Advanced Text Processing

Session 128

Douglas Davidson
Natural Languages Group

Introduction

- Applications often deal with large amounts of text
- Knowledge about that text can help the user
- Mac OS X and iOS have sophisticated APIs for analyzing text

What You'll Learn

- String objects and their structure
- Iteration, matching, searching
- Regular expressions, Data Detectors, and linguistic APIs
- Text checking
- Putting it all together

C String Handling

```
char *src = instring, *dst = outstring + len - 1;

while (*src) *dst-- = *src++;
```

Unicode Strings

```
NSString *              u’string’

java.lang.String        UnicodeString

wchar_t                 std::wstring
```

Unicode String Handling

```
for (i = 0; i < [string length]; i++) {
    unichar c = [string characterAtIndex:i];

    // do something for each character

}
```

Demo

Character Clusters

- The smallest processing unit for most tasks
- Sometimes one character, sometimes two or more
- Composition
  - é = e + ´
  - 각 = ㄱ + ㅏ + ㄱ
- Surrogate pairs
  - 𠀋 U+2000B = 0xD840 + 0xDC0B

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color() range:substringRange];
}];
```

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color() range:substringRange];
}];
```

Character Cluster Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByComposedCharacterSequences
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color() range:substringRange];
}];
```

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Character Cluster Iteration

San_José

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Character Cluster Iteration

San José

Words

- Appropriate processing unit for most transformation tasks
  - Letter-case mapping
  - Spell-checking
- Whitespace is not necessarily the only way to break “words”
  - 正しい日本語です = 正しい + 日本語 + です
  - ภาษาไทย = ภาษา + ไทย

Word Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByWords
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color() range:substringRange];
}];
```

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Word Iteration

Say “正しい日本語です”!

Paragraphs

- The maximum processing unit for all Unicode processing tasks
- Especially important for bi-directional languages like Arabic and Hebrew
- Each paragraph has an overall text flow direction

Paragraph Iteration

```
[string enumerateSubstringsInRange:range
  options:NSStringEnumerationByParagraphs
  usingBlock:^(NSString *substring,
               NSRange substringRange,
               NSRange enclosingRange,
               BOOL *stop){

        [text addAttribute:NSForegroundColorAttributeName
               value:color() range:substringRange];
}];
```

Searching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSCaseInsensitiveSearch |
                  NSDiacriticInsensitiveSearch |
                  NSWidthInsensitiveSearch
          range:range locale:locale];

if (matchRange.length > 0) {
  // found a match
}
```

Matching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSAnchoredSearch |
                  NSCaseInsensitiveSearch |
                  NSDiacriticInsensitiveSearch |
                  NSWidthInsensitiveSearch
          range:range locale:locale];

if (matchRange.length > 0) {
  // found a match
}
```

Matching

```
NSRange matchRange =
  [string rangeOfString:@"resume"
          options:NSAnchoredSearch |
                  NSCaseInsensitiveSearch |
                  NSDiacriticInsensitiveSearch |
                  NSWidthInsensitiveSearch
          range:range locale:locale];

if (matchRange.length > 0) {
  // found a match
}
```

Comparing

```
NSComparisonResult result =
  [string compare:@"resume"
          options:NSCaseInsensitiveSearch |
                  NSDiacriticInsensitiveSearch |
                  NSWidthInsensitiveSearch
          range:range locale:locale];

if (NSOrderedSame == result) {
  // found a match
}
```

Additional Methods

- `hasPrefix:`
- `hasSuffix:`
- `rangeOfCharacterFromSet:`
- `rangeOfComposedCharacterSequenceAtIndex:`
- `rangeOfComposedCharacterSequencesForRange:`

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSCaseInsensitiveSearch
    range:range];              // immutable strings
```

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSCaseInsensitiveSearch
    range:range];              // immutable strings

[mutableString replaceOccurrencesOfString:
    @"resume" withString:@"résumé"
    options:NSCaseInsensitiveSearch
    range:range];              // mutable strings
```

Demo

Regular Expressions

- Use standard ICU regular expression syntax
- Fully Unicode compliant
- All of the usual options (and more) are available

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
  [NSRegularExpression
   regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
   options:NSRegularExpressionCaseInsensitive
   error:&error];
```

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
  [NSRegularExpression
   regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
   options:NSRegularExpressionCaseInsensitive
   error:&error];
```

NSRegularExpression

```
NSError *error = nil;

NSRegularExpression *regex =
 [NSRegularExpression
  regularExpressionWithPattern:@"\\b(i|o)(f|n)\\b"
  options:NSRegularExpressionCaseInsensitive
  error:&error];
```

Regular Expression Iteration

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

      [text addAttribute:NSForegroundColorAttributeName
               value:color range:[match range]];

}];
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Regular Expression Iteration

```
If into in onto of often on and ON.
```

Match Objects

- Objects of class NSTextCheckingResult

```
@property NSTextCheckingType resultType;

@property NSRange range;
```

  - This is the overall range

```
- (NSRange)rangeAtIndex:(NSUInteger)idx;
```

  - These are the ranges of capture groups

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                          [match rangeAtIndex:1];
    NSRange secondHalfRange =
                          [match rangeAtIndex:2];

}];
```

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                          [match rangeAtIndex:1];
    NSRange secondHalfRange =
                          [match rangeAtIndex:2];

}];
```

Regular Expression Ranges

```
[regex enumerateMatchesInString:string
  options:0 range:range
  usingBlock:^(NSTextCheckingResult *match,
               NSMatchingFlags flags, BOOL *stop){

    NSRange matchRange = [match range];
    NSRange firstHalfRange =
                          [match rangeAtIndex:1];
    NSRange secondHalfRange =
                          [match rangeAtIndex:2];

}];
```

Additional Methods

- matchesInString:options:range:
- numberOfMatchesInString:options:range:
- firstMatchInString:options:range:
- rangeOfFirstMatchInString:options:range:

```
NSRange matchRange =
  [regex rangeOfFirstMatchInString:string
         options:0 range:searchRange];

if (matchRange.length > 0) {
  // found a match
}
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings

[regex replaceMatchesInString:mutableString
    options:0
    range:range
    withTemplate:@"$2$1"];    // mutable strings
```

Search and Replace

```
NSString *modifiedString =
  [regex stringByReplacingMatchesInString:string
    options:0
    range:range
    withTemplate:@"$2$1"];    // immutable strings

[regex replaceMatchesInString:mutableString
    options:0
    range:range
    withTemplate:@"$2$1"];    // mutable strings
```

Template Replacement

```
If into in onto of often on and ON.
```

Template Replacement

```
If into in onto of often on and ON.
```

```
fI into ni onto fo often no and NO.
```

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                  NSCaseInsensitiveSearch
          range:range];

if (matchRange.length > 0) {
  // found a match
}
```

String Searching

```
NSRange matchRange =
  [string rangeOfString:@"\\b(i|o)(f|n)\\b"
          options:NSRegularExpressionSearch |
                  NSCaseInsensitiveSearch
          range:range];

if (matchRange.length > 0) {
  // found a match
}
```

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];              // immutable strings
```

String Search and Replace

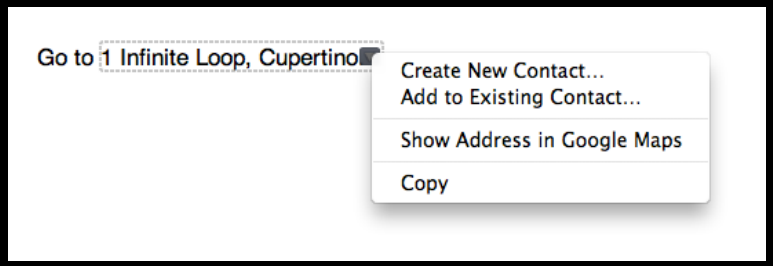
```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];              // immutable strings

[mutableString replaceOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];              // mutable strings
```

String Search and Replace

```
NSString *modifiedString =
  [string stringByReplacingOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];              // immutable strings

[mutableString replaceOccurrencesOfString:
    @"\\b(i|o)(f|n)\\b" withString:@"$2$1"
    options:NSRegularExpressionSearch
    range:range];              // mutable strings
```

Data Detectors

- Locate URLs, phone numbers, dates, addresses, etc.
- Can handle many international formats
- Made available via NSRegularExpression subclass



NSDataDetector

```
NSError *error = nil;

NSDataDetector *detector =
  [NSDataDetector
    dataDetectorWithTypes:
        NSTextCheckingTypeLink |
                    NSTextCheckingTypePhoneNumber
    error:&error];
```

NSDataDetector

```
NSError *error = nil;

NSDataDetector *detector =
  [NSDataDetector
    dataDetectorWithTypes:
        NSTextCheckingTypeLink |
                    NSTextCheckingTypePhoneNumber
    error:&error];
```

Data Detector Types

- Dates
  - `NSTextCheckingTypeDate`
- Addresses
  - `NSTextCheckingTypeAddress`
- URLs
  - `NSTextCheckingTypeLink`
- Phone numbers
  - `NSTextCheckingTypePhoneNumber`

Data Detector Iteration

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){

      [text addAttribute:NSForegroundColorAttributeName
              value:color range:[match range]];

}];
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Data Detector Iteration

```
Call 1-800-MY-APPLE or go to www.apple.com.
```

Getting Results

- NSTextCheckingResult objects with different resultType
- More NSTextCheckingResult properties:

```
@property NSDate *date;
@property NSDictionary *components;
@property NSURL *URL;
@property NSString *phoneNumber;
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Data Detector Results

```
[detector enumerateMatchesInString:string
 options:0 range:range
 usingBlock:^(NSTextCheckingResult *match,
              NSMatchingFlags flags, BOOL *stop){
  NSTextCheckingType t = [match resultType];

  if (t == NSTextCheckingTypeLink) {
    NSURL *url = [match URL];
    // do something with url
  } else if (t == NSTextCheckingTypePhoneNumber) {
    NSString *phoneNumber = [match phoneNumber];
    // do something with phone number
  }
}];
```

Additional Methods

- matchesInString:options:range:
- numberOfMatchesInString:options:range:
- firstMatchInString:options:range:
- rangeOfFirstMatchInString:options:range:

```
NSRange matchRange =
  [detector rangeOfFirstMatchInString:string
            options:0 range:searchRange];

if (matchRange.length > 0) {
  // found a match
}
```

Linguistic Tagging

Linguistic Tagging

```
Now          is          the          time     .
```

Linguistic Tagging

Now		is		the		time	.
word	space	word	space	word	space	word	punct

Linguistic Tagging

Now		is		the		time	.
word	space	word	space	word	space	word	punct
en		en		en		en	

Linguistic Tagging

Now		is		the		time	.
word	space	word	space	word	space	word	punct
en		en		en		en	
adverb		verb		determiner		noun	

Types of Tagging

- Token type
  - Word, punctuation, space, etc.
- Language
  - en, fr, de, ja, zh-Hans, etc.
- Script
  - Latn, Cyrl, Arab, Jpan, Hans, etc.

More Types of Tagging

- Lexical class
  - Noun, verb, adjective, etc.
- Named entity
  - Personal name, place name, organization name
- Lemma
  - Root form of word

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType, nil]
    options:0];

[tagger setString:string];
```

NSLinguisticTagger

```
NSLinguisticTagger *tagger =
  [[NSLinguisticTagger alloc]
    initWithTagSchemes:
        [NSArray arrayWithObjects:
            NSLinguisticTagSchemeTokenType,
            NSLinguisticTagSchemeLexicalClass,
            NSLinguisticTagSchemeNameType, nil]
    options:0];

[tagger setString:string];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
 scheme:NSLinguisticTagSchemeLexicalClass options:0
 usingBlock:^(NSString *tag, NSRange tokenRange,
              NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
 scheme:NSLinguisticTagSchemeLexicalClass options:0
 usingBlock:^(NSString *tag, NSRange tokenRange,
              NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Linguistic Tagger Iteration

```
[tagger enumerateTagsInRange:range
 scheme:NSLinguisticTagSchemeLexicalClass options:0
 usingBlock:^(NSString *tag, NSRange tokenRange,
              NSRange sentenceRange, BOOL *stop){

       if (tag == NSLinguisticTagNoun)
          [text addAttribute:NSForegroundColorAttributeName
               value:color range:tokenRange];

}];
```

Additional Methods

- tagAtIndex:scheme:tokenRange:sentenceRange:
- tagsInRange:scheme:options:tokenRanges:

```
NSString *tag =
  [tagger tagAtIndex:idx
              scheme:NSLinguisticTagSchemeLexicalClass
              tokenRange:NULL
              sentenceRange:NULL];

  if (tag == NSLinguisticTagNoun) {
      // found a noun
  }
```

Applications

- Improved text checking and correction
- Provide contextual information for words
- Identify names in text
- Improved indexing

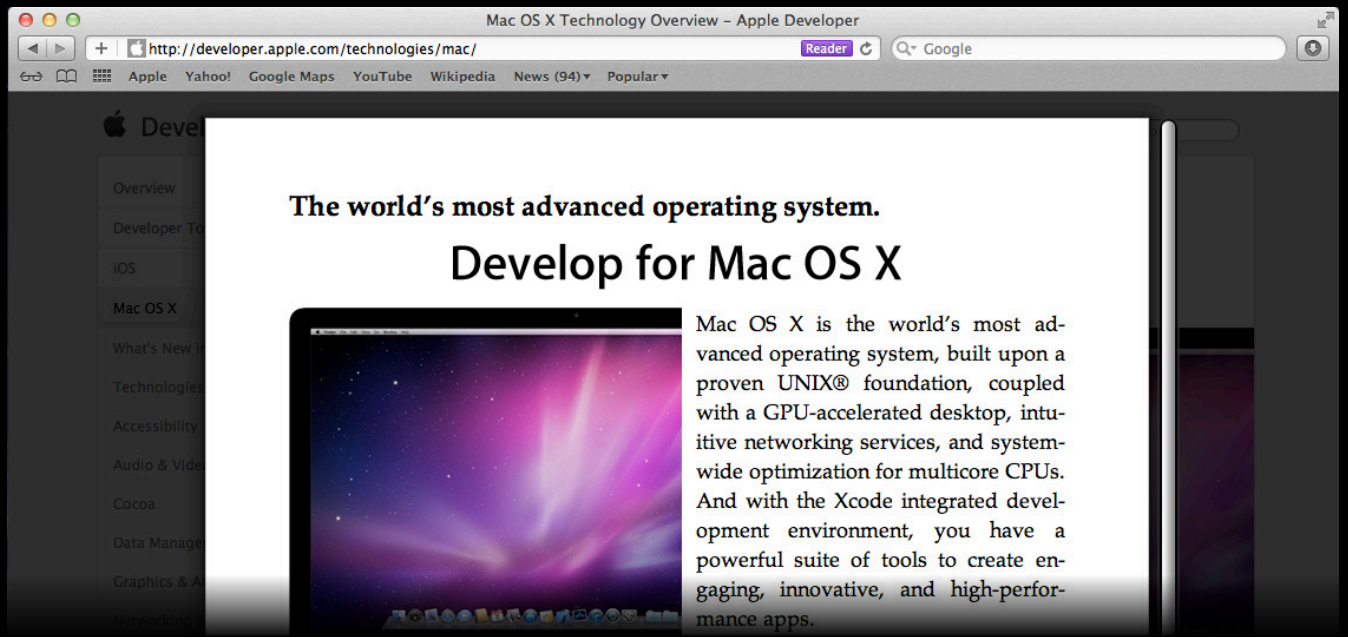
Demo

Jennifer Moore
Natural Languages Group

Hyphenation

- Available for low-level layout implementations
- Used by WebKit, iBooks

```
CFStringGetHyphenationLocationBeforeIndex()
```



Text Checking

- At the UI framework level (AppKit or UIKit)
- NSSpellChecker for AppKit
- UITextChecker for UIKit
- Used by UI text editing objects

NSSpellChecker

```
NSSpellChecker *checker =
                [NSSpellChecker sharedSpellChecker];
```

NSSpellChecker

```
NSSpellChecker *checker =
                [NSSpellChecker sharedSpellChecker];

NSArray *results =
  [checker checkString:string range:range
    types:NSTextCheckingTypeSpelling
    options:nil inSpellDocumentWithTag:0
    orthography:NULL wordCount:NULL];
```

NSSpellChecker

```
NSSpellChecker *checker =
                [NSSpellChecker sharedSpellChecker];

NSArray *results =
  [checker checkString:string range:range
    types:NSTextCheckingTypeSpelling
    options:nil inSpellDocumentWithTag:0
    orthography:NULL wordCount:NULL];

for (NSTextCheckingResult *result in results) {
  NSRange resultRange = [result range];
  // do something with misspelling
}
```

Spell Checker Results

Now is teh time for tihs.

Spell Checker Results

```
Now is teh time for tihs.
```

Spell Checker Results

```
Now is teh time for tihs.
```

Text Checking Types

- Spelling
- Grammar
- Smart quotes
- Smart dashes
- Text replacement
- Autocorrection

Additional Methods

```
NSRange misspelledRange =
  [checker checkSpellingOfString:string
    startingAt:range.location language:@"en_US"
    wrap:NO inSpellDocumentWithTag:0 wordCount:NULL];
```

Additional Methods

```
NSRange misspelledRange =
  [checker checkSpellingOfString:string
    startingAt:range.location language:@"en_US"
    wrap:NO inSpellDocumentWithTag:0 wordCount:NULL];

NSArray *guesses =
  [checker guessesForWordRange:misspelledRange
    inString:string language:@"en_US"
    inSpellDocumentWithTag:0];
```

Additional Methods

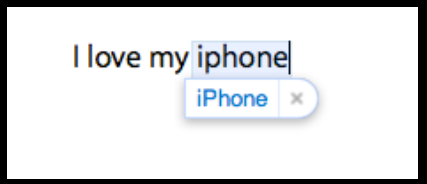
```
  [checker guessesForWordRange:misspelledRange
    inString:string language:@"en_US"
    inSpellDocumentWithTag:0];

NSString *correction =
  [checker correctionForWordRange:misspelledRange
    inString:string language:@"en_US"
    inSpellDocumentWithTag:0];
```

NSTextView

- Automatically requests checking and handles results
- All of these types can be turned on and off
- Default actions for each type
- Everything can be customized in subclasses

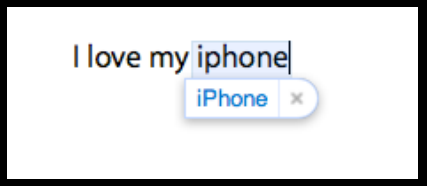
Autocorrection for Mac OS X

- New UI based on iOS autocorrection
- Can be turned on and off globally or per view
- Complete API on NSSpellChecker for custom text views



Autocorrection for Mac OS X

- New UI based on iOS autocorrection
- Can be turned on and off globally or per view
- Complete API on NSSpellChecker for custom text views
  - `showCorrectionIndicatorOfType:primaryString:alternativeStrings:forStringInRect:view:completionHandler:`
  - `dismissCorrectionIndicatorForView:`
  - `recordResponse:toCorrection:forWord:language:inSpellDocumentWithTag:`



UITextChecker

```
UITextChecker *checker =
                    [[UITextChecker alloc] init];
```

UITextChecker

```
UITextChecker *checker =
                    [[UITextChecker alloc] init];

NSRange misspelledRange =
  [checker rangeOfMisspelledWordInString:string
    range:range startingAt:range.location
    wrap:NO language:@"en_US"];
```

UITextChecker

```
UITextChecker *checker =
                    [[UITextChecker alloc] init];

NSRange misspelledRange =
  [checker rangeOfMisspelledWordInString:string
    range:range startingAt:range.location
    wrap:NO language:@"en_US"];

NSArray *guesses =
  [checker guessesForWordRange:misspelledRange
    inString:string language:@"en_US"];
```

Demo

Summary

- Text analysis based on ranges within NSStrings
- Use blocks to iterate over ranges of interest
- Different kinds of analysis provided by NSString, NSRegularExpression, NSDataDetector, and NSLinguisticTagger APIs
- Text checking for Mac OS X and iOS

More Information

Bill Dudney
Application Frameworks Evangelist
dudney@apple.com

Documentation
Mac OS X Dev Center
http://developer.apple.com/devcenter/mac

String Programming Guide for Cocoa
http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/Strings/introStrings.html

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

Related Sessions

What's New in Cocoa Touch	Presidio Tuesday 9:00–10:00AM
What's New in Cocoa	Presidio Tuesday 10:15–11:15AM
Getting Your Apps Ready for China and Other Hot Markets	Pacific Heights Friday 10:15–11:15AM

Labs

Mac OS and iOS Text Lab	Application Frameworks Lab C Thursday 4:30–6:00PM
Internationalization Lab	Application Frameworks Lab A Friday 11:30–1:30PM


WWDC2011