some languages, the order of parameters may need to change
  
  • Localizers can specify the order!

• Example

```
"HELLO_FIRST_AND_LAST_NAME_FORMAT" = "Hello %2$@ %1$@!";
```

```
NSString *fmt = NSLocalizedString(@"HELLO_FIRST_AND_LAST_NAME_FORMAT", @"");
NSString *helloString = [NSString stringWithFormat:fmt, @"Adam", @"Porter"]
```

Result: “Hello Porter, Adam!”
Genstrings Tool

- /usr/bin/genstrings

- Tool to scan your code and produce a .strings file
  - Looks for all uses of NSLocalizedString
  - Pulls out the comment field for use as clues to the localizer

- Run the tool over your *.m files

```
> genstrings -u Classes/*.m -o output/
> ls output/
Localizable.strings
```
Localized Sorting

• Correct sorting depends on the context
  ▪ Example: In Finnish, V and W are equal when sorting

• NSString

```c
- (NSComparisonResult)localizedCompare:(NSString *)s;
- (NSComparisonResult)localizedCaseInsensitiveCompare:(NSString *)s;

- (NSComparisonResult)compare:(NSString *)s options:(NSStringCompareOptions)o range:(NSRange)range locale:(NSLocale *)locale;
```

• NSStringCompareOptions
  ▪ Case Insensitive
  ▪ Numeric Search
  ▪ Others...
Other Resources

• Use NSBundle to load images, audio, plists,...

[[NSBundle mainBundle] pathForResource:@"CapitalCities" ofType:@"plist"]

• Nibs are loaded using NSBundle under the covers!

• NSBundle automatically uses the best localization
  • Looks for non-localized version first
  • If none, looks in appropriate .lproj directory
Internationalizing NIBs

• Plan for different string lengths in different languages
  ▪ Good idea to start with German...

• Options for localization
  ▪ Duplicate file, edit by hand (make sure to lock all non-localized editing)
  ▪ Use IBTool to extract strings, and generate new xibs
IBTool Process

- /usr/bin/ibtool
- Pull strings out of xib files
  
  ```ibtool --generate-stringsfile MyNib.strings MyNib.nib```

- Edit strings

- Create a new xib file with modified strings
  
  ```ibtool --strings-file MyNib.strings --write MyNewNib.nib MyNib.nib```

- IBTool has options for **incremental** localization
Localizing A Resource
Localizing A Resource
Localizing A Resource
Locale Based Information

- Information related to a user's language and region settings
Locale Based Information

- Information related to a user's language and region settings

![Image of two smartphones displaying language and region settings]
Locale Based Information

- Information related to a user's language and region settings
Locale Based Information

• Locale based information
  ▪ Measurements
  ▪ Currency
  ▪ Country information
  ▪ Formatting – Dates, Numbers

• NSLocale used to provide locale related information
  ▪ look up decimal separator, currency symbol
  ▪ find out if metric system is used

• NSFormatters used to format and interpret strings
International Code – Date Formatting

• Wrong

```c
int m = // get month from date
int d = // get day of month from date
int y = // get year from month
NSString *string = [NSString stringWithFormat: @”%d / %d / %d”, m,d,y];
```

• Correct

```c
NSDateFormatter *formatter = [[NSDateFormatter alloc] init];
[formatter setDateFormatStyle: NSDateFormatterNoStyle];
[formatter setFormatStyle: NSDateFormatterShortStyle];
NSString *string = [formatter stringFromDate: date];
```
International Code – Number Formatting

• Wrong

```swift
float floatValue = // some value to display...
NSString *string = [[NSString alloc] initWithFormat:@"%.2f", floatValue];
```

• Correct

```swift
NSNumberFormatter *formatter = [[NSNumberFormatter alloc] init];
(formatter setNumberStyle: NSNumberFormatterDecimalStyle]

NSNumber *number = [NSNumber numberWithFloat: floatValue];
NSString *string = [formatter stringFromNumber:number];
```

• Other Styles – Currency, Percent, Scientific, “Spell Out”
Calendar And Dates

• Year depends on what calendar a user prefers

• Gregorian calendar: Fri, 8 January 2010

• Others
  ▪ Hebrew – 22nd of Tevet, 5770
  ▪ Islamic Calendar – 1431

• Use NSCalendar data for the current locale

@interface NSCalendar : NSObject <NSCopying, NSCoding>

+ (id)currentCalendar; // users preferred calendar, tracks changes
Undo
### Undo / Redo

- **NSUndoManager** – general-purpose recorder of operations
  - Before changing a undoable value, tell NSUndoManager

- To register an **undo**
  - Provide the object that needs to change
  - Specify a method and arguments required to undo a change

- Registering a **redo**
  - During an undo, registering stored on a redo stack!
Managing Undo

• Usually your UIViewController will manage undo
  ▪ But, any UIResponder can

• Create an undo manager

```objective-c
NSUndoManager *undoMgr = [[NSUndoManager alloc] init]
```

• Register an undo/redo when a change is applied

```objective-c
- (void)changeBookTitle:(NSString *)bookTitle {
    [[undoMgr prepareWithInvocationTarget:self] changeBookTitle: [book title]];
    [undoMgr setActionName:@"Book Title"];
    [book setTitle: @"New Title"];
}
```
Some Implementation Details

[[undoMgr prepareWithInvocationTarget:self] changeBookTitle: [book title]];

- This shows of some cool Objective-C tricks
- What is prepareWithInvocationTarget: doing???
  - A “proxy” object is returned, which does not implement “changeBookTitle”
  - Objects that don’t respond to message are given a chance to forward it

- (void)forwardInvocation:(NSInvocation *)invocation {
  // save it for later!
  savedInvocation = [invocation retain];
}

  - The “proxy” simply saves the invocation for later
  - When an undo must execute, the saved invocation is replayed

  // replay the undo
  [savedInvocation invoke];
Performing Undo

- Device – shake it!
- Simulator – Hardware > Shake Gesture
- A few details you need to handle…

![Undo Typing, Redo Typing, Cancel]
Undo Notes

• Undo performed automatically by UIKit
  ▪ Searches responder chain to find the first -[UIResponder undoManager]
  ▪ Responder chain includes UIViews and their UIViewControllers

• NSUndoManager does not retain targets
  ▪ If an object is going away, make sure it’s not the target of any undo operations

[undoManager removeAllActionsWithTarget:object]
Copy / Paste
Copy / Paste

- Standard UIKit controls handle copy / paste for you
- Implement your own
  - Present the standard copy / paste menu
  - Implement `-copy:` and `-paste:` method in your UIResponder
    - Place data on the pasteboard
    - Retrieve data from the pasteboard
Presenting The Menu

- UIMenuController presents a menu interface for copy/paste
  - See UIResponder.h – cut, copy, paste, select, select all
- Get the shared UIMenuController
- Tell it where to show the menu

UIMenuController *theMenu = [UIMenuController sharedMenuController];
[theMenu setTargetRect:aRectangle inView:self];
[theMenu setMenuVisible:YES animated:YES];

- Menu automatically dismissed after the action method is called
Presenting The Menu

• Introspection discovers methods the first responder implements

• Menu displays items for only implemented methods

- (void)cut:(id)sender { }
- (void)copy:(id)sender { }
- (void)paste:(id)sender { }

• Additionally, can filter out items dynamically

- (BOOL)canPerformSelector:(SEL)action withSender:(id)sender {
  BOOL canPerform = NO;
  if (action == @selector(cut:) || action == @selector(copy:)) {
    canPerform = // YES if we really have some data!
  } else {
    canPerform = [super canPerformSelector:action ......
  }
}
Pasteboard

- Copy / Paste data lives on the pasteboard
- Multiple pasteboards
  - Shared system pasteboard – general copy / paste
  - Find pasteboard – holds the most recent search string
  - Custom named pasteboard – create any private pasteboard you want

- Pasteboard API
  - Determine type of data available
  - Get and set the pasteboard data
  - Pasteboards can be marked as persistent
Data On The Pasteboard

- Pasteboard stores items – type (UTI) and data
- Each item can consist of multiple representations
  - E.g. pictures – may want to supply a png, and a jpeg
- Example: one item, multiple representations
Data On The Pasteboard

• Convenience API for working with standard types

```objective-c
@property(nonatomic, copy, readwrite) NSString *string;
@property(nonatomic, copy, readwrite) NSArray *strings;
@property(nonatomic, copy, readwrite) UIImage *image;
@property(nonatomic, copy, readwrite) NSArray *images;
@property(nonatomic, copy, readwrite) NSURL *URL;
@property(nonatomic, copy, readwrite) NSArray *URLs;
...
```

• Example

```objective-c
- (void)copy:(id)sender {
    UIPasteboard *pboard = [UIPasteboard generalPasteboard];
    [pboard setString: [self title]];
}

- (void)paste:(id)sender {
    UIPasteboard *pboard = [UIPasteboard generalPasteboard];
    [self setTitle: [myPasteboard string]];
}
```
Data On The Pasteboard

• Binary data – NSData
  ▪ Your own custom data
  ▪ Possibly created using NSKeyedArchiver and NSCoding

```c
- (void)setData:(NSData *)data forPasteboardType:(NSString *)pboardType;
- (NSData *)dataForPasteboardType:(NSString *)pboardType;
```

• Property list objects – NSString, NSURL, NSNumber, NSArray, NSDate

```c
- (void)setValue:(id)value forPasteboardType:(NSString *)pasteboardType;
- (id)valueForPasteboardType:(NSString *)pasteboardType;
```
Accessibility
Accessibility

- Make it easier for people with visual, auditory, and physical disabilities to use their devices

- Features
  - Zoom – magnify the screen
  - White on Black – increases screen contrast
  - Mono Audio – easier to hear
  - Speech – speak text corrects and suggestions
  - Voice Control – control UI with voice commands
  - VoiceOver – screen reading technology
VoiceOver

- An accessible user interface element must provide accurate and helpful information about itself
  - Screen position
  - Name
  - Behavior
  - Value, and type
Accessibility – What You Need To Do

• Make sure your UI is accessible to VoiceOver

• The UI Accessibility APIs provides information to VoiceOver
  ▪ Standard controls and views handle much of the work for you
  ▪ Custom UI should provide information using UI Accessibility APIs

• **UIAccessibility** protocol – provide information about your UI
  ▪ label – succinct label used to identify the UI (e.g. @“Subject Field”)
  ▪ value – the value contained in the UI (e.g. @”RE: today’s game”)
  ▪ traits – is it a button, link, image, sound?
  ▪ frame – the location on screen
    ▪ (any NSObject can be an accessibility element)
Testing Accessibility

• Turn on VoiceOver
  ▪ Settings > General > Accessibility > VoiceOver

• Use “Accessibility Inspector” in the Simulator
  ▪ Settings > General > Accessibility
Testing Accessibility

- Turn on VoiceOver
  - Settings > General > Accessibility > VoiceOver
- Use “Accessibility Inspector” in the Simulator
  - Settings > General > Accessibility
Testing Accessibility

• Turn on VoiceOver
  - Settings > General > Accessibility > VoiceOver

• Use “Accessibility Inspector” in the Simulator
  - Settings > General > Accessibility
Reading

• Internationalization
  ▪ Developer.apple.com – BPInternational.pdf
    ▪ Guidelines for i18n – p.23 - 26
    ▪ Nibs and Strings – p. 31 - 37

• iPhone App Programming Guide.pdf
  ▪ Responder Chain – p. 78-80
  ▪ Cut, Copy, and Paste – p. 97 - 105